

# Rapid Generation of Pronunciation Dictionaries for new Domains and Languages

Zur Erlangung des akademischen Grades eines  
**Doktors der Ingenieurwissenschaften**  
von der Fakultät für Informatik  
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Tim Schlippe  
aus Ettlingen

Tag der mündlichen Prüfung:	10.12.2014
Erste Gutachterin:	Prof. Dr.-Ing. Tanja Schultz
Zweite Gutachterin:	Prof. Marelie Davel



## Acknowledgements

---

The road to a Doctorate is long and winding, and would be impossible were it not for the people met along the way. I would like to gratefully acknowledge my primary supervisor Prof. Tanja Schultz. She believed in my research and supported me with many useful discussions. Her great personality and excellent research skills had a very strong effect on my scientific career. All the travels to conferences, workshops and project meetings which are one of the best experiences in my life would be not possible without her support.

Tanja and my co-advisor Prof. Marelie Davel also deserve thanks for reading through my dissertation to provide constructive criticism. It was very kind of Marelie to take the long trip from South Africa to Karlsruhe to participate in the dissertation committee.

Special thanks to Dr. Stephan Vogel, Prof. Pascale Fung, Prof. Haizhou Li, Prof. Florian Metze, and Prof. Alan Black for their support regarding research visits for me and my students and for very close and beneficial collaboration.

Furthermore, I would like to thank all my friends and colleagues at the Cognitive Systems Lab for a great time. Their support is magnificent. Thanks to Ngoc Thang Vu, Michael Wand, Matthias Janke, Dominic Telaar, Dominic Heger, Christoph Amma, Christian Herff, Felix Putze, Jochen Weiner, Marcus Georgi, Udhyakumar Nallasamy, Dirk Gehrig for many great travel experiences and lovely activities after work. Moreover, thanks to Helga Scherer and Alfred Schmidt for their support. Even he did not work at CSL and left the Karlsruhe Institute of Technology in 2009, Prof. Matthias Wölfel was always an inspiring example. Thanks to all students I have supervised in their master, diploma, bachelor theses and student research projects for their collaboration and encouragement.

Finally, I am grateful for the patience and support of my family and friends. Especially, without my father Peter Schlippe who always supported me, my career would have never been possible. Sarah Kissel, Johannes Paulke and Norman Enghusen – thank you for always being there for me.



## Summary

---

Automatic speech recognition allows people an intuitive communication with machines. Compared to the keyboard and mouse as input modalities, automatic speech recognition enables a more natural and efficient way where hands and eyes are free for other activities. Applications, such as dictation systems, voice control, voice-driven telephone dialogue systems and the translation of spoken language, are already present in daily life.

However, automatic speech recognition systems exist only for a small fraction of the 7,100 languages in the world [LSF14] since the development of such systems is usually expensive and time-consuming. Therefore, porting speech technology rapidly to new languages with little effort and cost is an important part of research and development.

Pronunciation dictionaries are a central component for both automatic speech recognition and speech synthesis. They provide the mapping from the orthographic form of a word to its pronunciation, typically expressed as a sequence of phonemes.

This dissertation will present innovative strategies and methods for the rapid generation of pronunciation dictionaries for new domains and languages. Depending on various conditions, solutions are proposed and developed. – Starting from the straightforward scenario in which the target language is present in written form on the Internet and the mapping between speech and written language is close up to the difficult scenario in which no written form for the target language exists. We embedded many of the tools implemented in this work in the *Rapid Language Adaptation Toolkit*. Its web interface is publicly accessible and allows people to build initial speech recognition systems with little technical background. Furthermore, we demonstrate the potential of the developed methods in this thesis by many automatic speech recognition results.

## The main achievements of this thesis for languages with writing system

The main achievements of this thesis providing solutions for languages *with* writing system, can be summarized as follows:

**Rapid Vocabulary Selection:** Strategies for the collection of Web texts (*Crawling*) help to define the vocabulary of the dictionary. It is important to collect words that are time- and topic-relevant for the domain of the automatic speech recognition system. This includes the use of information from RSS Feeds and Twitter which is particularly helpful for the automatic transcription of broadcast news.

**Rapid Generation of Text Normalization Systems:** Text Normalization transfers writing variations of the same word into a canonical representation. For rapid development of text normalization systems at low cost we present methods where Internet users generate training data for such systems by simple text editing (*Crowdsourcing*).

**Analyses of Grapheme-to-Phoneme Model Quality:** Except for languages with logographic writing systems, *data-driven grapheme-to-phoneme (G2P) models* trained from existing word-pronunciation pairs can be used to directly provide pronunciations for words that do not appear in the dictionary. In analyses with European languages we demonstrate that despite varying grapheme-to-phoneme relationships word-pronunciations pairs whose pronunciations contain 15k phoneme tokens are sufficient to obtain a high consistency in the resulting pronunciations for most languages.

**Automatic Error Recovery for Pronunciation Dictionaries:** The manual production of pronunciations, e.g. collecting training data for grapheme-to-phoneme models, can lead to flawed or inadequate dictionary entries due to subjective judgements, typographical errors, and 'convention drift' by multiple annotators. We propose completely automatic methods to detect, remove, and substitute inconsistent or flawed entries (*filter methods*).

**Web-based Tools and Methods for Rapid Pronunciation Creation:** Crowdsourcing-based approaches are particularly helpful for languages with a complex grapheme-to-phoneme correspondence when data-driven approaches decline in performance or when no initial word-pronunciation pairs exist. Analyses of word-pronunciation pairs provided by Internet users on *Wiktionary (Web-derived pronunciations)*, demonstrate that many pronunciations are available for languages of various language families – even for flexions and proper names. Additionally, *Wiktionary's* paragraphs about the

word's origin and language help us to automatically detect foreign words and produce their pronunciations with separate grapheme-to-phoneme models. In the second crowdsourcing approach a tool *Keynounce* was implemented and analyzed. *Keynounce* is an online game where users can proactively produce pronunciations. Finally, we present efficient methods for a rapid and low-cost semi-automatic pronunciation dictionary development. These are especially helpful to minimize potential errors and the editing effort in a human- or crowdsourcing-based pronunciation generation process.

**Cross-lingual G2P Model based Pronunciation Generation:** For the production of the pronunciations we investigate strategies using grapheme-to-phoneme models derived from existing dictionaries of other languages, thereby substantially reducing the necessary manual effort.

### **The main achievements of this thesis for languages without writing system**

The most difficult challenge is the construction of pronunciation dictionaries for languages and dialects, which have no writing system. If only a recording of spoken phrases is present, the corresponding phonetic transcription can be obtained using a phoneme recognizer. However, the resulting phoneme sequence does not contain any information about the word boundaries. This thesis presents procedures where exploiting the written translation of the spoken phrases helps the word discovery process (*human translations guided language discovery*). The assumption is that a human translator produces utterances in the (non-written) target language from prompts in a resource-rich source language.

The main achievements of this thesis aimed at languages *without* writing system, can be summarized as follows:

**Word Segmentation from Phoneme Sequences through Cross-lingual Word-to-Phoneme Alignment:** We implicitly define word boundaries from the alignment of words in the translation to phonemes. Furthermore, our procedures contain an automatic error compensation.

**Pronunciation Extraction from Phoneme Sequences through Cross-lingual Word-to-Phoneme Alignment:** We gain a pronunciation dictionary whose words are represented by phoneme clusters and the corresponding pronunciations by their phoneme sequence. This enables an automatic “invention” of a writing system for the target language.

**Zero-resource Automatic Speech Recognition:** Finally, we tackle the task of bootstrapping automatic speech recognition systems without an a priori given language model, pronunciation dictionary, or transcribed speech data for the target language — available are only untranscribed speech and translations to resource-rich source languages of what was said.



## Zusammenfassung

---

Spracherkennung ermöglicht Menschen eine intuitive Kommunikation mit Maschinen. Im Vergleich zur Eingabe mit Tastatur und Maus hat man mithilfe der automatischen Spracherkennung eine natürlichere und effizientere Eingabemöglichkeit, bei der man die Hände und Augen für andere Tätigkeiten frei hat. Anwendungen mit Spracherkennung wie Diktiersysteme, Sprachsteuerung, natürlich-sprachliche Telefondialogsysteme und die Übersetzung gesprochener Sprache begegnen uns bereits im täglichen Leben.

Jedoch existieren Spracherkennungssysteme nur für einen kleinen Bruchteil der mehr als 7.100 Sprachen auf der Welt [LSF14], weil deren Entwicklung normalerweise teuer und zeitaufwändig ist. Daher ist die schnelle Portierung von Sprachtechnologie auf neue Sprachen mit geringem Aufwand und Kosten ein wichtiger Bestandteil der Forschung und industriellen Entwicklung.

Aussprachewörterbücher sind dabei ein zentraler Bestandteil, sowohl für die Spracherkennung als auch für die Sprachsynthese. Diese stellen die Zuordnung der orthographischen Form von Wörtern auf ihre Aussprachen dar, die typischerweise in Phonemsequenzen repräsentiert sind.

Diese Dissertation stellt innovative Strategien und Methoden für den schnellen Aufbau von Aussprachewörterbüchern für neue Anwendungsdomänen und Sprachen vor. Abhängig von verschiedenen Gegebenheiten werden Lösungsmöglichkeiten erarbeitet und präsentiert – ausgehend von dem einfachen Szenario, bei dem die Zielsprache in schriftlicher Form im Internet zu finden ist und in der eine direkte Zuordnung zwischen der gesprochenen Sprache und der Schriftsprache vorliegt, bis hin zum schwierigen Szenario, bei dem für die Zielsprache keine schriftliche Form existiert. Viele der in dieser Arbeit entwickelten Werkzeuge habe ich in das *Rapid Language Adaptation Toolkit* eingebettet, dessen Web-Interface öffentlich zugänglich ist und mit dem sogar Menschen mit geringem technischen Hintergrundwissen initiale Spracherkennungssysteme bauen können. Das Potential der in dieser Dissertation erarbeiteten Verfahren bei der Spracherkennung wird durch viele Spracherkennungsergebnisse belegt.

## Hauptergebnisse dieser Dissertation, die auf Sprachen mit Schriftsystem abzielen

Die Hauptergebnisse der Arbeit, die auf Sprachen mit Schriftsystem abzielen, lassen sich wie folgt zusammenfassen:

**Schnelle und kostengünstige Vokabularselektion:** Meine Strategien für die Sammlung von Webtexten (*Crawling*) helfen das Vokabular des Wörterbuchs zu definieren. Dabei ist es wichtig, Wörter zu sammeln, die zeitlich für die zu erkennenden Aufnahmen des Gesprochenen und die Domäne des späteren Spracherkenners relevant sind. Unter anderem werden dabei Informationen aus RSS Feeds und von Twitter verwendet, welche vor allem für die automatische Transkription von Nachrichtensendungen hilfreich sind.

**Schneller und kostengünstiger Aufbau von Textnormalisierungssystemen:** Eine Textnormalisierung überführt Schreibvariationen für dasselbe Wort in eine kanonische Repräsentation. Um mit möglichst geringem Aufwand sprachspezifische *Textnormalisierungssysteme* zu erstellen, präsentiere ich Verfahren, mit denen Internet-Nutzer durch einfaches Texteditieren Trainingsmaterial für solche Systeme generieren (*Crowdsourcing*).

**Qualitätsanalyse datengetriebener Graphem-zu-Phonem Modelle:** Mit Ausnahme von Sprachen mit logographischem Schriftsystem können *datengetriebene Graphem-zu-Phonem (G2P)-Modelle* aus existierenden Wort-Aussprache-Paaren direkt trainiert werden. Mit diesen kann man dann automatisch Aussprachen für weitere Wörter erzeugen. In Analysen mit europäischen Sprachen zeige ich, dass trotz unterschiedlicher Graphem-zu-Phonem-Zuordnung Trainingsmaterial mit 15.000 Phonemen und ihren zugehörigen Graphemen gesammelt werden sollte. Diese Mengen reichen um für die meisten Sprachen eine hohe Konsistenz bei den resultierenden Aussprachen zu erhalten.

**Automatische Detektion und Behebung von Fehlern und Inkonsistenzen in den Aussprachen:** Das manuelle Erzeugen von Aussprachen, z.B. beim Sammeln von Trainingsdaten für die Graphem-zu-Phonem-Modelle, kann im Fall von mehreren Personen aufgrund subjektiver Meinungen, typographischer Fehler und Abweichungen in der Vereinbarung zu inkonsistenten und fehlerhaften Aussprachen führen. Um solche problematischen Aussprachen aufzufinden und zu ersetzen, habe ich vollautomatische datengetriebene Verfahren (*Filtermethoden*) entworfen.

**Web-basierte Tools und Methoden für die Aussprachengenerierung:** Crowdsourcing-basierte Verfahren eignen sich besonders für Spra-

chen, deren Graphem-zu-Phonem-Zuordnung so komplex ist, dass datengetriebene Verfahren nicht gut funktionieren oder keine initialen Wort-Aussprache-Paare dafür existieren. Meine Analysen von Aussprachen, die Internet-Nutzer mit Crowdsourcing auf Webseiten zur Verfügung stellen (*Web-derived pronunciations*), zeigen, dass speziell in *Wiktionary* für Wörter und deren Flexion viele Aussprachen in Sprachen verschiedener Sprachfamilien vorhanden sind. Zusätzlich nutze ich Information in den Abschnitten über die Herkunft und die Sprache der Wörter in *Wiktionary* zur automatischen Detektion und Behandlung von Wörtern, die aus anderen Sprachen stammen und deren Aussprachen nicht den Ausspracheregeln der Zielsprache folgen. Bei dem zweiten Crowdsourcing-basierten Ansatz wurde *Keynounce* implementiert und analysiert, ein *Online-Spiel*, mit dem Spieler für gegebene Wörter Aussprachen durch das Aneinanderreihen von Phonemen per Mausklick erzeugen können. Schließlich habe ich ökonomische Verfahren für die semi-automatische Generierung von Aussprachen entwickelt, mit denen man potentielle Fehler und die Tipparbeit von Muttersprachlern, Crowdsourcern oder Linguisten in der manuellen Eingabe von Aussprachen reduzieren kann.

**Cross-linguale Graphem-zu-Phonem-Model-basierte Aussprache-generierung:** Weiterhin habe ich Strategien untersucht, um mithilfe von Graphem-zu-Phonem-Modellen aus verwandten Sprachen den manuellen Aufwand zu reduzieren.

### **Hauptergebnisse dieser Dissertation, die auf Sprachen ohne Schriftsystem abzielen**

Die schwierigste Herausforderung ist der Bau von Wörterbüchern für Sprachen und Dialekte, für die kein Schriftsystem existiert. Falls ausschließlich eine Aufnahme des Gesprochenen vorhanden ist, kann mithilfe eines Phonemkenners die zugehörige Phonemsequenz erkannt werden, die jedoch keine Wortgrenzen enthält. In dieser Dissertation wurden deshalb Verfahren erarbeitet, bei denen man die geschriebene Übersetzung des Gesprochenen ausnutzt, um die Wortgrenzen zu definieren. Dazu wird vorausgesetzt, dass ein Sprecher geschriebene Sätze aus der Quellsprache in eine schriftlose Zielsprache übersetzt und ausspricht.

Die Hauptergebnisse der Arbeit, die auf Sprachen ohne Schriftsystem abzielen, lassen sich wie folgt zusammenfassen:

**Wortsegmentierung aus Phonemsequenzen mithilfe von Cross-Lingual Word-to-Phoneme Alignment:** Meine Verfahren ermöglichen die

implizite Definition von Wortgrenzen aus der Zuordnung der Wörter in der Übersetzung zu den Phonemen (*Cross-lingual Word-to-Phoneme Alignment*). Außerdem kompensieren die Verfahren automatisch Fehler der Phonemerken-  
nung.

**Extraktion von Aussprachen aus Phonemsequenzen mithilfe von Cross-Lingual Word-to-Phoneme Alignment:** Am Ende gewinne ich ein Aussprachewörterbuch, dessen Wörter durch Phonemcluster repräsentiert sind und deren Aussprachen durch deren Phonemsequenz. Dies ermöglicht das automatische “Erfinden” eines Schriftsystems für die Zielsprache.

**“Zero-resource” Spracherkennung:** Schließlich befasse ich mich noch mit der Herausforderung, automatische Spracherkennungssysteme ohne gegebenes Sprachmodell, Aussprachewörterbuch oder transkribierte Sprachdaten zu erstellen. – Nur Audiodaten des Gesprochenen ohne Transkription und deren Übersetzungen in andere Ressourcen-reiche Sprachen liegen vor.

# Contents

---

<b>1</b>	<b>Introduction and Motivation</b>	<b>1</b>
1.1	Motivation . . . . .	1
1.2	Automatic Speech Recognition and the Role of the Pronunciation Dictionary . . . . .	3
1.3	Rapid Adaptation of Automatic Speech Recognition Systems to new Domains and Languages . . . . .	14
1.4	Pronunciation Generation . . . . .	19
1.4.1	The International Phonetic Alphabet . . . . .	19
1.4.2	Correspondence between Graphemes and Phonemes . . . . .	21
1.4.3	Manual Pronunciation Generation . . . . .	23
1.4.4	Automatic Pronunciation Generation . . . . .	24
1.4.5	Semi-Automatic Pronunciation Generation . . . . .	27
1.4.6	Graphemic Pronunciation Dictionaries . . . . .	27
1.5	Structure of this Thesis . . . . .	29
<b>2</b>	<b>Challenges of Dictionary Generation for new Domains and Languages</b>	<b>31</b>
2.1	Vocabulary Selection and Text Normalization . . . . .	31
2.1.1	Vocabulary . . . . .	31
2.1.2	Text Normalization . . . . .	35
2.2	Pronunciation Generation . . . . .	36
2.2.1	Lexical Resources on the Web . . . . .	38
2.2.2	No Lexical Resources of the Target Language . . . . .	39
2.2.3	In-house Lexical Resources . . . . .	39
2.2.4	Non-Written Languages . . . . .	40
2.2.5	Erroneous Pronunciations . . . . .	40
<b>3</b>	<b>Vocabulary Selection and Text Normalization</b>	<b>43</b>
3.1	Vocabulary Selection . . . . .	44
3.1.1	Traditional Methods . . . . .	44
3.1.2	Supervised and Unsupervised Vocabulary Adaptation . . . . .	45
3.1.3	Summary . . . . .	59

3.2	Text Normalization . . . . .	59
3.2.1	Traditional Methods . . . . .	60
3.2.2	SMT-based Text Normalization through Crowdsourcing . . . . .	61
3.2.3	Summary . . . . .	69
3.3	Vocabulary Extraction for Non-Written Languages . . . . .	69
3.3.1	Challenges . . . . .	70
3.3.2	Related Work . . . . .	71
3.3.3	Word Segmentation . . . . .	72
3.3.4	Alignment and Word Segmentation Performance . . . . .	77
3.3.5	Summary . . . . .	82
<b>4</b>	<b>Pronunciation Quality Assurance</b>	<b>83</b>
4.1	Grapheme-to-Phoneme Model Quality . . . . .	84
4.1.1	Consistency and Complexity . . . . .	85
4.1.2	Automatic Speech Recognition Performance . . . . .	87
4.1.3	Summary . . . . .	89
4.2	Detection and Recovery of Inconsistencies and Errors . . . . .	89
4.2.1	Detection of Inconsistencies and Errors . . . . .	89
4.2.2	Recovery of Inconsistencies and Errors . . . . .	91
4.2.3	Summary . . . . .	94
4.3	Phoneme-Level Combination . . . . .	95
4.3.1	Single G2P Converter Output . . . . .	96
4.3.2	Combination of G2P Converter Output . . . . .	98
4.3.3	Adding Web-driven G2P Converter Output . . . . .	99
4.3.4	Automatic Speech Recognition Experiments . . . . .	100
4.3.5	Summary . . . . .	104
<b>5</b>	<b>Crowdsourcing-based Pronunciation Generation</b>	<b>105</b>
5.1	Pronunciations from the Web . . . . .	106
5.1.1	Automatic Pronunciation Extraction . . . . .	110
5.1.2	Evaluation of the Web-derived Pronunciations . . . . .	111
5.1.3	Recovery of Inconsistencies and Errors . . . . .	121
5.1.4	Summary . . . . .	122
5.2	An Online Game for Pronunciation Generation . . . . .	123
5.2.1	Related Approaches . . . . .	124
5.2.2	Design and Backend Implementation of Keynounce . . . . .	126
5.2.3	Performance with Amazon Mechanical Turk . . . . .	131
5.2.4	Performance of the Keynounce Experiment . . . . .	133
5.2.5	Improving Pronunciation Quality . . . . .	136
5.2.6	Summary . . . . .	137
5.3	Semi-Automatic Pronunciation Generation . . . . .	138

---

5.3.1	Traditional Methods . . . . .	138
5.3.2	Semi-Automatic Pronunciation Generation Strategy . .	139
5.3.3	Word Selection Strategy . . . . .	141
5.3.4	Iterative G2P Model Building . . . . .	142
5.3.5	Single G2P Converters . . . . .	144
5.3.6	Combination of G2P Converter Output . . . . .	148
5.3.7	Resources for Initial Pronunciations . . . . .	149
5.3.8	Summary . . . . .	152
<b>6</b>	<b>Cross-lingual G2P Model-based Pronunciation Generation</b>	<b>155</b>
6.1	Cross-lingual Pronunciation Generation Strategy . . . . .	155
6.2	Which Languages are Appropriate? . . . . .	160
6.2.1	Language Characteristics . . . . .	160
6.2.2	Predicting the Accuracy . . . . .	162
6.3	Towards Universal Grapheme-to-Phoneme Conversion . . . . .	165
6.4	Summary . . . . .	167
<b>7</b>	<b>Pronunciation Generation for Foreign Words and Accents</b>	<b>169</b>
7.1	Pronunciation Generation for Foreign Words . . . . .	170
7.1.1	Traditional Methods . . . . .	171
7.1.2	Experimental Setup . . . . .	172
7.1.3	Detection of Foreign Words . . . . .	174
7.1.4	Summary . . . . .	196
7.2	Lexical Adaptation for Non-Native Speakers . . . . .	196
7.2.1	Related Work . . . . .	198
7.2.2	Pronunciation Variants Generation . . . . .	200
7.2.3	Combination of Lexical and Acoustic Model Adaptation	204
7.2.4	Summary . . . . .	205
<b>8</b>	<b>Dictionary Generation for Non-Written Languages</b>	<b>207</b>
8.1	Word Pronunciation Extraction Algorithm . . . . .	207
8.1.1	Source Word Dependent Clustering . . . . .	210
8.1.2	Source Word Independent Clustering . . . . .	210
8.2	Evaluation Metrics . . . . .	212
8.3	Evaluation of Extracted Pronunciations . . . . .	213
8.4	Which Source Language Is Favourable? . . . . .	221
8.5	Which Words Are Extracted Correctly? . . . . .	223
8.6	Automatic Speech Recognition in the Target Language . . . . .	225
<b>9</b>	<b>Conclusions and Future Directions</b>	<b>229</b>

---

9.1	Contributions . . . . .	229
9.1.1	Strategies For Rapid Vocabulary Selection and Text Normalization . . . . .	230
9.1.2	Techniques For Rapid Generation Of Pronunciations .	231
9.2	Potential Future Research Directions . . . . .	234
9.2.1	More Robust Grapheme-to-Phoneme Conversion . . . .	235
9.2.2	Language Technology for Non-Written Languages . . .	235
	<b>Bibliography</b>	<b>260</b>
	<b>My Publications</b>	<b>261</b>



## List of Figures

---

1.1	Block diagram of automatic speech recognition. . . . .	4
1.2	The levels of the language analysis. . . . .	5
1.3	Connected HMMs building the word 'be'. . . . .	7
1.4	Binary phonetic decision tree of the phone 't' in different contexts. . . . .	8
1.5	Search graph. . . . .	12
1.6	Possible errors of an automatic speech recognition system. . .	14
1.7	Steps to build automatic speech recognition and synthesis systems with the Rapid Language Adaptation Toolkit. . . . .	18
1.8	IPA vowel chart [IPA99]. . . . .	20
1.9	IPA consonant chart [IPA99]. . . . .	20
1.10	Sequitur G2P model performance over n-gram order $M$ . . . . .	26
1.11	Structure of the experiments and results. . . . .	29
2.1	Basic scenario for English-Klingon (non-written). . . . .	33
2.2	Basic idea of how resources are proceeded. . . . .	34
2.3	Reducing the vocabulary size with text normalization [SK06].	36
2.4	Dictionary generation for non-written languages on a German-English example. . . . .	41
3.1	Out-of-vocabulary rate (%) over days of text crawling. . . . .	48
3.2	Word error rate reduction (%) for five Eastern European languages. . . . .	51
3.3	Unsupervised language model and vocabulary adaptation. . .	52
3.4	Word error rates (%) of the baseline systems on the broadcast news task. . . . .	54
3.5	Word error rates (%) with language models containing RSS Feeds-based text data from different periods. . . . .	55
3.6	Word error rate (%) with language models containing RSS Feeds-related text compared to random text data. . . . .	56
3.7	Word error rates (%) for $Q$ -LM and $GP$ -LM. . . . .	57

3.8	Out-of-vocabulary rates (%) for <i>Q-LM</i> and <i>GP-LM</i> before and after vocabulary adaptation. . . . .	58
3.9	Web-based user interface for text normalization. . . . .	62
3.10	Text normalization systems. . . . .	63
3.11	Edit distance (%) and perplexity over amount of training data. . . . .	66
3.12	Edit distance reduction with <i>iterative-SMT/-hybrid</i> . . . . .	68
3.13	Word alignment between English and Spanish. . . . .	73
3.14	Generative process in IBM Model 3. . . . .	74
3.15	Generative process in Model 3P. . . . .	76
3.16	English-Spanish alignment in Model 3P. . . . .	76
3.17	Phoneme error rate over $\lambda$ for the Spanish phoneme recognizer. . . . .	78
3.18	Alignment error rate over phoneme error rate on BTEC (source language: English, target language: Spanish). . . . .	79
3.19	Word segmentation quality over phoneme error rate on BTEC (source language: English, target language: Spanish). . . . .	80
3.20	Word segmentation quality over phoneme error rate on BTEC (source language: Spanish, target language: English). . . . .	81
4.1	Grapheme-to-phoneme consistency across 10 languages. . . . .	86
4.2	<i>GlobalPhone</i> grapheme-to-phoneme model complexity. . . . .	87
4.3	Correlation between consistency and word error rate decrease. . . . .	88
4.4	2-stage filtering. . . . .	92
4.5	Phoneme error rate (%) of single grapheme-to-phoneme converter outputs over amount of training data. . . . .	97
4.6	Edit distances at the phoneme level (%) between grapheme-to-phoneme converter outputs (en / de / fr / es). . . . .	97
4.7	Phoneme error rate change (%) with <i>Phoneme Level Combination</i> using converters trained without and with Web-derived pronunciations. . . . .	98
4.8	Word error rate (%) with dictionaries from single grapheme-to-phoneme converter outputs over amount of training data. . . . .	101
4.9	Word error rate (%) change over training data size with and without Web-derived data for early ( <i>PLC</i> ) and late ( <i>CNC</i> ) fusion. . . . .	102
4.10	Correlation between phoneme error rate/I+D and word error rate. . . . .	102
5.1	Growth of <i>Wiktionary</i> entries over several years [wik14]. . . . .	108
5.2	English <i>Wiktionary</i> page for the word “vitamin”. . . . .	109
5.3	Pronunciation diversity in English and French <i>Wiktionary</i> . . . . .	109
5.4	<i>Wiktionary</i> editions with more than 1k pronunciations. . . . .	110

---

5.5	Ratio of searched words (first bar) and retrieved <i>Wiktionary</i> pronunciations (second bar). . . . .	111
5.6	Ratio of searched Chinese characters (segments) (first bar) and retrieved <i>Wiktionary</i> pronunciations (second bar). . . . .	112
5.7	Coverage for international city and country names. . . . .	113
5.8	Consistency of <i>GP</i> . . . . .	116
5.9	Consistency of <i>wikt</i> . . . . .	116
5.10	Consistency of <i>WiktOnGP</i> . . . . .	117
5.11	<i>Wiktionary</i> grapheme-to-phoneme model consistency variances. . . . .	118
5.12	<i>Wiktionary</i> grapheme-to-phoneme model complexity. . . . .	119
5.13	<i>GlobalPhone</i> grapheme-to-phoneme model complexity. . . . .	119
5.14	Game Peekaboom [vALB06]. . . . .	125
5.15	Keynounce user interface. . . . .	127
5.16	Welcome screen in Keynounce. . . . .	129
5.17	Final screen in Keynounce. . . . .	130
5.18	Phoneme error rate, time per word and number of users per session with German and English words. . . . .	133
5.19	Arithmetic mean and median time needed for single German words in first session. . . . .	135
5.20	Semi-automatic pronunciation generation. . . . .	140
5.21	Word selection strategies. . . . .	142
5.22	Cumulated phoneme error rate (%) over Sequitur n-gram order <i>M</i> . . . . .	145
5.23	Cumulated phoneme error rate (%) over Phonetisaurus context width. . . . .	146
5.24	Comparison of grapheme-to-phoneme converter strategies on English. . . . .	151
6.1	Correlation between characteristics (grapheme, digraph, tri-graph, phoneme, vowel, consonant coverage) and the phoneme accuracy. . . . .	163
6.2	Correlation between characteristics (diphone, triphone coverage) and the phoneme accuracy. . . . .	164
6.3	Correlation between predicted and the real phoneme accuracy. . . . .	165
6.4	Strategy for universal grapheme-to-phoneme conversion. . . . .	165
7.1	Foreign words in different word lists . . . . .	175
7.2	Anglicism detection system. . . . .	176
7.3	Classification with the <i>Grapheme Perplexity Feature</i> . . . . .	177
7.4	Detection performance in relation to an absolute perplexity threshold. . . . .	179

7.5	Detection performance (F-score) with different dictionary sizes for default threshold of zero. . . . .	182
7.6	Portion of words found in each dictionary. . . . .	184
7.7	Entry of German <i>Wiktionary</i> containing a paragraph about the word's origin and language. . . . .	185
7.8	Classification with the <i>Google Hits Count Feature</i> . . . . .	187
7.9	Anglicism detection performance (F-score) of all features. . . . .	190
7.10	Precision-recall chart of all features on the Microsoft-de and Spiegel-de test sets. . . . .	190
7.11	F-scores (%) of the feature combinations and the best single features. . . . .	193
7.12	Relative F-score change (%) of <i>Voting</i> if one feature is left out. . . . .	194
7.13	Performance (F-score) of <i>Voting</i> after removing difficult word categories from the test sets. . . . .	195
7.14	Statistical machine translation based pronunciation variant generation. . . . .	200
7.15	Parallel corpus for the phoneme-level statistical machine translation-based pronunciation variant generation. . . . .	202
8.1	Pronunciation extraction with German-English. . . . .	209
8.2	Source word dependent clustering. . . . .	210
8.3	German-English example with correct word boundaries but incorrect alignment. . . . .	211
8.4	Source word independent clustering. . . . .	211
8.5	Mapping between word labels and written words for evaluation. . . . .	212
8.6	Log-log plot of the word distribution in the ESV Bible. . . . .	214
8.7	Evaluation metrics over $k$ for source word independent clustering (source language: Spanish ( <i>es3</i> )). . . . .	220
8.8	Layout for Table 8.6 and 8.7. . . . .	221
8.9	Mean word frequency over phoneme error rate (Spanish-English). . . . .	223
8.10	Phoneme error rate over word length in phonemes (Spanish-English). . . . .	224
8.11	Training set generation for language modeling. . . . .	225
9.1	Pronunciation Generation. . . . .	232

## List of Tables

---

1.1	Word error rates (%) with phoneme-based and grapheme-based dictionaries. . . . .	28
3.1	Initial text corpus size for five Eastern European languages. . . . .	47
3.2	Additional data from various websites for five Eastern European languages. . . . .	49
3.3	Out-of-vocabulary rate (OOV) and perplexity (PPL) reductions for five Eastern European languages. . . . .	50
3.4	Quality of baseline language models on the broadcast news task.	53
3.5	Relative word error rate reduction (%) for the last five shows with our text collection and decoding strategy. . . . .	58
3.6	Language-independent and -specific text normalization. . . . .	64
3.7	Amazon Mechanical Turk experiments: Perplexity results. . . . .	67
3.8	Amazon Mechanical Turk experiments: Time and cost. . . . .	68
3.9	Accuracy improvement using <i>Model 3P</i> on the BTEC corpus. . . . .	80
3.10	F-score improvement using <i>Model 3P</i> on the BTEC corpus. . . . .	80
3.11	Accuracy improvement using <i>Model 3P</i> instead of GIZA++. . . . .	82
4.1	Average number of phones in a pronunciation. . . . .	85
4.2	Word error rates (%) of systems with dictionaries built completely with grapheme-to-phoneme generated pronunciations. . . . .	88
4.3	Word error rates (%) for Hausa with and without filtered pronunciation dictionary. . . . .	93
4.4	Mixed error rates (%) on the <i>SEAME</i> Mandarin-English Code-Switch Corpus development set. . . . .	94
5.1	The ten largest Wiktionary language editions, based on [Wik12].	107
5.2	Absolute and relative amount of <i>Wiktionary</i> pages containing pronunciations. . . . .	112
5.3	Amount of compared pronunciations, percentage of identical ones and amount of new pronunciation variants. . . . .	114
5.4	Consistency check setup. . . . .	115

5.5	Word error rates (%) with dictionaries built completely with grapheme-to-phoneme model generated pronunciations (w/o filtering). . . . .	121
5.6	Word error rates (%) with dictionaries built completely with grapheme-to-phoneme generated pronunciations (with filtering). . . . .	122
5.7	Phoneme error rate (%) over all words from German players in different word ranges. . . . .	136
5.8	Phoneme error rate (%) over all words from German players who worked on one or two sessions (5-10 words). . . . .	137
5.9	Phoneme error rate (%) over all words from German players who worked on more than two sessions (5-10 words). . . . .	137
5.10	Strategies for grapheme-to-phoneme model retraining. . . . .	143
5.11	Sequitur context width optimization: Total number of edits. . . . .	144
5.12	Sequitur context width: Time estimations on 10k dictionaries. . . . .	145
5.13	Phonetisaurus context width: Total number of edits on 10k dictionaries . . . . .	147
5.14	Cumulated phoneme error rates (cPERs) (%) for single grapheme-to-phoneme converters. . . . .	149
5.15	Phoneme error rate (%) and optimal cross-over for initial pronunciations. . . . .	151
5.16	Reductions in total number of human edits. . . . .	152
5.17	Reductions in cumulated phoneme error rate (cPER) (%). . . . .	152
6.1	Cross-lingual pronunciation production for <i>bir</i> . . . . .	156
6.2	Comparison of pronunciations from grapheme-based vs. phoneme-based dictionary. . . . .	157
6.3	Effort (#rules) and performance (in terms of PER, WER) for a Ukrainian dictionary using cross-lingual rules. . . . .	158
6.5	Comparison of cross-lingual generated dictionaries and grapheme-based dictionaries. . . . .	160
6.6	Performance of universal grapheme-to-phoneme conversion with Ukrainian as target language. . . . .	167
7.1	Annotation guidelines for the German test sets. . . . .	173
7.2	Detection performance (F-score) with different thresholds for the grapheme perplexity difference. . . . .	178
7.3	Performance of the Grapheme Perplexity Feature. . . . .	179
7.4	Detection performance (F-score) with different thresholds for the grapheme-to-phoneme model confidence difference. . . . .	181
7.5	Performance of the G2P Confidence Feature. . . . .	182

---

7.6	Detection performance of English and Matrix Language Hunspell Lookup. . . . .	183
7.7	Percentage of words found in Wiktionary. . . . .	186
7.8	Performance of the Wiktionary Lookup Feature. . . . .	186
7.9	Estimated size of the Web in different languages. . . . .	189
7.10	Performance of the Google Hit Counts Feature. . . . .	189
7.11	Portion false positives (Non-Anglicisms wrongly detected) . .	191
7.12	Detection performance of <i>Voting</i> with different thresholds. . .	192
7.13	Phoneme error rate (%) of different approaches to generate pronunciations for the German IT corpus. . . . .	196
7.14	Word error rate (%) of the baseline system tested with accented speech. . . . .	202
7.15	Word error rate (%) on test sets with lexical adaptation. . . .	203
7.16	Word error rate (%) on development sets with lexical and acoustic model adaptation. . . . .	204
7.17	Word error rate (%) on test sets with lexical and acoustic model adaptation. . . . .	205
8.1	Overview of the used Bible translations. . . . .	215
8.2	Acoustic models $AM_{13.1}$ and $AM_{45.1}$ applied in a phoneme recognizer on $EN_{filt}$ . . . . .	216
8.3	Dictionary evaluation with source word dependent clustering. .	217
8.4	Dictionary evaluation with source word independent clustering.	217
8.5	Error recovery through source word dependent and independent clustering (source language: Spanish ( <i>es3</i> )). . . . .	220
8.6	Absolute correlation coefficients between our evaluation metrics and different influencing factors for source word dependent clustering. . . . .	221
8.7	Absolute correlation coefficients between our evaluation metrics and different influencing factors for source word independent clustering. . . . .	222
8.8	Word error rate and multiWER. . . . .	225
8.9	Automatic speech recognition performance with a speaker dependent acoustic model ( $R_{13.1}$ , $AM_{13.1}$ ). . . . .	226
8.10	Automatic speech recognition performance with a speaker independent, corpus-mismatched acoustic model ( $R_{45.1}$ , $AM_{45.1}$ ). . . . .	227





# Introduction and Motivation

---

## 1.1 Motivation

The performance of speech and language processing technologies has improved dramatically over the past two decades with an increasing number of systems being deployed in a large variety of applications. Those applications are included in sectors such as automotive, healthcare, military, telephony, education, and daily life.

Speech driven in-car systems like Ford's Syn<sup>1</sup> and GMC's IntelliLink<sup>2</sup> enable to initiate phone calls, select radio stations or play music from compatible smartphones, MP3 players and flash drives by using natural speech. Automatic speech recognition leads to time and cost savings in clinical documentation by automatically turning clinician dictations into formatted documents, e.g. with Dragon Medical<sup>3</sup>. Examples for military applications are pilot's associate systems, air traffic control training systems, battle management command and control support systems, and spoken language translation systems [Wei91]. With Interactive Voice Response (IVR) components, automated telephone information systems speak to a caller using a combination of fixed voice menus and data extracted from databases in real time. The caller responds by speaking words or short phrases or pressing digits on the keypad.

---

<sup>1</sup><http://www.ford.com/technology/sync>

<sup>2</sup><http://www.gmc.com/intellilink-infotainment-system.html>

<sup>3</sup><http://www.nuance.com/products/physician-speech-recognition-solutions/index.htm>

An example is the IBM WebSphere Voice Response<sup>4</sup>. Furthermore, speech technology is used in computer-assisted language learning systems. For example, many studies [HLMW08, HLQM09, HMA<sup>+</sup>99, TDK04] approached the problem of mispronunciation detection by using the recognized phoneme sequence information directly. In daily life, intelligent personal assistants like Siri<sup>5</sup>, with natural language user interfaces, are used on smartphones and tablet computers to answer questions, send requests to a set of Web services, and make recommendations.

Speech technology potentially allows everyone to participate in today's information revolution. Moreover, it can bridge language barriers and facilitates worldwide business activities, simplifies life in multilingual communities, and alleviates humanitarian missions [Bre14]. For example, when a devastating earthquake hit Haiti in 2010 there was a demand for speech technology since many helpers neither spoke French, nor Haitian Creole; a lot of time and human resources were unnecessarily spent on translation issues instead of being used for more important issues like vetting the injured, searching missing people or rebuilding the infrastructure [Bre14].

With some 7,100 languages in the world, for decades data resources such as text files, transcribed speech or pronunciation dictionaries have been only available in the most economically viable languages and only in the last few years efforts have been made to tackle more languages. Africa itself has more than 2,000 languages [HN00] plus many different accents, e.g. there are more than 280 languages in Cameroon [LSF14]. For only a few of Africa's many languages, speech processing technology has been analyzed and developed so far [PGMSPL11, Ade09, SDV<sup>+</sup>12].

The biggest challenge today is still to rapidly port speech processing systems to new languages with low human effort and at reasonable cost. The reasons are that nowadays the majority of state-of-the-art automatic speech processing systems heavily rely on large amounts of data which are necessary to train such systems. Transcribed speech resources, large amounts of text for language modeling, and pronunciation dictionaries are of great importance to create such systems [SSVS12]. However, collecting them is usually not feasible. Authors in [SK06] estimate that transcription of one hour conversational speech data can take up to 20 hours of effort. Therefore, in recent years, automatic speech recognition research shifted its focus to low- and under-resourced settings [BBKS14]. Consequently, also less prevalent languages are addressed, e.g. by exploring new ways to col-

---

<sup>4</sup><http://www-03.ibm.com/software/products/en/voice-response-aix>

<sup>5</sup><http://www.apple.com/ios/siri/>

lect data [GJK<sup>+</sup>09, CCG<sup>+</sup>09, SOS14, GABP11], using grapheme-based approaches [KN02], or sharing information across languages [SW01].

Over the past years, the World Wide Web has been increasingly used as a text data source for rapid adaptation of automatic speech recognition systems to new languages and domains at low cost; e.g. websites are crawled to collect texts to build language models. Moreover, prompts which can be read by native speakers to receive transcribed audio data, are extracted from the crawled text [SBB<sup>+</sup>07].

The creation of pronunciation dictionaries can be time-consuming and expensive if they are manually produced by language experts. The dictionaries provide the mapping from the orthographic form of a word to its pronunciation, which is useful in both text-to-speech and automatic speech recognition systems. They are used to train speech processing systems by describing the pronunciation of words according to appropriate units, typically phonemes [MD07]. In this dissertation we present innovative strategies and methods for the rapid generation of pronunciation dictionaries for new domains and languages.

In the following sections, we give a short introduction to automatic speech recognition and describe the role of the pronunciation dictionary in the training and the decoding procedures.

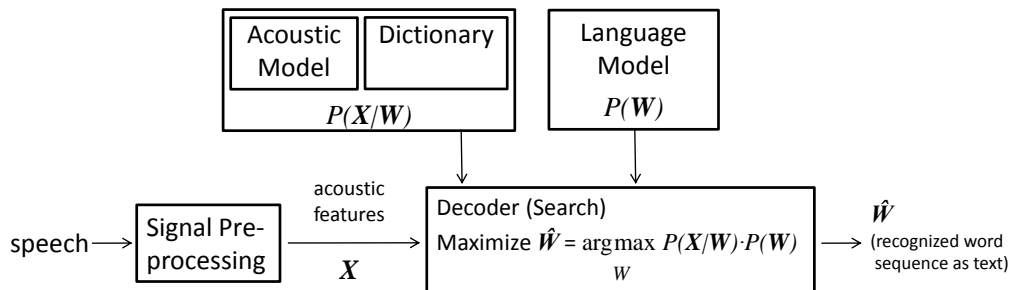
## 1.2 Automatic Speech Recognition and the Role of the Pronunciation Dictionary

### Automatic Speech Recognition

The fundamental problem of automatic speech recognition (ASR) is to find the most likely word sequence given a speech recording [HAH01]. Factors such as speaker variability, noisy environment and different properties of the recording equipment have negative influence on the recognition performance. The following equation (1.1) is transformed using Bayes' rule and summarizes the mathematical model commonly used for large vocabulary continuous speech recognition (LVCSR):

$$\hat{W} = \arg \max_W P(W|X) = \arg \max_W \frac{P(X|W) \cdot P(W)}{P(X)} = \arg \max_W P(X|W) \cdot P(W) \quad (1.1)$$

As a result of digital signal processing, the acoustic signal is represented as a sequence of feature vectors  $X = x_1 x_2 \dots x_n$  which capture the most important information of the speech signal for the classification task. The goal is to estimate the most likely word sequence  $\hat{W} = \hat{w}_1 \hat{w}_2 \dots \hat{w}_m$  depending on the prior probability  $P(W)$ , provided by a language model, and the conditional probability  $P(X|W)$  given by an acoustic model. When computing the most likely word sequence, the denominator from the Bayes' rule  $P(X)$  is not considered as it does not play a role in the maximization of the function. Since the language model usually works at the word level and the acoustic model with acoustic units such as phonemes, a pronunciation dictionary is required to bridge the gap between words and phonemes. The pronunciation dictionary is a mapping between words and their pronunciations. To find the word sequence with the highest probability, a search strategy has to be applied. The most widespread search algorithm in automatic speech recognition is the Viterbi search. Figure 1.1 illustrates a block diagram of a typical automatic speech recognition system.

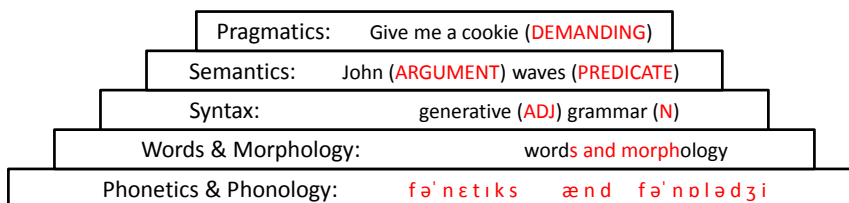


**Figure 1.1** – Block diagram of automatic speech recognition.

## Natural Language Structure

Language and speech are complex phenomena and can be analyzed at different levels [JM00]. Figure 1.2 demonstrates the levels of language analysis.

*Phonetics* examines how language sounds are produced. *Phonology* studies the rules of how a language sounds, and how and when certain sounds can



**Figure 1.2** – The levels of the language analysis.

be combined. A *phoneme* is the smallest contrastive unit in the phonology of a language, while a *phone* is the acoustic realization of a phoneme [O'S87]. The International Phonetic Alphabet (IPA) [IPA99] offers a standard way of describing and categorizing the sounds produced when speaking. We will introduce IPA in Section 1.4.1.

Spoken *words* are built from phonemes. *Morphology* studies the structure of a given language's morphemes and other linguistic units. Different languages have different morphological rules and therefore different degrees of morphological complexity.

At the higher levels, the language analysis deals with syntax, semantics and pragmatics. *Syntax* is the study of the principles and processes by which sentences are constructed in particular languages. *Semantics* refers to the meaning level of language. *Pragmatics* studies the ways in which context contributes to meaning. Such a context includes for example structural and linguistic knowledge of the speaker and listener, the context of the utterance, any pre-existing knowledge about those involved, and the inferred intent of the speaker. Although there is research on how syntactic, semantic and pragmatic knowledge can improve the automatic speech recognition performance, these areas are of greater interest in fields like natural language understanding and many branches of linguistics. A way of including simple syntactic and semantic knowledge in the process of automatic speech recognition is to use the word context information as done with the application of the language model.

## Signal Preprocessing

Goal of the signal preprocessing is to extract features from the speech signal providing a compact representation of speech. The speech signal is converted into a sequence of feature vectors. The feature vectors  $X = x_1x_2\dots x_n$  from equation 1.1 are calculated by dividing the speech signal into frames (typically  $16ms$ ). It is a common practice to let the blocks overlap (e.g.  $10ms$ ).

There are different ways of extracting features for automatic speech recognition. In LVCSR, commonly used features are the Mel-Frequency Cepstral Coefficients (MFCCs). MFCCs are the representation of the short-term cepstrum of a sound wave, transferred on the Mel scale by using overlapping triangular gitters.

## Acoustic Modeling

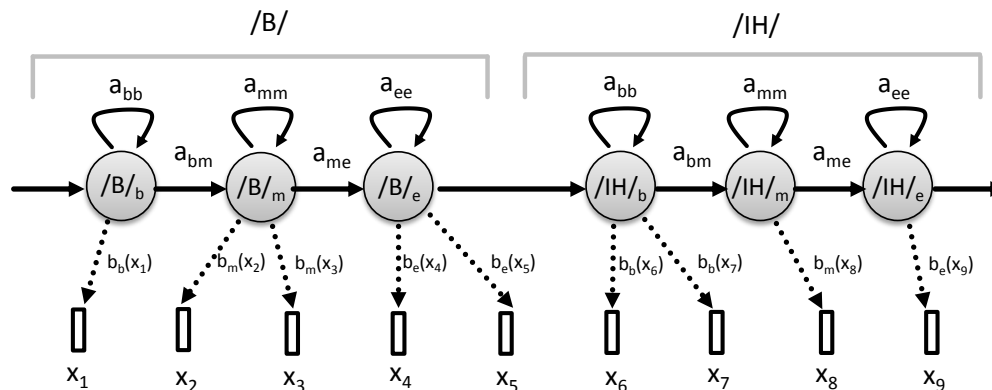
In LVCSR, the acoustic is usually modeled by using phonemes as fundamental speech units. The Hidden Markov Model (HMM) is currently the most widely used representation of a phoneme. An HMM is a 5-tuple with the following elements [Rab89]:

- Set of states  $S = \{s_1, s_2, \dots, s_N\}$ .
- A continuous alphabet  $V = \{v_1, v_2, \dots, v_M\}$  of possible emissions.
- State transition probability distributions  $A = \{a_{ij}\}$ , where  $a_{ij}$  is the probability of moving from state  $s_i$  to  $s_j$ .
- The emission probability distribution  $B = b_j(k)$  which gives the probability of emitting symbol  $v_k$  in state  $s_j$ .
- The probability distribution  $\pi$  which assigns a probability to each state  $S_i$  to be the initial state.

In any discrete moment, the system is in one state  $s_i$ . In comparison to a Markov Model, the current state in an HMM is unknown or “hidden”. Observing the system leads to an indirect conclusion in which particular state the system may be at a certain time.

The HMM representing a phoneme consists typically of three emitting states for begin, middle and end of the phoneme. To form a word, the HMMs are concatenated with the help of the pronunciation dictionary as illustrated in Figure 1.3. The probability of moving to state  $s_e$ , given the current state  $s_m$  is  $a_{me}$ . The output of the system depends on the emission probabilities.

For example, the acoustic vector  $y_3$  is emitted from state  $s_m$  with probability  $b_{s-m}(y_3)$ .

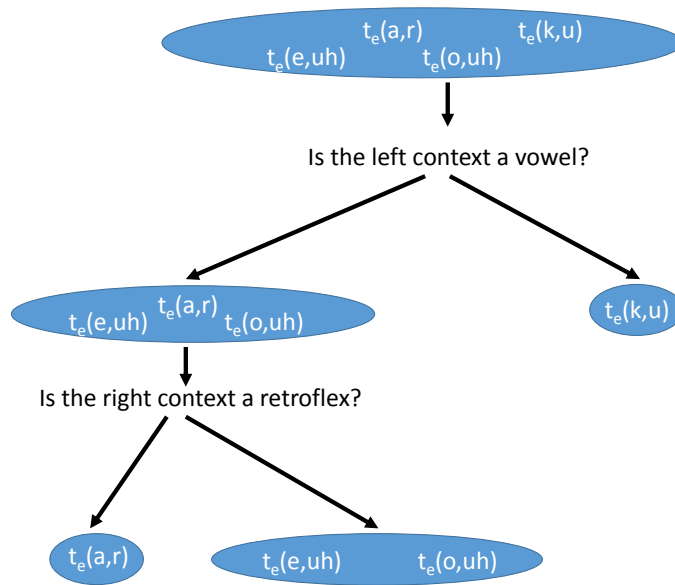


**Figure 1.3** – Connected HMMs building the word 'be'.

If the emission probabilities  $b$  are probability density functions, the HMM is called *continuous*. Gaussian mixture models (GMMs) are typically used to represent the emission probability  $b$ , which makes this approach very accurate. The problem is that it requires a large set of training data because there are many parameters to be estimated. As pointed out in [You96], in English, there are 45 phonemes used. Started with the motivation that phonemes sound different depending on the preceding and the following phonemes due to coarticulation effects, different acoustic models are trained for a phoneme dependent on the context of this phoneme [Lee88]. If the left and right neighbors of a phoneme are considered, i.e. using triphones or quintphones, much more than 45 HMMs must be trained. This will lead to models not robust if a small amount of speech data is used for training.

Semi-continuous HMMs take advantage of parameter tying. They still use GMMs but share those models when possible. For example, phonemes with different contexts share GMMs with different state weights. This approach reduces the amount of parameters to be estimated and offers a compromise between accuracy and trainability. However, to estimate which phones sound similar in different context, a binary phonetic decision tree is used. As proposed in [YOW94], binary phonetic decision trees are structures with a question on each node that can be answered with “yes” or “no”. The questions are related to the acoustic properties of the phoneme or its context. An example for a binary phonetic decision tree is depicted in Figure 1.4. The end of the phone 't' ( $t_b$  in the Figure) surrounded by 'o' from the left and 'uh'

from the right (as in 'otter'), sounds the same as 't' surrounded by 'e' and 'uh' (as in 'better'). Therefore, both phones are contained in the same leaf. The question splitting the clusters with lowest resulting entropy is placed on the top. The algorithm terminates if either no leaf node can be split since a split would result in not enough training data or the reduction in entropy by splitting is lower than a given threshold. The leaves contain phones which are acoustically similar despite their different context.



**Figure 1.4** – Binary phonetic decision tree of the phone 't' in different contexts.

Given the definition of an HMM, the following three basic problems have to be addressed to apply HMMs to speech applications [Rab89]: Evaluation problem, decoding problem and learning problem.

The *evaluation problem* of HMMs answers the question how likely is it for a given observation to be the result from a given model. This problem is solvable with the Forward algorithm [Rab89].

The *decoding problem* of HMMs needs to be solved in the decoding phase. Given an acoustic observation  $X$  and an HMM, the hidden state sequence has to be found. Computing the hidden state sequence is equivalent to finding the spoken phoneme sequence. This problem can be solved by using the Viterbi algorithm [Vit67, For73].



The *learning problem* of HMMs relates to the training phase. The idea is to adjust the parameters for a given HMM  $\lambda = (A, B, \pi)$ , which maximize the probability of observing an acoustic sequence  $X$ . Using the Forward-Backward algorithm, the acoustic sequence  $X$  is used as an input to train the model. The forward and backward variables needed for this algorithm can be computed with the Baum-Welch method.

## Language Modeling

The language model used in automatic speech recognition captures linguistic knowledge about the language. A language model helps to direct the search to find the most likely word sequence. Language and acoustic models are computed separately and then connected as illustrated in equation 1.1 to determine the most likely word sequence.

Statistical *n-gram* language models are common in LVCSR. An n-gram language model can be computed from a text corpus. It is a process of counting the occurrences of a given word  $W$  in some context called history  $H$ . The history contains the previous  $n - 1$  words from a text. Depending on  $n$ , the language model can be unigram (no history considered), bigram (a context of 2 words, i.e. history of one word considered), trigram, etc.

Given a trigram language model, the probability of a given sequence of  $k$  words can be computed with the help of equation 1.2.

$$P(W_1..W_k) = P(W_1) \cdot P(W_2|W_1) \cdot P(W_3|W_1W_2) \cdot \dots \cdot P(W_k|W_{k-2}W_{k-1}) \quad (1.2)$$

Equation 1.3 show how to compute the probability  $P(W_k|W_{k-2}W_{k-1})$  for the trigram  $W_{k-2} W_{k-1} W_k$ . In the numerator, all occurrences of the three words in the defined order are counted and divided by the number of occurrences of the two words in the history in the training text.

$$P(W_k|W_{k-2}W_{k-1}) = \frac{\text{count}(W_{k-2}W_{k-1}W_k)}{\text{count}(W_{k-2}W_{k-1})} \quad (1.3)$$

However, it is impossible to find training data for n-grams covering all possible word combinations of the language. To avoid the problem of assigning a zero probability to a phrase that actually can occur as valid language construct but does not appear in the training text, different language model smoothing techniques are used [CG98].

To measure the quality of a language model, the out-of-vocabulary rate and perplexity can be computed on a test set text. The out-of-vocabulary rate gives the number of tokens in a test set which are not covered by the vocabulary of the language model. Depending on the LVCSR application, the vocabulary can vary from a few thousand to millions of words. The *perplexity* of a language model is derived from the entropy  $H$  of the test set  $W$ . The entropy can be computed as follows:

$$H(W) = - \sum P(W) \log P(W) \quad (1.4)$$

The perplexity is obtained as  $2^{H(W)}$ . For a fixed out-of-vocabulary rate, language models with lower perplexity are usually preferred, although the perplexity is only loosely correlated with the performance of automatic speech recognition systems.

The quality of the n-gram language model in terms of out-of-vocabulary rate, perplexity and final impact to automatic speech recognition performance depends on the amount as well as the topic and time relevance of the text data used for training the model [SGVS13].

Two or more language models can be combined by applying a *linear interpolation* to the probabilities of the two models. A linear interpolation of two models is shown in equation 1.5.

$$P(\text{word}) = \lambda \cdot P_{M_1}(\text{word}) + (1 - \lambda) \cdot P_{M_2}(\text{word}) \quad (1.5)$$

$P_{M_1}$  denotes the probability provided by the first model and  $P_{M_2}$  the probability from the second model. The weight  $\lambda$  is usually optimized on a test set text.

To retrieve relevant texts from the Web for language modeling, search queries are often used [BOS03, SGG05]. Usually, search queries are made by extracting characteristic words of domain-specific documents or Web pages by calculating a *Term-Frequency Inverse-Document-Frequency (TF-IDF)* score which reflects how important a word is to a document in a collection or corpus [SGN, MK06, LGS08, LGP08]. To cover more context in the search queries, bigrams or higher order n-grams are typically scored, and those with the highest scores are used as search queries. A TF-IDF score for a bigram is computed as shown in equation 1.6.

$$score_i = \frac{tf_i}{\sum_j tf_j} \ln\left(\frac{N}{df_j}\right), \quad (1.6)$$

where  $tf_i$  is the frequency and  $df_i$  is the document frequency of bigram  $i$  and  $N$  is the total number of downloaded documents.

## Pronunciation Dictionary

The pronunciation dictionary is a mapping between lexical units – usually words – and their pronunciations. Different pronunciation dictionaries can be used for training and decoding. Pronunciation dictionaries are referred to as just “dictionary” or substituted with the term “lexicon”. There is no standard dictionary format. Usually it is a list with words as keys and pronunciation sequences as values.

As a word can have multiple pronunciations, the relationship between a word and its pronunciation is not always a 1:1 relation but due to pronunciation variations a 1:M relation.

The keys (words) of the dictionary are language-dependent and are words in the surface form of the language to be recognized. The pronunciations of the words is given as phonetic transcriptions defining how the words are pronounced. There are standard phonetic transcriptions systems such as IPA [IPA99] and ARPAbet [arp].

For example, dictionary entries of English words and their pronunciations in ARPAbet look as follows [Mih11]:

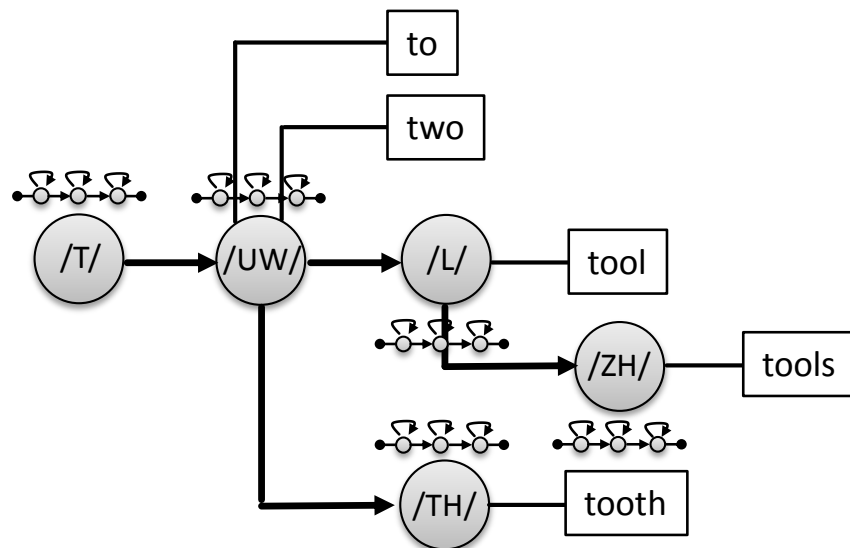
```
efficient      IH F IH SH AX N T
efficient(2)   IX F IH SH AX N T
efficiently    IH F IH SH AX N T L IY
```

In addition to the conventional use of phonemes in the pronunciations, graphemes or syllables can be used as subword modeling units. Such dictionaries are employed in automatic speech recognition systems of languages with a rather close grapheme-to-phoneme relationship and are cheap to set up, as we describe in Section 1.4.6.

Furthermore, dictionaries with weighted pronunciation variants can help to improve automatic speech recognition system performance in cases of large numbers of pronunciation variants as reported in [AHHL<sup>+</sup>09].

## Search

The task of the automatic speech recognition is to find the most likely word sequence  $\hat{W}$  corresponding to a spoken speech utterance. Generally, decoding is a search problem. A commonly employed algorithm is the Viterbi search. The search space can be represented as a graph. As it is usually very large, some techniques for accelerating the search performance are used. There are two widely employed techniques for faster search algorithms: Sharing and pruning. The key step in the construction of the search space is building a complete determined and minimized graph from the dictionary as shown in Figure 1.5. This graph is built from subword parts (phones) with the help of the pronunciations defined in the dictionary. Words that contain the same prefixes share arcs in the tree. The internal nodes represent phones, the leaf nodes contain words.



**Figure 1.5** – Search graph.

The deficiency of this model is that language model probabilities can be used after the word is known (once a word-node is reached). This issue can be solved with the language model probabilities distributed among the subword parts rather than only on path ends.

If a unigram language model is used, the structure in Figure 1.5 works well. However, for higher order language models, multiple copies of lexical trees have to be traversed. To handle the higher complexity of the search space, two possibilities are listed:

- Static decoder with precomputed search network containing language model, dictionary and acoustic model information (Finite State Transducer) [MPR01].
- Dynamic generation of the n-gram search space via graph search algorithms (dynamic decoder) [SSK10, RHL<sup>+</sup>11].

In this work, we conducted our automatic speech recognition experiments with the dynamic decoder of the Janus Recognition Toolkit (JRTk) [FGH<sup>+</sup>97, SMFW01].

## Measuring Automatic Speech Recognition Performance

The standard metric to evaluate an automatic speech recognition system is the word error rate (WER) which is based on the Levenshtein edit distance [Lev66]. The output of the decoding process is a hypothesis for what has been spoken. Comparing the hypothesis with the reference text which is the true transcription of what is said, gives a score in terms of the percentage of errors made. The following errors can occur after the alignment of the hypothesis and the reference text:

- Substitution: A word is misrecognized.
- Deletion: A word from the reference is missing in the hypothesis.
- Insertion: The recognizer inserts a word that is not actually spoken.

Figure 1.6 demonstrates the errors that an automatic speech recognition system can make. On the vertical axis is the reference, and horizontally the output sequence of the system.

To compute the word error rate after identifying these errors, the following equation is used:

$$WER [\%] = \frac{\#substitutions + \#deletions + \#insertions}{\#words (reference)} \cdot 100 \quad (1.7)$$

The equation above shows that the word error rate can exceed 100% in the case when automatic speech recognition systems tend to insert words.

Often, the performance of an automatic speech recognition system is reported as the word accuracy (WA):

$$WA [\%] = (1 - WER) \cdot 100 \quad (1.8)$$

	text						
	reference				substitution		
Reference	correct			deletion			
	a						
	is			insertion			
	this						
		this	is	the	a	hypothesis	text

Hypothesis

**Figure 1.6** – Possible errors of an automatic speech recognition system.

### 1.3 Rapid Adaptation of Automatic Speech Recognition Systems to new Domains and Languages

#### Efforts for Rapid Language Adaptation

With more than 7,100 languages in the world [LSF14] and the need to support multiple languages, it is one of the most pressing challenges for the speech and language community to develop and deploy speech processing systems in yet unsupported languages rapidly and at reasonable costs [Sch04, SK06, SVS13]. Major bottlenecks are the sparseness of speech and text data with corresponding pronunciation dictionaries, the lack of language conventions, and the gap between technology and language expertise.

Data sparseness is a critical issue due to the fact that today's speech technologies heavily rely on statistically based modeling schemes, such as Hidden Markov Models and n-gram language modeling. Although statistical modeling algorithms are mostly language independent and proved to work well for a variety of languages, reliable parameter estimation requires vast amounts of training data. Large-scale data resources for research are available for less than 100 languages and the costs for these collections are prohibitive to

all but the most widely spoken and economically viable languages [SVS13]. The lack of language conventions concerns a surprisingly large number of languages and dialects. The lack of a standardized writing system for example hinders Web harvesting of large text corpora and the construction of pronunciation dictionaries.

Despite the well-defined process of system building, it is cost- and time-consuming to deal with language-specific peculiarities, and requires substantial language expertise. Unfortunately, it is extremely difficult to find system developers who have both, the necessary technical background and the native expertise of a language in question. As a result, one of the pivotal issues for developing speech processing systems in multiple languages is the challenge of bridging the gap between language and technology expertise [Sch04].

As stated in [SVS13], the standard way of building speech applications for an unsupported language is to collect a sizable training corpus and to train statistical models for the new language from scratch. Considering the enormous number of languages and dialects in the world this is clearly a suboptimal strategy, which highlights the need for more sophisticated modeling techniques. It would be desirable to develop models that can take advantage of similarities between dialects and languages of similar type, and models which can share data across different varieties. This would have two benefits, first leading to truly multilingual speech processing which can handle common phenomena such as code switching, and second providing models which are likely to be more robust toward dialectal and cross-lingual accent variations. These multilingual shared models can then be used as seed models to jumpstart a system in an unsupported language by efficiently adapting the seeds using limited data from the language in question. We refer to this development strategy as *rapid language adaptation*. Particularly, since the IARPA<sup>6</sup> Babel program [GKRR14] develops technologies to enable rapid deployment of spoken term detection systems for low-resource languages, rapid language adaptation attracts more interest.

In the following sections, we introduce our Rapid Language Adaptation Toolkit [SBB<sup>+</sup>07, VSKS10, SVS13], which we use to rapidly build speech processing systems for unsupported languages and which we extended with many of the tools implemented in this work.

---

<sup>6</sup>Intelligence Advanced Research Projects Activity

## Rapid Language Adaptation Toolkit

The project SPICE (Speech Processing - Interactive Creation and Evaluation) (NSF, 2004-2008) performed at the Language Technologies Institute at Carnegie Mellon University (CMU) [SBB<sup>+</sup>07] and the Rapid Language Adaptation project at the Cognitive Systems Lab (CSL) at Karlsruhe Institute of Technology (KIT) [VSKS10, SVS13] aim at bridging the gap between the language and technology expertise. For this purpose the Rapid Language Adaptation Toolkit (RLAT<sup>7</sup>) provides innovative methods and interactive Web-based tools to enable users to develop speech processing models, to collect appropriate speech and text data to build these models, as well as to evaluate the results allowing iterative improvements. The toolkit significantly reduces the amount of time and effort involved in building speech processing systems for unsupported languages.

In particular, the toolkit allows the user to (1) design databases for new languages at low cost by enabling users to record appropriate speech data along with transcriptions, (2) to continuously harvest, normalize, and process massive amounts of text data from the Web, (3) to select appropriate phoneme sets for new languages efficiently, (4) to create vocabulary lists, (5) to automatically generate pronunciation dictionaries, (6) to apply these resources by developing acoustic and language models for automatic speech recognition, (7) to develop models for text-to-speech synthesis, and (8) to finally integrate the built components into an application and evaluate the results using online speech recognition and synthesis in a talk-back function [SBB<sup>+</sup>07].

RLAT and SPICE leverage the two projects *GlobalPhone* [SVS13] and *FestVox* [BL00] to implement bootstrapping techniques that are based on extensive knowledge and data sharing across languages, as well as sharing across system components [SBB<sup>+</sup>07]. Examples for data sharing techniques are the training of multilingual acoustic models across languages based on the definition of global phone sets. Sharing across components happens on all levels between recognition and synthesis, including phoneme sets, pronunciation dictionaries, acoustic models, and text resources.

RLAT and SPICE are a freely available online service which provide an interface to the Web-based tools and have been designed to accommodate all potential users, ranging from experts to users with little background knowledge in speech processing. Users with little background are able to read

---

<sup>7</sup><http://csl.ira.uka.de/rlat-dev>

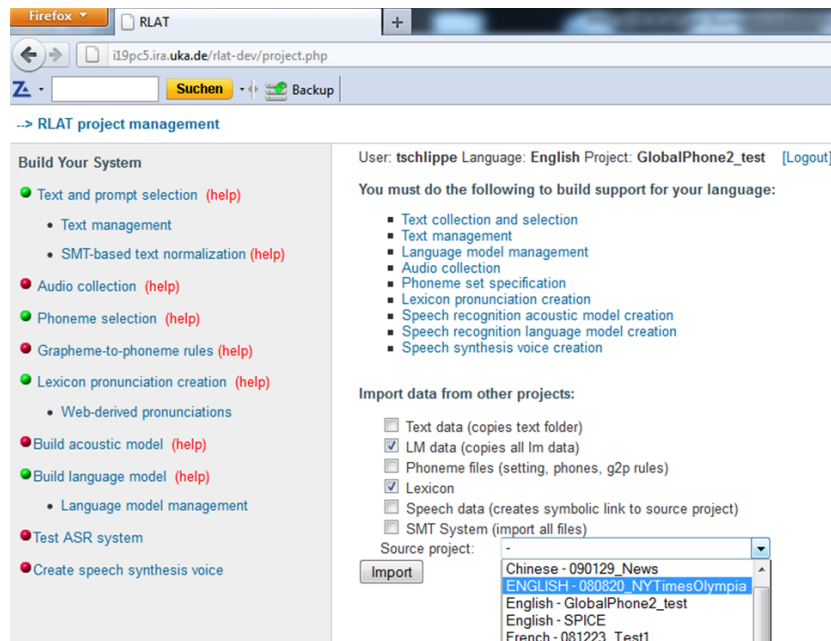


easy-to-follow, step-by-step guidelines as they build a speech processing system. Expert users can skip past these instructions. In addition, file-uploading routines allow for feeding the bootstrapping algorithms with available data and thus shortcut the process. As a result the tools collect information from the broadest array of people: A general audience of Internet users who may have little experience with speech tools, and a specific audience of speech and language experts, who can use data they already have. By keeping the users in the developmental loop, the RLAT and SPICE tools can learn from their expertise to constantly adapt and improve the resulting models and systems.

The tools are regularly used for training and teaching purposes at two universities (KIT and CMU). There, students are asked to rely solely on the tools when building speech processing systems and report back on problems and limitations of the system. Results indicate that it is feasible to build end-to-end speech processing systems in various languages (more than 15) for small domains within the framework of a six-week hands-on lab course [SBB<sup>+</sup>07].

Within this work, we applied, implemented, enhanced and embedded several tools of the Rapid Language Adaptation Toolkit: We used the Rapid Language Adaptation Toolkit to bootstrap automatic speech recognition systems for Eastern European languages and to optimize the language model qualities with large quantities of text material from the Internet crawled in only 20 days [VSKS10]. Furthermore, we included a Web-interface where native speakers can normalize crawled text, thereby providing a parallel corpus of normalized and non-normalized text [SZGS10, SZLS13]. With this corpus, statistical machine translation models are generated to translate non-normalized into normalized text. Moreover, we implemented the Automatic Dictionary Extraction Tool which is a component to retrieve Web-derived pronunciations [SOS14, SOS10]. Additionally, we implemented an RSS parser into the Rapid Language Adaptation Toolkit. As described in [SGVS13], it takes RSS Feeds, extracts the URLs with the publishing date and collects the texts of the corresponding Web pages preserving the time information. Recently, we developed *RLAT light* suited for entirely novices with simplified functionalities, step-by-step guidelines for an efficient task accomplishment and more precise feedback from the system [He14].

Figure 1.7 demonstrates the main page with a list of major components of our Rapid Language Adaptation Toolkit.



**Figure 1.7** – Steps to build automatic speech recognition and synthesis systems with the Rapid Language Adaptation Toolkit.

## GlobalPhone - A Multilingual Text and Speech Database

For most of our experiments described in this thesis we use the *GlobalPhone* data. *GlobalPhone* [Sch02, SVS13] is a multilingual data corpus developed in collaboration with the Karlsruhe Institute of Technology (KIT). The complete data corpus comprises (1) audio/speech data, i.e. high-quality recordings of spoken utterances read by native speakers, (2) corresponding transcriptions, (3) pronunciation dictionaries covering the vocabulary of the transcripts, and (4) baseline n-gram language models.

The entire *GlobalPhone* corpus provides a multilingual database of word-level transcribed high-quality speech for the development and evaluation of large vocabulary speech processing systems in the most widespread languages of the world [Sch02, SVS13]. *GlobalPhone* is designed to be uniform across languages with respect to the amount of data per language, the audio quality (microphone, noise, channel), the collection scenario (task, setup, speaking style), as well as the transcription and phoneme set conventions (IPA-based naming of phones in all pronunciation dictionaries). Thus, *GlobalPhone* supplies an excellent basis for research in the areas of (1) multilingual speech recognition, (2) rapid deployment of speech processing systems to yet unsupported languages, (3) language identification tasks, (4) speaker recognition

in multiple languages, (5) multilingual speech synthesis, as well as (6) monolingual speech recognition in a large variety of languages.

As a part of this thesis, we have collected Hausa in Cameroon [SDV<sup>+</sup>12], Ukrainian in Donezk [SVYS13], Vietnamese in Hanoi [VS09], Swahili in Nairobi, and an accented speech database [Mih11] with native speakers of Bulgarian, Chinese (Mandarin or Cantonese), German and some of the languages spoken in India (Hindi, Marathi, Bengali, Telugu, Tamil) speaking English [SVS13, SS14].

## 1.4 Pronunciation Generation

For the automatic and semi-automatic pronunciation generation of written languages, a correct mapping of a language’s graphemes to a sequence of symbols representing its corresponding acoustic units – mostly in terms of phonemes – is very important. An accurate mapping guarantees a higher recognition quality of an automatic speech recognition system [GN14]. Otherwise an acoustic model trained with suboptimal data is used during the decoding process for the calculation of scores. However, a grapheme-to-phoneme mapping is not always straightforward due to the fact that there is a variety of languages with differing types of writing systems and various degrees of complexity in the relationship between graphemes and phonemes.

In the following sections we present the different types of writing systems together with the correspondence between their graphemes and phonemes. Afterwards, we give an impression of a manual pronunciation generation and motivate the support of automatic methods. Finally, we present fully and semi-automatic state-of-the-art approaches as well as the economic application of graphemic pronunciation dictionaries.

### 1.4.1 The International Phonetic Alphabet

The International Phonetic Alphabet (IPA) [IPA99] offers a standard way of describing and categorizing the sounds produced when speaking. The current version of the IPA chart from 2005 contains 107 IPA symbols describing the basic consonants and vowels. These can be extended with the help of 31 diacritics that mark the place of articulation and 19 additional symbols for denoting sound properties such as length, tone, and stress.

IPA is conceptualized to represent one symbol for each verbalizable sound in all languages in the world. It reduces ambiguity in labeling phonemes and is used in many dictionaries, e.g. Oxford, Cambridge, Collins, and Langenscheidt. Several ASCII-based keyboard transliterations of IPA exist, e.g. SAMPA [Wel97], X-SAMPA, and Arpabet [arp].

In the IPA charts the different sounds are also classified as consonants and vowels. This mapping is shown in Figure 1.8 and Figure 1.9.

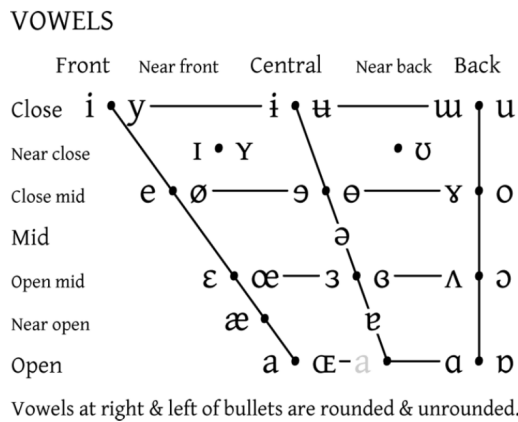


Figure 1.8 – IPA vowel chart [IPA99].

CONSONANTS (PULMONIC)

	Bilabial	Labio-dental	Dental	Alveolar	Post-alveolar	Retroflex	Palatal	Velar	Uvular	Pharyngeal	Epi-glottal	Glottal
Nasal	m	ɱ	n			ɳ	ɲ	ŋ	ɴ			
Plosive	p b	ɸ β	t d			ʈ ɖ	c ɟ	k ɡ	q ɢ			
Fricative	ɸ β	f v	θ ð	s z	ʃ ʒ	ʂ ʐ	ç ʝ	x ɣ	χ ʁ	ħ ʕ	ħ ʕ	h ɦ
Approximant		ʋ	ɹ			ɻ	j	ɰ				
Trill	ʙ		r						ʀ		ʀ	
Tap, Flap		ⱱ	ɾ			ɽ						
Lateral fricative			ɬ ɮ		ɮ	ɬ	ɬ	ɮ				
Lateral approximant			l		ɭ	ɭ	ʎ	ʎ	L			
Lateral flap			ɺ		ɺ	ɺ						

Where symbols appear in pairs, the one to the right represents a modally voiced consonant, except for murmured *ɦ*. Shaded areas denote articulations judged to be impossible. Light grey letters are unofficial extensions of the IPA.

Figure 1.9 – IPA consonant chart [IPA99].

## 1.4.2 Correspondence between Graphemes and Phonemes

The different types of writing systems with varying degree of complexity in the relationship between graphemes and phonemes are usually categorized as follows [omn14, GN14]:

- Alphabets
- Abjads
- Syllabic alphabets
- Semanto-phonetic writing systems

### Alphabets

In general, alphabets represent sounds of a language and consist of a set of letters, i.e. written or printed signs mainly each representing a sound in speech. Languages using alphabetic scripts are characterized by mainly three types of relationships between letters and sounds [GN14]: (1) a single letter or a group of letters represent a single sound, (2) a letter or a group of letters corresponds to many sounds depending on the context or specific rules and (3) a variety of letters or combination of letters represent a single sound.

Some alphabetic scripts have a rather close grapheme-to-phoneme relationship, e.g. Bulgarian, Czech, Polish, and Spanish. Others are characterized of a weaker relationship, e.g. Portuguese, French, and German. English is an example for a more ambiguous relationship: There one letter can represent a variety of sounds conditioned by complex rules and many exceptions [WSS<sup>+</sup>00]. In [SS14] and [SOS12b], we investigated the grapheme-to-phoneme relationships of ten languages with alphabetic scripts. Our analyses are described in Section 4.1.

### Abjads

The abjads, a subcategory of alphabets, consist of letters for consonants but vowels are often not marked [GN14]. If they are marked, it is typically done with diacritics below or above each letter. Abjads are usually written from right to left. They are used for example in Arabic, Hebrew and Syriac. In Arabic the letters even change their appearance depending on their position in a word. The major problem for the production of pronunciation dictionaries for those languages is the absence of the diacritics in most texts which

mark the vowelization. “shadda” is the only diacritic which appears in several modern Arabic scripts [SNV08]. Native speakers distinguish the right pronunciation and the correct meaning of a word without diacritic marks in an automatic way by considering the context and the position of the word in a sentence. Their knowledge of Arabic grammar and vocabulary enable them to correctly vocalize words in written texts based on the context. A full vowel indication is only used for some political and religious texts as well as in textbooks for beginners of the Arabic language [ZSS06].

For Arabic there are on average 2.9 possible vowelizations for a given written word. To generate pronunciations for Arabic words, it is essential to deal with this ambiguity. There are several approaches to restore the missing diacritics [SNV08]. However, the accuracy of the automatic diacritization systems stays low, being in the range of 15-25% word error rate [GN14]. [ANX<sup>+</sup>05] report that a phoneme-based approach which requires the missing diacritics performs 14% better than a grapheme-based approach which used the characters without diacritics as acoustic units. In 2009 [EGMA09] reported that this accuracy gap can be neutralized by increasing the number of Gaussian densities in context-dependent acoustic models or the amount of transcribed audio data for training. However, the diacritization is still important for speech synthesis and machine translation applications.

## Syllabic Alphabets

Syllabic alphabets are also called alphasyllabaries and abugidas. Each grapheme in such alphabets represents one syllable. The syllables contain consonants with a corresponding vowel which can be changed to another vowel or muted with the help of diacritics [GN14, omn14]. In addition, diacritics are used to separate vowel letters when they appear at the beginning of syllables or on their own. Syllabic alphabets are used by Thai, Tamil, Bengali, Khmer, Lao, Lanna and other languages in South and South East Asia. Unfortunately, automatic speech recognition literature is not yet very advanced for most of these languages.

More literature exists for Thai. [SCB<sup>+</sup>05] report a relatively poor grapheme-to-phoneme relationship for this language. However, [CHS06] built a grapheme-based system with an enhanced tree clustering which outperforms a phoneme-based one with an automatically generated dictionary and reaches almost the performance of a phoneme-based system with a handcrafted dictionary. Moreover, [SSB<sup>+</sup>08] show a comparable performance between a phoneme- and a grapheme-based Khmer system.

### Semanto-Phonetic Writing Systems

Semanto-phonetic writing systems are also called logophonetic, morphophonemic, logographic and logosyllabic. In such writing systems the symbols may represent both sound and meaning [GN14]. The following types of symbols are included: (1) Pictograms or pictographs, (2) logograms, (3) ideograms or ideographs, and (4) compound characters. *Pictograms* or *pictographs* resemble the what they represent. *Logograms* represent parts of words or whole words. *Ideograms* or *ideographs* represent graphically abstract ideas. *Compound characters* consist of a phonetic element and a semantic element [omn14].

The Japanese Kanji and Chinese are two semanto-phonetic writing systems. The Chinese script comprises ideograms and mostly compound characters [omn14]. Each character represents one syllable. However, multiple characters correspond to one syllable, each one with different meaning. Thus, semanto-phonetic scripts are a challenge for the pronunciation generation [SW01]. Existing tools which convert the semanto-phonetic characters can be used for Chinese and Japanese [RSW99, SW01, SCT04]. Then the pronunciation can be derived easier from the converted strings. In the case of Chinese the characters are often converted to the Pinyin transcription, a widely used transcription with the Latin alphabet. There the grapheme-to-phoneme relationship is much more straightforward. However, the transcription is a complex task since about 13% of the Chinese characters have more than one pronunciation [RSW99].

#### 1.4.3 Manual Pronunciation Generation

Generating pronunciations by hand, without any automatic support, can be very time-consuming. For example, [DB04c] report that producing a pronunciation for an Afrikaans word takes between 19.2 and 30 seconds on average. The average times were computed on a set of 1,000 words. 19.2 seconds was the fastest average time observed in their lab using a proficient phonetic transcriber, and represents an optimistic time estimate. Consequently, with only one annotator it would take between 8.9 and 13.9 days to compile a whole Afrikaans dictionary containing 40,000 words plus periods for pauses. Note that Afrikaans is a Germanic language with a fairly regular grapheme-to-phoneme relationship.

The annotation process can be done in parallel. However, multiple annotators may produce inconsistent and flawed pronunciations due to different subjec-

tive judgments, small typographical errors, and 'convention drift' [SDV<sup>+</sup>12]. Therefore, the manual work is usually supported with automatic methods.

#### 1.4.4 Automatic Pronunciation Generation

##### Rule-based Grapheme-to-Phoneme Conversion

In case of a close relationship between graphemes and phonemes, defining rules by hand can be efficient. The best case is a 1:1 relationship where the number of grapheme-to-phoneme conversion rules to be defined would correspond to the number of letters. If no wide context information is necessary for a good grapheme-to-phoneme conversion, it can be implemented by search and replace rules. Knowledge-based approaches with rule-based conversion systems can typically be expressed as finite-state automata [KK94, BLP98]. However, for languages with loose grapheme-to-phoneme relationships, these methods often require specific linguistic skills and exception rules formulated by human experts.

The advantage of a rule-based grapheme-to-phoneme conversion in contrast to data-driven approaches is that no sample word-pronunciation pairs are required to train the converters. However, if defining the rules for languages with a more complex grapheme-to-phoneme relationship becomes an elaborate task, it is better to generate training data in terms of sample word-pronunciation pairs and use data-driven approaches. For example, for the creation of the Ukrainian *GlobalPhone* dictionary, we elaborated and applied 882 search-and-replace rules based on [BMR08] to produce phoneme sequences for our Ukrainian words, as described in [SVYS13].

##### Data-driven Grapheme-to-Phoneme Conversion

In contrast to knowledge-based approaches, data-driven approaches are based on the idea that, given enough examples, it should be possible to predict the pronunciation of unseen words purely by analogy. The benefit of the data-driven approach is that it trades the time- and cost-consuming task of designing rules, which requires linguistic knowledge, for the much simpler one of providing example pronunciations.

[Bes94] propose data-driven approaches with heuristic and statistical methods. In [Kne00], the alignment between graphemes and phonemes is generated using a variant of the Baum-Welch expectation maximization al-

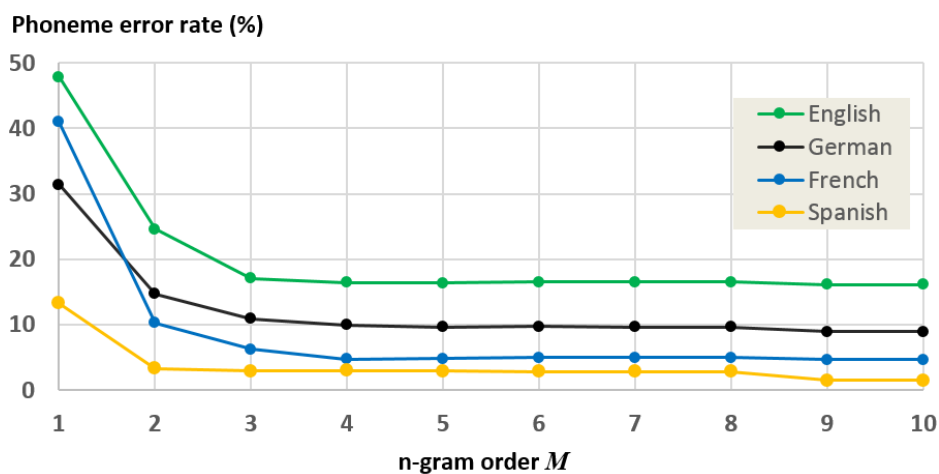


gorithm. [Che03, VALR03, JKS07, BN08] use a joint-sequence model to the grapheme-to-phoneme conversion task. [Nov11] and [NMH12] utilize weighted finite-state transducers for decoding as a representation of the joint-sequence model. [GF09], [LDM09], and [KL10], apply statistical machine translation-based methods for the grapheme-to-phoneme conversion. A good overview of state-of the art grapheme-to-phoneme conversion methods is given in [HVB12]. Data-driven methods are applied commonly and cross-checks of the generated pronunciations are often not performed [LCD<sup>+</sup>11] because a manual check is time-consuming and expensive, especially if native speakers or linguists need to be hired for this task. Following the literature, we denote a statistical model for the grapheme-to-phoneme conversion as *grapheme-to-phoneme model*.

As Sequitur G2P, which uses joint-sequence models for the grapheme-to-phoneme conversion task, usually gives the best performance [BN08, HVB12, SQS14], we used it for training and applying grapheme-to-phoneme models for most of our experiments. Therefore, we introduce Sequitur G2P in the following paragraph.

### Sequitur G2P

Sequitur G2P is a data-driven grapheme-to-phoneme converter developed at RWTH Aachen University [BN08]. It is open source software and in experiments it has shown favorable results when compared to other grapheme-to-phoneme tools. Sequitur G2P employs a probabilistic framework based on joint-sequence models. Such models consist of units called multigrams or graphones, which carry both input and output symbols (graphemes and phonemes). The effective context range covered by a model is controlled by the maximum length  $L$  of graphones and the n-gram order  $M$  of the sequence model. A graphone of length  $L$  carries 0 to  $L$  graphemes and 0 to  $L$  phonemes, with the non-productive case of both 0 graphemes and 0 phonemes being excluded. The simplest case of  $L=1$  is called a singular graphone and corresponds to the conventional definition of a finite state transducer. Several previous experiments have shown that singular graphones in combination with long-range  $M$ -grams yield the best performance [BN08]. Typically,  $M$  should be in the order of the average word length for maximum accuracy as described in [BN08]. This value naturally differs between our project languages, so we settled on the lowest common average value for the sake of comparability. Therefore, in most experiments we chose to train models using the parameters  $L=1$  and  $M=6$ .



**Figure 1.10** – Sequitur G2P model performance over n-gram order  $M$ .

Figure 1.10 also confirms that  $M=6$  is a reasonable choice. We illustrate the performance of grapheme-to-phoneme models of four languages with differing grade of grapheme-to-phoneme relationship and varying model sizes  $M$  trained with 10k word-pronunciation pairs. The performance is expressed by the edit distance between the hypotheses and the canonical reference pronunciations at the phoneme level denoted as phoneme error rate (PER). We regard pronunciations which are closer to our high-quality reference pronunciations as better since they minimize the human editing effort in a semi-automatic pronunciation dictionary development, as we show in Section 5.3. For all four languages, we observe that  $M$  larger than 3 tend to asymptote and give no much performance improvement. Consequently,  $M=6$  is a choice which is not too conservative.

During the iterative training process, Sequitur G2P uses an expectation maximization algorithm to estimate the model. To avoid the problem of overfitting, evidence trimming and discounting is performed. Since the discount values need to be optimized on a data set that is separate from the data that is used to compute the evidence values, we let Sequitur G2P select a random 5% of our training data as held-out development set. The removal of this data from the training set has a noticeable negative effect on model performance. Because of this, Sequitur G2P does a ‘fold-back’ training: The held-out data is added back to the training set after the training process has converged. The training then continues with the previously calculated discount parameters until further convergence.

### 1.4.5 Semi-Automatic Pronunciation Generation

In semi-automatic dictionary generation processes like [MBT04], [DM09], and [SBB<sup>+</sup>07] native speakers enter pronunciations as phoneme strings. To reduce difficulty and effort of pronunciation generation, the learned rules are iteratively updated and the user is provided with potential pronunciations. Additionally, he or she can listen to a synthesized sound file of the entered pronunciation.

[MBT04] and [DM09] display the words according to their occurrence frequencies in a text corpus. By covering common words early, word error rates in automatic speech recognition are minimized for an early training and decoding. [Kom06] and [KB06] order the words according to their n-gram coverage to learn many grapheme-to-phoneme relations early. [DB04b] and [DB04c] prefer short words over longer words to alleviate the correction effort for human editors. While [DM09] use a phoneme set defined by linguists, [Kom06] infers a phoneme set in an automatic way: An initial phoneme recognizer is trained on a grapheme-based dictionary. Based on audio recordings and transcriptions, acoustic model units are adapted based on Merge&Split.

Some approaches update the grapheme-to-phoneme model after each word [DM09, SBB<sup>+</sup>07]; others combine incremental updates and periodic rebuilding [Kom06, DB05]. In [MBT04] and [DFG<sup>+</sup>05], the user decides about the creation of new grapheme-to-phoneme models. [DB05] introduce a data-adaptive strategy, updating the grapheme-to-phoneme model after the pronunciations of 50 words needed to be corrected. While [DM09] start with an empty dictionary, [MBT04] manually generate pronunciations for the most frequent 200–500 words in a text corpus. [SBB<sup>+</sup>07] initializes the pronunciation generation with a substitution of graphemes with the most commonly associated phonemes (*1:1 G2P mapping*) entered by the user.

[Kom06] records 20 minutes of speech and builds an initial dictionary automatically based on the grapheme-based phoneme set, acoustic information and their transcriptions.

### 1.4.6 Graphemic Pronunciation Dictionaries

Whereas the phoneme-based techniques are based on pronunciation rules, which are either manually created by linguists or native speakers or statistically derived from available data sets [BD12], the grapheme-based (also called “graphemic”) method uses graphemes as subword modeling units.

Directly using graphemes as acoustic units is fully automatic and less costly. Generally, the grapheme-based method does not perform as well as the phoneme-based methods. However, some studies have demonstrated that the performance of grapheme-based automatic speech recognition systems depends on the nature of the grapheme-to-phoneme relationship of the target language [BD12, KSS03, KN02, SS04]. For example, [KN02] report a relative word error rate increase of 2% for Dutch, German and Italian using graphemes as acoustic units instead of phonemes, while for English 20%. [BD12] shows for Afrikaans that the performance of the grapheme-based approach is dependent on word categories: For spelled out words, proper names, spelling errors, partial words, and multi-category words, they observe lower word error rates with graphemic pronunciations than with pronunciations which have been generated automatically using Afrikaans grapheme-to-phoneme rules. The overall word error rate of the grapheme-based system converges to the performance of the phoneme-based system as the training data size in terms of transcribed audio data increases. Additionally, a study with Russian demonstrated that the grapheme-based system performed almost as well as the phoneme-based one [SS04]. Table 1.1 shows our performances of phoneme-based and grapheme-based systems in Czech (*cs*), German (*de*), English (*en*), Spanish (*es*), French (*fr*), Polish (*pl*) and the African language Hausa (*ha*) together with the relative word error rate (WER) increase of the grapheme-based systems compared to the phoneme-based ones. Since Spanish has a close grapheme-to-phoneme relationship, we expected a smaller word error rate difference between the grapheme- and the phoneme-based systems. A possible reason may be local accents and inconsistent pronunciations in the audio data recorded in Costa Rica. For some languages the grapheme-based approach even outperforms manually cross-checked phoneme-based dictionaries. For example, in [SDV<sup>+</sup>12] we report a slightly better grapheme-based *GlobalPhone* Hausa system even after several cross-checks.

	cs	de	en	es	fr	pl	ha
Phoneme-based WER	15.62	17.11	11.52	11.97	20.41	14.98	14.22
Grapheme-based WER	17.56	17.83	19.15	14.06	23.36	15.38	13.57
Rel. WER increase	12.42	4.21	66.23	17.46	14.45	2.67	-4.57

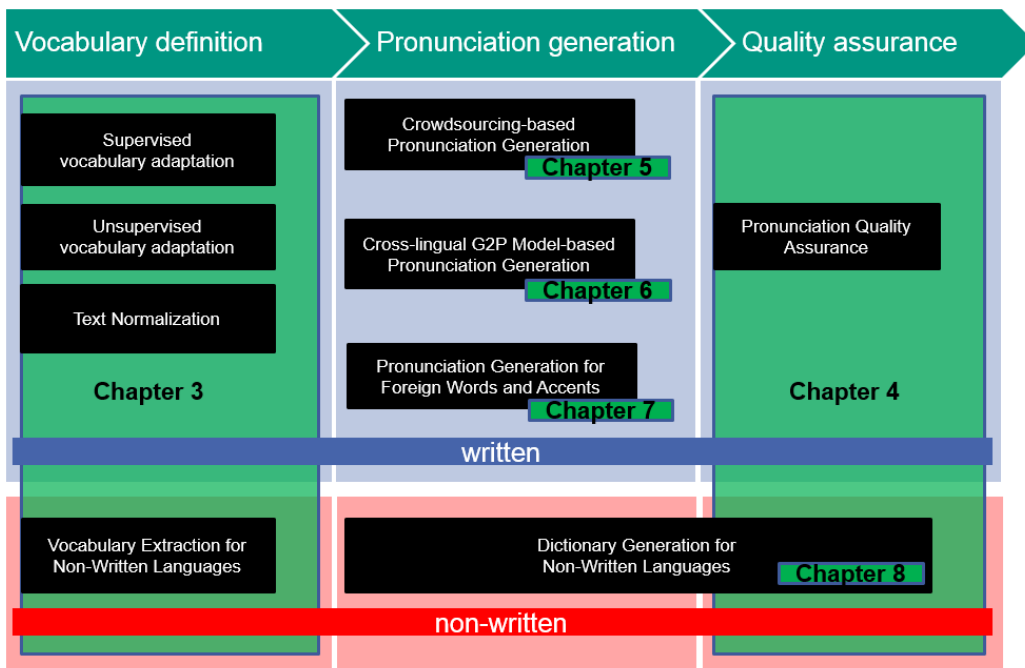
**Table 1.1** – Word error rates (%) with phoneme-based and grapheme-based dictionaries.

Furthermore, grapheme-based systems have proven to enhance automatic speech recognition system performance in a confusion network combination if the grapheme-based system is reasonable close in performance to the other

systems of the combination and produces an output with errors that are different to those of the other systems [LCD<sup>+</sup>11, SDV<sup>+</sup>12].

## 1.5 Structure of this Thesis

This thesis is structured as follows: In Chapter 1, we have introduced the reader to the purpose of this thesis and the required background. In Chapter 2, we will demonstrate the challenges to build pronunciation dictionaries for new domains and languages. Chapter 3 presents our strategies for an efficient vocabulary selection and text normalization. In Chapter 4, we propose our methods to assure the quality of the pronunciations. Then we demonstrate our solutions for the pronunciation generation in different scenarios from Chapter 5 to Chapter 8. We conclude the thesis in Chapter 9.



**Figure 1.11** – Structure of the experiments and results.

This thesis aims at investigating, analyzing, and improving all parts of the pronunciation dictionary generation processing chain. In order to obtain a structured description, the results are arranged in relation to this processing chain.

This concept is depicted in Figure 1.11. The top row shows the main building blocks of the dictionary development process, i.e. vocabulary definition, pronunciation generation, and quality assurance. Our contributions for languages with writing system are depicted in the upper part, for non-written languages in the lower part.

After introducing the challenges we tackle, we begin with the first building block “Vocabulary definition” in Chapter 3, where we show how to define the vocabulary for the pronunciation dictionary. For languages with writing system, we describe our methods to define the vocabulary of the dictionary with the help of Web texts and to rapidly develop text normalization systems at low cost with the help of crowdsourcing. For non-written languages and dialects, we exhibit our algorithm using cross-lingual word-to-phoneme alignment to segment phoneme sequences into word units and induce a vocabulary.

The second building block “Pronunciation generation” is divided into four chapters (Chapter 5, 6, 7, 8). We devote a lot of space to investigation since this process usually takes most effort and can be the biggest challenge. In Chapter 5 we demonstrate how dictionary entries from websites or derived on crowdsourcing platforms can contribute to a rapid and cheap semi-automatic pronunciation dictionary development. In Chapter 6 we present our approaches for a cross-lingual grapheme-to-phoneme modeling-based pronunciation generation. We address the problem of detecting foreign words for a separate handling and a lexical adaptation to speakers with accent in Chapter 7. Chapter 8 demonstrates our methods to gain a pronunciation dictionary for non-written languages which enables an automatic “invention” of a writing system for the target language.

The error recovery methods belong to the third building block “Quality assurance” and cover both languages with and without writing system. Before we characterize our contributions for the rapid pronunciation generation, we first introduce our methods to detect and revise faulty and inconsistent pronunciations and improve quality beyond state-of-the-art methods in Chapter 4 since we deploy these methods in our solutions to assure qualified pronunciations.

We conclude this thesis with a chapter of summary and future work.

## CHAPTER 2

# Challenges of Dictionary Generation for new Domains and Languages

---

In this chapter, we introduce the challenges to build pronunciation dictionaries for new domains and languages. For the pronunciation generation we describe scenarios depending on the available resources for which we demonstrate our solutions in this thesis.

## 2.1 Vocabulary Selection and Text Normalization

### 2.1.1 Vocabulary

It is an important task to find an appropriate vocabulary to train and decode automatic speech recognition systems due to the following reasons: In the automatic forced alignment process, feature vectors are assigned to the HMM states for training HMM-based systems. To avoid alignment errors in this process, covering the whole vocabulary of the training transcriptions together with corresponding pronunciations is essential. Furthermore, the decoding vocabulary should fit the expected vocabulary in the audio to decode.

How to select the search vocabulary for an unknown domain and an unseen test set, respectively? It is important to collect words which are time- and topic-relevant for the domain of the automatic speech recognition system. If the vocabulary is too small, the system has a high out-of-vocabulary rate, i.e. the missing words cannot be recognized which leads to errors and follow-up errors. A decoding vocabulary that contains words not matching the target domain may result in confusion errors. Researchers and developers have used the World Wide Web as a source to find relevant words (*Crawling*) based on word frequency and time relevance [SGVS13].

Appropriate vocabulary sizes are dependent on the language and on the task: For example, Slavic languages are characterized by a rich morphology, caused by a high inflection rate of nouns using various cases and genders. With 293k words in our *GlobalPhone* Russian pronunciation dictionary, we report an out-of-vocabulary rate of 3.9% in the domain of national and international political and economic news, while for Spanish a vocabulary size of only 19k is sufficient to obtain an out-of-vocabulary rate of 0.1% in the same domain [SVS13].

In Section 3.1.2, we present our strategies for the collection of Web texts to define the vocabulary of the dictionary.

### Vocabulary for Non-written Languages

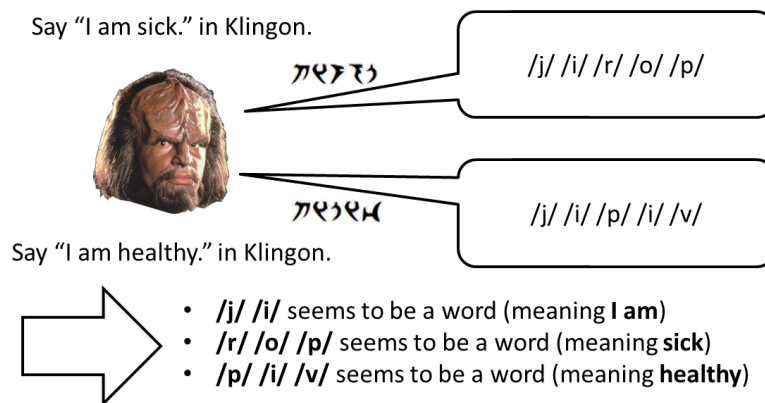
A large number of languages in the world do not have an acknowledged written form. [LSF14] report that approximately 50% of the 7,100 languages have a writing system. It is known that 696 are unwritten languages. For 2,839 languages, [LSF14] have no information if they are written or unwritten. These numbers show that if the goal is to provide speech technology for all languages in the world one day, solutions to rapidly establish systems for non-written languages are required.

If the target language does not have a written form, it is possible to define one. However, defining a writing system and training people to use it consistently is in itself very hard and prone to inconsistencies (e.g. Iraqi Arabic transcription techniques in the Transtac Speech to Speech Translation Project, see [BZG06]).

In contrast to the arduous process of inventing a writing system for non-written languages and dialects, we consider the following scenario to rapidly and economically induce a vocabulary [SW08, SBW09, SSVS12, SSVS13, SSVS14b]: Assuming a recording of spoken phrases is available, the corre-



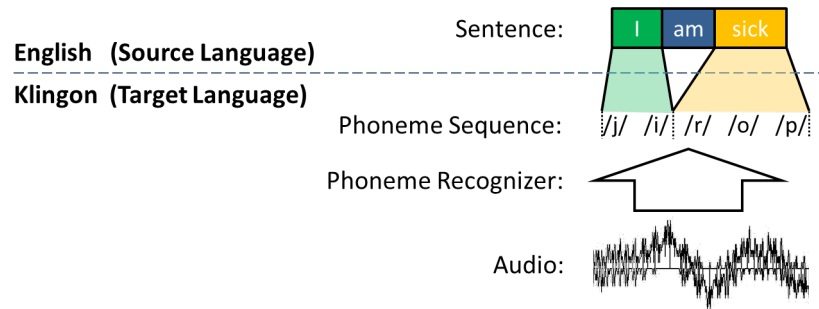
sponding phonetic transcription can be received using a phoneme recognizer. However, the resulting phoneme sequence does not imply any information about the word boundaries. Since exploiting the written translation of the spoken phrases has proven to outperform monolingual approaches, we use it to segment the phoneme sequence into word pseudo units. These word units may be represented by their phoneme sequence or in an orthographic representation after applying phoneme-to-grapheme rules of a related language. The assumption is that a human translator produces utterances in the (non-written) target language from prompts in a resource-rich source language. We align source language words to target language phoneme sequences across languages, i.e. cross-lingually. Based on this alignment, we induce phoneme sequences forming words of the target language.



**Figure 2.1** – Basic scenario for English-Klingon (non-written).

Figure 2.1 demonstrates the scenario that we investigate<sup>1</sup>, and shows which resources are assumed to be available [SSVS13]. English serves in this example as the resource-rich source language. We have chosen the constructed language Klingon [Okr08] spoken by the fictional Klingons in the Star Trek universe to represent a completely unknown under-resourced target language. We assume that we are able to peacefully convince a Klingon understanding English to speak Klingon translations of English sentences – in this case “I am sick” and “I am healthy”. These written English sentences may have been generated by an English automatic speech recognition system. Based only on this limited knowledge, an English speaker may hypothesize that /j//i/, /p//i//v/ and /r//o//p/ are phoneme clusters corresponding to Klingon words, although he or she has no knowledge of Klingon at all. We utilize this idea to build the vocabulary of the target language.

<sup>1</sup>Worf’s head is taken from <http://www.flickr.com/photos/datarknz/3442646145/> (Accessed on 15th November 2013)



**Figure 2.2** – Basic idea of how resources are processed.

As shown in Figure 2.2, we transfer this paradigm to machines as follows: Starting with the written word sequence in the source language on one side and the audio file of the spoken translation in the target language on the other side, the first step is to run a phoneme recognizer and transform the audio file into a phoneme sequence. However, this sequence does not contain any clues of where a word ends and the next word starts. In order to find such word boundaries, we have to consult other knowledge sources (such as other occurrences of the source language words in the corpus) and find a mapping between the words on the source language side and the phonemes on the target language side. In other words, we have to find a *word-to-phoneme alignment*. If we assume a 1:1 relationship between the words in the source and the target language and such a word-to-phoneme alignment is given, it is straightforward to extract word boundary markers: We insert a word boundary maker into the phoneme sequence wherever two neighboring phonemes are aligned to different source language words (i.e. wherever a black alignment line in Figure 2.2 ends). To deal with many-to-one or one-to-many relationships between the words in the source and the target language, we developed methods to post-process the found phoneme clusters which are applied in a later step of the whole pronunciation extraction algorithm described in Chapter 8.

To retrieve the phoneme sequences, we can use a phoneme recognizer of a language similar to the target language in terms of phonological characteristics. However, training acoustic models without transcriptions and recognizing phonemes in an unknown target language is by itself a complex and unsolved research topic. It may be bootstrapped using recognizers from other languages and adaptation techniques as presented in [VKS11] or in [SPC<sup>+</sup>13]. Phonetic language discovery in zero-resource speech technologies [JDG<sup>+</sup>13] (i.e. identifying phoneme like subword units for acoustic modeling in an unknown target language) is addressed among others in

[LG12, VKD08, CHR11]. In this work, we study the influence of phoneme recognition errors to our methods, but do not focus on the phoneme recognition itself.

At this point, we have fully described the available resources summarized again in the following overview:

- Source language sentences in written form,
- Their spoken translations in the target language,
- Phoneme recognizer.

We developed methods that

- extract the vocabulary of the target language and
- reconstruct target language **word** sequences uttered by the speaker.

These methods are described in Section 3.3.

### 2.1.2 Text Normalization

Text Normalization transfers spelling variants of the same word in a canonical representation. Such a processing of text helps to avoid that different representations of the same word are included and blow up the dictionary size and the search vocabulary. Furthermore, non-standard representations in the text such as numbers, abbreviations, acronyms, special characters, dates, etc. are typically normalized. A very nice example from [SK06] for the reduction of the vocabulary size in an English dictionary with text normalization is given in Table 2.3. They reduce the vocabulary of the four example sentences by removing punctuation marks, splitting “s” from the corresponding word and converting all words to lowercase.

In addition to the reduction of the lexical variability, more robust language models can be estimated after text normalization as probability distributions of the same word are not distributed across several representations.

For language-specific text normalization, knowledge of the language in question, which engineers of language and speech technology systems do not necessarily possess, is usually helpful [SZGS10, SZLS13]. Without sufficient language proficiency, engineers need to consult native speakers or language experts. Hiring those people to normalize the text manually can be expensive, and they do not necessarily have the computer skills to implement rule-based text normalization systems. In Section 3.2, we present methods

Mrs. Green is a member of the Greens Garden Club.  
 Bob Green's car is green.  
 The Greens all like eating greens.  
 Green's her favorite color.

<i>Normalization 1</i>		<i>Normalization 2</i>	
<i>graphemic form</i>	<i>phonemic form</i>	<i>graphemic form</i>	<i>phonemic form</i>
green	grin	green	grin
Green	grin	greens	grinz
Green's	grinz	's	z
greens	grinz		
Greens	grinz		

**Figure 2.3** – Reducing the vocabulary size with text normalization [SK06].

where Internet users generate training data for such systems by simple text editing.

Spoken languages without established writing conventions, e.g. Amharic or Luxembourgian before the second half of the 20th century, allow any written form that reflects what is said. The question how to define a word for those languages is traditionally a concern of linguistic research but can be supported with automatic processing.

For some languages with high morphology, a word segmentation in subword units such as morphemes allows to reach a better lexical coverage while keeping the same size of the pronunciation dictionary or even reduce it. This is especially helpful for agglutinative languages like Turkish or Korean or compounding languages like German. Furthermore, there are scripts that completely lack word segmentation, e.g. Chinese and Thai, and segmentation algorithms are required to define lexical units [TMWW00, HKD08].

## 2.2 Pronunciation Generation

### Phoneme Set

The pronunciations of words in a dictionary are described according to manageable units, typically phonemes. Information about the phoneme set of

many languages can be derived from language-specific International Phonetic Alphabet (IPA) charts or the chart of the full IPA [IPA99].

Many developers of automatic speech recognition or text-to-speech systems in new languages use IPA charts to adopt the defined phoneme set of the target language or a subset of it. In our Rapid Language Adaptation Toolkit, the phoneme selection component provides the user with an interface for phonemes in IPA notation, where the user can listen to the target phoneme and select it [SBB<sup>+</sup>07]. Methods to identify and define phonetic units from audio data are still a challenge. For example, [LG12] try to reconstruct the 48 phonetic units in the TIMIT corpus and report 76% F-score. Thus, our phonetic unit selection for new languages is based on IPA and done by native speakers. Only for the experiments where we tackle non-written languages, we use phoneme recognizers of other languages and methods to discover the phonetic units from audio data.

For automatic speech recognition it sometimes makes sense to combine similar phonemes depending on the amount of samples of the phonemes in the audio data in order to model them together in the acoustic model (*phoneme sharing*). For example, with almost 12 hours audio data for training an Ukrainian automatic speech recognition system, we obtain lower word error rates by representing semi-palatalized and non-palatalized phonemes together in the dictionary, thereby modeling them together in the acoustic model [SVYS13].

## Pronunciation Generation

For the rapid generation of pronunciations there is no general solution. “Rapid” means to save expenses and time which is usually coupled with an automatic process for the pronunciation generation, since in a complete manual creation native speakers or linguists need to spend more time on the task which can be time-consuming and expensive. Depending on the situation, one is faced with different challenges for the pronunciation generation. The questions are how to obtain correct pronunciation entries with a certain degree of automation and where to get the linguistic knowledge for the automation from. In the next sections, we demonstrate several scenarios – starting from the simple scenario where a target language dictionary is already present, we have a simple mapping between speech and written language and a dictionary for a new domain is required up to the difficult scenario in which no written form for the target language exists.

### 2.2.1 Lexical Resources on the Web

Except for languages with logographic writing systems, data-driven grapheme-to-phoneme converters trained from existing sample dictionary entries can be used to provide pronunciations for words which we have collected with a vocabulary selection strategy. Those sample dictionary entries can directly be taken from existing dictionaries. If not already available, it usually needs time and money to manually generate word-pronunciation pairs as training data for grapheme-to-phoneme converters. A lower bound for the grapheme-to-phoneme converter training data size gives information how much training data is sufficient to obtain a reliable consistency in the resulting pronunciations.

Crowdsourcing facilitates inexpensive collection of large amounts of data from users around the world [CBD10]. A collection of pronunciations with crowdsourcing methods or from websites where Internet users have entered word-pronunciation pairs supports the process of pronunciation generation. In addition to a cheap way of gathering grapheme-to-phoneme converter training material, using pronunciations provided by Internet users is helpful for languages with a complex grapheme-to-phoneme correspondence where data-driven approaches decrease in performance.

Challenges of crowdsourcing-based approaches for the generation of training data for grapheme-to-phoneme converters are: How to find websites where Internet users have already entered word-pronunciation pairs? And if they cannot be found or do not contain pronunciations in sufficient amount – how to setup the task of pronunciation generation for crowdsourcers. Finally, it is a challenge to assure quality of the crowdsourcing-derived pronunciations, as we show in Chapter 5.

Sample dictionary entries from websites or derived on crowdsourcing platforms can contribute to a rapid and economic semi-automatic pronunciation dictionary development and particularly help to reduce the editing effort to produce pronunciations. As shown in Section 5.3, an optimal strategy for the word selection, the optimal period for the grapheme-to-phoneme converter retraining as well as a phoneme-level combination of the output of multiple grapheme-to-phoneme converters can further benefit the whole process.

### 2.2.2 No Lexical Resources of the Target Language

In this scenario, we cannot obtain sample dictionary entries for grapheme-to-phoneme modeling – maybe since no native speakers are available or it is too expensive to hire them and we cannot find word-pronunciation pairs on the Web.

Therefore, we elaborated a strategy to use grapheme-to-phoneme models from other related languages to bootstrap a dictionary in a new language, as demonstrated in Section 6. To cross-lingually retrieve qualified pronunciations, selecting optimal related languages based on certain criteria such as the language family or grapheme and phoneme coverage of both languages is relevant. Furthermore, the question raises if the statistical rules of one related language achieve success, would a mix of more than one related language perform even better. Our techniques can reduce the effort to produce pronunciations manually and thus to make the manual dictionary generation faster by showing first pronunciations to the native speakers or linguists.

### 2.2.3 In-house Lexical Resources

The easiest scenario is to already possess target language word-pronunciation pairs in a given dictionary and the grapheme-to-phoneme relationship of the language is close. Then, a remaining challenge is to select a vocabulary, i.e. words that are time- and topic-relevant for the new application domain of the automatic speech recognition system [SGVS13, VSKS10].

For the pronunciation generation, we investigate the following two problems:

Due to the close grapheme-to-phoneme relationship, it is possible to train an automatic grapheme-to-phoneme converter from the given word-pronunciation pairs and automatically generate with it the pronunciations for the words of the new domain. Then, the new produced pronunciations may be manually cross-checked subsequently and given enough training data for the grapheme-to-phoneme converter only slight manual modifications are necessary. However, with the globalization more and more words from other languages come into a language without assimilation to the phonetic system of the new language [LB02]. To economically generate pronunciations with automatic or semi-automatic methods, it is important to detect and treat foreign words separately. Due to the strong increase of Anglicisms, especially from the information technology (IT) domain, features for their automatic detection are required.

Pronunciations in existing dictionaries are traditionally those of native speakers. Therefore, a challenge is the use of the pronunciation dictionary where speech of speakers with accent is to be transcribed, e.g. at the airport, in radio communication of aircrafts or where parliamentary speeches in multilingual communities are transcribed, e.g. the cantonal parliament of Valais [IBC<sup>+</sup>12].

We address the problem of detecting foreign words for a separate handling and a lexical adaptation to speakers with accent in Chapter 7.

### 2.2.4 Non-Written Languages

Given is our scenario which assumes a recording of spoken phrases available, the corresponding phonetic transcription received using a phoneme recognizer and the vocabulary derived by exploiting the written translation. To build a pronunciation dictionary, we apply several steps to take phoneme clusters derived with the help of the cross-lingual word-to-phoneme alignment and gain word labels which may be wordIDs or orthographic representations after applying phoneme-to-grapheme rules from a related language together with corresponding pronunciations [SSVS13].

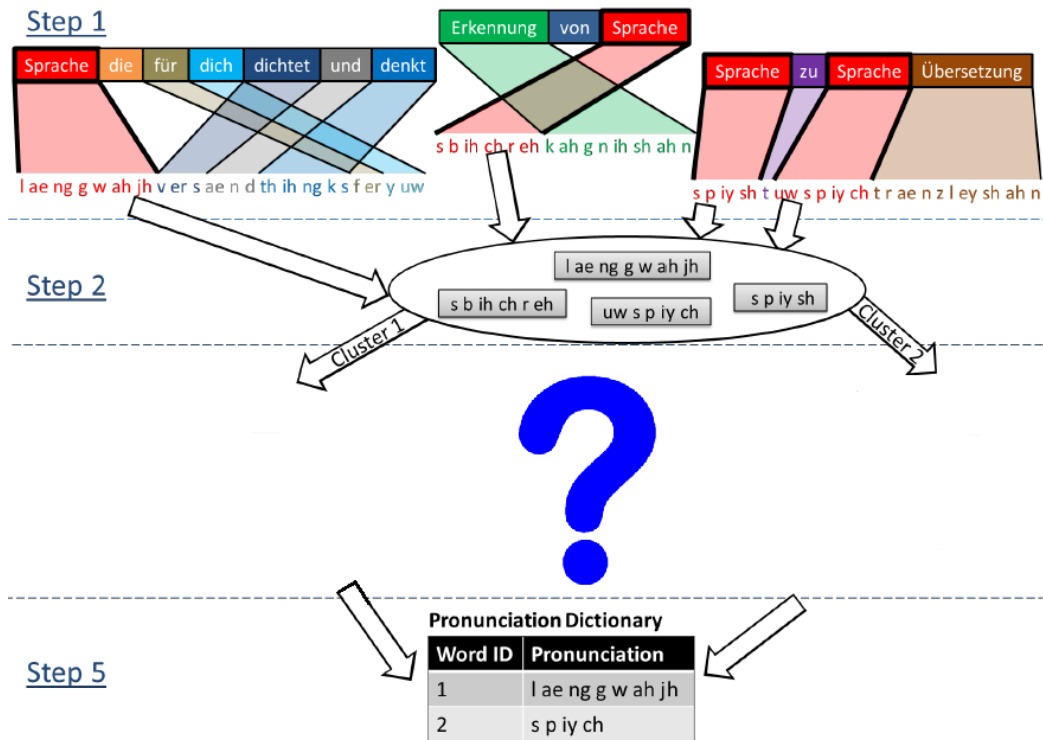
This procedure is visualized in Figure 2.4 with German as source language and English as target language. In Chapter 8, we analyze several steps which include clustering procedures to separate phoneme clusters from each other, which do not represent the same target language word and reconstructing correct phoneme sequences from phoneme sequences with alignment and phoneme recognition errors.

### 2.2.5 Erroneous Pronunciations

As pronunciation dictionaries are so fundamental to speech processing systems, much care has to be taken to select a dictionary that is as free of errors as possible. For automatic speech recognition systems, faulty pronunciations in the dictionary may lead to incorrect training of the system and consequently to a system that does not function to its full potential.

Flawed or inadequate dictionary entries can originate from different subjective judgments, small typographical errors, and 'convention drift' by multiple annotators [SOS12a, Lem13]. One example is our initial *GlobalPhone* Hausa pronunciation dictionary. It has been created in a rule-based fashion and





**Figure 2.4** – Dictionary generation for non-written languages on a German-English example.

was then manually revised and cross-checked by native speakers but still contained a few errors and inconsistencies [SDV<sup>+</sup>12].

As pronunciations from the World Wide Web often lack information about the corresponding word or language, it may happen that inappropriate word-pronunciation pairs are collected which is confirmed by our analyses in [SOS10, SOS14].

Correct pronunciations that do not match the target domain or accent can also lead to worse automatic speech recognition performance. For example, our straightforward insertion of new valid pronunciation variants into our existing Mandarin-English SEAME code-switch dictionary resulted in automatic speech recognition performance degradations since the new pronunciations did not match the target domain and accent [SOS12a].

For grapheme-to-phoneme extraction algorithms the correctness of the dictionary is equally important as each erroneous entry can cause incorrect grapheme-to-phoneme models to be generated, thereby compromising the created dictionary as we show in [SOS12b, SOS14].

To support a rapid but error-less dictionary development, we present methods to detect and revise pronunciations completely without or without much manual effort for written languages in Chapter 4. For our scenario with the non-written languages, we demonstrate methods to compensate for alignment and phoneme recognition errors in Chapter 8.

## CHAPTER 3

# Vocabulary Selection and Text Normalization

---

To select the vocabulary for the pronunciation dictionary, the goal is to collect words that are time- and topic-relevant for the application domain of the automatic speech recognition system. If the vocabulary is too small, the system's out-of-vocabulary rate is high, i.e. the missing words are not able to be recognized, which leads to errors and follow-up errors. In contrast, a decoding vocabulary that contains words not fitting the target domain may result in confusion errors. Researchers and developers have used the World Wide Web as a source to find relevant words based on word frequency and time relevance. In Section 3.1, we describe those methods and extend them providing a supervised and an unsupervised vocabulary adaptation.

Moreover, text normalization steps are applied to prohibit that the same words have different representations and increase the dictionary size and with it the search space in the decoding. To define text normalization rules without knowledge of a language and without much effort, we demonstrate our system in Section 3.2, where Internet users or crowdsourcers who are able to speak the target language deliver resources to build text normalization systems.

Finally, we present our methods to tackle the challenge of defining a vocabulary for non-written languages or dialects in Section 3.3.

## 3.1 Vocabulary Selection

In our supervised language model and vocabulary adaptation strategy, we combine innovative techniques to rapidly bootstrap and extend automatic speech recognition systems in a few days. With our unsupervised language model and vocabulary adaptation strategy [SGVS13], we present methods for the automatic speech recognition of broadcast news as they mostly contain the latest developments, new words emerge frequently and different topics get into the focus of attention.

### 3.1.1 Traditional Methods

Principally, there has been much work to turn to the World Wide Web as an additional source of training data for language modeling. The idea is to collect text from websites (*Crawling*) in order to build statistical language models and adapt existing ones to new topics, domains, or tasks for which little or no training material is available [BOS03]. A vocabulary adaptation is usually associated with a language model adaptation since new words in the dictionary whose probability of occurrence and context is not modeled in the language model, would receive the probability of the language model class for unknown words [Sto02]. Since the language model probability for this class is very low, the chance that the word is recognized is very small. In some works the focus is rather on language model adaptation with the existing vocabulary even if there is potential to reduce the out-of-vocabulary rate with adding new words from the crawled text. However, the authors of the related work report how to retrieve time- and topic-relevant texts, which is relevant for our vocabulary adaptation methods.

For example, in [BOS03] the authors achieve significant word error rate reductions by supplementing training data with text from the Web and filtering it to match the style and topic of a meeting recognition task. Although adding in-domain data is an effective mean of improving language models [Ros95], adding out-of-domain data is not always successful [IO99]. To retrieve relevant texts from the Web, search queries are used [BOS03, SGG05]. Usually, bigram search queries are made by extracting characteristic words of domain-specific documents or Web pages after calculating a Term-Frequency Inverse-Document-Frequency (TF-IDF) score which we introduced in “Language Modeling” in Section 1.2. This score reflects how important a query is to a document in a collection or corpus [SGN, MK06, LGS08, LGP08]. [LDHM12], [Kem99] and [YTTW00] extract topic words from the 1st pass

hypothesis of a show in question and then pass them as a query to a Web search engine. From the retrieved documents, they extract a list of words to adapt the vocabulary for a 2nd-pass decoding.

There has not been much work on vocabulary adaptation based on word frequency and time relevance. Usually, an initial basic vocabulary is iteratively enriched with the most frequent words from crawled online newspapers for each broadcast news show to transcribe. [AGFF00] and [ON07] determine an optimal period of crawling news websites and an optimal frequency-based vocabulary for the broadcast news shows they transcribe in a supervised way, regarding the automatic speech recognition system's performance.

Twitter is an online social networking and microblogging service from Web 2.0 which enables its users to send and read text-based messages of up to 140 characters, known as "Tweets". Tweets are usually updated faster than traditional websites and there is a large amount of data available. However, a restriction is that currently it not possible to get Tweets that are older than 6-8 days with the Twitter REST API<sup>1</sup>. [FR12] adapts a general language model by interpolating it with a language model trained on normalized TV Tweets and improved automatic speech recognition accuracy for a voice-enabled social TV application.

Another paradigm from Web 2.0 are RSS Feeds. They are small automatically generated XML files that contain time-stamped URLs of the published updates. RSS Feeds can easily be found on almost all online news websites. [ABP09] uses RSS Feeds to fetch the latest news texts from the Web. [Mar08] subscribe the RSS news Feeds services of six Portuguese news channels for daily vocabulary and language model adaptation for a broadcast news automatic speech recognition system.

### 3.1.2 Supervised and Unsupervised Vocabulary Adaptation

Compared to related approaches which investigated single techniques, we combine innovative methods in our strategies with the goal to bootstrap and adapt automatic speech recognition systems in rapid and cost-efficient ways.

In our *supervised* language model and vocabulary adaptation strategy [VSKS10], we combine (1) the collection of massive amounts of text data from the Internet, (2) frequency-based vocabulary selection to reduce the out-of-vocabulary

---

<sup>1</sup><https://dev.twitter.com/docs/api/1.1>

rate, (3) language-specific text normalization, (4) a daywise linear language model interpolation, and (5) adding text data diversity. This technique is particularly helpful to rapidly bootstrap and extend automatic speech recognition systems in a few days. This work is described in Section “Supervised Vocabulary Adaptation” below.

With our *unsupervised* language model and vocabulary adaptation strategy [SGVS13], we elaborated a strategy with time- and topic-relevant text from RSS Feed-related websites and Tweets thereby using (1) TF-IDF-based topic word retrieval, (2) linear language model interpolation, (3) 2-pass decoding, and (4) frequency-based vocabulary adaptation. Depending on the quality of an underlying generic baseline language model on our test data, we optimize the vocabulary adaptation technique. We advanced the modules in our Rapid Language Adaptation Toolkit for the text normalization, the collection of RSS Feeds together with the text on the related websites, a TF-IDF-based topic words extraction, as well as the opportunity for language model interpolation. This technique is particularly helpful for an economic recurrent automatic speech recognition of broadcast news because they mostly contain the latest developments, new words emerge frequently and different topics get into the focus of attention. Details are given in Section “Unsupervised Vocabulary Adaptation” below.

### Supervised Vocabulary Adaptation

For a supervised language model and vocabulary adaptation we propose the following steps [VSKS10]:

1. Collection of massive amounts of text data in the target domain from the Internet.
2. Daywise frequency-based vocabulary selection to reduce the out-of-vocabulary rate.
3. Language-specific text normalization.
4. Daywise linear language model interpolation.
5. Adding of text data diversity.

This procedure was evaluated with experiments where we rapidly bootstrapped baseline large vocabulary continuous speech recognition systems in Eastern European languages from the *GlobalPhone* corpus, namely Bulgarian, Croatian, Czech, Polish, and Russian [VSKS10]. A challenge for those languages

is the rich morphology resulting in large vocabulary growth and high out-of-vocabulary rates. The initial dictionary and language model vocabulary of the baseline systems consisted of the words in all utterances of the training transcriptions. The language models of the baseline systems were trained from all utterances of the training data. The out-of-vocabulary rates on our development set were 11.2% for Bulgarian, 12.1% for Croatian, 13.9% for Czech, 16.9% for Polish, and 22.3% for Russian.

We applied the rapid bootstrapping and acoustic model training function in the Rapid Language Adaptation Toolkit, which is based on a multilingual acoustic model inventory [SBB<sup>+</sup>07]. This inventory was trained earlier from seven *GlobalPhone* languages. The results were 63% word error rate for Bulgarian, 60% for Croatian, 49% for Czech, 72% for Polish, and 61% for Russian.

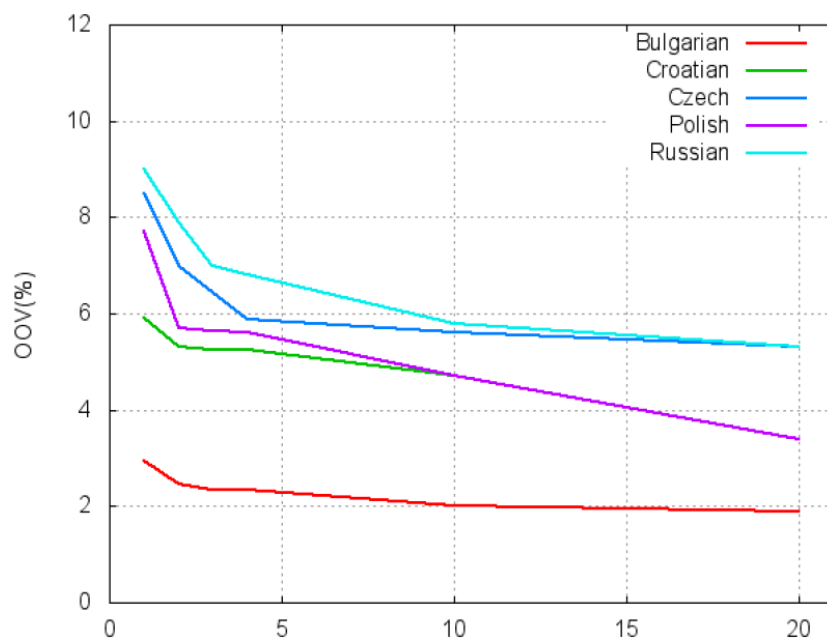
To build a large text corpus in the *GlobalPhone* domain for these languages, we first triggered a crawling process with link depth of 20 for one online newspaper website per language using our Rapid Language Adaptation Toolkit as listed in Table 3.1 [VSKS10]. We applied a link depth of 20, i.e. we captured the content of the given webpage, then followed all links of this page to crawl the content of the successor pages. The process was continued with the respective successors of these pages until the specified link depth was reached.

Languages	Website	# words tokens	# words types
Bulgarian	dariknews.bg	302M	560k
Croatian	www.hrt.hr	124M	248k
Czech	www.lidovky.cz	790M	1,250k
Polish	wiadomosci.wp.pl	347M	815k
Russian	www.rian.ru	565M	1,000k

**Table 3.1** – Initial text corpus size for five Eastern European languages.

To adapt the vocabulary, we selected the 100K most frequent words derived from the collected data and defined the frequency of the last occurring word in the list as threshold. All words occurring more often than this threshold were selected. Day by day we increased the threshold by one, but only if the number of entries in the vocabulary increased. If not, we used the previous threshold. After 20 days, the decoding vocabulary for Bulgarian was 140K, for Croatian 160K, for Czech 197K, for Polish 179K, and for Russian 196K words. This method works quite well in order to control the growth of vocabulary and perplexity on one side and to decrease the out-of-vocabulary rate on the other side as shown in Figure 3.1.

We generated the pronunciations for all words with a grapheme-to-phoneme model that was trained on the initial *GlobalPhone* word-pronunciation pairs using *Sequitur G2P* [BN08], a data-driven grapheme-to-phoneme converter developed at RWTH Aachen University, and complemented the baseline dictionary. An introduction to grapheme-to-phoneme conversion methods is given in Chapter 1.4.



**Figure 3.1** – Out-of-vocabulary rate (%) over days of text crawling.

Furthermore, we improved our language-independent text normalization in the Rapid Language Adaptation Toolkit which includes a removal of HTML tags, code fragments, and empty lines with a language-specific text normalization consisting of (1) special characters were deleted, (2) digits, cardinal numbers, and dates were mapped into text form to match the dictionary, (3) punctuation was deleted, (4) all text data were converted to lowercase. Particularly, the second step involved some linguistic knowledge as in Slavic languages the textual form of numbers changes with gender, numerus, and case of the referring noun. To reduce the human effort, we later developed a crowdsourcing-based text normalization, which we introduce in Section 3.2.2. The four postprocessing steps gave significant relative word error rate reductions of 15% for Bulgarian, 7% for Croatian, 10% for Czech, 12% for Polish, and 6% for Russian on the *GlobalPhone* development sets compared to the systems with initial dictionary and language model.



Websites (#days)	OOV rate	PPL	#words	Vocab
Bulgarian				
24chasa.bg (2)	2.1	904	66M	153K
dnes.bg (2)	2.2	1,099	77M	169K
capital.bg (5)	1.7	808	262M	174K
Interpolated LM	1.2	543	405M	274K
Czech				
halonoviny.cz (5)	5.2	2,699	127M	166K
respek.ihned.cz (5)	6.6	3,468	118M	173K
hn.ihned.cz (5)	5.2	2,600	127M	63K
aktualne.centrum (5)	9.5	3,792	136M	102K
Interpolated LM	3.8	2,115	508M	277K
Croatian				
index.hr (5)	4.5	1,006	71M	218K
ezadar.hz (5)	5.6	1,333	87M	187K
tportal.hr (5)	5.7	1,084	49M	143K
vecernji.hr (5)	6.3	1,884	124M	158K
Interpolated LM	3.6	813	331M	362K
Polish				
fakt.pl (5)	8.2	3,383	79M	136K
nowosci.com.pl (5)	9.0	4,824	45M	90K
wyborcza.pl (5)	3.1	1,673	100M	225K
Interpolated LM	2.9	1,372	224M	243K
Russian				
pravda.ru (3)	4.0	2,039	84M	216K
news.ru (4)	4.6	2,330	91M	222K
bbc.ru (4)	14.5	3,015	23M	34K
news.mail.ru (5)	7.2	3,098	136M	129K
Interpolated LM	3.4	1,675	334M	293K

**Table 3.2** – Additional data from various websites for five Eastern European languages.

We observed that in the crawled text there are parts which match better the target domain represented by the development set text, while others match worse. Consequently, we propose to build individual language models from parts of the crawled text and combine them with weights in a language model interpolation, as introduced in Section 1.2. To keep the computational effort for the interpolations low and avoid data sparseness, we built the individual language models from the text crawled on a day. For 20 days, every day

one language model was built, based on the text data crawled on that day. Based on a linear interpolation of the collection of 20 daily language models, the final language model was created. The interpolation weights were computed using the SRI Language Modeling Toolkit [Sto02], optimized on the development set transcriptions.

So far we had started from one particular website per language to harvest text data. However, this crawling process may be fragile, especially if the starting page is poorly chosen not matching the target domain. In our experiments we observed that for Croatian the crawling process prematurely finished after 10 days, retrieving only a relatively small amount of text data [VSKS10]. Furthermore, an increasing word error rate for Croatian after the third day of crawling indicated that the crawled text data was suboptimal. Therefore, we increased the text diversity by choosing additional websites as starting points for our Rapid Language Adaptation Toolkit crawling process. The days of crawling were limited to up to five days. Based on these additional data, interpolated language models were built and the vocabulary was selected in the same way as described above. Table 3.2 summarizes the URLs of websites, days of crawling, and the performance of the resulting language models on our development set transcriptions.

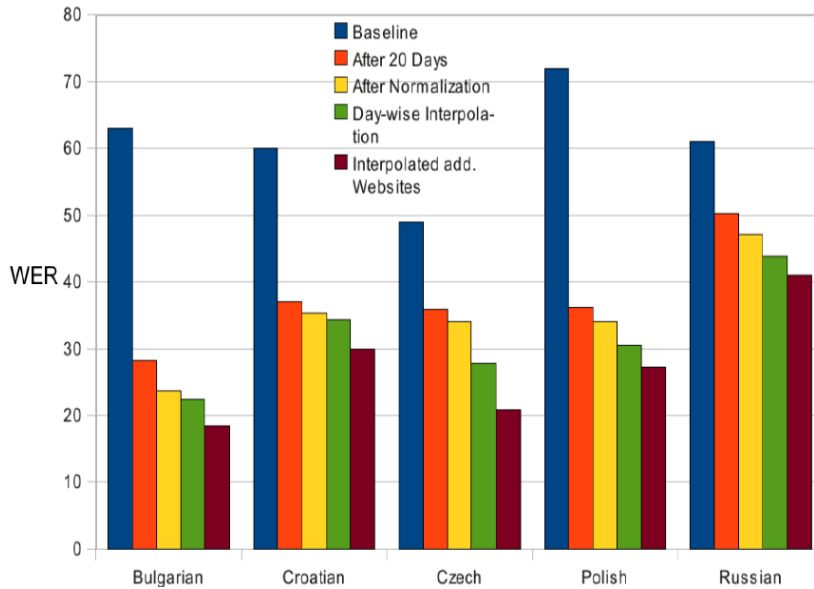
Languages	baseline		final		rel. reduction	
	PPL	OOV (%)	PPL	OOV (%)	PPL	OOV (%)
Bulgarian		11.2	543	1.2		89
Croatian		12.1	813	3.6		70
Czech		13.9	2,115	3.8		73
Polish		16.9	1,372	2.9		83
Russian		22.3	1,675	3.4		85

**Table 3.3** – Out-of-vocabulary rate (OOV) and perplexity (PPL) reductions for five Eastern European languages.

Table 3.3 shows the overall out-of-vocabulary rate and perplexity reductions on the development set transcriptions. We achieve up to 89% relative out-of-vocabulary rate reduction for our Eastern European *GlobalPhone* systems after 20 days using our proposed steps.

The word error rate improvements on our *GlobalPhone* test set achieved with the proposed 5-step procedure are summarized in Figure 3.2. Our results indicate that initial automatic speech recognition systems can be built in very short time and with moderate human effort. We observe that using the texts from the Web has a huge impact on the word error rate. For Bulgarian

and Polish, the word error rates even halve. The text normalization, the interpolation of daily language models plus adding texts from additional websites have less impact but still reduce the word error rate further by approximately 30% relative.

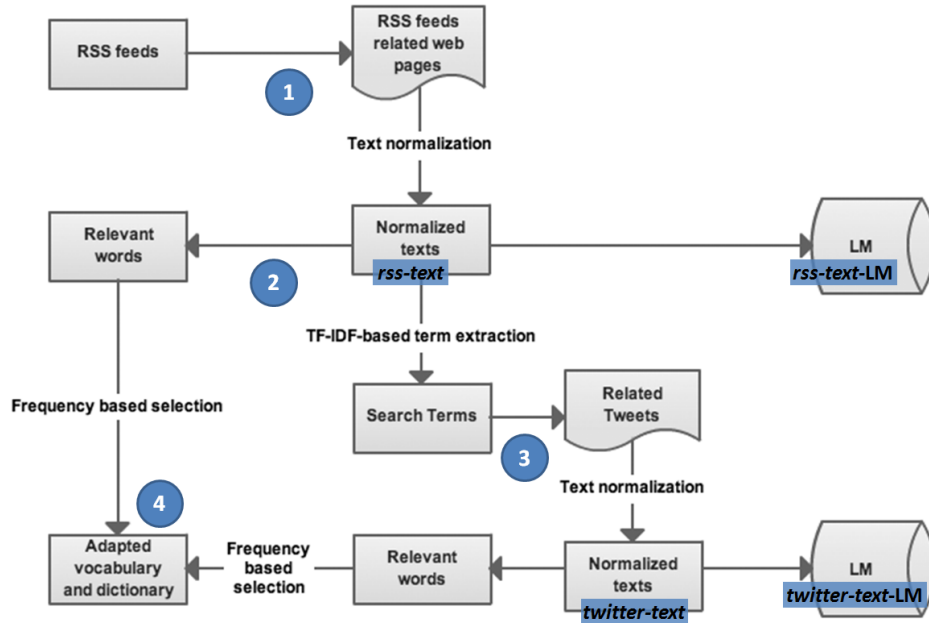


**Figure 3.2** – Word error rate reduction (%) for five Eastern European languages.

### Unsupervised Vocabulary Adaptation

In the method described above for collecting large amounts of text data for language model and vocabulary adaptation, we used recursive crawling in our Rapid Language Adaptation Toolkit. This implementation is good for crawling large amounts of text data. However, it has no functionality to pick out exclusively text material that is relevant for up-to-date broadcast news shows [SGVS13, Gre11, Gre13]. We extended the Rapid Language Adaptation Toolkit with RSS Feeds-based crawling methods and developed a strategy to improve the automatic speech recognition of broadcast news using paradigms from Web 2.0 to obtain time- and topic-relevant text data [SGVS13, Gre11, Gre13].

As shown in Figure 3.3, our strategy for time- and topic-relevance starts with the collection of text which is in near temporal proximity to the date of the news show in focus based on time-stamped URLs of the published updates in RSS Feeds ①. From this text (*rss-text*), we extract topic bigrams based



**Figure 3.3** – Unsupervised language model and vocabulary adaptation.

on a TF-IDF score after text normalization steps as follows ②:

1. Remove stop words<sup>2</sup> in *rss-text*.
2. Compute the frequency of the bigrams in all downloaded documents where the stop words have been removed.
3. For each bigram, compute the number of documents in which the bigram occurs.
4. The bigrams are scored and sorted in decreasing order according to their TF-IDF score.
5. Extract the bigrams with the highest TF-IDF scores<sup>4</sup>.

Then, we search appropriate Tweets with the resulting bigrams using the Twitter API and normalize them (*twitter-text*) ③. *rss-text* and *twitter-text* are used to build language models which are interpolated with an initial generic baseline language model (*base-LM*). To determine optimal interpolation weights, we decode a show in a 1st pass with *base-LM*. As a next step the combination of weights is adopted which reduces most the perplexity on

<sup>2</sup>Stop words are words which are filtered out to optimize search engines. For our test set of French radio broadcasts from Europe 1, 126 French stop words recommended by `ranks.nl`, a Search Engine Optimization organization<sup>3</sup>, worked better than the most frequent words from crawled text.

<sup>4</sup>15 bigrams as search queries worked out to be optimal for our test set.

the 1st pass hypothesis. Based on the most frequent words in *rss-text* and *twitter-text*, the vocabulary of the final language model is adapted ④. A 2nd pass decoding with our final language model results in our news show transcription.

To elaborate and evaluate our strategy in terms of automatic speech recognition performance, perplexity (PPL) and out-of-vocabulary (OOV) rate, we downloaded radio broadcasts of the 7 a.m. news from Europe 1<sup>5</sup> in the period from January 2011 to end of February 2012. Each show has a duration of 10-15 minutes. We evaluated our experiments where we included *rss-text* on ten of these episodes. Validating the impact of *twitter-text* was done only on the last five shows since we decided to include *twitter-text* in August 2011 and it is not possible to retrieve Tweets older than 6-8 days. Reference transcriptions have been created by a French native speaker. In total, all ten broadcast news shows contain 691 sentences with 22.5k running words, the last five shows 328 sentences with 10.8k running words.

To investigate the impact of our strategy on baseline language models with different quality, we adapted two different baseline 3-gram language models (*Base*) that have been successfully applied in French large vocabulary continuous speech recognition but match the domain of our audio data with varying degrees: The French language model from the *GlobalPhone* corpus [SVS13] (*GP-LM*) and a language model which we generated within the Quaero Programme (*Q-LM*) [LCD<sup>+</sup>11]. Their average perplexities and out-of-vocabulary rates on the reference transcriptions of all ten news shows as well as their vocabulary sizes are outlined in Table 3.4.

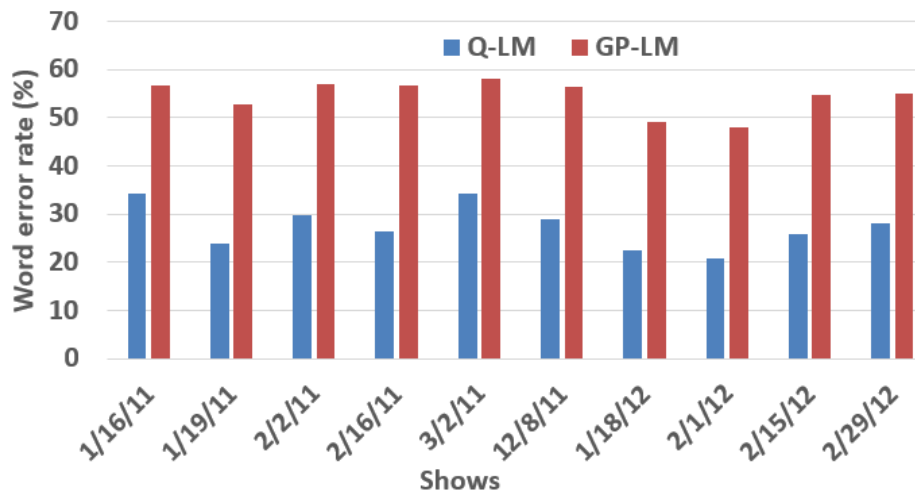
	GlobalPhone ( <i>G-LM</i> )	Quaero ( <i>Q-LM</i> )
Ø PPL	734	205
Ø OOV (%)	14.18	1.65
Vocabulary size	22k	170k

**Table 3.4** – Quality of baseline language models on the broadcast news task.

We collected text data using the information in the RSS Feeds of the four French online news websites from *Le Parisien*, *Le Point*, *Le Monde*, and *France24*. All articles, which were published up to five days before each tested news show, were crawled with the Rapid Language Adaptation Toolkit crawler. A total of on average 385k lines from the RSS Feeds-related websites were collected for each show. Further we gathered Tweets containing 38k lines

<sup>5</sup><http://www.europe1.fr>

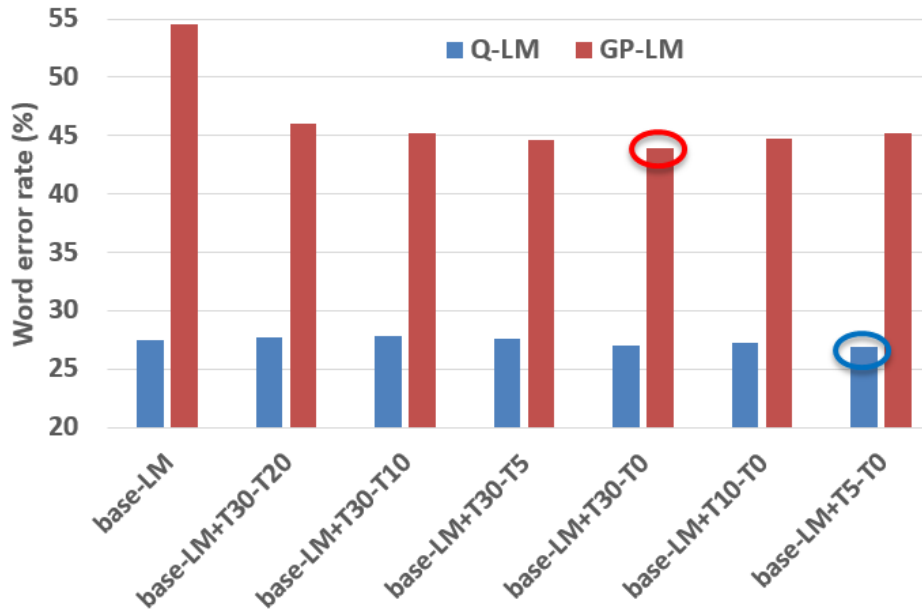
on average for each show. For our experiments we used the acoustic model of our KIT 2010 French Speech-to-Text System, which we describe in [LCD<sup>+</sup>11]. Before the vocabulary adaptation we used the Quaero pronunciation dictionary, which has 247k dictionary entries for 170k words. Figure 3.4 illustrates the word error rates of each Europe 1 show with our *base-LMs*. *Q-LM*, which better matches the domain of the shows in terms of perplexity and out-of-vocabulary rate, also outperforms *GP-LM* in automatic speech recognition performance.



**Figure 3.4** – Word error rates (%) of the baseline systems on the broadcast news task.

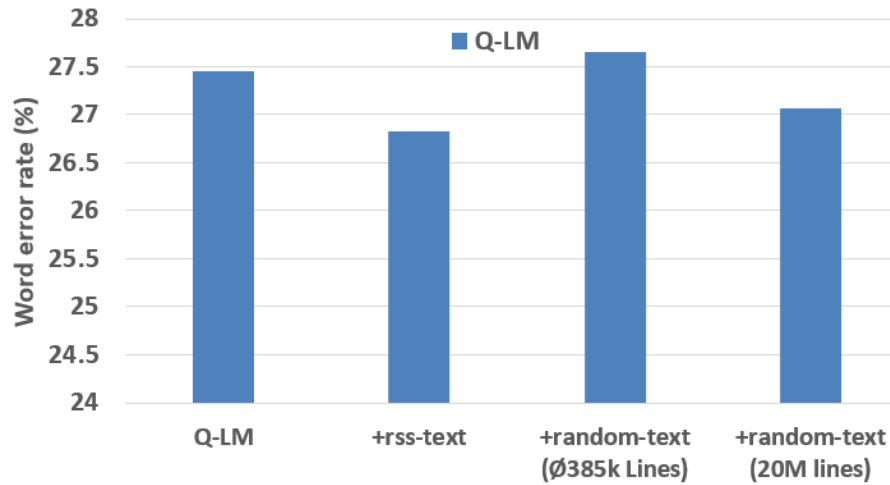
Our 2-pass decoding strategy works as follows: With the help of the SRI Language Modeling Toolkit [Sto02], we train individual 3-gram language models with *rss-text* and *twitter-text* for each show. By interpolating these two Web 2.0-based language models for the show in question with *base-LM*, we create the language model we use for the final decoding of the corresponding show (*adapted-LM*). To determine the language model interpolation weights, the following approach is used:

1. Decoding with *base-LM* (*1st pass*)
2. Tuning of the interpolation weights for *rss-text-LM*, *twitter-text-LM* and *base-LM* on the *1st pass* transcription by minimizing the perplexity of the model.
3. Creation of *adapted-LM* from the interpolation of *rss-text-LM*, *twitter-text-LM* and *base-LM* based on these weights.
4. Re-decoding with the *adapted-LM* (*2nd-Pass*).



**Figure 3.5** – Word error rates (%) with language models containing RSS Feeds-based text data from different periods.

We implemented an RSS parser into the Rapid Language Adaptation Toolkit, which takes RSS Feeds, extracts the URLs with the publishing date and collects them preserving the time information. Then, only the pages corresponding to the listed URLs are crawled. After crawling, HTML tags are removed and the text data are normalized. Figure 3.5 demonstrates the average word error rates with interpolated language models consisting of RSS Feeds-based text data from different periods of time before the shows. Our analyses to find the optimal time period for the texts indicate that most relevant texts are from 30 days to the date of the show with *GP-LM* and from five days before to the date of the show with *Q-LM*. Using text data from less than five days to the date of the show decreased the performance [SGVS13]. Not directly time-relevant text from longer time ago seems to have a positive impact on weaker language models like our *GP-LM* since the text covers at least parts of completely lacking relevant word sequences. In contrast, text from longer time ago seems to harm already relevant word sequences in the stronger language models. Although for *GP-LM* an *rss-text* collection from 30 days to the date of the show (*baseLM+T30-T0*) is better than gathering from five days before the date of the show, we used only *rss-text* from five days before for further experiments with *GP-LM*. The reason is that we had to extract topic words from *rss-text* which are relevant for the search for



**Figure 3.6** – Word error rate (%) with language models containing RSS Feeds-related text compared to random text data.

Tweets and it is not possible to get Tweets that are older than 6-8 days with the Twitter API.

Figure 3.6 illustrates the average word error rates of all ten tested shows. We see that on average 385k lines of *rss-text* for the adaptation of each show improved automatic speech recognition performance compared to *Q-LM*, while using the same number of lines of randomly selected texts from a recursive crawl of a news website (*+randomText*) decreased the performance. Even 20 million lines of randomly selected texts from traditional recursive crawls did not outperform *rss-text*, which indicates its high relevance.

From *rss-text*, we extracted topic words based on TF-IDF to search relevant French Tweets with the Twitter API in the period from five days before to the date of the show. Then we applied the following text normalization steps to the selected Tweets similar to [FR12]:

1. Remove URLs plus retweet (“RT:”) and mention markers (“@username”).
2. Remove very short Tweets.
3. Remove Tweets being only in uppercase.
4. Remove Tweets containing more than 50% unknown or misspelled words according to French GNU aspell<sup>6</sup>.
5. Extend abbreviations.

<sup>6</sup>aspell.net



Figure 3.7 shows that for the last five shows on average 1.5% relative word error rate reduction is achieved by incorporating the *twitter-text*-LM (*Base+RSS+Tweets*) besides the *rss-text*-LM with both underlying *base*-LMs.

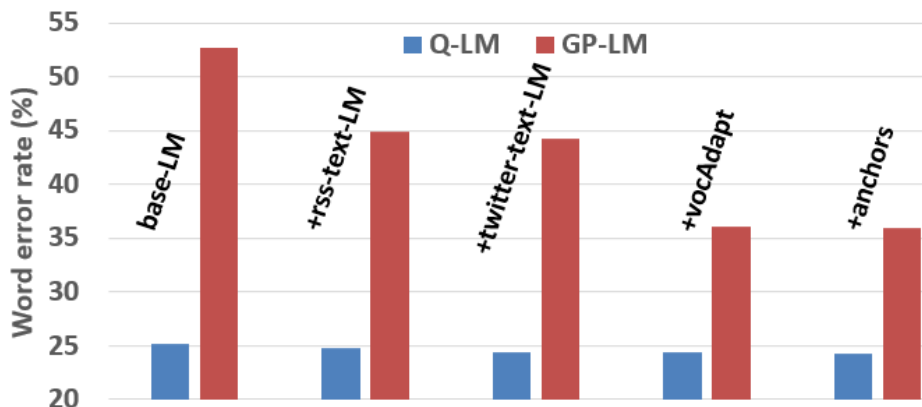


Figure 3.7 – Word error rates (%) for *Q-LM* and *GP-LM*.

To gain additional performance improvements, we adapt the vocabulary of our language models (*vocAdapt*) and our decoding dictionary. We experimented with different vocabulary adaptation strategies. The missing French pronunciations were generated with the grapheme-to-phoneme converter *Sequitur G2P*, which was trained with the known word-pronunciation pairs from the Quaero dictionary.

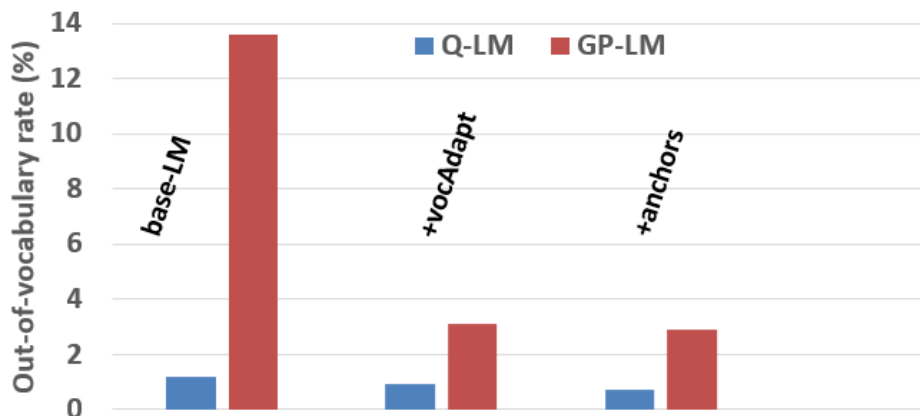
For *GP-LM* which has a high out-of-vocabulary rate, the following strategy performs best:

1. Generate a list of words present in the concatenation of *rss-text* and *twitter-text* with the corresponding number of occurrences.
2. From this list, remove all words present only once in our text data.
3. The remaining words still not present in the search vocabulary are added.

With this strategy on average 19k words are added to the vocabulary for each show. Due to their considerably lower out-of-vocabulary rates, we worked out another strategy for *Q-LM*:

1. Reduce words in the language model to improve the perplexity by removing the words with the lowest probability.
2. Remove those words in the decoding dictionary as well.
3. Add the most frequent new words present in the concatenation of *rss-text* and *twitter-text*.

We experimented with different vocabulary sizes to find a good balance between an increased out-of-vocabulary rate and a lower perplexity. Optimal is a new baseline vocabulary with 120k words plus the 1k most frequent words from the concatenation of *rss-text* and *twitter-text*.



**Figure 3.8** – Out-of-vocabulary rates (%) for *Q-LM* and *GP-LM* before and after vocabulary adaptation.

	Q-LM	GP-LM
Adding <i>rss-text</i>	1.59	14.77
Adding <i>twitter-text</i>	1.53	1.51
Vocabulary adaptation based on <i>rss-text+twitter-text</i>	0.08	18.41
Adding names of news anchors	0.66	0.39
Total WER rate improvement	3.81	31.78

**Table 3.5** – Relative word error rate reduction (%) for the last five shows with our text collection and decoding strategy.

Moreover, we manually added the names of the news anchors to the vocabulary as their names were still not present in the adapted vocabulary (*+anchors*). Listening to only one of the shows gives information about the names since they occur in each show. The word error rate reduction with the vocabulary adaptation is shown in Figure 3.7. The out-of-vocabulary rate decrease is illustrated in Figure 3.8. We achieved up to 71% relative out-of-vocabulary rate reduction using our vocabulary adaptation strategies.

As summarized in Table 3.5, the word error rate of the five tested French broadcast news shows from Europe 1 are reduced by almost 32% relative

with an underlying language model from the *GlobalPhone* project and by almost 4% with an underlying language model from the Quaero project.

### 3.1.3 Summary

First, we have investigated a strategy for the crawling of massive amounts of text data and investigated the impact of text normalization and text diversity on the vocabulary, the language model and its influence on automatic speech recognition performance. Our results indicate that initial automatic speech recognition systems can be built in very short time and with moderate human effort.

Second, we have presented an unsupervised strategy to automatically adapt generic language models to the several topics that can be encountered during a transcription, especially in broadcast news. We crawled appropriate texts from RSS Feeds, complemented it with texts from Twitter, performed a language model and vocabulary adaptation, as well as a 2-pass decoding. For that, we advanced the modules in our Rapid Language Adaptation Toolkit for the text normalization, the collection of RSS Feeds together with the text on the related websites, a TF-IDF-based topic words extraction, as well as the opportunity for language model interpolation. We have shown the relevance of RSS Feeds-based text and Tweets. Future work may include further paradigms from Web 2.0 such as social networks to obtain time- and topic-relevant text data.

## 3.2 Text Normalization

In the previous sections we have shown that text normalization can give significant improvements within our demonstrated language model and vocabulary adaptation methods. For example, Table 3.2 shows that applying language-specific text normalization steps in our supervised adaptation strategy reduces the word error rate in the range of 3% to 18% relative. For rapid development of text normalization systems at low cost, we propose methods where Internet users generate training data for such systems by simple text editing.

### 3.2.1 Traditional Methods

A text normalization for French and its impact on automatic speech recognition was investigated in [AADGL97]. The authors use 185 million words of a French online newspaper and propose different steps such as processing of ambiguous punctuation marks, processing of capitalized sentence starts as well as number normalization.

[GJWW06] treat the text normalization in a similar way to machine translation with the normalized text being the target language. A transfer-based machine translation approach is described which includes a language-specific tokenization process to determine word forms.

A statistical machine translation approach for text normalization is proposed in [HH09] where an English chat text was translated into syntactically correct English. First, some preprocessing steps are applied containing an extraction of <body> tag content, removal of HTML characters, conversion into lower case, line split after punctuation marks as well as language-specific text normalization such as correction of some word forms and tokenization of the text. From the remaining 400k sentences, 1,500 sentences are utilized for tuning and another 1,500 for testing, while the other lines are used for training. [HH09] report an edit distance of 0.3% on the News Commentary corpus data and Web data.

[AZXS06] apply a phrase-based statistical machine translation for English SMS text normalization. With a corpus of 3k parallel non-normalized and normalized SMS messages, they achieve a BLEU score of 80.7%.

In addition to a statistical machine translation-based text normalization system, [KYD08] present an “ASR-like” system that converts the graphemes of non-normalized text to phonemes based on a dictionary and rules, creates a finite state transducer for transducing phoneme sequences into word sequences with an inverted dictionary and finally searches the word lattice for the most likely word sequence incorporating language model information. The “ASR-like” approach is worse than the SMT approach and leads only to slight improvement in a system combination scheme. Alternative “noisy channel” approaches treat the text normalization problem as a spelling correction problem [CG91, BM00, TM02].

Since the statistical machine translation-based approach has comparable results to the other approaches and requires less memory and training time than other “noisy channel” approaches such as Conditional Random Fields, we decided to select this approach for our experiments. Another advantage is

that the Moses Package [KHB<sup>+</sup>07], GIZA++ [ON03] and the SRI Language Modeling Toolkit [Sto02] provide a framework to automatically create and apply statistical machine translation systems. However, our crowdsourcing-based approach is also applicable with other methods which require a parallel corpus for training. For more information about statistical machine translation we recommend [Kni99].

### 3.2.2 SMT-based Text Normalization through Crowdsourcing

Our research interest was to output text in high quality for automatic speech recognition and speech synthesis with statistical machine translation systems [SZGS10, SZLS13]. However, the statistical machine translation systems are supposed to be built with training material, which does not need much human effort to create it. To keep the human effort low, we use rules for the non-language-specific part of the text normalization and employ humans only for parts which requires language proficiency.

For a rapid development of speech processing applications at low cost, we propose text normalization systems constructed with the support of Internet users. The users normalize sentences displayed in a Web interface. In contrast to the grammatical definition, we use the term “sentence” for all tokens (characters separated by blanks) located in one line of the crawled text. Based on the normalized text generated by the user and the original non-normalized text, statistical machine translation models [Kni99] such as translation model, language model and distortion model can easily be created. With these models, we treat the text normalization as a monotone machine translation problem, similar to the way we have tackled the diacritization problem in [SNV08].

The main goal was to investigate if the development of normalization tools can be performed by breaking down the problem into simple tasks which can be performed in parallel by a number of language proficient users without the need of substantial computer skills. Furthermore, the work examines the performance of normalization as a function of the amount of data.

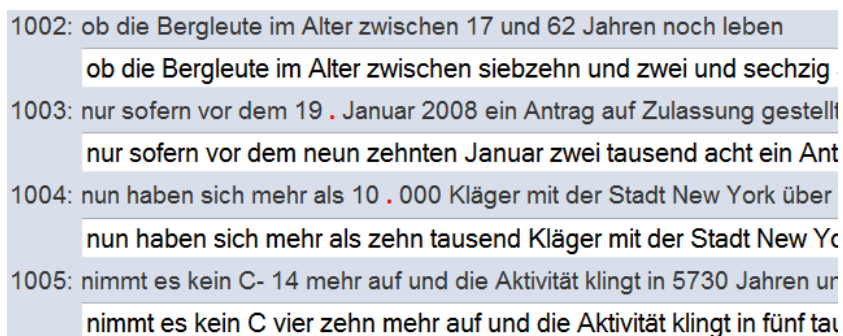
#### Crowdsourcing

The crowdsourcing platform Amazon Mechanical Turk facilitates an inexpensive collection of large amounts of data from users around the world [CBD10].

For the NAACL 2010 Workshop, the platform has been analyzed to collect data for human language technologies. For example, it has been used to judge machine translation adequacy as well as to build parallel corpora for machine translation systems [DL10, AV10]. As our annotation work can be parallelized to many users, we provided tasks to normalize English texts to Turkers and checked the resulting quality [SZLS13].

### Web Interface and Backend

The user is provided with a simple readme file that explains how to normalize the sentences, i.e. remove punctuation, remove characters not occurring in the target language, replace common abbreviations with their long forms, apply a number normalization, etc. In our web-based interface, sentences to normalize are displayed in two lines [SZGS10, SZLS13]: The upper line shows the non-normalized sentence, the lower line is editable. Thus, the user does not have to write all words of the normalized sentence. An excerpt of the Web-based front-end is shown in Figure 3.9. After editing 25 sentences, the user presses a save button and the next 25 sentences are displayed. We present the sentences in random order to the user. For our French system, we observed better performances by showing sentences with dates, ordinal, cardinal and nominal numbers to the user first in order to enrich the phrase table with normalized numbers early [SZGS10]. However, for our Bulgarian, English, and German systems, displaying the sentences in random order, thereby soon inserting normalization of numbers, casing and abbreviations into the phrase table in equal measure, performed better [SZLS13]. Except for our Amazon Mechanical Turk experiments, we take the user output for granted and perform no quality cross-check for simplicity.



1002: ob die Bergleute im Alter zwischen 17 und 62 Jahren noch leben  
ob die Bergleute im Alter zwischen siebzehn und zwei und sechzig

1003: nur sofern vor dem 19 . Januar 2008 ein Antrag auf Zulassung gestellt  
nur sofern vor dem neun zehnten Januar zwei tausend acht ein Ant

1004: nun haben sich mehr als 10 . 000 Kläger mit der Stadt New York über  
nun haben sich mehr als zehn tausend Kläger mit der Stadt New Yc

1005: nimmt es kein C- 14 mehr auf und die Aktivität klingt in 5730 Jahren ur  
nimmt es kein C vier zehn mehr auf und die Aktivität klingt in fünf tau

**Figure 3.9** – Web-based user interface for text normalization.

To generate phrase tables containing phrase translation probabilities and lexical weights, the Moses Package [KHB<sup>+</sup>07] and GIZA++ [ON03] are used. By default phrase tables containing up to 7-gram entries are created. The 3-gram language models are generated with the SRI Language Modeling Toolkit [Sto02]. A minimum error rate training to find the optimal scaling factors for the models based on maximizing BLEU scores as well as the decoding are performed with the Moses Package.

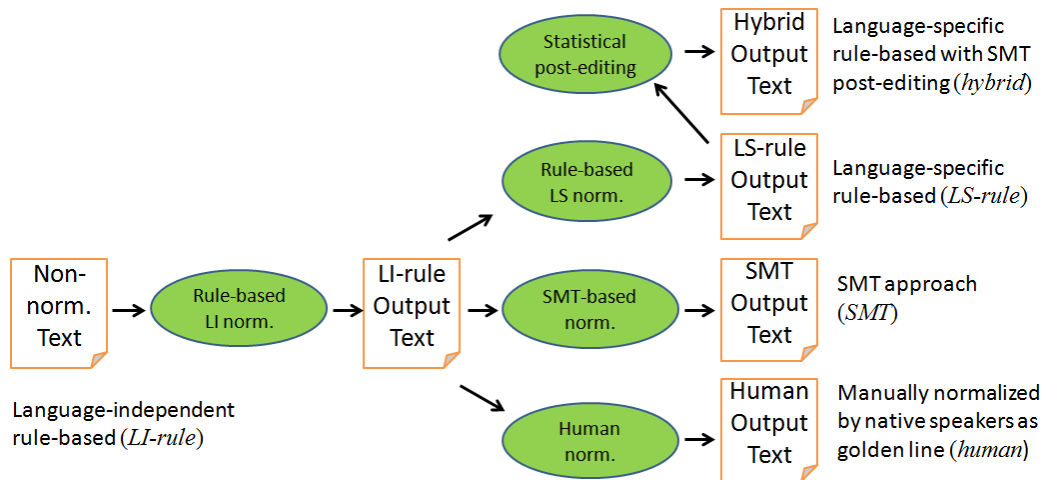


Figure 3.10 – Text normalization systems.

### Performance over Training Data

Figure 3.10 demonstrates the text normalization systems which we evaluated. For the evaluation, we compared text normalized ...

- with a language-independent rule-based text normalization system (*LI-rule*)
- with a language-specific rule-based text normalization system (*LS-rule*)
- with a pure statistical machine translation-based text normalization system (*SMT*).
- with a language-specific rule-based text normalization system with statistical phrase-based post-editing (*hybrid*)
- with a manual text normalization by native speakers (*human*) [SZGS10, SZLS13].

We evaluated our systems for Bulgarian, English, French, and German with different amounts of training data. For English, French, and German, the

quality of 1k output sentences derived from the systems is compared to text which was normalized by native speakers in our lab (*human*). With the Levenshtein edit distance, we analyzed how similar both texts are. As we are interested in using the processed text to build language models for automatic speech recognition tasks, we created 3-gram language models from the systems' hypotheses and evaluated their perplexities (PPLs) on 500 sentences manually normalized by native speakers. For Bulgarian, the set of normalized sentences was smaller: We computed the edit distance of 500 output sentences to *human* and built a language model. Its perplexity was evaluated on 100 sentences manually processed by native speakers. The sentences were normalized with *LI-rule* in our Rapid Language Adaptation Toolkit. Then *LS-rule* was applied to this text by the Internet users. *LI-rule* and *LS-rule* are itemized in Table 3.6.

<b>Language-independent Text Normalization (<i>LI-rule</i>)</b>
1. Removal of HTML, Java script and non-text parts.
2. Removal of sentences containing more than 30% numbers.
3. Removal of empty lines.
4. Removal of sentences longer than 30 tokens.
5. Separation of punctuation marks which are not in context with numbers and short strings (might be abbreviations).
6. Case normalization based on statistics.
<b>Language-specific Text Normalization (<i>LS-rule</i>)</b>
1. Removal of characters not occurring in the target language.
2. Replacement of abbreviations with their long forms.
3. Number normalization (dates, times, ordinal and cardinal numbers, etc.).
4. Case norm. by revising statistically normalized forms.
5. Removal of remaining punctuation marks.

**Table 3.6** – Language-independent and -specific text normalization.

As demonstrated in Figure 3.11, text quality improves with more text used to train the statistical machine translation system for Bulgarian, English, and German [SZLS13]. This reproduces our conclusions we reported for French in [SZGS10]. Exceeding a certain amount of training sentences, we gained lower perplexities with *SMT* than with *LS-rule* for Bulgarian, English and German. This originates from the fact that human normalizers are better in correcting typos and casing as well as detecting the correct forms in the number normalization (especially the correct gender and number agreement) due to their larger context knowledge which is more limited in our rule-based



normalization systems. While for our French texts, a performance saturation started at already 450 sentences used to train the statistical machine translation system, we observe a saturation at approximately 1k for Bulgarian, 2k for English, and 2k for German sentences. Consequently, the amount of required training data to obtain a static quality varies depending on the input text quality and the language. *hybrid* obtained a better performance than *SMT* and converges to the quality of *human* for all languages. This shows that with the statistical phrase-based post-editing in *hybrid* it is possible to automatically correct systematic errors made by rule-based systems.

### Performance with Amazon Mechanical Turk

The development of our normalization tools can be performed by breaking down the problem into simple tasks which can be performed in parallel by a number of language proficient users without the need of substantial computer skills. Everybody who can speak and write the target language can build a text normalization system due to the simple self-explanatory user interface and the automatic generation of the statistical machine translation models. Amazon's Mechanical Turk service facilitates inexpensive collection of large amounts of data from users around the world. However, Turkers are not trained to provide reliable annotations for natural language processing tasks, and some Turkers may attempt to cheat the system by submitting random answers. Therefore, Amazon provides requesters with different mechanisms to help ensure quality [CBD10]. With the goal to find a rapid solution at low cost and to get over minor errors creating statistical rules for our statistical machine translation systems, we did not check the Turker's qualification. We rejected tasks that were obviously spam to ensure quality with minimal effort. Initially, the Turkers were provided with 200 English training sentences which had been normalized with *LI-rule* together with the readme file and example sentences. Each Human Intelligence Task (HIT) was to annotate eight of these sentences with all requirements described in the readme file. While the edit distance between *LI-rule* and our ground truth (*human*) is 34% for these 200 sentences, it could be reduced to 14% with the language-specific normalization of the Turkers (*mT-all*). The analysis of the confusion pairs between *human* and *mT-all* indicates that most errors of *mT-all* occurred due to unrevised casing. As the focus of the annotators was rather on the number normalization with *mT-all*, we decided to provide two kinds of HITs for each set of eight sentences which contain numbers (*mT-split*): The task of the first HIT was to normalize the numbers, the second one to correct wrong cases in the output of the first HIT together with the other requirements. The

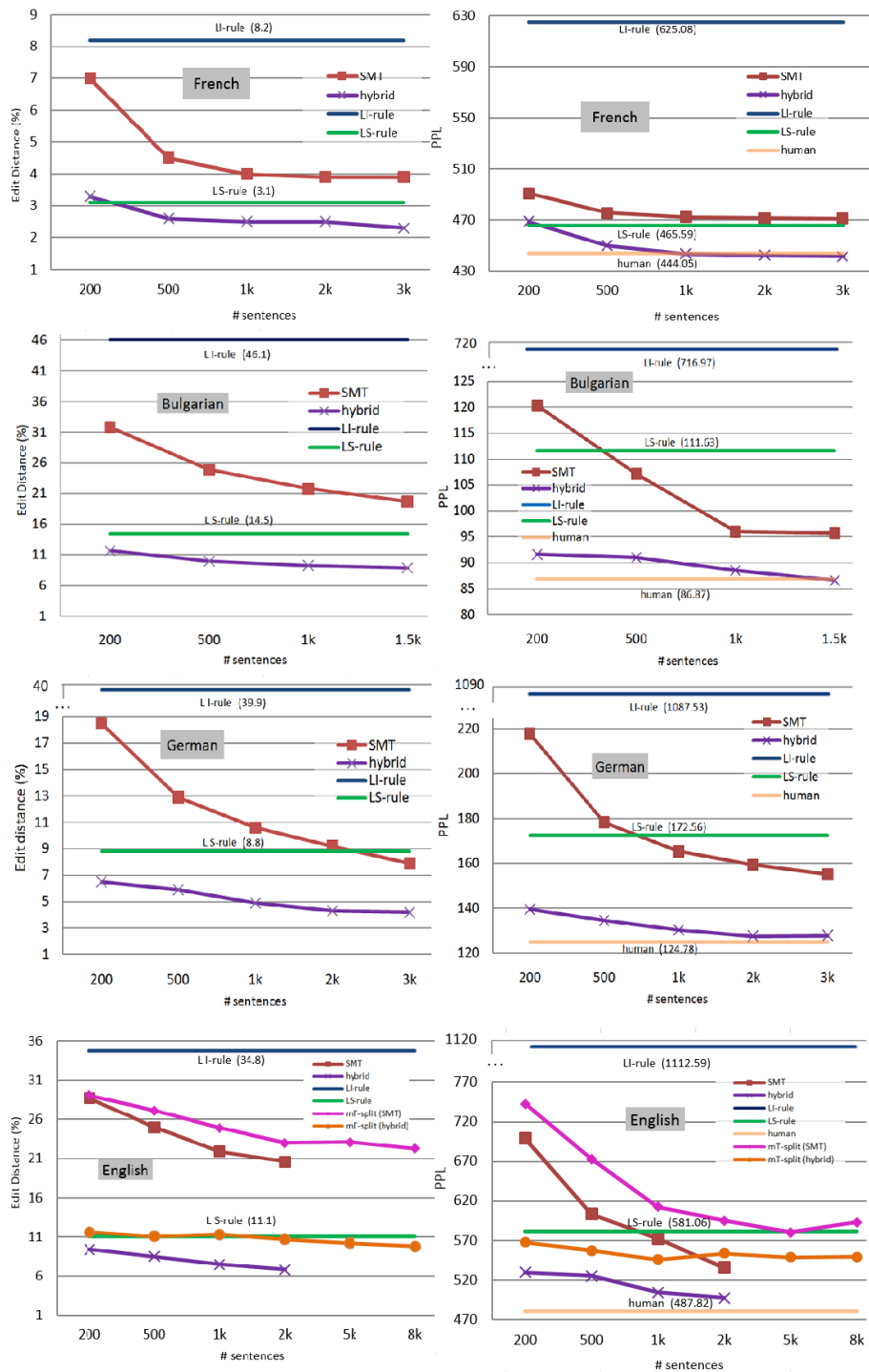


Figure 3.11 – Edit distance (%) and perplexity over amount of training data.

benefit of concentrating either on the numbers or on the other requirements resulted in an edit distance of 11% between *mT-split* and *human* which is 21% relative lower than with *mT-all*.

Finally, all 8k English training sentences were normalized with *mT-split* and used to build new statistical machine translation systems as well as to accumulate more training sentences for our existing system built with 2k sentences thoroughly normalized in our lab. As shown in Figure 3.11, the quality of *mT-split* is worse with the same training sentences than those created with our thoroughly normalized sentences (*SMT*) in terms of edit distance and perplexity. While the different normalizers in our lab came to an agreement if diverse number representations were possible, the Turkers selected different representations to some extent, e.g. “two hundred five”, “two hundred and five” or “two oh five”, depending on their subjective interpretation of what would be said most commonly. We explain the fluctuations in *mT-split* (*hybrid*) with such different representations plus incomplete normalizations in the annotated training sentences. We recommend a thoroughly checked tuning set for the minimum error rate training if available since we could build better statistical machine translation systems with a tuning set created in our lab (*tune-lab*) than with one created by the Turkers (*tune-mT*). Revising the sentences normalized by the Turkers, which requires less editing effort than starting to normalize the sentences from scratch, would further improve the systems.

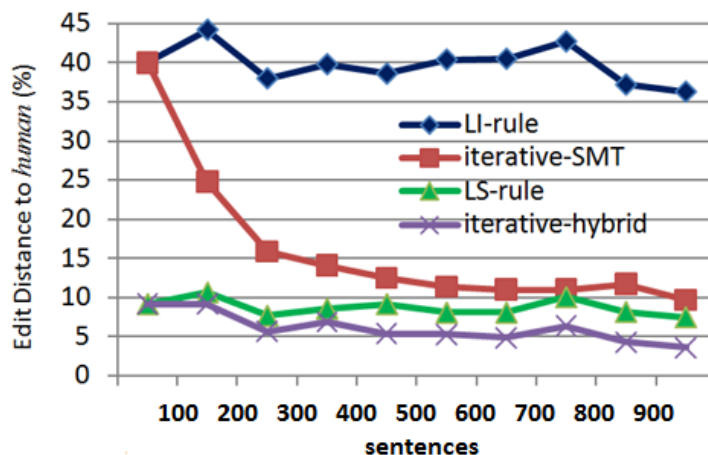
More information about our Amazon Mechanical Turk experiment is summarized in Table 3.7 and 3.8. For \$17, the Turkers worked about 10 hours to normalize 2k sentences. A statistical machine translation-based text normalization system trained with these sentences outputs text which can be used to generate language models with half the perplexity than those generated with text normalized only in a language-independent way. More than 2k edited training sentences resulted in only slight performance improvements. The parallel processing led to a speedup of 1.5.

# training sentences	<i>LI</i> -PPL	<i>LS</i> -PPL	<i>SMT</i> -PPL ( <i>mT-split</i> )	<i>hybrid</i> -PPL ( <i>mT-split</i> )
after 2k	1112.59	581.06	<b>595.06</b>	<b>551.33</b>
after 8k	1112.59	581.06	<b>592.98</b>	<b>549.03</b>

**Table 3.7** – Amazon Mechanical Turk experiments: Perplexity results.

# training sentences	hybrid-PPL worktime $T_{mT}$	Effective by 1 Turker $T_1$	$\emptyset$ time 1 sent. $T_{seq} (n * T_1)$	Sequence time $(T_{seq}/T_{mT})$	Speedup $S_{mT}$ costs
after 2k	10.1 hrs.	27.3 sec.	14.7 hrs.	1.45	\$17.01
after 8k	30.5 hrs.	19.7 sec.	45.0 hrs.	1.48	\$48.62

**Table 3.8** – Amazon Mechanical Turk experiments: Time and cost.



**Figure 3.12** – Edit distance reduction with *iterative-SMT*/*-hybrid*.

### User Effort Reduction

To reduce the effort of the Internet users who provide normalized text material, we iteratively used the sentences normalized so far to build the statistical machine translation system and applied it to the next sentences to be normalized (*iterative-SMT*) [SZLS13]. With this approach, we were able to reduce the edit distance between the text to be normalized and the normalized text, resulting in less tokens the user has to edit. If a language-specific rule-based normalization system is available, the edit distance can also be reduced with that system (*LS-rule*) or further with a language-specific rule-based text normalization system with statistical phrase-based post-editing (*iterative-hybrid*). As corrupted sentences may be displayed to the user due to shortcomings of statistical machine translation system and rule-based system, we propose to display the original sentences to the user as well.

Figure 3.12 shows lower edit distances for the first 1k German sentences with *iterative-SMT* and *iterative-hybrid* compared to the previous system where text, exclusively normalized with *LI-rule*, was displayed to the user. After each 100 sentences, the training material for the statistical machine transla-

tion system was enriched and the statistical machine translation system was applied to the next 100 sentences.

### 3.2.3 Summary

We have presented a crowdsourcing approach for statistical machine translation-based language-specific text normalization which came close to our language-specific rule-based text normalization (*LS-rule*) with French online newspaper texts and even outperformed *LS-rule* with the Bulgarian, English, and German texts. The statistical machine translation system which translates the output of the rule-based system (*hybrid*) performed better than *SMT* and came close to the quality of text normalized manually by native speakers (*human*) for all languages. The annotation process for English training data could be realized fast and at low cost with Amazon Mechanical Turk. The results with the same amounts of text thoroughly normalized in our lab are slightly better which shows the need for methods to detect and reject Turkers' spam. Due to the high ethnic diversity in the U.S. where most Turkers come from and Turkers from other countries [RIS<sup>+</sup>10], we believe that a collection of training data for other languages is also possible. Finally, we have proposed methods to reduce the editing effort in the annotation process for training data with *iterative-SMT* and *iterative-hybrid*. Instead of statistical machine translation, other "noisy channel" approaches can be used in our back-end system. Future work may include an evaluation of the systems's output in automatic speech recognition and text-to-speech systems.

## 3.3 Vocabulary Extraction for Non-Written Languages

To extract the vocabulary for non-written languages, we assume a written translation and a recording of spoken phrases being available. We can derive the corresponding phonetic transcription using a phoneme recognizer. However, the resulting phoneme sequence does not imply any information about the word boundaries. We use the written translation of the spoken phrases to segment the phoneme sequence into word units.

### 3.3.1 Challenges

To be processed in language technology, spoken words need a written representation. For non-written languages methods to extract the vocabulary from text are not possible unless an artificial writing system is defined. Therefore, we automatically extract phoneme sequences from audio data and define word units, which build up the vocabulary.

The phoneme sequence which is obtained using a phoneme recognizer does not contain any information about the word boundaries. Since exploiting the written translation of the spoken phrases has proven to outperform monolingual approaches where the segmentation is only estimated on the whole phoneme sequence [SSVS12], we use the written translation to segment the phoneme sequence into word units. These word units may be represented by their phoneme sequence or in an orthographic representation after applying phoneme-to-grapheme rules of a related language.

We align source language words to target language phoneme sequences cross-lingually. Based on this alignment, we induce phoneme sequences forming words of the target language. This word units are used for our methods to build pronunciation dictionaries for non-written language, which we describe in Chapter 8.

The approach has two major challenges [Sta11]:

- **High phoneme error rates.** One great advantage of our approach is that no a priori knowledge about the target language is necessary. At the same time, this is a severe drawback due to the fact that no acoustic models and therefore no phoneme recognizer is available for the target language – even the phoneme set might be unknown. Consequently, we have to run a phoneme recognizer trained on speech data of one or more languages to obtain the phoneme sequences in the target language. This introduces high phoneme recognition errors.
- **Inaccurate alignments.** Another major challenge is to compensate for frequent alignment errors. We use alignment models derived from statistical machine translation to find word-to-phoneme mappings. However, as in general the solution space for word-to-phoneme alignments is significantly larger than for word-to-word alignments, finding such alignments automatically is even harder. Furthermore, high phoneme error rates interfere with statistical alignment models.

### 3.3.2 Related Work

If the target language does not have a written form, an option is to define one. However, defining a writing system manually is very hard and expensive. As we operate only at the phoneme level on the target side, we implicitly introduce an artificial writing system, where the words are represented by their pronunciations (*word labels*) [SSVS14b]. This enables to bypass the written form for the purpose of speech-to-speech translation.

Unsupervised word or morphology segmentation in machine learning relies primarily on statistical models [Gol10]. In addition to bilingual approaches as ours, the following monolingual methods for word segmentation have been used in the past: First, Minimal Description Length analysis [Kit00, Gol06] approximates the optimal compression of a (phoneme-)string (corresponding to its Kolmogorov complexity). Assuming that a word sequence of a language is the optimal compression of the corresponding phoneme sequence, the data segmentation induced by such compression methods is taken as the word segmentation. The second approach uses adaptor grammars, which are context free grammars that learn new rules from the training data [Joh08]. Since recent studies underline the feasibility of applying adaptor grammars to the word segmentation problem [JG09], we use them in Section 3.3.4 representatively for all monolingual word segmentation methods.

In our approach, we use information of a parallel corpus between word sequences in the source language and phoneme sequences in the target language similar to [BZG06]. In an oracle experiment, they replace the words in the target language with their pronunciations and remove word boundary markers. For segmenting these phoneme sequences into words, however, they run a monolingual unsupervised algorithm in contrast to us using cross-lingual word-to-phoneme alignment. After applying the monolingual segmentation to an Iraqi phoneme sequence, they use the resulting word sequences in the training process of a machine translation system that translates Iraqi phoneme segments to English words. Their results show that even with low word accuracy of their word segmentation algorithm (55.2%), vocabulary extraction efforts are applicable to machine translation.

The authors in [SW08] use the word-to-word aligner GIZA++ [ON00] to align English word sequences to Spanish phoneme sequences from the BTEC corpus [KSTY03] and extracted training data for language technology from human simultaneously spoken translations. According to our approach, they insert a word boundary maker into the phoneme sequence wherever two

neighboring phonemes are aligned to different source language words for the word segmentation.

In [SBW09] they combined the monolingual and the GIZA++-based approach by taking the most frequent word segments from the monolingual segmentation output, replacing them and then using GIZA++ for segmenting the remaining phoneme sequences.

In [SSVS12] we show that even if the found word-to-phoneme alignments in [SW08] have acceptable quality on perfect phonetic transcriptions, the word segmentation precision is not significantly higher as a monolingual approach when phoneme recognition errors are more common, only in F-score and accuracy. Therefore, we proposed a new alignment model for word-to-phoneme alignment (see Section 3.3.3) and achieve significantly higher word segmentation and alignment quality in F-score, precision, and accuracy. In [SSVS13] and [SSVS14b], we conducted first experiments with parallel data from the Christian Bible to extract pronunciations from the segmented phoneme sequences with English as target language.

We simulated phoneme recognition errors instead of using real phoneme recognizers because we intended to focus on the algorithms for the word segmentation and the pronunciation extraction. Bootstrapping a phoneme recognizer without information of the target language is not a trivial task. It may be bootstrapped using recognizers from other languages and adaptation techniques as presented in [VKS11] or in [SPC<sup>+</sup>13].

### 3.3.3 Word Segmentation

Cross-lingual word-to-phoneme alignments introduced in [BZG06, SW08, SBW09] and tackled by us with our alignment model *Model 3P* [SSVS12] are the basis for our word segmentation and pronunciation extraction algorithm. In the following sections, we describe our approach for finding word-to-phoneme alignments automatically with English taking the role as the resource-rich source language and Spanish as the “under-resourced” target language. Afterwards we demonstrate the performance of *Model 3P* on a English-Spanish portion of the BTEC corpus [KSTY03] and written translations from the Christian Bible.

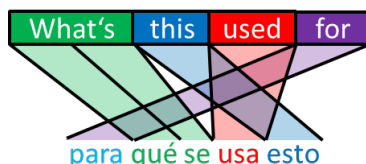


### IBM Model 3

As our approach is highly inspired by ideas originating from statistical machine translation, we briefly discuss those in this section. The central data structure in statistical machine translation is the word alignment, which identifies word pairs in parallel corpora that are translations of each other. For instance, the alignment in Figure 3.13 indicates that the Spanish word **esto** is a possible translation of the English word **this**. Various statistical models for estimating the probability of such alignments exist in literature, such as the HMM Model [VNT96], the IBM Model hierarchy 1-5 [BPPM93], and their variations [ON03, ON00]. They differ in the set of parameters, forms of restrictions or deficiency. GIZA++ [ON00] is an implementation of the IBM Models and the HMM Model widely used in statistical machine translation for automatically finding word-to-word alignments.

Our proposed alignment model (*Model 3P*) is an extension of the IBM Model 3 [BPPM93]. The parameters of the latter model are composed of a set of *fertility probabilities*  $n(\cdot|\cdot)$ ,  $p_0$ ,  $p_1$ , a set of *translation probabilities*  $t(\cdot|\cdot)$ , and a set of *distortion probabilities*  $d(\cdot|\cdot)$ . According to IBM Model 3, the following generative process produces the target language sentence  $f$  from a source language  $e$  with length  $l$  [Kni99].

1. For each source word  $e_i$  indexed by  $i = 1, 2, \dots, l$ , choose the fertility (how many target words  $e_i$  produces)  $\phi_i$  with probability  $n(\phi_i|e_i)$  (**fertility** step).
2. Choose the number  $\phi_0$  of “spurious” target words to be generated from  $e_0 = \text{NULL}$ , using probability  $p_1$  and the sum of fertilities from *step 1*.
3. Let  $m = \sum_{i=0}^l \phi_i$ .
4. For each  $i = 0, 1, 2, \dots, l$ , and each  $k = 1, 2, \dots, \phi_i$ , choose a target word  $\tau_{ik}$  with probability  $t(\tau_{ik}|e_i)$  (**lexical translation** step).



**Figure 3.13** – Word alignment between English and Spanish.

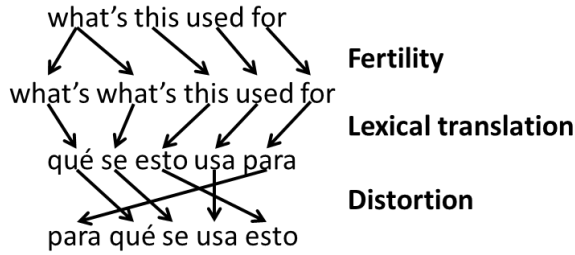


Figure 3.14 – Generative process in IBM Model 3.

5. For each  $i = 1, 2, \dots, l$ , and each  $k = 1, 2, \dots, \phi_i$ , choose target position  $\pi_{ik}$  with probability  $d(\pi_{ik}|i, l, m)$  (**distortion** step).
6. For each  $k = 1, 2, \dots, \phi_0$ , choose a position  $\pi_{0k}$  from the  $\phi_0 - k + 1$  remaining vacant positions in  $1, 2, \dots, m$ , for a total probability of  $1/\phi_0!$ .
7. Output the target sentence with words  $\tau_{ik}$  in positions  $\pi_{ik}$  ( $0 \leq i \leq l, 1 \leq k \leq \phi_i$ ).

Figure 3.14 illustrates the generation of the Spanish sentence *Para qué se usa esto* from the English sentence *What's this used for*. Equation 3.1 states the process as a general formula:

$$P(a, f|e) = \binom{m - \phi_0}{\phi_0} \cdot p_0^{m-2\phi_0} \cdot p_1^{\phi_0} \cdot \prod_{i=1}^l n(\phi_i|e_i) \cdot \prod_{j=1}^m t(f_j|e_{a_j}) \cdot \prod_{j:a_j \neq 0}^m d(j|a_j, l, m) \prod_{i=1}^l \phi_i! \quad (3.1)$$

The alignment  $a$  is represented as a vector of integers, in which  $a_i$  stores the position of the source word connected to the target word  $f_i$ . For instance,  $a = (4, 1, 1, 3, 2)$  for the alignment in Figure 3.13.

### Model 3P

Statistical alignment models for word-to-word alignment are well-studied in statistical machine translation literature. However, for the word segmentation, aligning words in the source language with phonemes in the target language is required. One method for automatically obtaining such word-to-phoneme alignments is to use word-to-word alignment models from statistical machine translation. Authors in [SW08] use a perfect phoneme tran-

scription on the target side, and run the word-to-word aligner GIZA++ to align English words to Spanish phonemes. The alignment error rate of the found alignments is comparable to similar experiments with words instead of phonemes on the target side. Our experiments in Section 3.3.4 suggest that significantly better results can be achieved by using our new alignment model *Model 3P*<sup>7</sup> for word-to-phoneme alignment, in particular with respect to word segmentation quality. *Model 3P* extends the IBM Model 3 by additional dependencies for the *translation probabilities*  $t(\cdot|\cdot)$  and a set of *word length probabilities*  $o(\cdot|\cdot)$ . The generative process upon which it is based can be described as follows:

1. For each source word  $e_i$  indexed by  $i = 1, 2, \dots, l$ , choose the fertility  $\phi_i$  with probability  $n(\phi_i|e_i)$  (**fertility** step).
2. Choose the number  $\phi_0$  of “spurious” target words to be generated from  $e_0 = \text{NULL}$ , using probability  $p_1$  and the sum of fertilities from *step 1*.
3. Let  $m = \sum_{i=0}^l \phi_i$ .
4. For each  $i = 1, 2, \dots, l$ , and each  $k = 1, 2, \dots, \phi_i$ , chose a target word position  $\pi_{ik}$  with probability  $d(\pi_{ik}|i, l, m)$  (**distortion** step).
5. For each  $k = 1, 2, \dots, \phi_0$ , choose a word position  $\pi_{0k}$  from the  $\phi_0 - k + 1$  remaining vacant positions in  $1, 2, \dots, m$ , for a total probability of  $1/\phi_0!$ .
6. For each  $i = 0, 1, \dots, l$ , and each  $k = 1, 2, \dots, \phi_i$ , choose the word length  $\psi_{ik}$  with probability  $o(\psi_{ik}|e_i)$  (**word length** step).
7. For each  $i = 0, 1, \dots, l$ , and each  $k = 1, 2, \dots, \phi_i$ , and each  $j = 1, 2, \dots, \psi_{ik}$ , choose a target phoneme  $\tau_{ikj}$  with probability  $t(\tau_{ikj}|e_i, j)$  (**lexical translation** step).
8. Output the target phoneme sequence with phonemes  $\tau_{ikj}$  in positions  $\pi_{ik}$  ( $0 \leq i \leq l, 1 \leq k \leq \phi_i, 1 \leq j \leq \psi_{ik}$ ).

Besides the fact that *Model 3P* skips *step 4* of IBM Model 3 (lexical translation), both models are identical until applying the distortion model (*step 5* or *step 6*, respectively). At this point, we can regard the target sequence in *Model 3P* as a sequence of anonymous tokens, each is a placeholder for a target word. In *step 6*, we choose the number of phonemes in the final phoneme sequence according to the *word length probabilities*  $o(\cdot|\cdot)$ . The next step fills in the phonemes itself, depending on the source word  $e_i$  and their phoneme position  $j$  in the target word. Figure 3.15 illustrates an instance of the generative process of *Model 3P*.

<sup>7</sup>A multi-threaded implementation is available at <http://pisa.googlecode.com/>

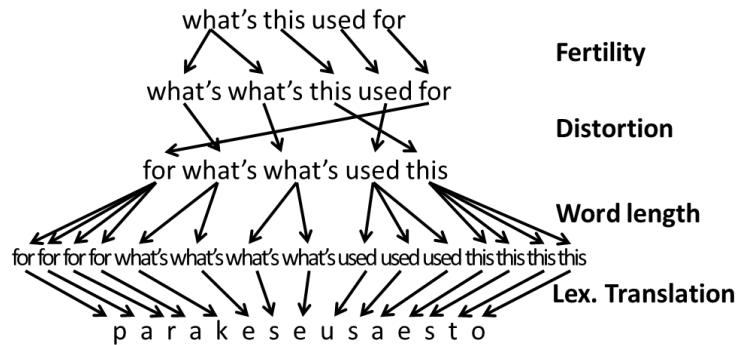


Figure 3.15 – Generative process in Model 3P.

In *Model 3P*, an alignment  $A \in \{\mathbb{N} \cup \{-\}\}^{4 \times m}$  is a matrix rather than an integer vector like in IBM Model 3. It captures additional model decisions made in the fertility and word length step, which would be hidden in an integer vector representation:

- $A_{0j}$ : English word position connected to the  $j$ -th target phoneme
- $A_{1j}$ : Position of the target word belonging to the  $j$ -th target phoneme
- $A_{2j}$ : Word length in phonemes of the target word  $A_{1j}$
- $A_{3j}$ : Phoneme position of the  $j$ -th target phoneme in the corresponding target word

English word position ( $i$ )	4	4	4	4	1	1	1	1	3	3	3	2	2	2	2
Target word position ( $\pi_{ik}$ )	1	-	-	-	2	-	3	-	4	-	-	5	-	-	-
Target word length ( $\psi_{ik}$ )	4	-	-	-	2	-	2	-	3	-	-	4	-	-	-
Phoneme position in target word ( $j$ )	1	2	3	4	1	2	1	2	1	2	3	1	2	3	4

Figure 3.16 – English-Spanish alignment in Model 3P.

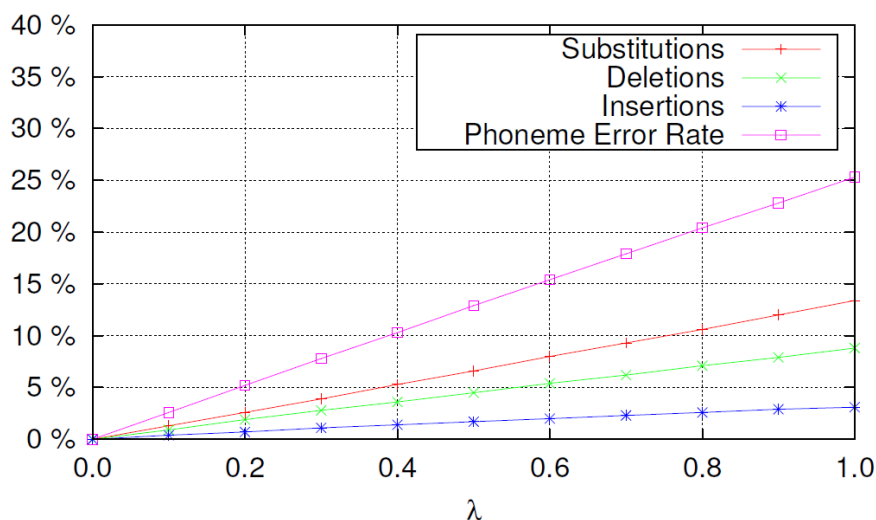
An example of an alignment is shown in Figure 3.16. The word boundary between the 6th and 7th phoneme could not be reconstructible in a simple integer vector representation. Equation 3.2 expresses *Model 3P* as a general formula:

$$\begin{aligned}
 P(A, f|e) = & \binom{k - \phi_0}{\phi_0} \cdot p_0^{k-2\phi_0} \cdot p_1^{\phi_0} \\
 & \cdot \prod_{i=1}^l (n(\phi_i|e_i) \cdot \phi_i!) \cdot \prod_{j=1}^m t(f_j|e_{A_{0j}}, A_{3j}) \\
 & \cdot \prod_{j: A_{0j} \neq 0, A_{1j} \neq -}^m (d(A_{1j}|A_{0j}, l, k) \cdot o(A_{2j}|e_{A_{0j}}))
 \end{aligned} \tag{3.2}$$

### 3.3.4 Alignment and Word Segmentation Performance

We have shown in [SSVS12, SSVS14b] that unsupervised learning of word segmentation is more accurate when information of another language is used – for example by using *Model 3P* word-to-phoneme alignments. Given such an alignment, we insert a word boundary marker into the phoneme sequence wherever two neighbouring phonemes are aligned to different source language words (i.e. wherever a black alignment line in Figure 2.2 ends).

We intersperse the perfect phoneme sequences with phoneme errors to investigate the performance of different word segmentation approaches depending on the underlying phoneme error rate. In order to imitate recognition errors realistically, we trained a phoneme recognizer on the Spanish *GlobalPhone* corpus [SVS13] and used the NIST sclite scoring and evaluation tool [Fis07] to create its confusion matrix  $R \in \mathbb{R}^{36 \times 36}$ .  $R_{so}$  contains the probability  $P_R(o|s)$  that the phoneme recognizer confuses the stimulus phoneme  $s$  with the observed phoneme  $o$  (substitution). An additional row and a column model insertions and deletions, so that all elements in a row sum up to 1 and induce a probability distribution. The Spanish phoneme recognizer has a phoneme error rate of 25.3%. We smooth  $R$  with  $\lambda \in [0, 1]$  to control the phoneme error rate. We obtain a disturbed phoneme sequence by replacing each phoneme  $s$  in the perfect phoneme transcription with a phoneme  $o$  with the probability  $P_\lambda(o|s)$ . Figure 3.17 shows that the resulting phoneme error rate is linear in  $\lambda$ .



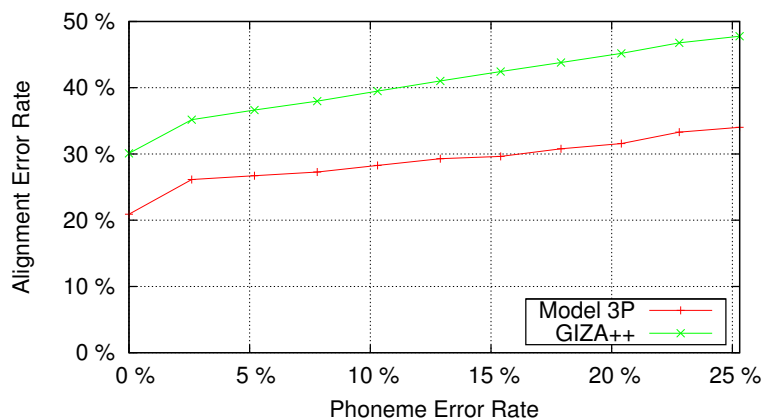
**Figure 3.17** – Phoneme error rate over  $\lambda$  for the Spanish phoneme recognizer.

## Settings

We found that setting the following GIZA++ parameters enhances the alignment quality:

- `maxfertility = 12` (default=10). This adjustment addresses the large discrepancy between source and target sentence length in the word-to-phoneme alignment scenario. However, larger values did not improve alignment quality.
- `deficientdistortionmodelforemptyword = 1` (default=0). As reported in [ON03], the distortion model for NULL in IBM 3 and IBM 4 is not deficient. Since IBM 3 and IBM 4 are deficient, the EM training can maximize probabilities by aligning more and more words to NULL. Using a deficient distortion model for NULL turned out to increase the alignment performance significantly.
- `emprobforemptyword = 0.1` (default=0.4). This also slows down the process of aligning more and more words to NULL.

Our PISA Alignment Tool implements our word-to-phoneme alignment model *Model 3P*. PISA requires the initial alignments generated by GIZA++. As described in [SSVS12], GIZA++ first calculates initial alignments with our optimal settings which are then further refined by the PISA Alignment Tool applying *Model 3P*. More information and practical advices for using the PISA Alignment Tool are given in [Sta14].



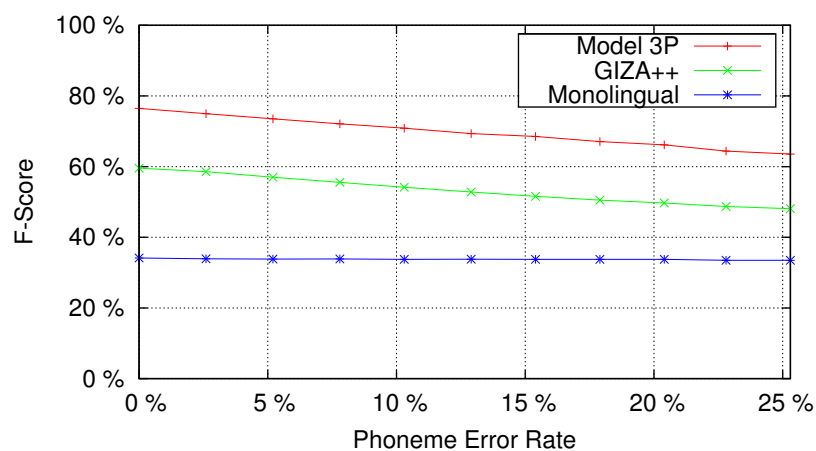
**Figure 3.18** – Alignment error rate over phoneme error rate on BTEC (source language: English, target language: Spanish).

### Performance

Figure 3.18 shows the alignment performance of both, GIZA++ and *Model 3P* over the phoneme error rate for the alignment between English words and Spanish phoneme sequences on the BTEC corpus. The reference alignments were generated by running GIZA++ with default parameters at the word level, and then replacing the Spanish words with their pronunciations afterwards. The alignment error rate [ON03] of *Model 3P* alignments is up to 13.6% lower than for GIZA++’s word-to-phoneme alignments. The alignment error rate for both systems increase proportionally with the phoneme error rate, GIZA++ slightly more rapidly than *Model 3P*.

The quality of the found word segmentations for both cross-lingual approaches and a monolingual approach [Joh08] is summarized in Figure 3.19. Using *Model 3P* for the alignment between English words and correct Spanish phoneme sequences resulted in 90.0% segmentation accuracy [VIM04] (76.5% in F-score) and thus outperformed a state-of-the-art monolingual word segmentation approach by 24.72% relative in accuracy (124.11% in F-score). Furthermore, we still report 83.9% segmentation accuracy on a phoneme sequence containing 25.3% errors produced by our simulated phoneme recognizer (see Table 3.9 and 3.10).

In a second experiment, we switched English and Spanish such that Spanish took the role as resource-rich source language that was aligned to the “under-resourced” target language English to check if *Model 3P* also outperforms GIZA++ in another language combination and with higher phoneme



**Figure 3.19** – Word segmentation quality over phoneme error rate on BTEC (source language: English, target language: Spanish).

		Accuracy (in %)			Relative Model 3P Improvement (in %)	
		Monoling.	GIZA++	Model 3P	Monoling.	GIZA++
en-es	0% PER	72.21	80.45	90.06	24.72	11.95
	25.3% PER	72.16	70.30	83.92	16.30	19.37
es-en	0% PER	69.18	86.36	88.60	28.07	2.59
	45.5% PER	68.18	68.48	76.18	11.73	11.24

**Table 3.9** – Accuracy improvement using *Model 3P* on the BTEC corpus.

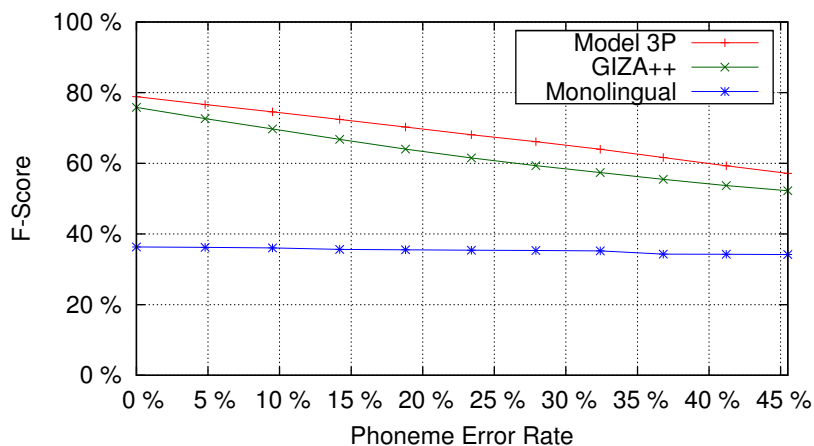
		F-score (in %)			Relative Model 3P Improvement (in %)	
		Monoling.	GIZA++	Model 3P	Monoling.	GIZA++
en-es	0% PER	34.13	59.57	76.49	124.11	28.40
	25.3% PER	33.47	48.11	63.57	89.93	32.13
es-en	0% PER	36.33	75.83	78.85	117.04	3.98
	45.5% PER	34.18	52.26	57.17	67.26	9.40

**Table 3.10** – F-score improvement using *Model 3P* on the BTEC corpus.

error rates error rates [SSVS14b]. Phoneme errors were simulated as before, with a smoothed confusion matrix of an English phoneme recognizer with 45.5% phoneme error rate (39 phonemes). Figure 3.20 shows that the results are similar to the first experiment. However, the achieved gain by using *Model 3P* instead of GIZA++ is smaller than before. Using *Model 3P* for the alignment between Spanish words and correct English phoneme sequences



resulted in 88.6% segmentation accuracy (78.9% in F-score) and thus performed better than the monolingual word segmentation approach by 28.1% relative in accuracy (117.0% in F-score). We report still 68.5% segmentation accuracy on a phoneme sequence containing 45.5% errors produced by our simulated phoneme recognizer.



**Figure 3.20** – Word segmentation quality over phoneme error rate on BTEC (source language: Spanish, target language: English).

In addition to the experiments on the BTEC corpus, we applied both cross-lingual word segmentation approaches GIZA++ and *Model 3P* to the Christian Bible corpus<sup>8</sup>, which we use in our pronunciation extraction experiments. English (*en*), Spanish (*es*), and Portuguese (*pt*) served as target languages, whereas the remaining 14 translations represented the source languages [SSVS13, SSVS14b]. As demonstrated with the BTEC corpus, *Model 3P* substantially improved the GIZA++ alignments for all investigated language pairs. An overview of the used Bible translations is given in Table 8.1.

Table 3.11 gives an overview of the *Model 3P* improvements. On the Bible corpus we report for each of the target languages both the biggest and smallest relative gain over all available source languages. For example, selecting Portuguese (*pt*) as target language we achieve between 20.07% with a Spanish translation (*es3*) and 76.95% relative gain with a Swedish translation (*sw*). We observe that the lower the GIZA++ segmentation accuracy, the larger the relative gain achieved with *Model 3P*.

<sup>8</sup>Extracted from <http://www.biblegateway.com/> (Accessed on 15th November 2013)

		Accuracy (in %)		Improvement (in %)	
		GIZA++	Model 3P	Absolute	Relative
BTEC en-es	0% PER	80.45	90.06	9.61	<b>11.95</b>
	25.3% PER	70.30	83.92	13.62	<b>19.37</b>
BTEC es-en	0% PER	86.36	88.60	2.24	<b>2.59</b>
	45.5% PER	68.48	76.18	7.70	<b>11.24</b>
Bible *_en	Biggest gain (se-en)	47.74	74.72	26.98	<b>56.52</b>
	Smallest gain (es3-en)	73.01	84.52	11.51	<b>15.76</b>
Bible *_es	Biggest gain (se-es)	33.74	71.51	37.77	<b>111.92</b>
	Smallest gain (pt2-es)	53.61	81.18	27.57	<b>51.41</b>
Bible *_pt	Biggest gain (se-pt)	41.01	72.57	31.56	<b>76.95</b>
	Smallest gain (es3-pt)	67.65	81.22	13.57	<b>20.07</b>

**Table 3.11** – Accuracy improvement using *Model 3P* instead of GIZA++.

### 3.3.5 Summary

The word segmentation problem describes the task of segmenting phoneme sequences into word units. We have investigated three different unsupervised algorithms for automatically finding word boundaries in phonetic transcriptions. We showed that using information from another language rather than a pure monolingual approach helps to find better segmentations on perfect phoneme sequences. A simple way to incorporate cross-lingual information is to apply word-to-word alignment models from SMT to align words of the other language to the phonemes of the target language. However, when phoneme recognition errors are common, the word segmentation precision is not significantly higher than with the monolingual approach. Therefore we proposed the new alignment model *Model 3P* for cross-lingual word-to-phoneme alignment, which extends the generative process of IBM Model 3 by a word length step and additional dependencies for the lexical translation probabilities. With this new model, we obtain considerably better word segmentations than with both previous methods.

# Pronunciation Quality Assurance

---

Due to the fact that high-quality pronunciation dictionaries are so important to speech processing systems, much care has to be taken to produce a dictionary as free of errors as possible [SOS12a]. For automatic speech recognition systems, error-prone pronunciations in the training dictionary may decrease the quality of the acoustic models and result in higher word error rates. Therefore, we developed methods which aim to assure the quality of pronunciations.

As mentioned in Section 1.4, data-driven methods are applied commonly if reliable word-pronunciation pairs are available and a relationship between graphemes and phonemes is given. However, cross-checks of the pronunciations generated in a data-driven way are often not performed [LCD<sup>+</sup>11] since manual checks are time-consuming and expensive, especially if native speakers or linguists need to be hired for this task. Particularly, if not enough examples in terms of word-pronunciation pairs are given to train the grapheme-to-phoneme converters, the quality of the produced pronunciations is suboptimal. However, the manual production of examples is expensive. In Section 4.1 we demonstrate lower bounds for the grapheme-to-phoneme converter training data to allow a rapid pronunciation production with a minimum of effort.

Due to different subjective judgments, small typographical errors, and 'convention drift' by multiple annotators, the manual production of pronunciations can result in erroneous or inadequate dictionary entries. Browsing

through all entries without any automatic support to find inappropriate entries is not possible with little manual effort. Consequently, we propose completely automatic methods to detect, remove, and substitute inconsistent or flawed entries in Section 4.2.

In Section 4.3 we show that grapheme-to-phoneme converters trained with the same example word-pronunciation pairs are reasonably close in performance but at the same time produce an output that differs in its errors. The outputs provide complementary information which can lead in combination to grapheme-to-phoneme converter performance improvements and thus to better pronunciations.

## 4.1 Grapheme-to-Phoneme Model Quality

In the commonly used data-driven methods statistical grapheme-to-phoneme models are trained, which are used to generate missing pronunciations or to produce pronunciation variants. To achieve optimal pronunciation quality, we need to ensure high-quality grapheme-to-phoneme models. For our quality analyses, we built grapheme-to-phoneme models for European languages from ten *GlobalPhone*-based dictionaries (Bulgarian, Croatian, Czech, English<sup>1</sup> [cmu], French, German, Spanish, Polish, and Russian). First, we check the grapheme-to-phoneme accuracy as an indicator of dictionary consistency similar to [WEH02] and [DB06]. For this purpose, we built grapheme-to-phoneme models with increasing amounts of word-pronunciation pairs from *GlobalPhone* as training material. We applied them to test sets from the respective source and computed the phoneme error rate to the original canonical pronunciations. We report the phoneme error rate since it gives information about the required edits to achieve qualified *GlobalPhone* quality. The higher the phoneme error rate, the higher the necessary post-editing effort.

Then we selected grapheme-to-phoneme models which had all been trained with a comparable number of training material. With these, we investigate their relations among grapheme-to-phoneme consistency, complexity and their usage for automatic speech recognition. For the automatic speech recognition experiments, we replaced the pronunciations in the dictionaries of six *GlobalPhone* automatic speech recognition systems (Czech, English, French, Spanish, Polish, and German) and investigated the change in performance by using exclusively pronunciations generated from *GlobalPhone*

---

<sup>1</sup>For English, we used the English CMU dictionary (<http://www.speech.cs.cmu.edu/cgi-bin/cmudict>).

grapheme-to-phoneme models for training and decoding. Further analyses including relations between grapheme-to-phoneme consistency, model complexity, model confidence, grapheme n-gram coverage, and phoneme perplexity are published in [SOS12b], [SOS14] and [SS14].

### 4.1.1 Consistency and Complexity

For training and applying grapheme-to-phoneme models, we use Sequitur G2P [BN08] which was introduced in Section 1.4.4. To quantify the *complexity* of a grapheme-to-phoneme model, we use the model size, which corresponds to the amount of non-pruned  $M$ -grams.

	bg	cs	de	en	es	fr	hr	pl	pt	ru
#phones / pron.	8.6	7.8	9.0	6.6	8.4	7.0	8.5	8.2	7.4	9.0

**Table 4.1** – Average number of phones in a pronunciation.

To verify the pronunciation quality with the *consistency* similar to [WEH02] and [DB06], we performed a 6-fold cross validation as follows: For each *GlobalPhone*-based dictionary, we randomly selected 30% of the total number of word-pronunciation pairs for testing. From the remainder, we extracted increasing amounts of entries based on their accumulated phoneme count and used them for training the grapheme-to-phoneme models in each fold. Figure 4.1 demonstrates differences in grapheme-to-phoneme consistency among the languages Bulgarian (*bg*), Croatian (*hr*), Czech (*cs*), English (*en*), French (*fr*), German (*de*), Spanish (*es*), Polish (*pl*), Portuguese (*pt*) and Russian (*ru*). To compare the languages, we expressed the amount of grapheme-to-phoneme converter training data on the x-axis in terms of the total number of phones in the dictionary entries. Table 4.1 illustrates the average number of phones per word entry in the *GlobalPhone* pronunciations and in the English CMU dictionary [cmu]. We observe a strong phoneme error rate decrease for amounts of training data between 100 and 7k phonemes [SOS12b]. For more than 7k phonemes, the phoneme error rates decrease less with more training data. Creating pronunciations for languages with a close grapheme-to-phoneme relationship like Bulgarian, Czech, Polish, and Spanish, 5k-10k phonemes with corresponding graphemes are sufficient for training a well-performing converter. For example, the phoneme error rate on Polish is lower than 4% and drops to 3.2% with 30k phonemes for training [SS14]. However, as the relationship gets weaker (Portuguese, French, German), significantly more training examples are required to achieve similar performances.

For example, the German grapheme-to-phoneme converter requires six times more training material (30k phones) than the Portuguese one (5k phones) to achieve the same quality (10% phoneme error rate) on the generated pronunciations. We learn that for the 10 languages word-pronunciation pairs containing 15k phonemes were sufficient to have stable quality, as the curves start to saturate at 15k phonemes for all 10 languages. However, we conclude that grapheme-to-phoneme conversion for languages like Portuguese, French, German, and in particular English, will not reach the performance level of those with close grapheme-to-phoneme relationship [SOS12b, SOS14, SS14].

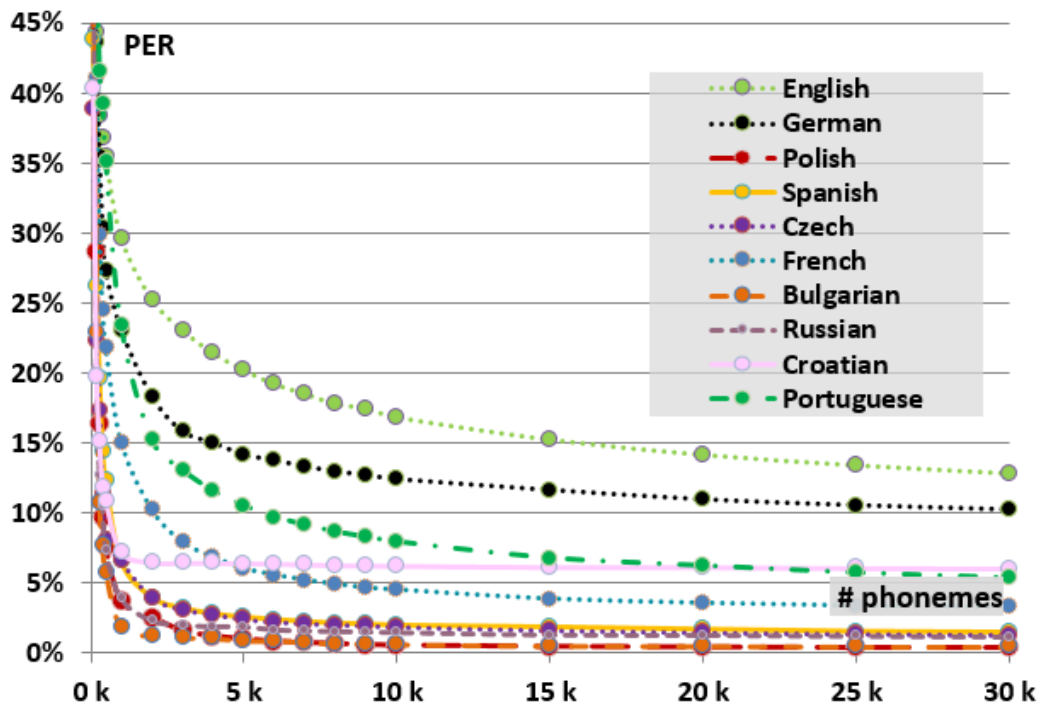


Figure 4.1 – Grapheme-to-phoneme consistency across 10 languages.

We investigated the complexity of the grapheme-to-phoneme models over training data and among languages and compared the complexity change to the consistency change. Figure 4.2 shows the increase in complexity of the grapheme-to-phoneme models with the increase of training material between 100 and 30k phonemes with corresponding graphemes. A comparison of Figure 4.1 with 4.2 indicates that although the consistency saturates at 15k phonemes, the model complexity keeps increasing for larger amounts of training data. However, this has minor impact on quality in terms of consistency as the model increases with the new  $M$ -grams which, however, represent seldom rules and rather exceptions after 15k phonemes.

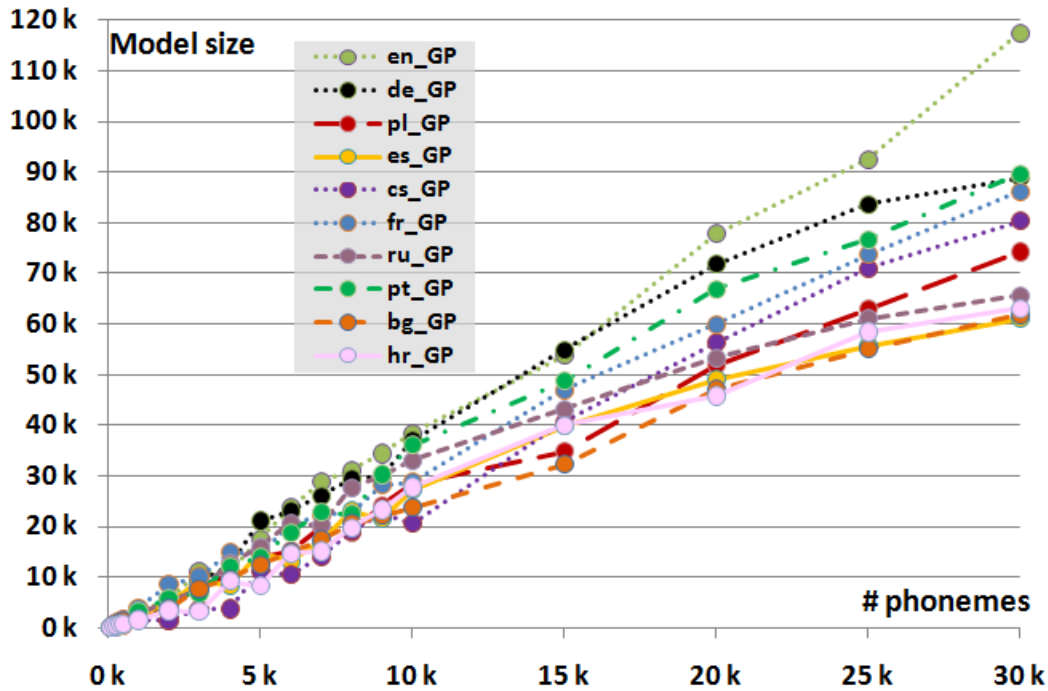


Figure 4.2 – *GlobalPhone* grapheme-to-phoneme model complexity.

For the automatic speech recognition performance checks, we selected grapheme-to-phoneme models which were trained with 30k phonemes and their corresponding graphemes since this number reflects a saturated grapheme-to-phoneme model consistency, 30k phonemes are contained in all *GlobalPhone*-based dictionaries and in many *Wiktionary* editions which are a good source for pronunciations on the Web, as we show in Section 5.1.

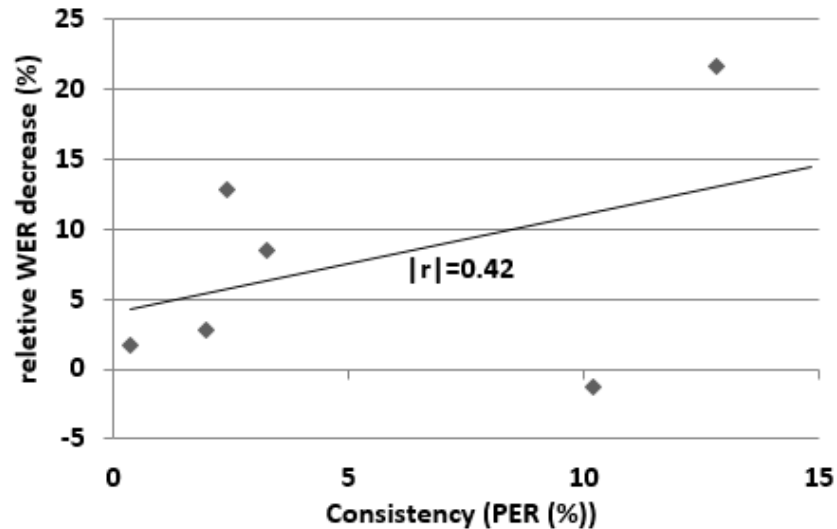
#### 4.1.2 Automatic Speech Recognition Performance

Finally, we analyzed if we can use the pronunciations generated with our *GlobalPhone* grapheme-to-phoneme models in automatic speech recognition [SOS12b]. Furthermore, we were interested if our information about the pronunciation quality correlates with their impact on automatic speech recognition performance. We replaced the pronunciations in the dictionaries of six *GlobalPhone*-based automatic speech recognition systems with pronunciations generated with *GlobalPhone* grapheme-to-phoneme models which were trained with 30k phonemes and the corresponding graphemes. Then, we trained and decoded the systems completely with those pronunciation dictionaries. We built and decoded automatic speech recognition systems with

dictionaries where only the most likely (1-best) pronunciation for each *GlobalPhone* word was produced with our grapheme-to-phoneme models. We compared these to *GlobalPhone* systems, which were also limited to the first pronunciation (base form). The results of the automatic speech recognition experiments together with the consistency results of the used grapheme-to-phoneme models are listed in Table 4.2.

	GlobalPhone (base form)	GlobalPhone G2P (1-best)	relative WER change (%)	GlobalPhone ( <i>GP</i> ) Consistency (PER) at 30k
cs	15.59	17.58	12.76	2.41
de	16.71	16.50	-1.26	10.21
en	14.92	18.15	21.65	12.83
es	12.25	12.59	2.78	1.99
fr	20.91	22.68	8.46	3.28
pl	15.51	15.78	1.74	0.36

**Table 4.2** – Word error rates (%) of systems with dictionaries built completely with grapheme-to-phoneme generated pronunciations.



**Figure 4.3** – Correlation between consistency and word error rate decrease.

Using exclusively pronunciations generated from grapheme-to-phoneme models for automatic speech recognition training and decoding resulted in reasonable performance degradations given the cost and time efficient generation process. Figure 4.3 indicates that the severeness of degradation slightly correlates with the grapheme-to-phoneme consistency in terms of phoneme error rate with a Pearson’s correlation coefficients  $|r|$  [RN88] of 0.42.



### 4.1.3 Summary

We have investigated the grapheme-to-phoneme model generation for European languages with pronunciations from 10 *GlobalPhone* dictionaries. We analyzed and compared their quality with regard to consistency and complexity and detected a saturation at 15k phonemes with corresponding graphemes as training material. Using exclusively pronunciations generated from grapheme-to-phoneme models for automatic speech recognition training and decoding resulted in reasonable performance degradations given the cost and time efficient generation process. The severeness of degradation correlates with the grapheme-to-phoneme consistency.

## 4.2 Detection and Recovery of Inconsistencies and Errors

The manual production of pronunciations by multiple annotators can result in erroneous or inadequate dictionary entries. Therefore, cross-checks and post-editing are essential. However, browsing through all entries without any automatic support to find inappropriate entries is not possible with little manual effort.

### 4.2.1 Detection of Inconsistencies and Errors

Different approaches to automatically detect flawed entries have been described in the past [SOS12a]. [VALR03] apply a stochastic grapheme-to-phoneme model to the task of dictionary verification and detect spurious entries, which can then be examined and corrected manually. [MD07] focus on mechanisms to identify incorrect entries which require limited human intervention. The techniques for verifying the correctness of a dictionary include word-pronunciation length relationships, grapheme-to-phoneme alignment, grapheme-to-phoneme rule extraction, variant modeling, duplicate pronunciations, and variant analysis. The automated correction of these entries is not investigated and erroneous entries are simply removed. [DM09] propose a semi-automated development and verification process. The extraction of grapheme-to-phoneme rules provides an immediate avenue for error detection: By cross-validating the dictionary, errors made by the grapheme-to-phoneme predictor can be flagged for verification [DdW10]. Grapheme-to-phoneme rules themselves may also be able to identify highly

irregular training instances [MD07] or provide an indication of the likelihood of a specific pronunciation [BN08, VALR03] in order to flag possible errors. In [WEH02] and [DB06], grapheme-to-phoneme accuracy is considered an indicator of dictionary consistency, especially where variants are concerned. Inconsistencies lead to unnecessarily complex pronunciation models, and consequently, suboptimal generalization. [DB06] generate pronunciations with rules and flag pronunciations with alternative generated pronunciations. [DdW10] describe a technique that does not only flag specific words for verification, but also presents verifiers with example words that produce pronunciation patterns conflicting with the flagged instances.

The previous approaches show isolated methods to detect inconsistent dictionary entries. To correct those entries, they substitute them manually in separate processes. The annotation of flagged entries may be still costly and time-consuming. Therefore, we investigated the performance of different fully automatic data-driven methods to detect, remove and substitute such entries. We determine the thresholds for removing word-pronunciation pairs completely on the data in the dictionaries and restore removed pronunciations with pronunciations generated with validated grapheme-to-phoneme models. For better filtering, we experimented with single and 2-stage approaches.

Most approaches reported in related work have not been evaluated in automatic speech recognition experiments. We investigate the performance of our methods on different tasks and check their impact on automatic speech recognition: First, we analyze their impact on the *GlobalPhone* Hausa pronunciation dictionary, which had been manually cross-checked but still contains a few errors [Djo11]. Then, we use our methods to select pronunciations from an additional dictionary to enhance the *SEAME* code-switch dictionary which contains entries of Mandarin and English with Singaporean and Malayan accent to transcribe Mandarin-English code-switching conversational speech [VLW<sup>+</sup>12].

Additionally, we apply our methods to pronunciations from the World Wide Web. We present the results in Section 5.1, where we describe how we retrieve Web-derived pronunciations and analyze their quality. Since they often lack information about the corresponding word or language, it may happen that inappropriate word-pronunciation pairs are collected. Further results are demonstrated in Section 5.3, where we show how to use the Web-derived pronunciations in a rapid and economic semi-automatic pronunciation dictionary development strategy.

### 4.2.2 Recovery of Inconsistencies and Errors

Our investigated methods, which filter erroneous word-pronunciation pairs and substitute the filtered pronunciations with more reliable ones, fall into the following categories [SOS12a]:

1. Length Filtering (*Len*)
  - (a) Remove a pronunciation if the ratio of grapheme and phoneme tokens exceeds a certain threshold.
2. Epsilon Filtering (*Eps*)
  - (a) Perform a 1-1 grapheme-to-phoneme alignment [MD07, BLP98] which involves the insertion of graphemic and phonemic nulls (epsilons) into the lexical entries of words.
  - (b) Remove a pronunciation if the proportion of graphemic and phonemic nulls exceeds a threshold.
3. m-n Alignment Filtering (*M2NAlign*)
  - (a) Perform an M-N grapheme-to-phoneme alignment [MD07, BLP98].
  - (b) Remove a pronunciation if the alignment score exceeds a threshold.
4. Grapheme-to-phoneme Filtering (*G2P*)
  - (a) Train grapheme-to-phoneme models with “reliable” word-pronunciation pairs.
  - (b) Apply the grapheme-to-phoneme models to convert a grapheme string into a most likely phoneme string.
  - (c) Remove a pronunciation if the edit distance between the synthesized phoneme string and the pronunciation in question exceeds a threshold.

The threshold for each filter method depends on the mean ( $\mu$ ) and the standard deviation ( $\sigma$ ) of the measure in focus (computed on all word-pronunciation pairs), i.e. the ratio between the numbers of grapheme and phoneme tokens in *Len*, the ratio between the numbers of graphemic and phonemic nulls in *Eps*, the alignment scores in *M2NAlign*, and the edit distance between the synthesized phoneme string and the pronunciation in question in *G2P*. Those word-pronunciation pairs whose resulting number is shorter than  $\mu - \sigma$  or longer than  $\mu + \sigma$  are rejected. Experiments with more or less restrictive thresholds performed worse.

To provide “reliable” pronunciations for *G2P*, we propose to prefilter the word-pronunciation pairs by applying *Len*, *Eps* or *M2NAlign*, as shown in

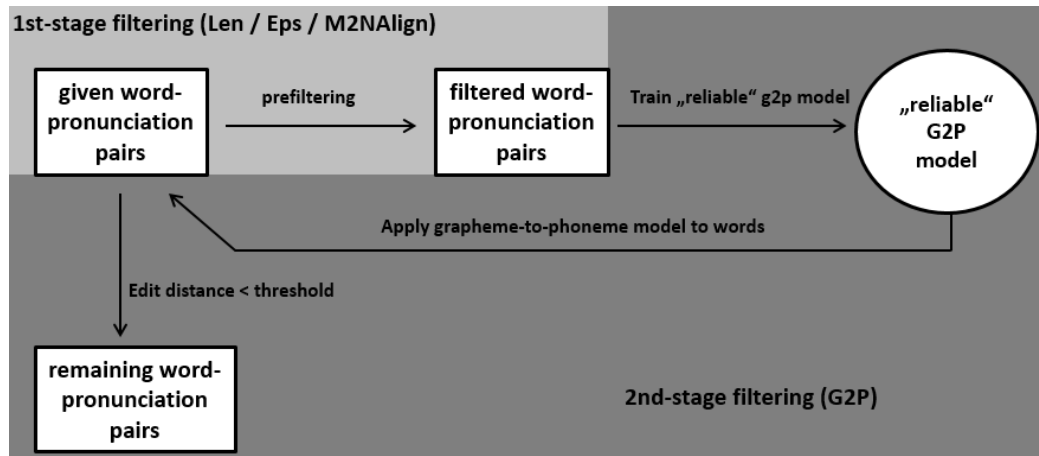


Figure 4.4 – 2-stage filtering.

Figure 4.4 (*1st stage*). On the remaining word-pronunciation pairs, *G2P* is applied (*2nd stage*). Those *2-stage filter* methods are represented as  $G2P_{Len}$ ,  $G2P_{Eps}$ , and  $G2P_{M2NAlign}$  in the following sections. Experiments with two stages in the prefiltering before *G2P* (*3-stage filtering*) were too restrictive and did not leave enough training data for reliable grapheme-to-phoneme models. If a validated dictionary already exists,  $\mu$  and  $\sigma$  can be computed on its entries. All our single and 2-stage data-driven methods expect more good than bad pronunciations in the data to obtain good estimates for  $\mu$  and  $\sigma$ . With our pure statistical methods, no manual labor and linguistic knowledge is required.

In this section we demonstrate the performance of our proposed methods by addressing the following two possible tasks occurring in the development of automatic speech recognition systems:

1. An LVCSR dictionary, which has been manually checked under supervision, can still have a few errors and inconsistencies. We apply our methods on the *GlobalPhone* Hausa dictionary, which represents such a dictionary.
2. The straightforward insertion of new pronunciation variants into an existing dictionary can lead to automatic speech recognition performance degradations if the new pronunciations do not match the target domain or accent. We filter English pronunciation variants from a new dictionary which do not match the existing Singaporean/Malayan English pronunciations in our English-Mandarin code-switch dictionary.

For all experiments with *M2NAlign*, we report the error rates of the alignment which performed best on the particular dictionary.

### Correction Of Handcrafted Pronunciations

For the African language Hausa, we collected almost 9 hours of speech from 102 Hausa speakers reading newspaper articles as a part of our *GlobalPhone* corpus [SDV<sup>+</sup>12].

We evaluate our filter methods on the initial *GlobalPhone* dictionary, which has been created in a rule-based fashion and was then manually revised and cross-checked by native speakers and causes a word error rate of 23.49% (*baseline*) [SDV<sup>+</sup>12]. After filtering, we used the remaining word-pronunciation pairs to build new grapheme-to-phoneme models and applied them to the words with rejected pronunciations. Then we trained and decoded the Hausa system with each processed dictionary. Table 4.3 shows that we reduce the word error rate with all filtered dictionaries but *G2P* by 1.5% relative on average. *G2P<sub>Len</sub>* performs best with 2.6% relative improvement.

	baseline	<i>G2P</i>	<i>Len</i>	<i>G2P<sub>Len</sub></i>	<i>Eps</i>	<i>G2P<sub>Eps</sub></i>	<i>M2NAlign</i>	<i>G2P<sub>M2NAlign</sub></i>
ha	23.49	23.68	23.20	<b>22.88</b>	23.30	23.15	23.17	23.11

**Table 4.3** – Word error rates (%) for Hausa with and without filtered pronunciation dictionary.

### Filtering New Pronunciation Variants

*SEAME* [VLW<sup>+</sup>12, AKT<sup>+</sup>14] contains 157 speakers and approximately 52k intra-sentential English-Mandarin code-switching utterances. The recorded speakers speak Singaporean/Malayan English, which differs strongly from American English. In addition to our previous pronunciation dictionary (*prev*), our partners generated a new dictionary for Singaporean English (*new*) by applying 160 rules to the pronunciations in the American CMU dictionary, which they had derived in a data-driven way. With this dictionary a word error rate of 16.89% was achieved on texts from the English Aurora 4 [PPPH04, AYS04] corpus which were read by Singaporean speakers. With the American CMU dictionary the word error rate was 75.19% on the same test set.

As an evaluation measure, we use the mixed error rate [VLW<sup>+</sup>12]. It is a combination of word error rates for English segments and character error rates for Mandarin segments. Due to this, the performance can be compared across

different segmentations of Mandarin. Our system with the existing dictionary (*prev*) has a mixed error rate (MER) of 36.89% on *SEAME* [VLW<sup>+</sup>12]. To improve *prev*, our goal was to enrich it with pronunciations from *new*. However, adding all almost 5k English pronunciations as pronunciation variants which are not equal to the pronunciations in *prev* for decoding led to a performance degradation of 0.23% absolute (*prev+new*). Therefore, we applied our filter methods to select only those pronunciations from *new* which fit the pronunciations which have been successfully used before. The mean ( $\mu$ ) and the standard deviation ( $\sigma$ ) of the measure in focus were computed based on the word-pronunciation pairs of *prev* for *Len*, *Eps*, *G2P*, and *M2NAlign1*. The alignments of *M2NAlign2* were computed on those from *new*. Table 4.4 shows that we slightly reduce the mixed error rate compared to *prev* by 0.2% relative on average with a decoding using the filtered new pronunciations. *M2NAlign1* and *M2NAlign2* marginally outperform the other methods and result in a mixed error rate reduction of 0.3% relative. Our 2-stage filtering approaches were too restrictive and removed almost all pronunciations from *new*.

	prev	prev+new	<i>Len</i>	<i>Eps</i>	<i>G2P</i>	<i>M2NAlign1</i>	<i>M2NAlign2</i>
MERs	36.89	37.12	36.89	36.89	36.84	<b>36.79</b>	<b>36.79</b>
prons./word	1.78	1.94	1.85	1.84	1.88	1.89	1.90

**Table 4.4** – Mixed error rates (%) on the *SEAME* Mandarin-English Code-Switch Corpus development set.

### 4.2.3 Summary

We have presented completely automatic error recovery methods for pronunciation dictionaries. The methods are based on the means and deviations of certain characteristics computed on the word-pronunciation pairs of the dictionaries and on grapheme-to-phoneme model generation plus their application. Our methods improved the automatic speech recognition performances in each language and task slightly but not significantly.

More results with our filter methods are demonstrated in Section 5.1 and 5.3, where we describe how we retrieve Web-derived pronunciations, analyze their quality and investigate how to use them to reduce the manual effort in our rapid and economic semi-automatic pronunciation dictionary development strategy.

All experiments show that no particular filter methods outperforms the others. Future work may include an analysis which method works how good

on which kind of errors to find faster the best method for a dictionary in question.

### 4.3 Phoneme-Level Combination

We investigated if a combination of grapheme-to-phoneme converter outputs outperforms the single converters [Qua13, SQS14]. This is particularly important for the rapid bootstrapping of speech processing systems if not many manual created example word-pronunciation pairs are available and therefore a single grapheme-to-phoneme converter has a poor performance. In the case of semi-automatic pronunciation generation, enhanced pronunciations derived from the combination would reduce the editing effort and speed up the annotation process. We combine the grapheme-to-phoneme converter outputs based on a voting scheme at the phoneme-level. Our motivation is that the converters are reasonably close in performance but at the same time produce an output that differs in its errors. This provides complementary information which in combination leads to performance improvements.

We analyze five common grapheme-to-phoneme conversion approaches and their combination:

- SMT-based with Moses Package [KHB<sup>+</sup>07, ON03] (*Moses*)
- Grapheme-based with Sequitur G2P [BN08] (*Sequitur*)
- WFST-driven with Phonetisaurus [Nov11, NMH12] (*Phonetisaurus*)
- CART-based with t2p: Text-to-Phoneme Converter Builder [Len97, BLP98] (*Carttree*)
- Simple grapheme-to-phoneme conversion based only on the most frequently uttered phoneme for each grapheme<sup>2</sup> (*Rules*).

To investigate our methods for languages with different grade of regularity in grapheme-to-phoneme relationship, our experiments are conducted with German (*de*), English (*en*), Spanish (*es*), and French (*fr*).

For evaluating our grapheme-to-phoneme conversion methods, we use *GlobalPhone* dictionaries for *de* and *es* as reference data [SS14]. For *fr*, we employ our dictionary developed within the Quaero Programme. The *en* dictionary is based on the CMU dictionary.

---

<sup>2</sup>It represents a knowledge-based approach.

For each language, we randomly selected 10k word-pronunciation pairs from the dictionary for testing. From the remainder, we extracted 200, 500, 1k, and 5k word-pronunciation pairs for training to investigate how well our methods perform on small amounts of data. To evaluate the quality of the grapheme-to-phoneme converter outputs, we apply them to the words in the test set and compute their phoneme error rate (PER) to the original pronunciations.

As we find *en*, *fr*, *de*, and *es* word-pronunciation pairs in *Wiktionary*<sup>3</sup> (see also Section 5.1), we additionally built a grapheme-to-phoneme converter with these data for each language. The quality of Web-derived pronunciations is usually worse than handcrafted pronunciations as we show in Section 5.1. However, the word-pronunciation pairs from the Web can include complementary information to our given training data and we can find word-pronunciation pairs even for languages with no or very limited lexical resources, as we have shown in [SOS14]. First, we naively extracted word-pronunciation pairs without any filtering (5-30k phoneme tokens with corresponding grapheme tokens to reflect a saturated grapheme-to-phoneme consistency [SOS12b]). Second, we filtered them before we built the grapheme-to-phoneme converter with the methods which we have described in Section 4.2.

### 4.3.1 Single G2P Converter Output

For all grapheme-to-phoneme converters, we use context and tuning parameters resulting in lowest phoneme error rates on a development set with 1k word-pronunciation pairs as training data. For *Sequitur* 6-grams gave best performance, for *Phonetisaurus* 7-grams and for *Moses* 3-grams in phrase table and language model. Figure 4.5 demonstrates the phoneme error rates of the single grapheme-to-phoneme converter outputs. The converters with lowest phoneme error rate serve as a baseline for us and we compute the relative phoneme error rate change compared to their phoneme error rate in Section 4.3.2 and 4.3.3. We observe lower phoneme error rates with increasing amount of training data. Lowest phoneme error rates are achieved with *Sequitur* and *Phonetisaurus* for all languages and data sizes. *Sequitur* is slightly better than *Phonetisaurus* except for *en* with 5k training data and *es* with 200. *Carttree* results in worse performance. *Moses* is always worse than *Sequitur* and *Phonetisaurus*, even it is very close for *de*. Only for 200 *en* and *fr* word-pronunciation pairs, *Rules* outperforms *Moses*.

---

<sup>3</sup><http://www.wiktionary.org>



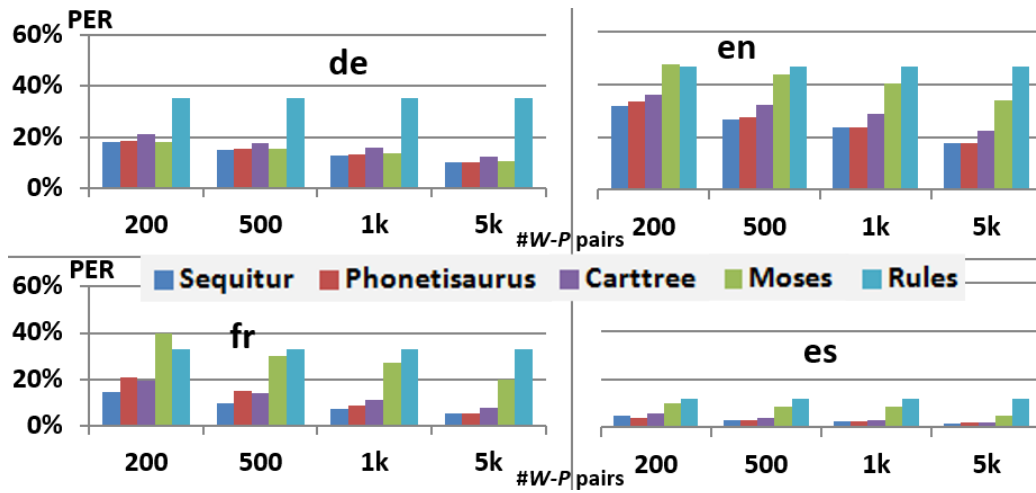
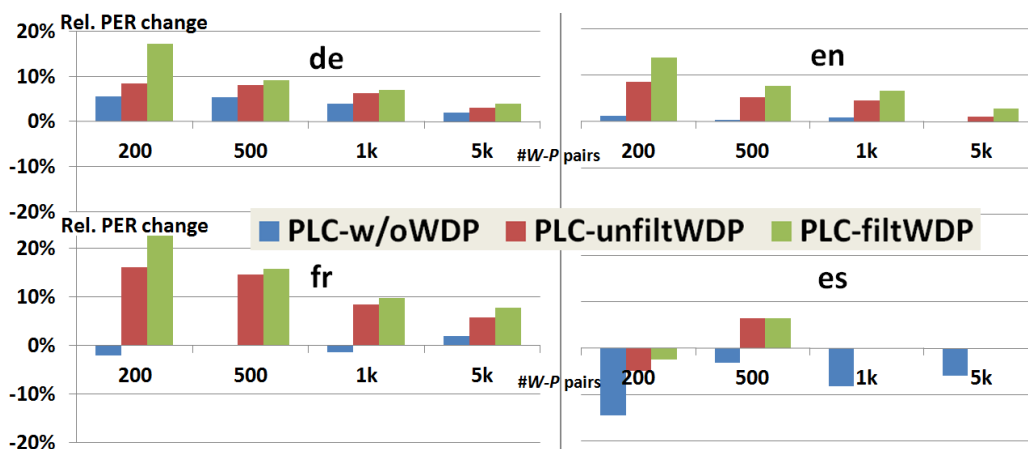


Figure 4.5 – Phoneme error rate (%) of single grapheme-to-phoneme converter outputs over amount of training data.

To show that the grapheme-to-phoneme converters produce different outputs, we present the edit distances at the phoneme-level between the grapheme-to-phoneme converter outputs trained with 1k word-pronunciation pairs in Figure 4.6. How much they differ depends on the similarity of the corresponding technique. For example, the smallest distances are between the grapheme-based converters *Sequitur* and *Phonetisaurus*, while *Rules* has the highest distances to the other approaches. It also depends on the grapheme-to-phoneme relationship: While the *en* outputs differ most for all amounts of training data, the *es* ones are closest. The distances of *fr* and *de* are located in between.

	[0-5[				[5-10[				[10-20[				[20-30[				[30-40[				[40-50[			
Rules																								
Moses																	40	35	36	11				
Carttree													40	13	28	7	46	34	34	11				
Phonetisaurus									24	12	11	2	36	8	26	8	46	34	33	12				
Sequitur					11	4	5	1	23	12	9	2	36	8	26	8	46	34	33	11				
	en	de	fr	es	en	de	fr	es	en	de	fr	es	en	de	fr	es	en	de	fr	es	en	de	fr	es
	Sequitur				Phonetisaurus				Carttree				Moses				Rules							

Figure 4.6 – Edit distances at the phoneme level (%) between grapheme-to-phoneme converter outputs (en / de / fr / es).



**Figure 4.7** – Phoneme error rate change (%) with *Phoneme Level Combination* using converters trained without and with Web-derived pronunciations.

### 4.3.2 Combination of G2P Converter Output

For the phoneme-level combination (*PLC*), we apply *nbest-lattice* at the phoneme-level, which is part of the SRI Language Modeling Toolkit [Sto02]. From each grapheme-to-phoneme converter we select the most likely output phoneme sequence (1st-best hypothesis). Then, we use *nbest-lattice* to construct a phoneme lattice from all converters’ 1st-best hypotheses and extract the path with the lowest expected phoneme error rate. We detected that in some cases the combination of subsets of grapheme-to-phoneme converter outputs improved the phoneme error rate slightly. In other cases worse 1st-best grapheme-to-phoneme converter outputs even helped to improve quality. As the impact is usually not clear, we continued our experiments with the combination of all 1st-best converter outputs.

The left blue bars in Figure 4.7 (*PLC-w/oWDP*) show the change in phoneme error rate compared to the grapheme-to-phoneme converter output with the highest quality. In 10 of 16 cases the combination performs equal or better than the best single converter. For *de*, we observe improvements for all training data sizes, for *en* slight improvements in four of five cases. Therefore, we selected these languages for our automatic speech recognition experiments (see Section 4.3.4). For *es*, the language with the most regular grapheme-to-phoneme relationship, the combination never results in improvements. The quality of the single *es* grapheme-to-phoneme converters seems already to be of such a good quality that minor disagreements have a negative impact. While for *de* the improvement is higher with less training data, the best *fr* improvement can be found with 5k training data. Further approaches of

weighting the 1st-best grapheme-to-phoneme converter outputs could only reach the quality of the best single converter and not outperform it [SQS14].

### 4.3.3 Adding Web-driven G2P Converter Output

We used *Sequitur* to build additional grapheme-to-phoneme converters based on pronunciations, which we found in *Wiktionary* together with corresponding words (*WDP*) (see also Section 5.1) and analyzed their impact to the combination quality. The single *de* Web-driven converter trained with unfiltered word-pronunciation pairs has a phoneme error rate of 16.74%, the *en* one 33.18%, the *fr* one 14.96%, and the *es* one 10.25%. The *de* one trained with filtered word-pronunciation pairs has a phoneme error rate of 14.17%, the *en* one 26.13%, and the *fr* one 13.97%. However, the phoneme error rate of the *es* one slightly increased to 10.90%. We assume that the *es* grapheme-to-phoneme converter performs worse after filtering the training data from the Web due to the following reason: Compared to the other three languages, we found only one-third of the amount of *es* grapheme-to-phoneme model training data in *Wiktionary*. The filter method which removes further 15% of the amount of found data provides a small training corpus size which is not enough to exceed the quality of the unfiltered data.

Figure 4.7 shows the changes without (*PLC-w/oWDP*) and with additional converter outputs compared to the best single converter. A positive value reflects the relative improvement compared to the the best single converter output trained with the corresponding amount of training data. A negative value shows relative degradations. First, we built grapheme-to-phoneme converters after we extracted word-pronunciation pairs from *Wiktionary* without any filtering (*PLC-unfiltWDP*) [SOS12a]. Second, we filtered them before we built the grapheme-to-phoneme converters as described in Section 5.1 (*PLC-filtWDP*). The *en* and *de* Web-driven grapheme-to-phoneme converters even perform better than converters trained with our 200 and 500 qualified word-pronunciations. For *es*, the Web-driven converters outperform our converter trained with 200 qualified word-pronunciation pairs.

We observe that *PLC-unfiltWDP* outperforms the best single converter output in 15 of 16 cases. In all cases it is better than *w/oWDP*. Like *PLC-unfiltWDP*, *PLC-filtWDP* outperforms the best single method in 15 cases. However, it is in all cases better than *PLC-unfiltWDP* and better than *PLC-w/oWDP*. With 23.1% relative phoneme error rate improvement, we report the largest improvement for *fr* where only 200 French word-pronunciation pairs and Web data are given as training. We also observed the high quality

of the pronunciations from the French *Wiktionary* edition in other experiments described in Section 5.1.

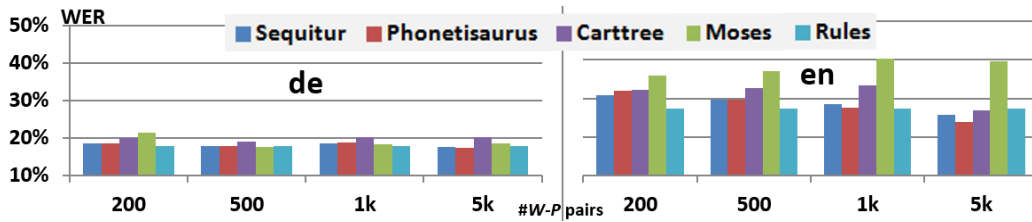
Where our *PLC* method improves the phoneme error rate, a linguist or native speaker has to change less phonemes to meet a validated pronunciation quality. Therefore, *PLC* has potentials to enhance the processes of semi-automatic pronunciation dictionary creation described in [MBT04], [DM09], [Kom06], [SBB<sup>+</sup>07], and [DFG<sup>+</sup>05]. In Section 5.3, we describe how we use the *PLC* method in our semi-automatic pronunciation generation strategy.

#### 4.3.4 Automatic Speech Recognition Experiments

For *de* and *en*, we have illustrated that we can approximate validated pronunciations using *PLC*, which can be helpful for speech synthesis and lowers the editing effort in the semi-automatic dictionary generation. In the following sections we investigate if the impact of the phoneme-level combination has a positive impact on automatic speech recognition performance as well. Furthermore, we compare *PLC* (*early fusion*) to a combination at lattice-level (*late fusion*) from the output of individual automatic speech recognition systems.

For the automatic speech recognition experiments, we replace all pronunciations in the dictionaries of our *de* and *en* *GlobalPhone*-based speech recognition systems with pronunciations generated with the grapheme-to-phoneme converters. Thereby, we replace 39k pronunciations for *de* and 64k for *en*. Then, we use them to build and decode LVCSR systems. The transcribed audio data and language models for *de* come from the *GlobalPhone* project [SVS13], those for *en* from the *WSJ0* corpus [GGPP93]. Finally, we decode their test sets with the resulting systems. For each amount of word-pronunciation pairs, the best performing single system serves as baseline. Then, we evaluated the combination approaches with the relative change in word error rate (WER) compared to the best performing system that is trained with a dictionary which has been built with a single converter. A positive value reflects the relative improvement compared to the the best single converter output trained with the corresponding amount of training data. A negative value shows relative degradations.

Figure 4.8 depicts the word error rate over the amounts of training data. The values for *en* are significantly higher than for *de*, while the automatic speech recognition systems trained and decoded with handcrafted dictionaries are closed in word error rate with 16.71% for *de* and 14.92% for *en*. This fact



**Figure 4.8** – Word error rate (%) with dictionaries from single grapheme-to-phoneme converter outputs over amount of training data.

shows that the *en* dictionaries which were also worse in the phoneme error rate evaluation still have negative impact on the word error rate performance. For both languages, there is a small general decrease with more training data using our data-driven grapheme-to-phoneme converters except for *en* with *Moses*. Whereas in our phoneme error rate evaluation *Sequitur* and *Phonetisaurus* outperform the other approaches, *Rules* results in lowest word error rates for most scenarios with less than 5k training data. The reason is that the pronunciations generated with less than 5k training data are more inconsistent than pronunciations generated with *Rules* which has a negative impact on the acoustic model training.

### Automatic Speech Recognition Systems with Dictionaries from Phoneme-Level Combination

Figure 4.9 shows that the word error rate changes compared to the best single converter, using dictionaries generated from *PLC* without (*PLC-w/oWDP*) and with (*PLC-filtWDP*) the additional Web-driven grapheme-to-phoneme converter outputs. *PLC-w/oWDP* is only in one case better than the best single method, whereas *PLC-filtWDP* outperforms the best single system in four cases. This shows that the Web data can also have a positive impact on automatic speech recognition performance.

Moreover, we learn that getting closer to the qualified reference pronunciations with *PLC* does not necessarily mean the word error rate of the automatic speech recognition systems improves. Figure 4.10 indicates that the correlation between the percentage of insertion and deletion errors (*I+D*) to the reference pronunciations at the phoneme-level correlates stronger with the word error rate than the phoneme error rate to the reference pronunciations. The automatic speech recognition systems usually deal better a little better with substitution errors in the pronunciations than insertion and deletion errors. Additionally, the fact that errors in the pronunciations of words

occurring frequent in training and test set have a bigger impact on the word error rate than less frequent ones blurs the correlation between word error rate and phoneme error rate.

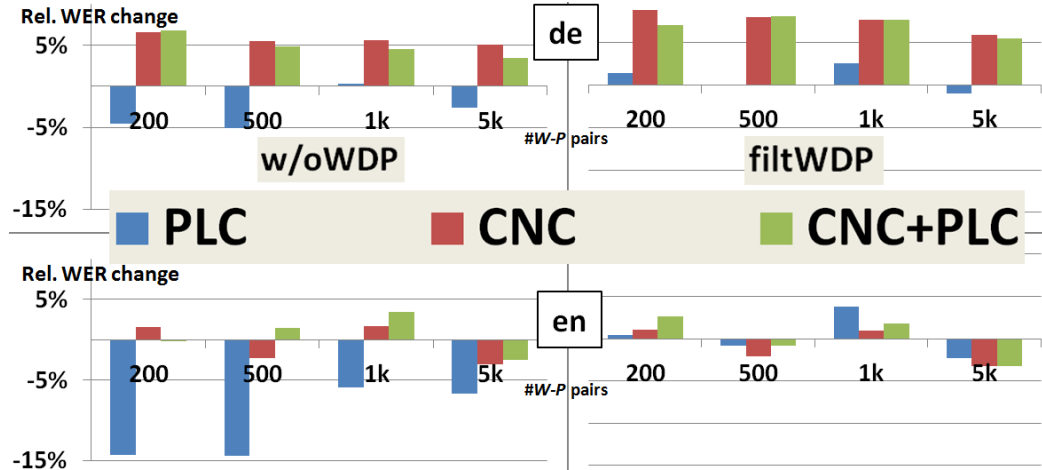


Figure 4.9 – Word error rate (%) change over training data size with and without Web-derived data for early (*PLC*) and late (*CNC*) fusion.

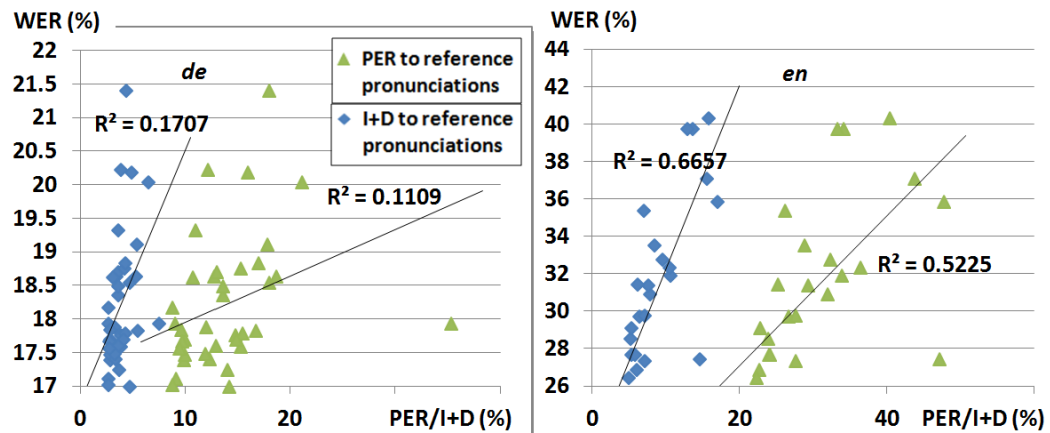


Figure 4.10 – Correlation between phoneme error rate/I+D and word error rate.

### Confusion Network Combinations

Before we compared the *PLC* approach to the confusion network combination, we compared our combination approach to two approaches of building the training and decoding dictionary where the outputs of the single

grapheme-to-phoneme converters are used as pronunciation variants: In the first approach we used all pronunciations for the training and decoding dictionary as in [JFI12]. Assuming that speakers in training and test set use similar speaking styles and vocabulary, and that our training process automatically selects the most likely pronunciations. In the second approach we removed all pronunciations in the dictionary that were not used in the training for decoding. However, both methods did outperform neither the corresponding best single converter nor the combination approaches in most cases.

Combination methods are known to lower the word error rate of automatic speech recognition systems [SFBW06]. They require systems which are reasonably close in performance but benefit from the fact that the output differs in its errors. This provides complementary information leading to performance improvements. As the individual automatic speech recognition systems with the dictionaries from different grapheme-to-phoneme converter outputs are close in performance, we combine them with a confusion network combination (CNC) (*late fusion*) and compare it to the *PLC* performance.

Figure 4.9 illustrates that a *late fusion* with *CNC* outperforms the *early fusion* approach with *PLC*. Including the systems with pronunciation dictionaries having been built with *PLC* in the *CNC* combination (*CNC+PLC*), outperformed *CNC* in six systems. While for *de* *CNC* gave improvement for all amounts of grapheme-to-phoneme training material, it outperformed the best single system in only half of the cases for *en*. With 8.8% relative word error rate improvement, we report the largest improvement for *de* where only 200 German word-pronunciation pairs and Web data are given as training data. We believe that the advantage of *CNC* is more context since the combination is at the word level and that language model information benefits the combination which is missing in the *PLC* approach.

Since the phoneme-level combination has potentials to generate a better pronunciation out of different pronunciations of the same word, we deploy it in crowdsourcing-based approaches where multiple annotators create pronunciations for the same word. In Section 5.2 we show how the *PLC* method can help to improve pronunciations created with our online game *Keynounce*. Moreover, we describe in Section 5.3 how we use the *PLC* method in our semi-automatic pronunciation generation strategy. Additionally, we apply *PLC* to obtain qualified word-pronunciation pairs as part of our whole pronunciation extraction algorithm for non-written languages, as specified in Chapter 8.

### 4.3.5 Summary

The automatic pronunciation generation can be improved through the combination of grapheme-to-phoneme converter outputs, especially where low resources are available to train the single converters. Our experiments showed that in most cases the phoneme-level combination approaches validated reference pronunciations more than the single converters. We detected that the output of grapheme-to-phoneme converters built with Web-derived word-pronunciation pairs can further improve pronunciation quality.

Pronunciations which are closer to high-quality pronunciations minimize the human editing effort in a semi-automatic pronunciation dictionary development, as we show in Section 5.3. In additional automatic speech recognition experiments we showed that the resulting dictionaries can lead to performance improvements. In Section 5.2, we demonstrate that *PLC* can also improve the quality of pronunciations which have been created by multiple annotators.



# Crowdsourcing-based Pronunciation Generation

---

As shown in [CBD10], crowdsourcing enables an inexpensive collection of large amounts of data from users around the world. For example, in [DL10, AV10] the crowdsourcing platform Amazon Mechanical Turk has been analyzed to collect data for human language technologies like judging machine translation adequacy and building parallel corpora for machine translation systems. In Section 3.2.2, we described how we assigned text normalization tasks to Turkers.

A collection of pronunciations on websites where Internet users have entered word-pronunciation pairs or with online games supports the process of pronunciation generation. In addition to a cheap way of gathering grapheme-to-phoneme converter training material, using pronunciations provided by Internet users is helpful for languages with a complex grapheme-to-phoneme correspondence where data-driven approaches have shortcomings.

In Section 5.1 we describe *Wiktionary*, a crowdsourcing-based online encyclopedia, where Internet users enter word-pronunciation pairs for several languages. We analyze the quality of the existing pronunciations and improve the quality of Web-derived pronunciations with our methods for automatic detection of inconsistencies and errors, which we have presented in Section 4.2.

In the second crowdsourcing approach, we let users proactively produce pronunciations in an online game [Lem13]. For this, we implemented Keynounce. The implementation and analyses are presented in Section 5.2.

## 5.1 Pronunciations from the Web

Our intention was to research if Web-derived pronunciations can be used to build pronunciation dictionaries from scratch or at least enhance existing ones. We extended our Rapid Language Adaptation Toolkit to extract pronunciations from the World Wide Web and collected pronunciations from *Wiktionary* [He09].

Initial investigations to leverage off pronunciations from the World Wide Web have been described in [GJK<sup>+</sup>09, CCG<sup>+</sup>09]. [GJK<sup>+</sup>09] retrieve English pronunciations in IPA and ad-hoc transcriptions from the World Wide Web and compare the pronunciations to the Pronlex dictionary<sup>1</sup> based on phoneme error rates. To extract, validate and normalize the Web-derived pronunciations, they apply English unigram grapheme-to-phoneme rules, grapheme-to-grapheme rules and phoneme-to-phoneme rules learned from the Pronlex dictionary. In [CCG<sup>+</sup>09], they apply their methods to English spoken term detection.

*Wiktionary*<sup>2</sup> is a community-driven free online lexical database providing rich information about words, such as explanations of word meanings, etymologies, and their pronunciations. As of today *Wiktionary* lists more than 13 million word entries covering more than 150 languages. Each of these languages has a dedicated *Wiktionary* edition. Parts of the lexical entries in an edition comprise pronunciations in the International Phonetic Alphabet (IPA). While phonetic notations are given mostly in IPA, some editions may also contain other notations such as SAMPA [Wel97]. An active community continuously contributes modifications and additions to the database entries. For the ten most popular languages in *Wiktionary*, Table 5.1 (June 2012) summarizes the number of cross-checked “good” database entries, total entries and entries with pronunciations, together with the number of administrators and active users. “Good entries” refer to content pages, excluding talk pages, help, redirects, etc. As can be seen, the community support and the number of available pronunciation entries vary greatly with language, while the amount of available resources (“Good” Entries) neither correlates with the amount of supporting people (Admins, Active Users) nor with the amount of speakers of that language.

Overall, the *Wiktionary* database is growing rapidly, both in the coverage of languages and in the amount of entries as Figure 5.1 indicates [SOS14].

---

<sup>1</sup>CALLHOME American English Lexicon, LDC97L20

<sup>2</sup>[www.wiktionary.org](http://www.wiktionary.org)

No.	Edition / Language	“Good” Entries	Total Entries	Entries with Prons.	Admins	Active Users
1	English	3,017k	3,180k	223k	98	1013
2	French	2,205k	2,320k	1,018k	23	353
3	Malagasy	1,516k	1,518k	591k	2	20
4	Chinese	826k	1,264k	4k	8	50
5	Lithuanian	608k	680k	1k	4	15
6	Russian	439k	621k	23k	7	143
7	Korean	342k	354k	35k	1	35
8	Turkish	308k	323k	4k	3	54
9	Greek	308k	333k	11k	7	44
10	Polish	287k	298k	80k	26	99

**Table 5.1** – The ten largest Wiktionary language editions, based on [Wik12].

Figure 5.1 has been created by the *Wiktionary* community in 2010 and illustrates the article count over time for eight languages from January 2003 to July 2010: English (*en*), French (*fr*), Lithuanian (*lt*), Turkish (*tr*), Chinese (*zh*), Russian (*ru*), Vietnamese (*vi*), and Ido (*io*). For some languages (e.g. French, English), the growth has almost been exponential, for others it is slower (e.g. Chinese). In rare cases, slightly negative growth might also occur, caused by merging or deletion of articles (e.g. Vietnamese). Due to the fast growth in language presence on *Wiktionary*, there is a future potential of harvesting pronunciations even for under-resourced languages from this source.

The *Wiktionary* online database is organized in a well structured format, which is nearly identical across languages and editions. Figure 5.2 displays an example page for the word “vitamin” in the English *Wiktionary* edition. The red circles in the screen shot marks the position where the pronunciations can be found. Pronunciations are given in terms of IPA symbols.

As shown in Figure 5.2 *Wiktionary* pages may contain more than one pronunciation per word. These additional pronunciations reflect alternative pronunciations, dialects, or even different languages. To gain some insights into this “language-mix” we performed a brief analysis on the English, French, German, and Spanish *Wiktionary* editions. For German *Wiktionary*, for example, we found that only 67% of the detected pronunciations are for German words, the remainder is for the languages Polish (10%), French (9%), English (3%), Czech (2%), Italian (2%), etc. Figure 5.3 (June 2012) shows this “language-mix” in the English and the French editions.

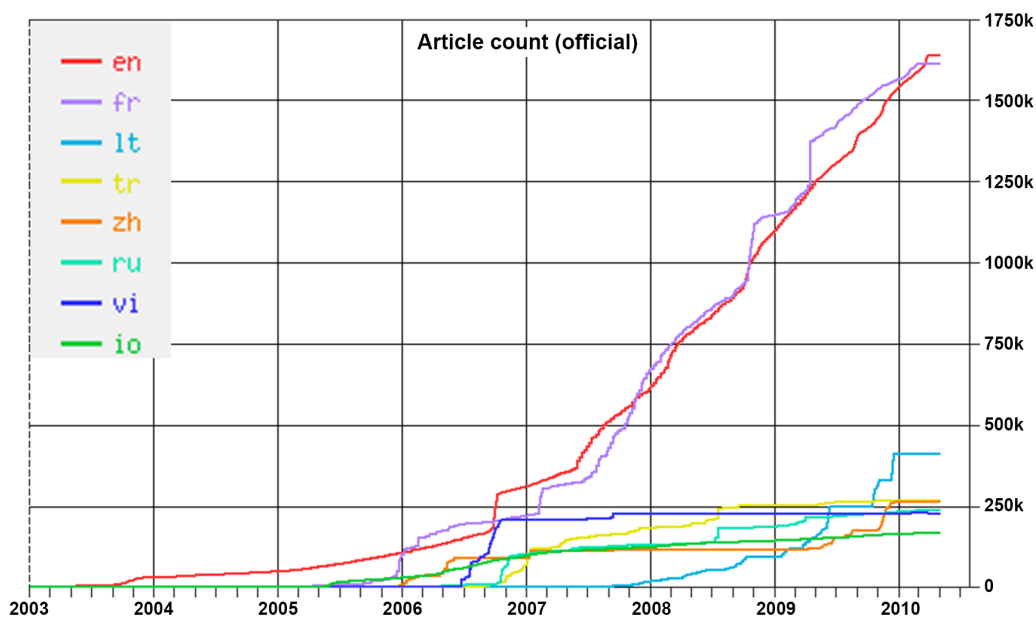


Figure 5.1 – Growth of *Wiktionary* entries over several years [wik14].

For some *Wiktionary* pages no information is given which language the pronunciations belong to. However, per convention the first given pronunciation on a *Wiktionary* page of a certain language edition usually refers to that language. In the experiments described later, we therefore always retrieve the first pronunciation, if multiple candidates exist.

As mentioned above, the number of data entries in *Wiktionary* varies widely across languages. Thus, we selected those *Wiktionary* editions which provide 1,000 or more pronunciation pages. In total, we found 39 editions which qualify. In Figure 5.4 we show the amount of pronunciations retrieved from those 39 editions, marked with their language families for illustration (June 2012). So far, the Indo-European languages dominate. However, other language families such as Basque, Altaic, Uralic, Sino-Tibetan, Japonic, Austronesian, Korean, and Austro-Asiatic are covered as well. Given the rapid growth and active community, other languages may catch up quickly.

Since we explore grapheme-to-phoneme based approaches in this study, it is also worth examining the coverage of writing systems. With Roman (e.g. English) and Cyrillic (e.g. Russian) the most prominent phonographic-segmental scripts are covered, with Arabic the most prominent abjad is covered, phonographic-syllabic scripts are represented (e.g. Japanese kana), phonographic-featural scripts are covered (e.g. Korean) and logographic scripts are covered by Hanzi (Chinese) and Japanese (Kanji). Therefore, it is rea-

vitamin

---

See also [Vitamin](#)

**Contents** [hide]

1 English

- 1.1 Etymology
- 1.2 Pronunciation
- 1.3 Noun
  - 1.3.1 Hyponyms
  - 1.3.2 Derived terms
  - 1.3.3 Translations
  - 1.3.4 See also

2 Danish

- 2.1 Pronunciation
- 2.2 Noun
  - 2.2.1 Inflection
  - 2.2.2 Related terms
  - 2.2.3 External links

**English** [edit]

---

**Etymology**

1920, originally *vitamine* (1912), from Latin *vīta* "life" (see *vital*) + *amine* (see *amino acids*). *Vitamine* coined by Polish biochemist *Casimir Funk* after the initial

⋮

**Pronunciation** [edit]

This entry needs **audio files**. If you have a microphone, please **record some** and **upload** them. (For audio required quickly, visit [WT:APR](#).)

- *(UK)* IPA: /vɪ.tə.mɪn/
- *(US)* IPA: /ˈvɑː.tə.mɪn/
- *(Australia)* IPA: /ˈvɑː.tə.mən/

⋮

**Danish** [edit]

---

**Pronunciation** [edit]

- IPA: /vitamiːn/, [vita miːʔn]

⋮

Figure 5.2 – English *Wiktionary* page for the word “vitamin”.

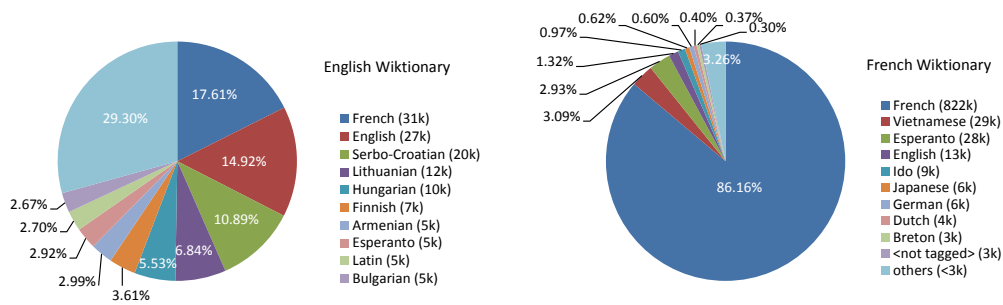


Figure 5.3 – Pronunciation diversity in English and French *Wiktionary*.

sonable to assume that *Wiktionary* covers most of the wide-spread writing systems of the world.

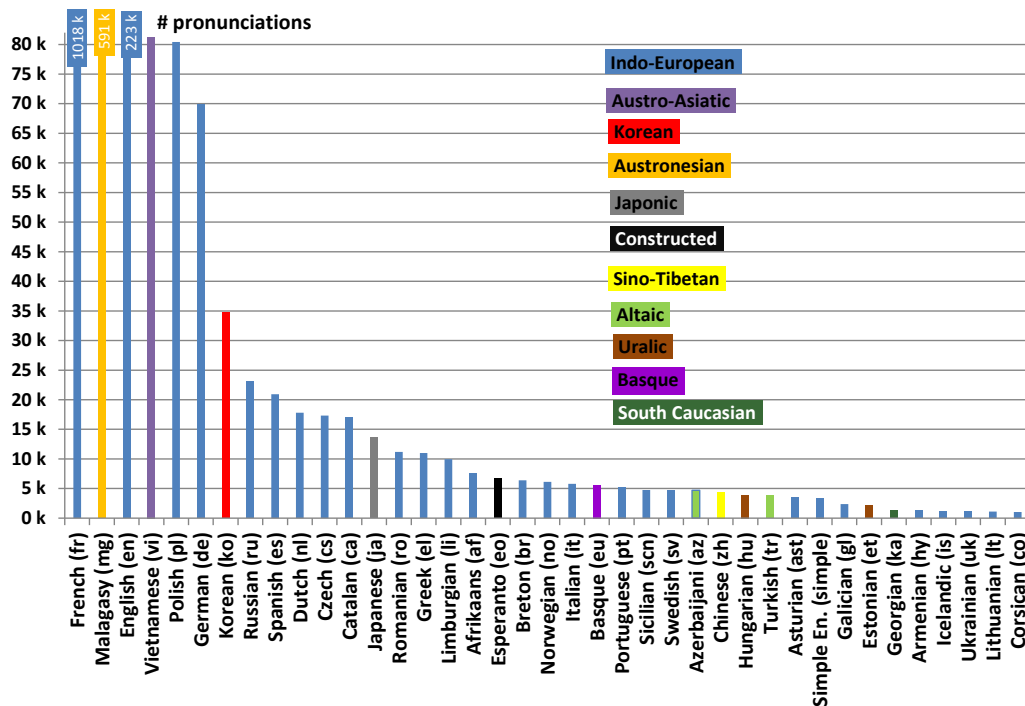


Figure 5.4 – *Wiktionary* editions with more than 1k pronunciations.

### 5.1.1 Automatic Pronunciation Extraction

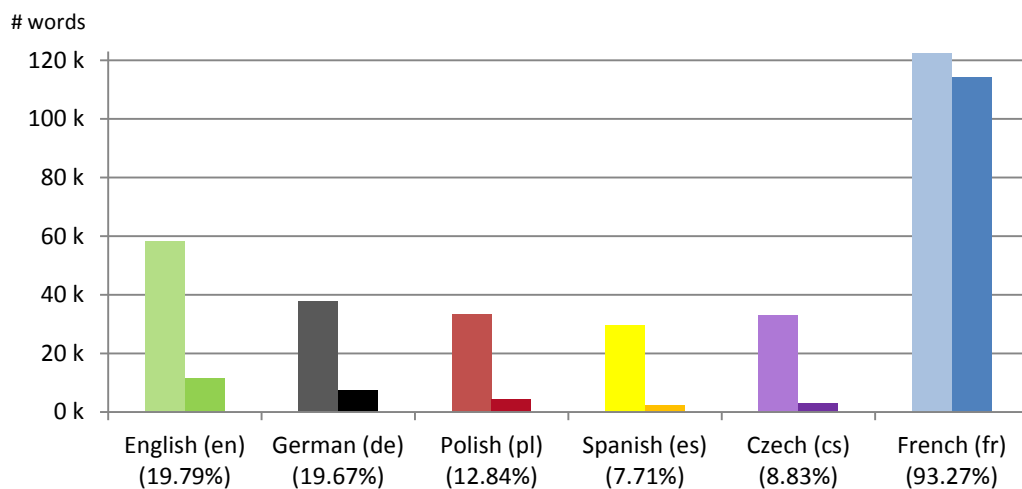
We implemented an *Automatic Dictionary Extraction Tool* and integrated this component into our Rapid Language Adaptation Toolkit [SOS10, SOS12b, SOS14]. The new component allows to extract pronunciations given in IPA from one or more online dictionaries that have separate pages for individual words, such as *Wiktionary*. The tool takes a vocabulary list as input and finds matching pages in the online dictionaries. If no matching page can be found, the tool tries the upper and lower case alternatives. We are aware that this procedure may cause flaws since for some languages such as German, casing can have an impact on the pronunciation (e.g. “Weg” is pronounced with a long /e/, but “weg” with a short /ɛ/).

Each retrieved Web page is then parsed for IPA-based pronunciations by searching for strings which contain at least one character in the IPA Extensions Unicode block (U+0250..U+02AF), surrounded by delimiters such

as “/ /” and “[ ]”, similar to [GJK<sup>+</sup>09]. This procedure allows a website-independent collection of pronunciations. As mentioned above, a *Wiktionary* page may contain more than one pronunciation. The current implementation picks the first variant if multiple exist. Finally, the tool outputs a pronunciation dictionary in IPA notation for those words in the vocabulary lists which were found in the online dictionary, and reports back for which words no match could be found.

### 5.1.2 Evaluation of the Web-derived Pronunciations

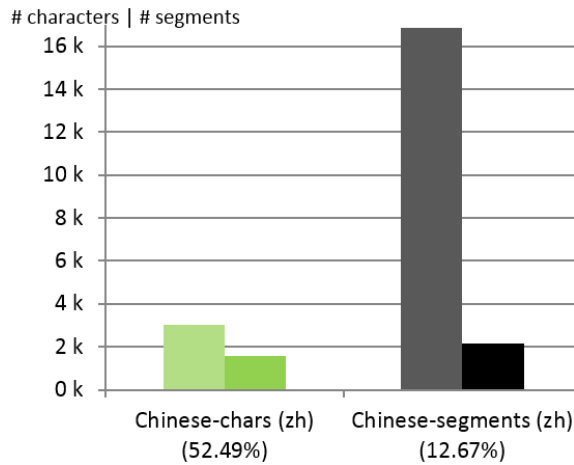
For evaluating the quantity and quality of pronunciations in *Wiktionary* as a seed resource for multilingual pronunciations, we use *GlobalPhone*-based dictionaries as reference data as they have been successfully used in LVCSR in the broadcast domain. Furthermore, we build automatic speech recognition systems with *GlobalPhone* dictionaries, speech data and language models for the evaluation.



**Figure 5.5** – Ratio of searched words (first bar) and retrieved *Wiktionary* pronunciations (second bar).

#### Quantity

With the vocabulary of the English, German, Polish, Spanish, Czech, and French *GlobalPhone*-based dictionaries as input to the Automatic Dictionary Extraction, we analyzed the coverage of corresponding pronunciations



**Figure 5.6** – Ratio of searched Chinese characters (segments) (first bar) and retrieved *Wiktionary* pronunciations (second bar).

found in *Wiktionary*. Figure 5.5 shows the outcome of this quantity check. French *Wiktionary* outperforms the other languages with a 93.27% coverage for the French broadcast-domain vocabulary list, while the coverage of the *Wiktionary* editions for English, German, Polish, Spanish, and Czech were surprisingly low. Possible explanations are the strong French internet community and there have been several imports of entries from freely licensed dictionaries in the French *Wiktionary* edition<sup>3</sup>. Additionally, we analyzed the coverage of pronunciations for Chinese. As depicted in Figure 5.6, we found pronunciations for 52.49% of the 3k Chinese characters in our Mandarin *GlobalPhone* dictionary. We also search for Chinese segments in the *GlobalPhone* dictionary and detected a coverage of 12.67%.

Language	Total # pages	# pages with prons	% pages with prons
English (en)	3,180k	223k	7.0%
French (fr)	2,319k	1,018k	43.9%
Polish (es)	298k	80k	26.8%
German (de)	228k	70k	30.7%
Spanish (es)	90k	21k	23.3%
Czech (cs)	37k	17k	45.9%

**Table 5.2** – Absolute and relative amount of *Wiktionary* pages containing pronunciations.

<sup>3</sup><http://en.wikipedia.org/wiki/FrenchWiktionary>



Particularly, the English coverage was discouraging given the large amount of *Wiktionary* pages for English. Therefore, we analyzed how many of the English *Wiktionary* pages actually include pronunciations in IPA notation. Table 5.2 shows that English is outnumbered by a factor of 4.5 compared to French. Furthermore, as shown in Figure 5.3, only 15% of the pronunciations in the English *Wiktionary* are tagged as belonging to English. To confirm that the poor coverage was not solely caused by our English vocabulary, we additionally computed the coverage with the vocabulary list of the English *Pronlex* dictionary and obtained a similarly low result of 22.67%.

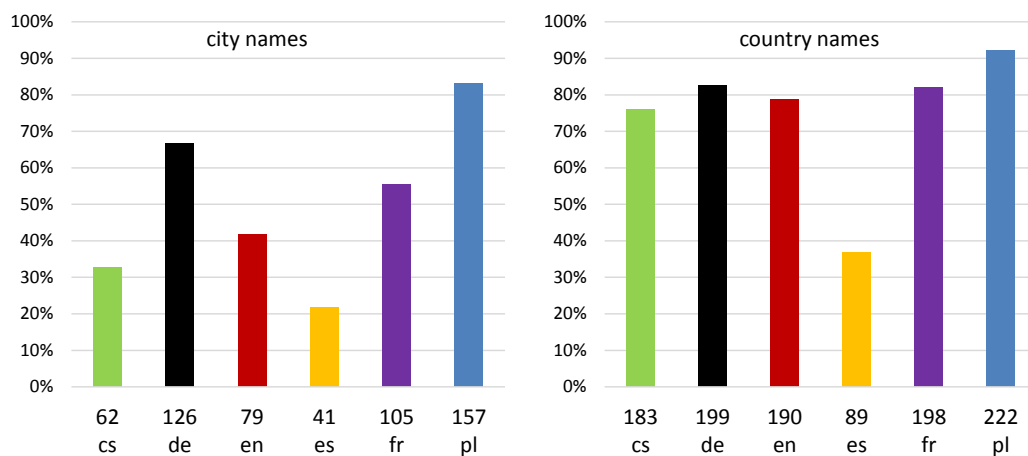


Figure 5.7 – Coverage for international city and country names.

We also investigated the coverage of pronunciations for proper names using a list of 241 country and 189 city names. Proper names are often of diverse etymological origin and may surface in various languages without undergoing the process of assimilation to the phonetic system of the new language [LB02]. Consequently, the pronunciations of proper names are difficult to generate automatically by grapheme-to-phoneme rules. For the quantity check using *Wiktionary*, the city and country names were translated into the respective language prior to each check. For example, the German city name “München” was translated into “Munich” for the analysis of the English *Wiktionary*. The results of this quantity check are illustrated in Figure 5.5 and 5.7. The coverage of country names is significantly higher than for city names in all language editions but Polish. Polish *Wiktionary* outperforms the other languages with a 92.1% coverage for country names and 83.0% coverage for city names. Compared to previously published data in [SOS10], the coverage of country and city names increased significantly within two years, reflecting the growth of pronunciations in *Wiktionary*.

## Quality

We compared pronunciations from *Wiktionary* to pronunciations from our dictionaries based on the *GlobalPhone* project. Table 5.3 shows that the highest percentages of identical pronunciations can be found in the French and Czech *Wiktionary* editions. For Spanish and Polish, half of the retrieved pronunciations are identical, for German and English almost a third. In [SOS10], we removed the identical pronunciations, filtered the remaining ones, and used them successfully in automatic speech recognition as pronunciation variants.

No.	Language	# prons.	% equal	# new
1	French	114k	74%	30k
2	Czech	3k	73%	1k
3	Polish	4k	52%	2k
4	Spanish	2k	50%	1k
5	German	7k	28%	5k
6	English	12k	26%	9k

**Table 5.3** – Amount of compared pronunciations, percentage of identical ones and amount of new pronunciation variants.

For our quality analysis, we built grapheme-to-phoneme models for European languages from six *Wiktionary* editions and compared them to the grapheme-to-phoneme models from the ten *GlobalPhone*-based dictionaries which we used in Section 4.1.1. To train the setup, we used the parameters with a maximum grapheme length  $L$  of 1 and an n-gram order of 6, as described in Section 1.4.4. To check the grapheme-to-phoneme model consistency, we built grapheme-to-phoneme models with increasing amounts of word-pronunciation pairs from *GlobalPhone* and *Wiktionary* as training material. We applied them to test sets from the respective source and computed the phoneme error rate to the original pronunciations. Furthermore, we evaluated the *Wiktionary* grapheme-to-phoneme models on the *GlobalPhone* test sets to appraise if the Web-derived data meet the quality of validated dictionaries. Then, we selected grapheme-to-phoneme models which had all been trained with a comparable number of training material. We investigated their grapheme-to-phoneme consistency, complexity and their usage for automatic speech recognition. For the automatic speech recognition experiments, we replaced the pronunciations in the dictionaries of six *GlobalPhone* automatic speech recognition systems (Czech, English, French, Spanish, Polish, and German) and studied the change in performance by us-

ing exclusively pronunciations generated from *Wiktionary* and *GlobalPhone* grapheme-to-phoneme models for training and decoding.

### Consistency Check

Our consistency check reflects the generalization ability of the grapheme-to-phoneme models. Table 5.4 shows how we analyzed the consistency within the *GlobalPhone*-based dictionaries (*GP*) and the *Wiktionary* editions (*wikt*) as well as between *Wiktionary* and the validated *GlobalPhone*-based dictionaries (*wiktOnGP*). For *GP* and *wikt*, we built grapheme-to-phoneme models with increasing amounts of word-pronunciation pairs in the dictionaries. Then we applied these to words from the same dictionary and compute the phoneme error rate between the new and the original pronunciations. For *wiktOnGP*, we evaluated the pronunciations generated with *Wiktionary* grapheme-to-phoneme models on the original *GlobalPhone* pronunciations. The resulting phoneme error rates tell us how close to validated pronunciations we can get with *Wiktionary* grapheme-to-phoneme models.

	Train	Test
<i>GP</i>	GlobalPhone	GlobalPhone
<i>wikt</i>	Wiktionary	Wiktionary
<i>wiktOnGP</i>	Wiktionary	GlobalPhone

**Table 5.4** – Consistency check setup.

As in Section 4.1.1, we performed a 6-fold cross validation to verify the pronunciation quality: For each *Wiktionary* edition and each *GlobalPhone* dictionary, we randomly selected 30% of the total number of word-pronunciation pairs for testing. From the remainder, we extracted increasing amounts of entries based on their accumulated phoneme count and used them for training the grapheme-to-phoneme models in each fold.

Figure 5.8 and 5.9 demonstrate differences in grapheme-to-phoneme consistency among the languages. As grapheme-to-phoneme accuracy is also a measure of the regularity of the grapheme-to-phoneme relationship of a language, we additionally see the grade of regularity of the ten languages in Figure 5.9. Comparing Figure 5.8 and 5.9 shows that for all tested languages except for German, *GP* is more consistent internally than *wikt*. *GP* comes closer to the validated *GlobalPhone* pronunciations than *wiktOnGP* for all languages. Figure 5.10 reveals noticeable differences between the phoneme error rates of *GP* and *wiktOnGP*. For Czech, English, and Spanish, the phoneme error rates of *wiktOnGP* are located between *wikt* and *GP* of the same language. However, for German, French, and Polish, the dictionaries

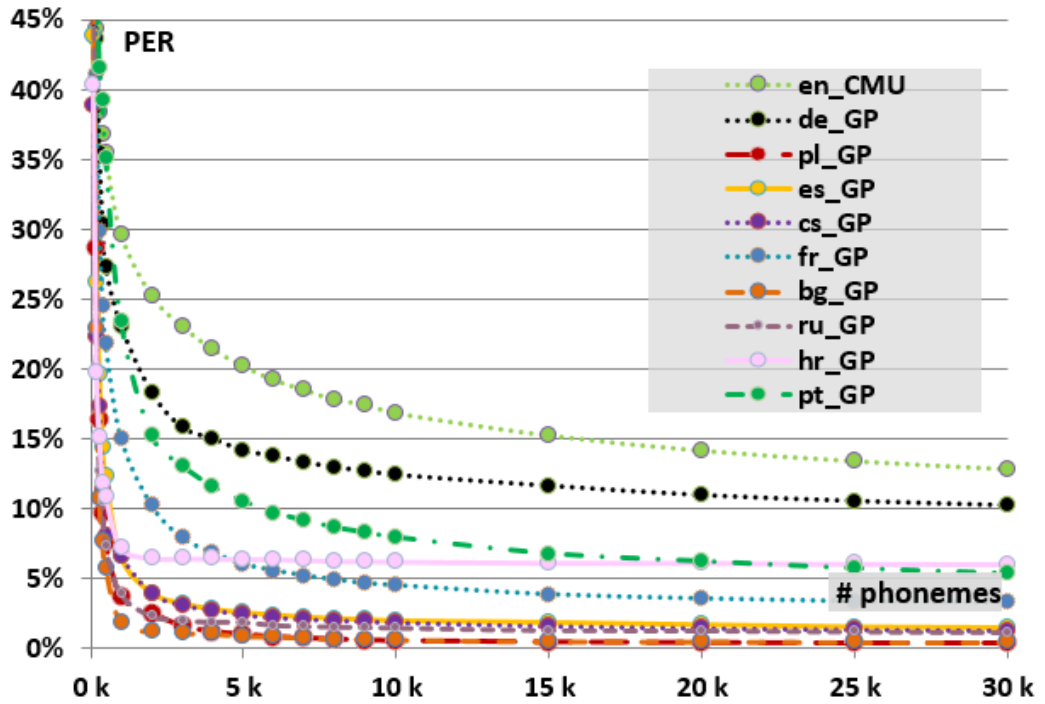


Figure 5.8 – Consistency of *GP*.

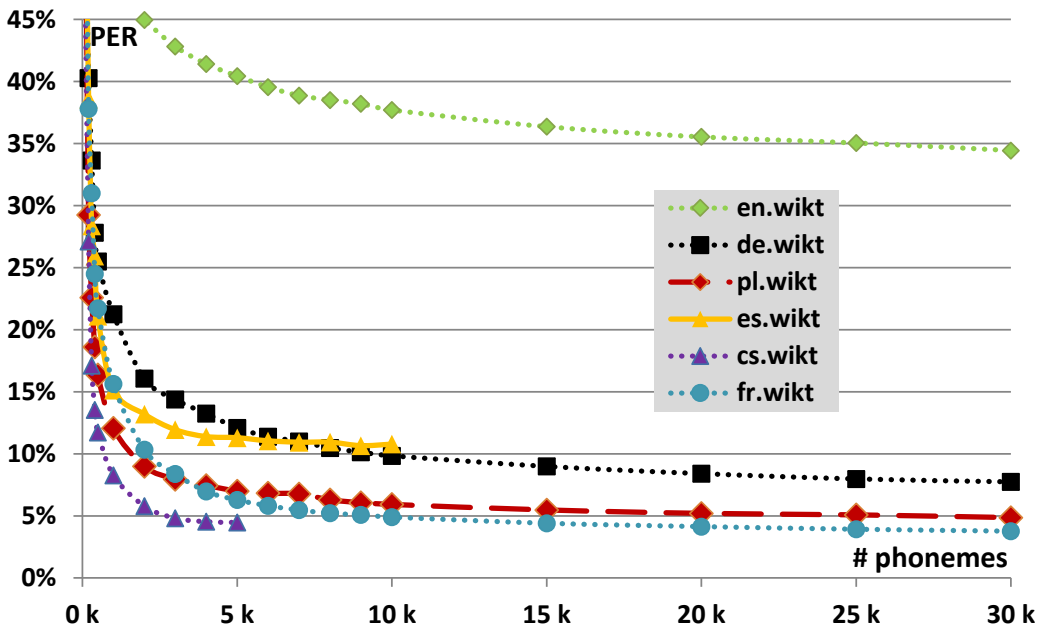


Figure 5.9 – Consistency of *wikt*.

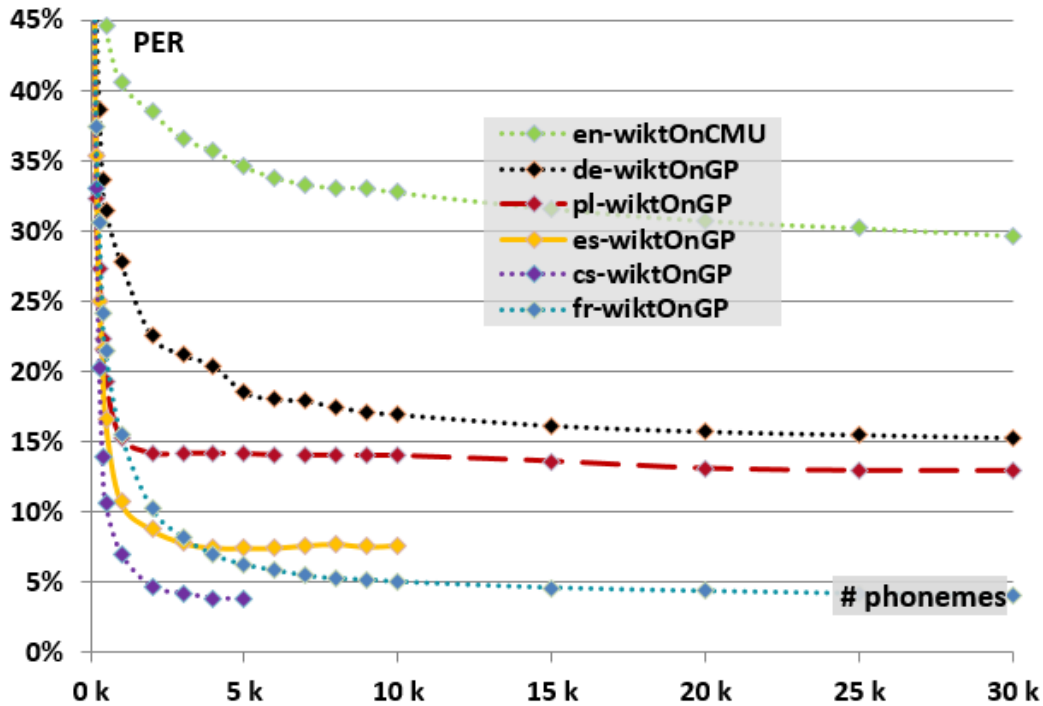


Figure 5.10 – Consistency of *WiktOnGP*.

are consistent internally but do not match in the cross-dictionary evaluation. Figure 5.8 and 5.9 show variations in phoneme error rates for amounts of training data below 7k phonemes. Like in Section 4.1.1, for more than 7k phonemes, the phoneme error rates decrease with more training data. As with the *GlobalPhone* pronunciations, we learn that *Wiktionary* word-pronunciation pairs containing 15k phonemes are sufficient to have constant quality as the curves start to saturate at 15k phonemes.

As demonstrated in Figure 5.11, the standard deviations in consistency for *wikt* are rather small. For *GP*, we detect even smaller deviations. Generally, a trend to smaller deviations with more training material for the grapheme-to-phoneme models can be observed. A deviation of less than 1% can be reached with only 1k phonemes with corresponding graphemes for most languages tested.

#### Complexity Check

For the second category, we investigated the complexity of the grapheme-to-phoneme models over training data and among languages and compare the complexity change to the consistency change. As described in Section 4.1.1, we defined the grapheme-to-phoneme model sizes represented by the number

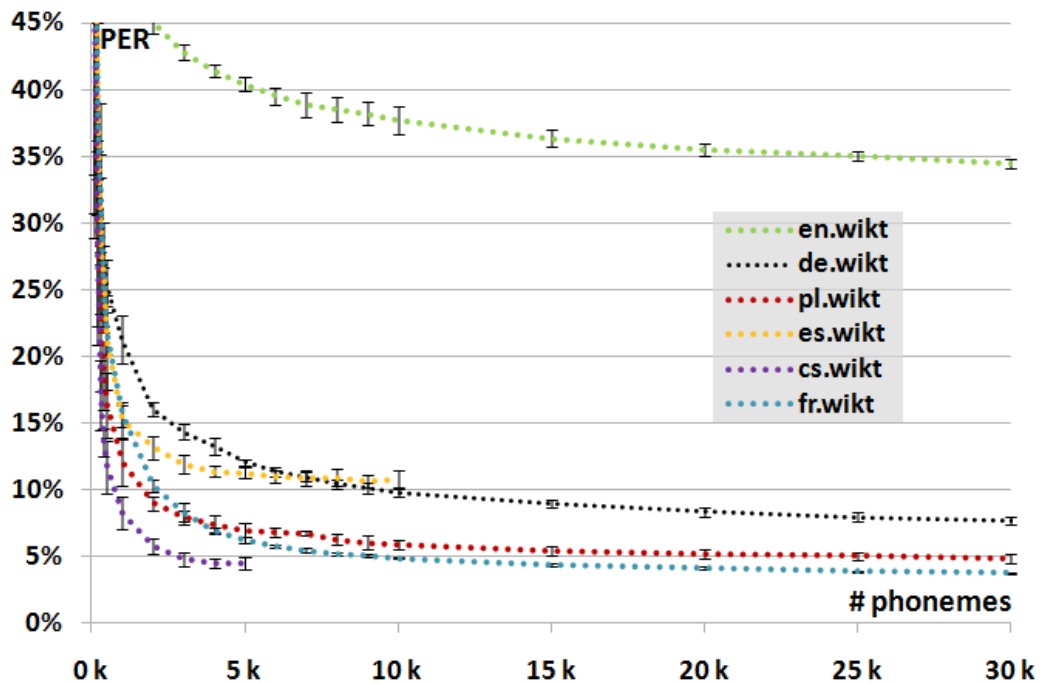


Figure 5.11 – *Wiktionary* grapheme-to-phoneme model consistency variances.

of non-pruned 6-grams as the measure for complexity. Figure 5.12 and 5.13 show the increase in complexity of the grapheme-to-phoneme models with the increase of training material between 100 and 30k phonemes with corresponding graphemes. A comparison of Figure 5.8 and 5.9 with Figure 5.12 and 5.13 indicates that although the consistency saturates at 15k phonemes, the model complexity keeps increasing for larger amounts of training data. However, as also observed in Section 4.1.1, this has minor impact on quality in terms of consistency as the model increases with the new M-grams which, however, represent seldom rules and rather exceptions after 15k phonemes.

For further analyses of the automatic speech recognition performance, we selected grapheme-to-phoneme models which were trained with 30k phonemes and their corresponding graphemes to reflect a saturated grapheme-to-phoneme model consistency. 30k phonemes are contained in all *GlobalPhone*-based dictionaries and in most of the 6 *Wiktionary* editions. For the Czech and Spanish *Wiktionary* and *GlobalPhone* grapheme-to-phoneme models, we used the maximum number of phonemes (5k and 10k) which we could find in *Wiktionary*.

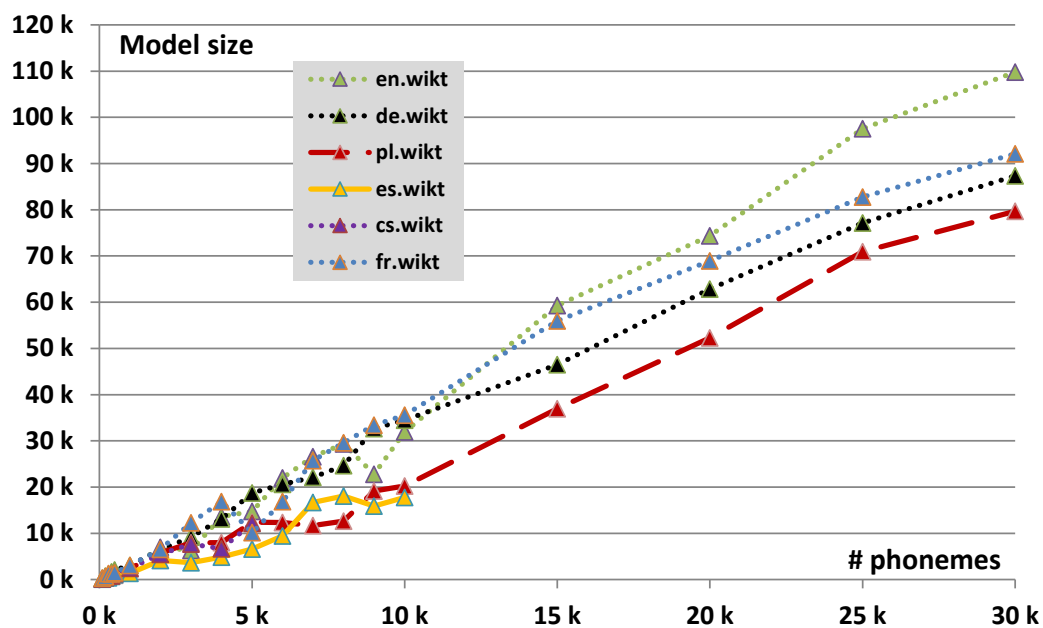


Figure 5.12 – *Wiktionary* grapheme-to-phoneme model complexity.

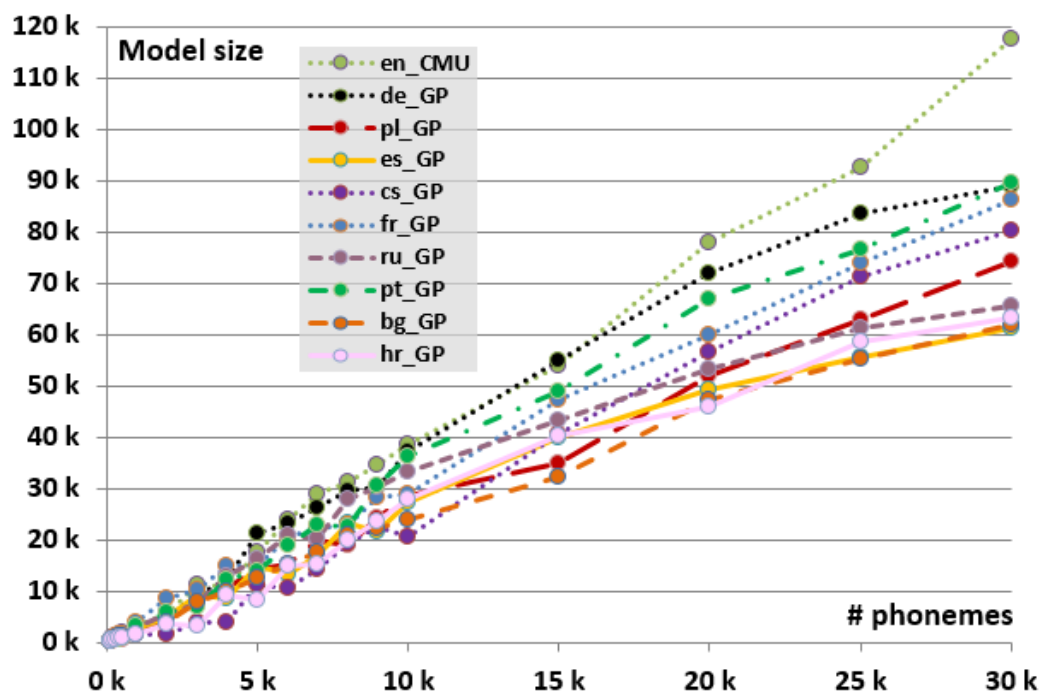


Figure 5.13 – *GlobalPhone* grapheme-to-phoneme model complexity.

### Evaluation by Automatic Speech Recognition

Finally, we analyzed how the pronunciations generated with our *wikt* and *GP* grapheme-to-phoneme models perform in automatic speech recognition. We replaced the pronunciations in the dictionaries of six *GlobalPhone* automatic speech recognition systems with pronunciations generated with *wikt* and *GP* grapheme-to-phoneme models. Then we trained and decoded the systems with those pronunciation dictionaries and evaluated their word error rates. First, we built and decoded automatic speech recognition systems with dictionaries consisting of only the most likely (1-best) pronunciation for each *GlobalPhone* word as produced by our grapheme-to-phoneme models. We compared these to *GlobalPhone* systems which were also limited to one pronunciation per word (base form). Furthermore, we established systems with dictionaries where pronunciation variants (n-best) were also produced. For each word, we generated exactly as many pronunciations as occurred in the *GlobalPhone*-based dictionaries. The results of the automatic speech recognition experiments in terms of word error rates are listed in Table 5.5.

For all languages but Spanish and French, the systems built with the 1-best grapheme-to-phoneme models performs better than those with pronunciation variants. With the *Wiktionary* grapheme-to-phoneme models, we come close to the word error rates of the *GlobalPhone* systems for all languages with on average a difference of 12% relative except for English. However, the *GlobalPhone* grapheme-to-phoneme systems perform slightly better as they did in our other evaluation categories. We explain the high word error rates in English with a difficult grapheme-to-phoneme correspondence and corrupted training material from *Wiktionary*. European languages are written in segmental phonographic scripts, which display a somewhat close grapheme-to-phoneme relationship, with one grapheme roughly corresponding to one phoneme. Therefore, we also trained and decoded automatic speech recognition systems with pure graphemic dictionaries (*Grapheme-based*) for comparison, an approach which gave encouraging results in former studies [KN02, KSS03, SS04] for large vocabulary speech recognition on several European languages. For some languages the grapheme-based approach even outperforms manually cross-checked phoneme-based dictionaries. Table 5.5 illustrates that for Czech, English, French, and Polish the grapheme-based systems outperform the automatic speech recognition systems with pronunciations generated with *wikt* grapheme-to-phoneme models. This shows the need for solutions to raise the quality of the data obtained from the Web.



	cs	de	en	es	fr	pl
GlobalPhone baseform	15.59	16.71	14.92	12.25	20.91	15.51
GP 1-best	17.58	16.50	18.15	12.59	22.68	15.78
wikt 1-best	18.72	16.81	28.86	12.82	25.79	17.21
GlobalPhone with variants	15.62	17.11	11.52	11.97	20.41	14.98
GP n-best	18.06	17.06	18.66	12.32	22.68	15.68
wikt n-best	19.32	17.40	37.82	12.81	25.17	17.34
Grapheme-based	17.56	17.83	19.15	14.06	23.36	15.38

**Table 5.5** – Word error rates (%) with dictionaries built completely with grapheme-to-phoneme model generated pronunciations (w/o filtering).

### 5.1.3 Recovery of Inconsistencies and Errors

To enhance the quality of the Web-driven grapheme-to-phoneme converters, we applied the methods to filter erroneous word-pronunciation pairs from *Wiktionary* and improve the resulting grapheme-to-phoneme models [SOS12a], as presented in Section 4.2. First, we applied our filter methods to remove faulty word-pronunciation pairs. Then, we took the remaining pairs to build new grapheme-to-phoneme models, which in turn were used to generate new pronunciations for the entire vocabulary.

For *baseline*, we used grapheme-to-phoneme models which originate from randomly unfiltered word-pronunciation pairs of the same amount as the filtered ones. Table 5.6 shows that we are able to reduce the word error rate of all tested systems, while the success of each method differs among the language editions. The best results are in bold. We believe that how good a method is depends on the number and kind of errors. Future work may include an analysis to find faster the best method for a dictionary in question.

We see improvements with  $G2P_{Len}$ ,  $Eps$ ,  $M2NAlign$ , and  $G2P_{M2NAlign}$ . With a relative word error rate reduction of 37.6% compared to the system with the unfiltered dictionary (*wikt 1-best*), most improvement is achieved on the English *Wiktionary* word-pronunciation pairs with  $M2NAlign$ . Samples in the original English data indicate a high number of pronunciations from other languages, pronunciations for stem or ending instead of the whole word or completely different pronunciations, which result in a bad initial dictionary. Without this outlier, the average improvement is 2.5% relative.  $G2P$ ,  $Len$ , and  $G2P_{Eps}$  do not improve the systems. Table 5.6 illustrates that for Czech, English, and Polish the grapheme-based systems still performed slightly

	cs	de	en	es	fr	pl
GP 1-best	17.58	16.50	18.15	12.59	22.68	15.78
wikt 1-best	18.72	16.81	28.86	12.82	25.79	17.21
wikt <i>G2P</i>	17.86	17.18	30.00	13.14	25.62	17.00
wikt <i>Len</i>	18.24	17.13	23.68	13.50	25.48	17.38
wikt <i>G2P<sub>Len</sub></i>	17.85	<b>16.79</b>	24.74	13.05	25.59	17.31
wikt <i>Eps</i>	<b>17.74</b>	17.12	22.85	12.99	<b>23.19</b>	16.98
wikt <i>G2P<sub>Eps</sub></i>	18.15	17.08	22.90	12.86	25.44	16.68
wikt <i>M2NAlign</i>	18.20	17.53	<b>20.97</b>	<b>12.25</b>	25.70	16.87
wikt <i>G2P<sub>M2NAlign</sub></i>	17.93	17.18	23.73	13.64	25.03	<b>16.57</b>
Grapheme-based	17.56	17.83	19.15	14.06	23.36	15.38

**Table 5.6** – Word error rates (%) with dictionaries built completely with grapheme-to-phoneme generated pronunciations (with filtering).

better than the automatic speech recognition systems with pronunciations generated with filtered *Wiktionary* grapheme-to-phoneme models. While Czech and Polish follow the pattern of languages with a close grapheme-to-phoneme relationship whereby grapheme-based systems can perform better than phoneme-based ones, the English *Wiktionary* word-pronunciation pairs are too poor for our data-driven methods that expect more good than bad pronunciations in the data to obtain good estimates for  $\mu$  and  $\sigma$ . However, it is possible to enhance automatic speech recognition performance with a confusion network combination of phoneme- and grapheme-based systems, even if the grapheme-based system is slightly better than the phoneme-based system. Both systems need to be reasonably close in performance but at the same time produce an output with different errors. This provides complementary information which leads to performance improvements as we have demonstrated with the African language Hausa in [SDV<sup>+</sup>12]. Furthermore, we show in Section 5.3 that the filtered Web-derived pronunciations can support the semi-automatic pronunciation generation process.

#### 5.1.4 Summary

We have demonstrated that pronunciations retrieved from the World Wide Web are an economical data source which supports the rapid creation of pronunciation dictionaries. With our Automatic Dictionary Extraction Tool, part of the Rapid Language Adaptation Toolkit, we developed a system for automatically extracting phonetic notations in IPA from any Web source

which has separate pages for individual words. The tool takes a vocabulary list as input and finds matching pages in the online dictionaries.

We analyzed the quantity and quality of pronunciations in *Wiktionary*, as it is available in many languages. The quantity checks with lists of international cities and countries demonstrated that even proper names whose pronunciations might not be found in the phonetic system of a language are detectable together with their phonetic notations in *Wiktionary*. Due to the fast growth in language presence on *Wiktionary*, there is a future potential of harvesting pronunciations for under-resourced languages from this source.

Moreover, we have investigated the grapheme-to-phoneme model generation for European languages with pronunciations from 6 *Wiktionary* editions. We analyzed and compared their quality and detected the saturation at 15k phonemes with corresponding graphemes as training material which we also described in Section 4.1.

Designers of automatic speech recognition systems are supposed to ensure that they use exclusively pronunciations of good quality for dictionary creation. Therefore, we applied our completely automatic error recovery methods which we have introduced in Section 4.2. Our methods improved the automatic speech recognition performances in each language.

## 5.2 An Online Game for Pronunciation Generation

With the online game *Keynounce* we analyzed, if it is possible to generate pronunciations with the help of an unknown and possibly unskilled crowd on the Internet [Lem13]. Providing a keyboard of sounds with a synthesizer, users may have fun transcribing words into phoneme strings, which significantly reduces the cost of creating pronunciations compared to paying skilled linguists. In addition to reducing costs, creating pronunciations for languages for which linguists are rare or not available, is possible. Thus, pronunciation generation simply depends on finding enough willing users with a sufficient knowledge (fluent speakers) of the target language. The languages in our testing scenario are English and German. By providing a gaming experience, users voluntarily create pronunciations. Consequently, the pronunciation creation is reduced to automatically filtering the given content to get the best results.

The main goal of our study was to analyze if anonymous Internet users can create or enlarge pronunciation dictionaries with an online game. The resulting dictionaries should have the following qualities [Lem13]:

- Good quality compared to dictionaries created by linguists.
- Low cost in creation.
- Little to no postprocessing.

To successfully accomplish this task, the main requirements are a stable Internet connection and an application which prompts the user and records his or her decisions. Other requirements for the system itself concern usability, stability and performance of the user to achieve qualified pronunciations.

### 5.2.1 Related Approaches

To the best of our knowledge, no group analyzed crowdsourcing in the field of pronunciation generation [Lem13]. However, three projects had an impact on our work:

- Peekaboom [vALB06].
- Human Factors during Bootstrapping [DB04c].
- LexLearner [SBB<sup>+</sup>07, KB06].

#### Peekaboom

At Carnegie Mellon University researchers investigate how to harness the computational power of human brains by letting people play online games. They call them *Games with a Purpose* or *GWAP* [vA06]. Out of the different games on their website there was one, *Peekaboom* [vALB06], which influenced our work. This game focuses on collecting meta information for images by trying to determine which area of the image a keyword refers to. They use images from the Internet and associate keywords with them.

Two random players are paired up and are given a word-picture pair. One player is in the role of *Peek*, while the other is *Boom*. Only *Peek* sees the image and the associated word. Then, he or she reveals a part of the image so that *Boom* can guess the correct word. This setup is demonstrated in Figure 5.14.

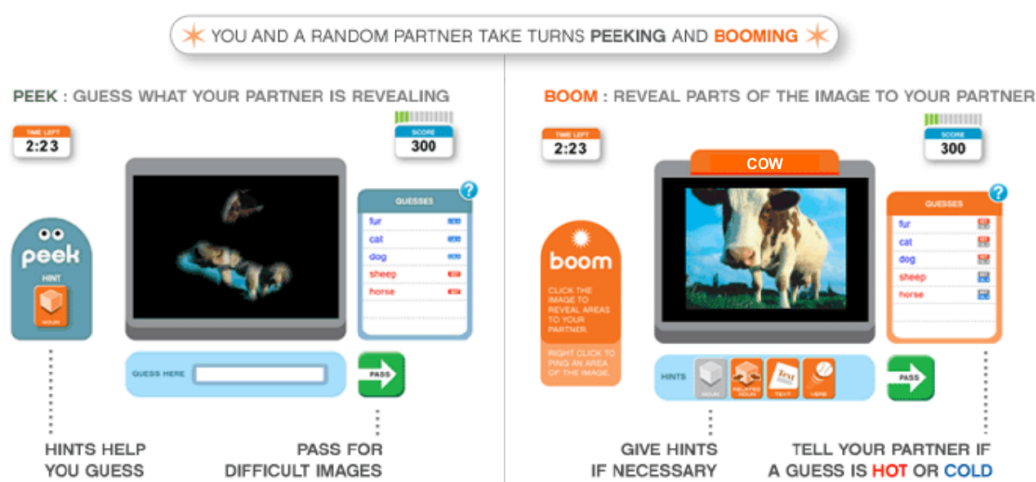


Figure 5.14 – Game Peekaboom [vALB06].

Points are awarded for the size of the uncovered area, whereby smaller is better. Moreover, the *Peek* gives several predefined hints, such as verb or noun, for which bonus points are awarded. This seems counterintuitive; but the purpose of the game is to collect meta information. Knowing if the keyword is a noun or a verb, is useful for the purpose of the game. After the word is guessed, their roles are reversed and a new image-word pair is given. Players have four and a half minutes to complete as many pairs as possible. Both players get the same amount of points at the end of one session, and then get paired with other players if they want to continue.

Apart from the fun of playing the game, players get on high score lists and can improve their standing there, which is a further incentive to keep playing. The amount of data which was collected reflects the success: In about 11 months (August 2005 to June 2006) nearly 30,000 users generated over two million pieces of data. On average, players spent about 72 minutes on the game in a single session.

To prove that their collected data was accurate and to show what could be done with it, the authors extracted the area in each image, referenced by the keyword. After a couple of players have used the same image-word pair, it was possible to make an accurate guess about the exact area the keyword was referring to. This area was calculated from the several areas revealed by different *Boom* players. The authors could then show that the collected data from the game was as good or even better in pinpointing the desired object than bounding boxes manually annotated by volunteers.

### Human Factors during Bootstrapping

[DB04c] evaluate the effect of the human factor in bootstrapping a pronunciation dictionary. It is analyzed if helpers with limited or no linguistic background using an semi-automatic toolkit can produce high-quality pronunciations in less time than it would take a trained expert to do using conventional methods.

For this purpose a system was designed, which presents the user with a written version of a word of the target language, a prediction for a pronunciation made by the system and the ability to play a sound for each phoneme of the prediction. These features speed up the process of creating a pronunciation dictionary considerably, compared to conventional methods. It is even possible for unskilled users to produce high quality dictionaries, rivaling those made by linguistic experts.

### LexLearner

A similar method to [DB04c] is the Lexicon Learner (LexLearner). The LexLearner is a part of our Rapid Language Adaption Toolkit, responsible for generating word pronunciations with the aid of users. The users do not have to be linguistic experts but need to be proficient in the target language.

In an initialization stage, rules to predict pronunciations are seeded by asking the user to match phonemes with their most commonly associated grapheme. These rules are used to present the user with a pronunciation prediction for a word. As suggested by [DB04c], these predictions are accompanied by a wavefile, which is generated through a phoneme-concatenation synthesizer. The user corrects the pronunciation and the system updates the rules accordingly. Further details are described in [KB06].

## 5.2.2 Design and Backend Implementation of Key-nounce

There should be little to no technological barriers imposed by the online game Keynounce [Lem13]. Therefore, we defined the following requirements to be fulfilled by Keynounce:

- Accessible on the World Wide Web.
- Usable for a majority of World Wide Web users.

- No additional software for users to install on their hardware.
- Small demands on the user's hardware.
- Graphical user interface (GUI).
- Ability to play sounds.
- Ability to quickly change interface depending on user input.
- Open source or freeware components to reduce costs.
- Reduced loading time during game-playing.
- Database interaction.

To meet the requirements of easy access for a majority of World Wide Web users without additional software installation and making small demands on the user's hardware, we relied on Java [Jav] and Flash [Fla] which are mainly used today to create small online games and graphical user interfaces, as well as dynamic commercials on the Internet. Both are widely used and most Internet users have everything required already installed on their machines.



Figure 5.15 – Keynounce user interface.

We decided to use Flash and Actionscript as it is an almost perfect fit to our needs. Flash was designed to bring moving pictures, sounds and interactivity to Web pages. It supports bidirectional streaming of audio content and supports user input by mouse and keyboard. The corresponding Flash Player is already installed on many personal computers and smartphones as a lot of motion picture content on the Internet is provided with Flash.

The chosen synthesizer was eSpeak [eSp] since it provides fast and easily adapted text synthesis and moderate quality of speech. The slightly robotic quality of speech should help the user to accept slight imperfections resulting from a lack of prosody and missing accentuation. As long as a correct combination of sounds is there, a user should accept the word as fine. A high quality synthesis would suggest to the user that the lack of prosody is his or her fault and he or she might try to remedy that.

We built a specific user interface, which is shown in Figure 5.15. The most important part of the user interface is the symbol keyboard. It is divided into two groups. The division into vowels and consonants is chosen in light of our premise of untrained users. As described in Section 1.4.1, in the IPA charts the different sounds are also classified as consonants and vowels. Although this is a good classification for the knowledgeable user, it would need special explanations to convey the concept to an uninformed user.

For this reason we decided to arrange the symbols according to their corresponding letters in the target languages [Lem13]. As a user without any background in linguistics has no further clue what the symbols stand for, the resemblance of the symbols to the letters of the Latin alphabet is helpful. Since one of the prerequisites for our interface was to be intuitive, we decided to use this resemblance as a clue how the symbols are arranged. We ordered the vowels by A-E-I-O-U and for the consonants we chose the normal alphabetical order of their associated letters. The user probably tries the most familiar symbols first. If the resulting audio does not meet the user's expectations, our explanation of how the symbols are arranged should lead him or her to try the symbols close to the first guess. The symbols in close proximity were arranged to represent variations of the same sound. The more distant two symbols are, the less similar their sounds. This order was established at a subjective level with a couple of students at our lab.

Preliminary testing showed that fine-tuning the pronunciation resulted in a cumbersome repetition of add - listen - delete. This is due to the fact that the users found an approximation quickly, but they were not satisfied with it. To get it almost right, mostly one or two symbols had to be replaced. However, to hit the appropriate nuances, a user sometimes has to listen to



the sound of different versions repeatedly. To alleviate this procedure, the SHIFT button was installed. This button can be toggled on screen or held on the physical keyboard. Clicking on a symbol while having activated SHIFT, results in the instant addition of that symbol to the string and playing of the resulting audio. The symbol itself is temporarily added to the string at the left side of the current cursor position. The symbols are not added permanently. Consequently, different symbols can be tried with the current string in rapid succession. As a bonus, the button also helps to get a feeling for the sounds. The vowel sounds can be played by themselves. In case of the consonants, a vowel has to be added first for the synthesizer to create a recognizable sound.

## Welcome to Keynounce



Figure 5.16 – Welcome screen in Keynounce.

After performing usability tests in our lab, we added a QUIT button to the interface. Some of the users intended to prematurely end a session and still get the results for the finished words. Consequently, they repeatedly hit either ACCEPT or SKIP. Below the option to quit the session, we installed a countdown showing the user how many words he or she has already finished and how many are still left in the session. This may incite a user to finish one or two more words since they are the final ones.

In a user's first interaction with the system a tutorial appears. For a returning user the tutorial is automatically hidden. But he or she can turn it on and refresh his or her memory about certain aspects. The tutorial gives hints on how to use the interface and the discussed part of the interface is highlighted

### Results

If you want to listen to your results or those of others, just **click on the words** in the table below. The reference words are only a possibility and sometimes not very good. Listen to the other results and form your own opinion which one is best.

default					
	your pronunciation	reference pronunciation	pronunciations by others		
			most often	second most	third most
leaflet	li:flɛt	li:flɛt	2x li:flɛt	1x li:flɪt	1x li:vlet
accuracy	ækju:ræsi:	ækjəəsi:	1x æku:reɪsɪ	1x ækɪu:reɪsɪ	1x ækju:ræsi:
kiss	kɪs	kɪs	4x kɪs	0x N.A.	0x N.A.
page	peɪdʒ	peɪdʒ	2x peɪdʒ	2x pɛtʃ	0x N.A.

### Feedback

You can help improve Keynounce by sharing your feedback with us in the form below.

Your native language:

Feedback

your email for answers:

### play again

[Back to the game](#)

**Figure 5.17** – Final screen in Keynounce.

with a red ring around it. To enlarge the potential userbase, we added the Facebook LIKE button and Google's 1+ button to connect with social networks. With these, our initial users had an easy way of propagating our experiment to their friends.

For a more complete feeling, we created a start and a final screen. As shown in Figure 5.16, the start screen allows a choice between the two available languages German and English and explains why this game was created. It also features a ranking system showing the names of participants and how many words they have submitted. This creates an incentive for the players since they either want to see their name in that top 10 list or are reminded of what effort other people put in. As demonstrated in Figure 5.17, the final screen lists the submitted strings of the current player. It also shows the reference pronunciation provided by the German *GlobalPhone* dictionary and the English CMU dictionary together with the top 3 submissions by other players. The top 3 are chosen by the simple way of counting the number

of matching strings for one word. The string submitted most often by all players is on the first position. All these strings can be listened to by the player. This provides the player with feedback to learn from.

### 5.2.3 Performance with Amazon Mechanical Turk

First, we set up a reduced version of Keynounce with Amazon’s Mechanical Turk (mTurk) platform [Lem13]. It is a quick way to reach a large userbase at affordable prices. The experiment on Amazon’s Mechanical Turk was set up on 10 May 2011. We used the rather limited requester website provided by Amazon. This was due to the fact that the more advanced forms to create Human Intelligent Tasks (HITs) were barely documented at that time. In a short document on the requester website, we explain the Turkers that their HIT is to transform two written words to their pronunciations. We also gave some hints how to complete the task. Then, they were required to click on a link opening another tab with the Keynounce website. This was necessary as Amazon had made it impossible to use external Web pages through the requesters’ website.

The Keynounce version used in this experiment lacked some of the gaming qualities. More specifically the following parts had been removed from the interface:

- Start Screen
- Tutorial
- Skip Button
- End Button
- Social Network Buttons

We removed these parts since HITs in mTurk are supposed to be short, clear and to the point. We had given all hints on the starting page of the HIT itself, which could be read by the Turkers before starting the task. The buttons to prematurely end the session or skip a word were removed because this was not an option we wanted to give the Turkers. Additionally, the final screen was modified to show a short “Thank You” message and an authentication code for later verification with mTurk. The Turkers were supposed to paste this code into an input field on the starting form of the HIT. With this authentication we were able to match the user in our database to the Turker

and could thus grant or reject the work. This was necessary to verify that the Turkers had actually done any work.

For evaluation we picked 100 random words from the English CMU dictionary and used the corresponding pronunciations in the CMU dictionary as reference. Our goal was to get 10 individual answers for each word and thus 1,000 phoneme strings in total. This number was chosen as it offered a good first impression on the work of the Turkers. We created 50 HITs with 2 words each. The Amazon services guaranteed that a single user could not work on two assignments of an identical HIT. This means that the users could finish any number of words between 2 and 100 without being able to work twice on the same word. Since we did not use the more advanced but more expensive API of the Mechanical Turk services, we had to manually approve the assignments. We did not simply approve every finished HIT since a large number of Turkers refused to do any work and just gave back the authentication code. This meant we had to supervise the results more closely than expected.

We initiated the experiment on May 10th and stopped on May 23rd with 387 approved and 531 rejected assignments. These assignments generated 1,902 words in total but 1,062 of these were spam. Comparing the mTurk experiment to the volunteer experiment, which we describe in the next Section 5.2.4, shows that there is a difference in the average time spent on completing the pronunciation for a word. The average time the Turkers spent on the Keynounce website was 53 seconds – not even half as long as the volunteers who took 113 seconds. The main incentive for the Turkers was getting paid. However, Keynounce was created in a way that people could use a trial and error system to find a near perfect solution. The majority of Turkers had no interest to spend a lot of time on it – at least not for the little amount of money that we offered.

In summary, the following three points made this line of mTurk experiments unsuitable for our purposes:

- Fast but sloppy (1,902 pronunciations within 5 days).
- High amount of spam (55% unusable input).
- No incentive to make an answer “the best answer”.

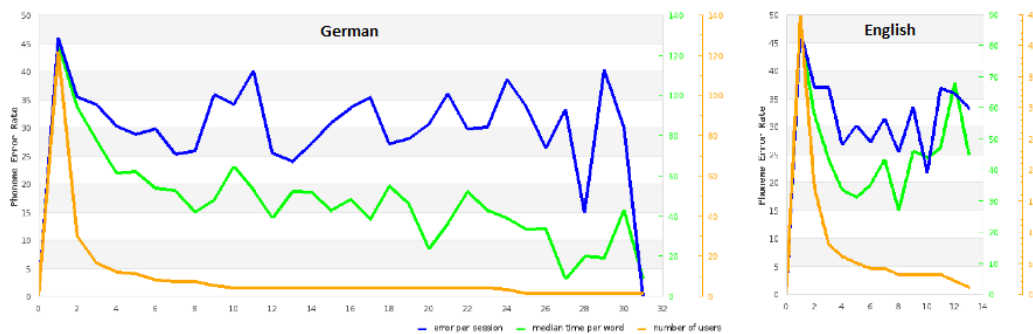
It would be possible to devise anti-spam procedures to clean the results. However, from our test subjects we expect to search for the best solution and not settle for a medium solution. In contrast, the Turkers try to finish as many tasks as possible in as little time as possible. This is why we aban-

done the mTurk approach and concentrated on voluntary participants for pronunciation generation.

### 5.2.4 Performance of the Keynounce Experiment

Since the mTurk experiment did not meet our expectations about the quality of the results, we decided to engage volunteers in our next experiment [Lem13]. To engage the users' interest, we used the full interface of Keynounce. Differing from the mTurk experiment, we decided to split the experiment and provide the choice between German and English words. This was motivated by the fact, that most of our users' first language is German. For the evaluation we picked 100 random words from the German *Global-Phone* dictionary and used their pronunciations as reference. We also added a choice of language to the starting screen. The language would then be set for a batch of words and could be changed after each batch. The batch size was five for the volunteers, in contrast to the two words in the mTurk experiment. This way users would be implicitly finish at least five words. To alleviate the task, we gave them the option to skip single words or abort a batch prematurely.

We asked colleagues and students and friends to help with this experiment. Additionally, we also encouraged everyone to tell other people about Keynounce. We let the experiment run generally unsupervised and interceded only once, when someone tried to submit one-phoneme words.



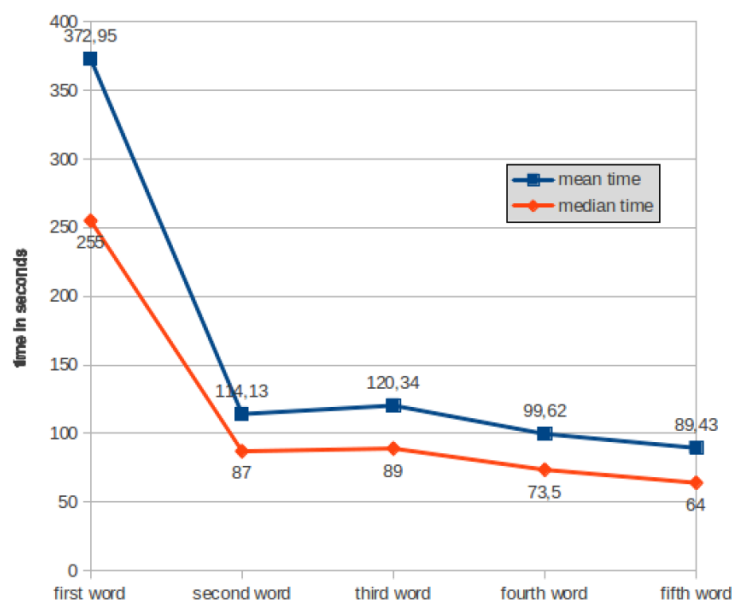
**Figure 5.18** – Phoneme error rate, time per word and number of users per session with German and English words.

We collected pronunciations for 1,595 words from 10/25/2011 to 04/17/2012. 89% of these pronunciations were collected within the first month. The retrieved pronunciations were evaluated with the phoneme error rate to our

reference pronunciations. As shown in the blue lineplot of Figure 5.18 of the German language results, the users learned quickly. After the initial session the phoneme error rate is 45.86%. For those users who looked at the results screen and then started a second session, the phoneme error rate drops to 35.33%. Until session 8 the phoneme error rate declines to 25.80%. At this point only the high incentive gamers are left. The phoneme error rate then becomes more erratic as there are so few users left. The green lineplot visualizes the median time it took all users to build a single word in the German language results. It starts with 125.4 seconds per word, which is almost 10.5 minutes for a session of five words. In the second session there is already a reduction of 30 seconds per word (94.4 seconds/word, 7.9 minutes/session). Users staying with the game after two sessions apparently became more familiar with the keyboard and what kind of sounds the symbols stand for. The time needed to produce the lowest phoneme error rate of 25.80% is 41.8 seconds per word, or just a little under 3.5 minutes per session. The pronunciation quality also gets more erratic with only a few more high incentive gamers playing, but in general keeps declining.

For the English users the same behavior can be observed in Figure 5.18 (blue lineplot). The phoneme error rate starts at 46.33% in the first session and is reduced to 35.34% in the second session. Then it drops to about 27% in session 4 and 6 and becomes increasingly erratic afterwards. For the English users the time per word also closely resembles the one for the German users. They start with 84.4 seconds per word, which is 7 minutes per session. The median time goes down to 57.5 seconds per word or 4.8 minutes per five words in the second session. The lowest phoneme error rate of 27% in sessions 4 and 6 are archived with less than 3 minutes or 33.4 seconds per word. This duration is comparable to [DB04c] who report 30 seconds for a manual generation of an Afrikaans word by developers with limited linguistic experience. In session 5, the users completed five words in a median time of 2.6 minutes or 31 seconds per word, but the phoneme error rate decreased slightly. With only the remaining high incentive gamers, it also turns erratic and the time even starts to increase with just four players remaining.

We have used the median time in Figure 5.18 to avoid outliers which result from some users who probably left the game open for a while and finished the session later. One user needed 3 hours to finish a session, another a little less than 2 hours. These outliers do not have an impact on the median, which results in a more reliable plot. The reason for the initial rise and subsequent decline of the phoneme error rate is probably a growing familiarity with the keyboard. Initially, most users do not know what each symbol stands for. After some trial and error they start to understand what they are supposed



**Figure 5.19** – Arithmetic mean and median time needed for single German words in first session.

to do plus how to do that with the given keys. Better understanding leads to faster results. This can be observed in the decline of time used per session in general. Additionally, we observe this behavior in the duration for each word in the first session, as illustrated in Figure 5.19. The figure shows the arithmetic mean (blue lineplot) and the median time (red lineplot) for each German word in first session of each user. On average 373 seconds (over 6 minutes) are needed to find the pronunciation for the first word, followed by 114 and 120 seconds for the next two words. The last two words are then found in 99 and 89 seconds. The median is lower: A user needs 255 seconds or 4.25 minutes to complete the first word. The following two words are found in about 89 seconds. For the last two words, it takes only 73.5 and 64 seconds. Unfortunately, the data for the first session do not include all participants of the study. Some participants had their computers configured to delete cookies which made it impossible to track, which words were generated in which session. As mentioned above, the median time needed for the last words in the first session is below the median for the second sessions. This can be explained as follows: After the first session, the users can evaluate their own solutions and compare them against the reference pronunciations and the three most frequent solutions of all other users for this word. Those users continuing with a second session afterwards, are, on the one hand, more familiar with the keyboard. On the other hand, they might have seen on the

result screen that better solutions are possible, which might lead to a little more time spent on the fine-tuning.

### 5.2.5 Improving Pronunciation Quality

To further improve the pronunciation quality, we applied our phoneme-level combination scheme (*PLC*), which we have presented in Section 4.3. We only evaluated the German results since we did not gather enough English data to present valid information.

PER all	PER nBest	#users
34.7%	22.7%	122

**Table 5.7** – Phoneme error rate (%) over all words from German players in different word ranges.

As illustrated in Table 5.7, all gathered hypotheses have a phoneme error rate of 34.7%, compared to our *GlobalPhone* reference pronunciations. After applying *PLC*, hypotheses remain for each word which have a phoneme error rate of only 22.7%, which is a relative improvement of 34.6%.

We have shown that it takes users a lot more time to complete the first session than any other session. Table 5.8 shows the initial phoneme error rate and the phoneme error rate (PER) after *PLC* for the 92 users who have submitted five words or less. This group archives a phoneme error rate of 42.7% and a phoneme error rate of 37.6% after *PLC*. The second line shows the 45 users having generated pronunciations for four words or less. They have either abandoned the game or skipped words. The achieved phoneme error rate for this group is 54.5% and about the same when calculated after *PLC*. This already shows that those users not having enough incentive to finish the first session properly, do generally not contribute valuable output. The phoneme error rate significantly decreases for those users who have done a second session or at least six but no more than ten words. These 15 users produced a phoneme error rate of 34.3%. The result after *PLC* is 4.5% absolute lower. This means, even though there are fewer hypotheses to work with, there are more correct phonemes in common for *PLC*.

The last line in Table 5.8 shows the overall phoneme error rate for users who did any amount of words within the first two session. The phoneme error rate is 40.6% and the lowest phoneme error rate after *PLC* is 29.4%.



PER all	PER after <i>PLC</i>	users, who processed	#users
42.7%	37.6%	5 or less words	92
54.5%	54.6%	4 or less words	45
34.3%	29.8%	6 to 10 words	15
40.6%	<b>29.4%</b>	10 or less words	107

**Table 5.8** – Phoneme error rate (%) over all words from German players who worked on one or two sessions (5-10 words).

PER all	PER after <i>PLC</i>	users, who processed	#users
31.6%	21.6%	10 or more words	25
31.2%	20.9%	15 or more words	14
31.0%	<b>20.8%</b>	20 or more words	12
32.6%	25.4%	100 or more words	4

**Table 5.9** – Phoneme error rate (%) over all words from German players who worked on more than two sessions (5-10 words).

After finishing the first two sessions the gamers slightly reduce the phoneme error rate but show a significant decrease in the phoneme error rate after *PLC*, as illustrated in Table 5.9. The pronunciations of those who did more than two sessions have a phoneme error rate of 31.6%, which is 2.7% absolute better than those of the users stopping after the second session. The phoneme error rate after *PLC* drops to 21.6%. Users doing a fourth or fifth session still have a slight decrease in both phoneme error rate and phoneme error rate after *PLC* to a minimum of 31% and 20.8%.

The four high incentive gamers who did more than 100 words produced pronunciations which slightly increase the phoneme error rate with 1.6% and even increased it with 4.4% after *PLC*. The fact that the phoneme error rate does not further drop with training might imply that just showing the final screen with the results has its limits. Users looking often enough at the final screen might not be as interested in the results anymore. In future work, it might be a goal to find some further means to let users improve themselves, even after they have gotten into some kind of routine.

### 5.2.6 Summary

Given the right kind of incentive in providing a game-like experience, anonymous users can help to generate pronunciations for free. We used the human ability to compare synthesized speech with what the user knows to be a cor-

rect articulation. As the human brain is able to decide relatively accurately, if slight differences are based on the synthesizer or the chosen pronunciation, most users can create good results.

Using the power of crowdsourcing for our benefit, we were able to produce pronunciations out of the sum of user outputs which are as good or even better than every single user output itself. Due to the fact that crowdsourcers' answers tend to agree on a large percentage of the answer, the difficult parts of the pronunciation can be identified. If there is enough input, it may even be possible to calculate two or more pronunciation variants due to dialect or regional differences.

Our project also showed that it may be easier to get people to help when contacted via a micropayment system like Amazon's Mechanical Turk. However, our experiment demonstrated that micropayment workers in general are not overly fond of investing a lot of time in fine-tuning answers. Our simple approach of just posting the problem and getting the answers was by far more effective without any monetary incentive.

## 5.3 Semi-Automatic Pronunciation Generation

Automatic methods to learn from seen data and to provide possible pronunciations are helpful to minimize potential errors as well as the editing effort in a human- or crowdsourcing-based pronunciation generation process. Therefore, we developed efficient methods which contribute to a rapid and economic semi-automatic pronunciation dictionary development [Mer14].

### 5.3.1 Traditional Methods

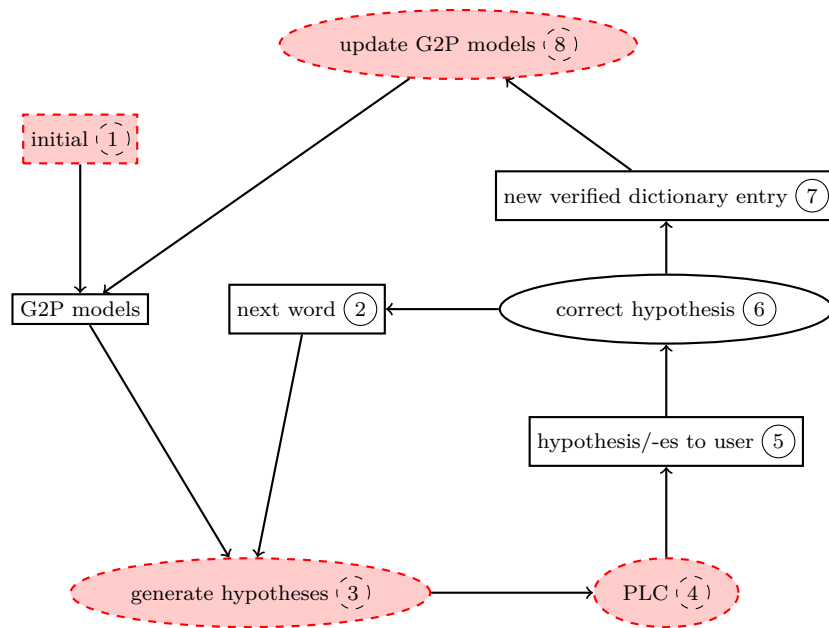
In semi-automatic dictionary generation processes like [MBT04], [DM09], and [SBB<sup>+</sup>07] native speakers enter pronunciations as phoneme strings. To reduce the difficulty of pronunciation generation, the user can listen to a synthesized wavefile of the entered pronunciation. Like [SBB<sup>+</sup>07], we present a list of available phonemes to the users, automatically reject pronunciations containing invalid phoneme labels and enable the user to listen to a synthesized wavefile of the pronunciation.

[MBT04] and [DM09] display the words according to their occurrence frequencies in a text corpus. By covering common words early, word error rates in automatic speech recognition are minimized. [Kom06] and [KB06] order the words according to their grapheme coverage to learn many grapheme-to-phoneme relations early. [DB04c] and [DB04b] prefer short words over longer words to alleviate the correction effort for human editors. We follow the principles of [Kom06] and [KB06] and additionally prefer short words over longer words like [DB04c] and [DB04b]. While [DM09] use a phoneme set defined by linguists, [Kom06] infers a phoneme set in an automatic way: An initial phoneme recognizer is trained on a grapheme-based dictionary. Based on audio recordings and transcriptions, acoustic model units are adapted, based on Merge&Split. In [SBB<sup>+</sup>07] as well as in our approach no additional audio recordings are required because users manually specify the phonemes from an International Phoneme Alphabet chart, guided by audio recordings of each phone.

Some approaches update the grapheme-to-phoneme model after each word [DM09, SBB<sup>+</sup>07]. Others combine incremental updates and periodic rebuilding [Kom06, DB05]. In [MBT04] and [DFG<sup>+</sup>05], the user decides the creation of new grapheme-to-phoneme models. [DB05] introduce a data-adaptive strategy, updating the grapheme-to-phoneme model after the pronunciations of 50 words needed to be corrected. While [DM09] start with an empty dictionary, [MBT04] manually generate pronunciations for the most frequent 200–500 words in a text corpus. [SBB<sup>+</sup>07] initializes the pronunciation generation with a *1:1 G2P mapping* entered by the user. [Kom06] records 20 minutes of speech and builds an initial dictionary automatically based on the grapheme-based phoneme set, acoustic information and their transcriptions. Since Web-derived pronunciations proved to be helpful for the dictionary generation process [GJK<sup>+</sup>09, SOS10], we used them to obtain initial training data. While conventional approaches use only one grapheme-to-phoneme converter, we use multiple grapheme-to-phoneme converters with similar performances and combine their outputs [SQS14].

### 5.3.2 Semi-Automatic Pronunciation Generation Strategy

Figure 5.20 illustrates our pronunciation generation strategy. The components which deviate from state-of-the-art methods are highlighted. Starting with an empty dictionary, our process consists of the following steps:



**Figure 5.20** – Semi-automatic pronunciation generation.

1. Initial word-pronunciation pairs are used to train an initial grapheme-to-phoneme model (1).
2. A next word is determined to generate a pronunciation for (2).
3. Each grapheme-to-phoneme converter generates a hypothesis for the pronunciation of that word (3).
4. The hypotheses are combined at phoneme level (4), which produces one hypothesis to be presented to the user.
5. Optionally, the 1st-best hypotheses of the each grapheme-to-phoneme converter are additionally offered to the user (5).
6. The user edits the best-matching hypothesis to the correct pronunciation for the requested word (6).
7. Word and corrected pronunciation are added to the dictionary (7).
8. After a certain number of words (8), the grapheme-to-phoneme converters update their grapheme-to-phoneme models based on the word-pronunciation pairs in the dictionary.

As it is expensive to assess real human editing times, we simulate the annotation process assuming the editor changes the displayed phoneme sequence to the phoneme sequence of the reference pronunciation. To measure the hu-

man editing effort, we introduce the cumulated phoneme error rate (cPER) as follows:

$$cPER(n) := \frac{\sum_{i=1}^n sub(w_i) + ins(w_i) + del(w_i)}{\sum_{i=1}^n phonemes(w_i)} \quad (5.1)$$

We compute the cPER from the beginning of the editing process to a current word  $w_n$  as follows: We accumulate the number of edits (substitution (*sub*), insertion (*ins*) or deletion (*del*) of single phonemes), a developer would have done in each processed word  $w_i$ , up to the current word  $w_n$  to reach the pronunciations of our reference dictionaries and set these edits in relation to the total number of phonemes seen in the dictionary. This value is the counterpart to the *phoneme correctness* in [DB08]. As the values contain the initialization phase with bad hypotheses, reading these numbers as phoneme error rate, which reflects only the editing effort for the current word  $w_n$  would be misleading. According to [DB04c], we assume a human dictionary developer to take 3.9 seconds on average for an edit to a predicted hypothesis.

As our methods should work for languages with different grade of regularity in grapheme-to-phoneme relationship, our experiments are conducted with German (*de*), English (*en*), Spanish (*es*), Vietnamese [VS09] (*vi*), Swahili (*sw*), and Haitian Creole (*ht*). For evaluating our methods, we use *Global-Phone* dictionaries [SS14] for *de*, *es* and *sw* as reference. The *en* dictionary is based on the CMU dictionary. For *ht*, we employ a dictionary also developed at Carnegie Mellon University. We created six random excerpts of 10k words from each dictionary to conduct all experiments in a 6-fold cross-validation, except for *vi*, as the Vietnamese dictionary contains only 6.5k word-pronunciation pairs.

### 5.3.3 Word Selection Strategy

Based on [Kom06] and [KB06], our word list is sorted by graphemic 3-gram coverage and based on [DB04b] with a preference for short words (*ngram sorted*) to speed up the annotation process.

Figure 5.21 shows that our proposed strategy outperforms a *random* order in cPER for English dictionary extracts (evaluated on 10k English word-pronunciation pairs with Phonetisaurus). Like [KB06], we plot an *alphabetical* selection for comparison. The impact of *ngram sorted* is higher in

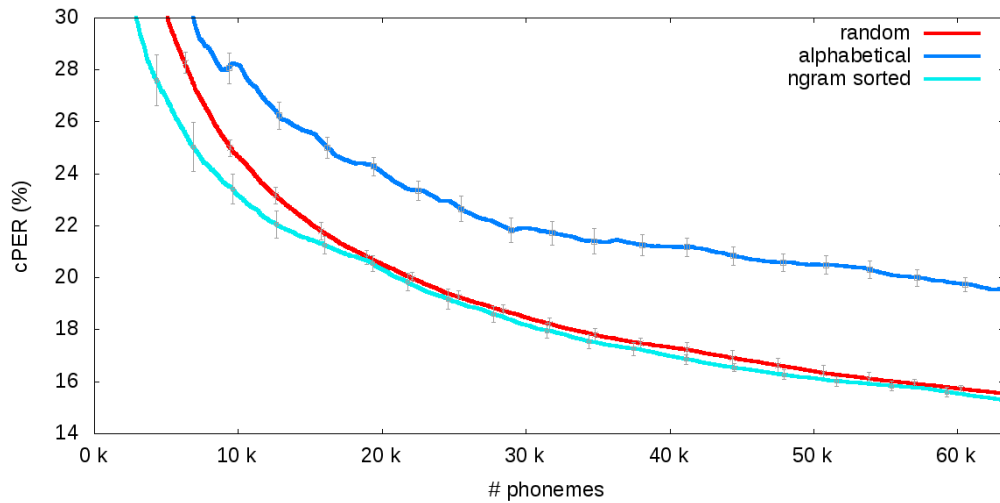


Figure 5.21 – Word selection strategies.

the beginning of the process when less training data for the grapheme-to-phoneme models are given. In all three curves we updated the grapheme-to-phoneme model according to logistic growing intervals, which we describe in Section 5.3.4. *ngram sorted* outperforms *random* and *alphabetical* for the other languages as well.

### 5.3.4 Iterative G2P Model Building

If the grapheme-to-phoneme model is updated after a new pronunciation is created by the user, no knowledge about the grapheme-to-phoneme relationship is skipped. In such a case, the model is always up-to-date and produces a pronunciation of optimal quality for the next word which results in lowest post-editing effort by the user to reach a qualified pronunciation. However, frequent grapheme-to-phoneme model generation results in a notable increase in CPU time. For example, the slowest grapheme-to-phoneme converters in our selection take approximately one hour for a grapheme-to-phoneme model re-training pass of 10k *en* word-pronunciation pairs on a computer equipped with a 2.6GHz AMD Opteron processor. Since parallel or incremental grapheme-to-phoneme model training is not possible for some of the methods, our goal is to train grapheme-to-phoneme models more frequently in the beginning, when it does not take much time and grapheme-to-phoneme model quality still increases rapidly with more training data. Consequently, [DB05] propose a data-adaptive training interval (*Adaptive*).

In a first phase they re-train their grapheme-to-phoneme model after each added word. When the dictionary reaches a size of 1,500 words, they switch to the second phase, where the grapheme-to-phoneme model is re-created after 50 predicted pronunciations needed corrections.

We compared *Adaptive* to a re-training at a fixed number of new dictionary entries (*Fixed*) and linearly growing intervals (*Linear*) with different parameter values. 10% dictionary growth in terms of new dictionary entries proved to be a sensible value for *Linear* with better results in less time than *Fixed*. However, *Linear* exhibits the disadvantage of a boundless increase of the training intervals for large dictionaries. To ensure a maximum size for the interval, we propose a logistic growth function (*Logistic*). This function starts with training interval 1 after word 1 and enables a smooth increase from 0 to 10k words, where we observed a notable slowdown in grapheme-to-phoneme model improvement even for the languages with a high complexity in the grapheme-to-phoneme relation. In our case we limit the maximum training interval to 3k words.

Interval	# edits	Editing time	CPU time	Total time
Logistic	22,983	89,634 s	2,055 s	<b>91,689 s</b>
Linear	22,657	88,362 s	3,282 s	<b>91,644 s</b>
Adaptive	22,530	87,867 s	28,928 s	116,795 s
Fixed	23,658	92,266 s	17,529 s	109,795 s

**Table 5.10** – Strategies for grapheme-to-phoneme model retraining.

Table 5.10 shows the raw editing time it would take a human supported by Phonetisaurus to generate the 10k dictionaries of all six languages (*de*, *en*, *es*, *vi*, *sw*, *ht*) plus the CPU time consumed (average in the 6-fold cross-validation without parallelization on a computer equipped with a 2.6GHz AMD Opteron processor and 32GByte RAM). Since Phonetisaurus is by far the fastest of our grapheme-to-phoneme converters, the training interval is more crucial for the other converters. Even though *Logistic* only consumes 60% of the CPU time of *Linear* and 7% of the CPU time of *Adaptive*, the editing effort results are comparable to the best-performing *Adaptive* and *Linear*. Thus, we decided to continue our experiments with *Logistic*. In a real human scenario, the user can additionally start the grapheme-to-phoneme model generation process manually before an extended break.

### 5.3.5 Single G2P Converters

We use four grapheme-to-phoneme converters for our experiments: Sequitur G2P [BN08], Phonetisaurus [Nov11, NMH12], Default&Refine [Dav05, DB04a, DB08] and CART trees [Len97]. For all grapheme-to-phoneme converters, we use context and tuning parameters resulting in an optimal tradeoff between performance and CPU time for the grapheme-to-phoneme model training of all six tested languages.

#### Sequitur G2P

We investigated the considered graphemic context width by varying the number of iterative “ramp-up” steps between 2 and 9. In contrast to Phonetisaurus, where the context widening from 1 to 10 increases computation time by a factor of 4.4 for the English dictionary, the added model iteration steps for Sequitur G2P have a greater CPU impact: The computation time increases by a factor of 15 between model-2 and model-9.

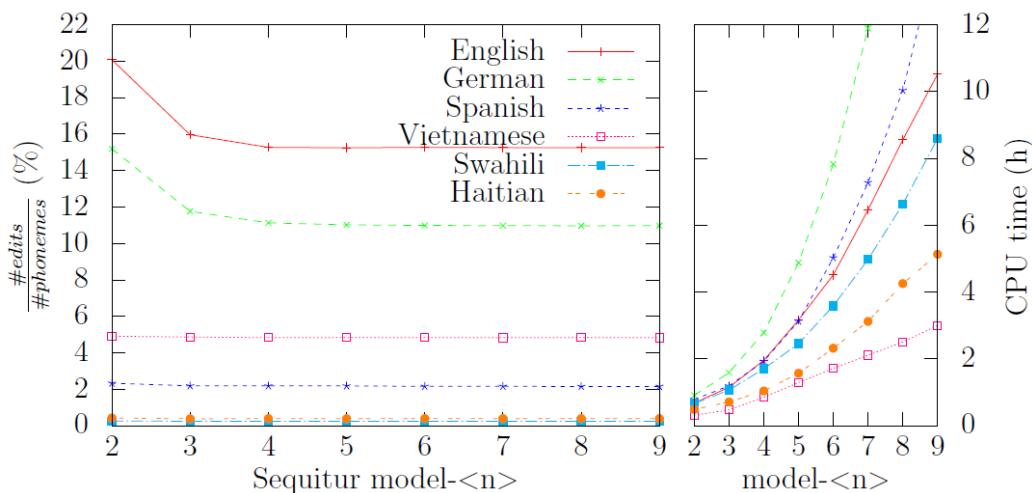
Table 5.11 illustrates the human editing effort with varying “ramp-up iterations”. The minimum number of human edits is reached for model-8.

Model	en	de	es	sw	ht	$\sum edits$	
2	12,719	13,853	1,925	935	240	258	29,930
3	10,114	10,728	1,806	926	<b>225</b>	<b>231</b>	24,030
4	9,678	10,150	1,803	920	227	238	23,016
5	<b>9,664</b>	10,053	1,805	922	227	236	22,905
6	9,676	10,023	1,786	920	228	242	22,874
7	9,667	10,013	1,784	<b>919</b>	226	237	22,845
8	9,667	<b>10,004</b>	<b>1,779</b>	920	229	240	<b>22,839</b>
9	9,667	10,011	1,780	919	229	238	22,843

**Table 5.11** – Sequitur context width optimization: Total number of edits.

But comparing the cumulated PER and CPU time in Figure 5.22 and Table 5.12 shows that CPU time dominates the total dictionary generation time starting with model-6. Following the total estimated times, model-2 would be an optimal choice. Since the number of edits stays nearly constant from model-5 on, we decided to proceed with training model-5 for our evaluation runs.





**Figure 5.22** – Cumulated phoneme error rate (%) over Sequitur n-gram order  $M$ .

Model	$\sum edits$	editing time (h)	CPU time (h)	total time (h)
2	29,930	33.3	<b>3.8</b>	37.1
3	24,030	26.7	6.2	<b>32.9</b>
4	23,016	25.6	10.3	35.8
5	22,905	25.5	16.5	41.9
6	22,874	25.4	25.0	50.4
7	22,845	25.4	35.8	61.2
8	<b>22,839</b>	<b>25.4</b>	48.6	74.0
9	22,843	25.4	62.6	88.0

**Table 5.12** – Sequitur context width: Time estimations on 10k dictionaries.

## Phonetisaurus

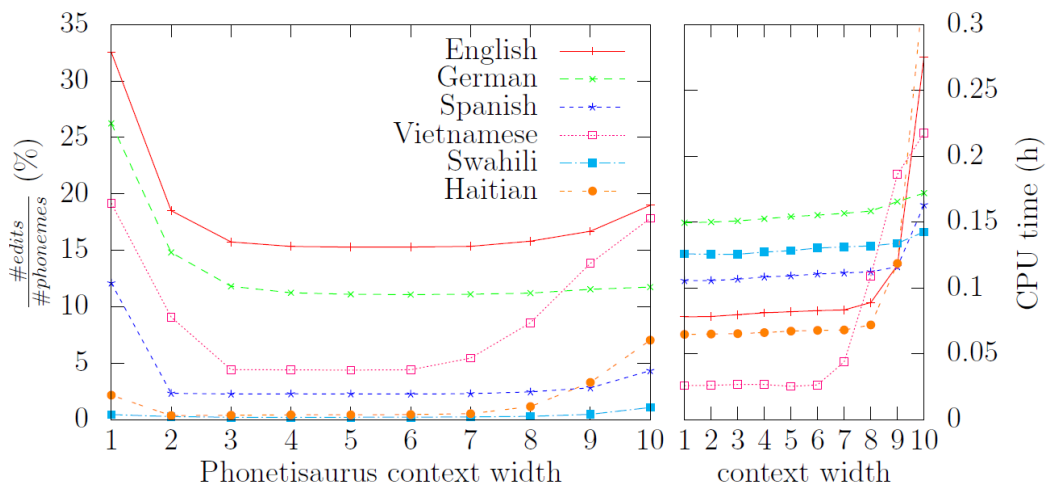
The implementation of Phonetisaurus [Nov11, NMH12] relies on the OpenFst library [ARS<sup>+</sup>07] to build a Weighted Finite-State Transducer (WFST) based graphemeto-phoneme conversion framework. It makes use of a modified EM algorithm and joint-sequence language models. During training, only one-to-many and many-to-one transformations are considered. According to [NMH12], this leads to small but consistent improvements in word accuracy in grapheme-to-phoneme conversion tasks.

The pronunciation model is implemented as a joint n-gram model. The input sequence of graphemes and phonemes is converted to aligned joint label pairs

using the MIT Language Modeling Toolkit [HG08]. From these aligned pairs, an n-gram model is trained, which is then converted to a WFST.

For pronunciation generation, the input word is combined with the model. Then a projection to the output symbols is done and the shortest path through this lattice is extracted [NMH12]. In cases where multiple hypotheses are requested via the nbest option, the hypotheses along the requested number of shortest paths are extracted.

To tune the quality of the generated hypotheses, we experimented with the context width in the n-gram estimation step and measured the resulting sum of human edits required to reach the reference pronunciations. The total number of edits across all languages reached an optimum, before the CPU time changed notably, as shown in Figure 5.23 and Table 5.13. Thus, we stayed with the minimum number of edits at a context width of 5. The other parameters were kept at the default values recommended in the documentation.



**Figure 5.23** – Cumulated phoneme error rate (%) over Phonetisaurus context width.

### Default & Refine

The Default&Refine algorithm [Dav05, DB04a, DB08] searches a default for each grapheme: One phoneme which is the most common for that grapheme. Based on this information, pronunciations diverging from the default rule are modeled as additional rules in a rule chain. This approach is suited better,

Context	en	de	es	sw	ht	$\sum edits$	
1	20,624	23,936	9,982	1,730	407	1,311	57,989
2	11,729	13,510	1,930	843	249	<b>218</b>	28,478
3	9,972	10,775	<b>1,876</b>	840	<b>193</b>	237	23,894
4	9,721	10,254	1,886	<b>836</b>	<b>193</b>	261	23,151
5	<b>9,686</b>	10,126	1,879	841	194	258	<b>22,984</b>
6	9,689	<b>10,111</b>	1,877	1,037	204	273	23,190
7	9,727	10,138	1,897	1,632	220	320	23,935
8	10,022	10,226	2,027	2,633	263	699	25,869
9	10,574	10,530	2,331	3,391	432	1,994	29,251
10	12,028	10,719	3,584	3,648	982	4,257	35,218

**Table 5.13** – Phonetisaurus context width: Total number of edits on 10k dictionaries

the more regular the grapheme-to-phoneme relationship of a language is. In less regular languages, the number of exception-rules increases fast.

First a Viterbi grapheme-to-phoneme alignment [DB04b] is computed. Then for each grapheme, a set of pronunciation rules is created based on left and right grapheme context. Rules are added iteratively, building a chain of rules with increasing specialization. The considered grapheme context is expanded until a rule can be found which maps the pronunciation exactly.

When generating pronunciations, this chain is traversed backwards: First the special rules with a large grapheme context are evaluated, resorting to more general rules when no exact grapheme context match is found. In contrast to statistical methods, the user can rely on correct recall of all pronunciations from the training set since every special case is mapped to a specialized rule [DM09].

The current implementation of the Default&Refine algorithm does not support the generation of multiple output hypotheses (n-best). Furthermore, it supports only one-character phoneme names. To circumvent the latter limitation for our existing dictionaries, we implemented a phoneme translation layer in our module interface. All phonemes in the universal input are internally mapped to one-letter phonemes for Default & Refine. Each predicted hypothesis is then mapped back to the original phoneme names before further processing.

## CART trees

*t2p: Text-to-Phoneme Converter Builder* [Len97] implements a grapheme-to-phoneme conversion approach based on Classification and Regression Trees (CART) [BFOS84, BLP98]. The authors recommend to generate a set of allowable pronunciations for each grapheme or multi-grapheme cluster manually. Based on this table, an alignment of high quality between graphemes and phonemes can be achieved. In our strategy, this is not possible in every case since we do not require a-priori knowledge about the target language – although we provide options to integrate such knowledge for our initial pronunciations. In such cases, the authors use the so-called epsilon scattering method, an application of the EM algorithm [DLR77]. In [BLP98], this leads to a slightly larger phoneme error rate than the manually generated set.

From this alignment, a CART decision tree is trained for each grapheme given its context. A fixed context size of 3 left and 3 right is used in this implementation. Each grapheme can predict epsilon or one or two phonemes. Building separate trees is reported to be faster and allows parallelization without significant difference in accuracy [BLP98]. The generation of multiple output hypotheses (n-best) is currently not supported.

The current implementation does not support Unicode graphemes. However, we added Unicode support to the program code in a simple way. The evaluations for all languages were done with this extended version.

### 5.3.6 Combination of G2P Converter Output

Table 5.14 lists the editing effort in terms of cPER to generate pronunciations for 10k words with *Logistic*. *Sequitur* performs best in three of six cases, closely followed by *Phonetisaurus*. *Phonetisaurus*, *Default&Refine* (*D&R*) and *CART tree* perform best in one case each. While *D&R* and *CART tree* seem to be better suited for languages with a regular grapheme-to-phoneme relation, the statistical approach with smoothing seems to be better for languages with less regular pronunciations. The best single grapheme-to-phoneme converter for each language provides our baseline to which we compare all improvements. As we observed for the phoneme-level combination (*PLC*) [SQS14] that the order of pronunciations is of great importance for the results, we ordered the 1st-best hypotheses according to the average performance of the different grapheme-to-phoneme converters in our baseline scenario: *Sequitur*, *Phonetisaurus*, *D&R*, *CART tree*. As demonstrated in Table 5.14, *PLC* leads to a statistically significant reduction in cPERs ( $\Delta PLC$ )

for all languages between 1.9% and 38.1% relative. The improvements are considerably higher in terms of lower cPERs.

	en	de	vi	es	ht	sw
Sequitur	<b>15.24</b>	<b>11.02</b>	4.83	<b>2.19</b>	0.39	0.25
Phonetis.	15.28	11.10	<b>4.42</b>	2.28	0.43	<b>0.21</b>
D&R	16.80	12.85	5.12	2.23	0.42	<b>0.21</b>
CART	20.01	13.89	5.20	2.56	<b>0.36</b>	0.26
<b>PLC</b>	14.47	10.81	3.78	2.00	0.28	0.13
<b>ΔPLC</b>	5.05	1.91	14.48	8.68	22.22	38.10

**Table 5.14** – Cumulated phoneme error rates (cPERs) (%) for single grapheme-to-phoneme converters.

### 5.3.7 Resources for Initial Pronunciations

In addition to the traditional substitution of graphemes with the most commonly associated phonemes (*1:1 G2P Mapping*), we show that grapheme-to-phoneme models from Web data and from other languages can help to reduce the initial human editing effort.

#### 1:1 G2P Mapping

As in [SBB<sup>+</sup>07], we created initial pronunciations with *1:1 G2P Mapping*. This mapping can be compiled by a native speaker but also derived from existing word-pronunciation pairs, e.g. from the Web. How close the pronunciations with the *1:1 G2P Mapping* come to our validated reference pronunciations in terms of PER is illustrated in Table 5.15.

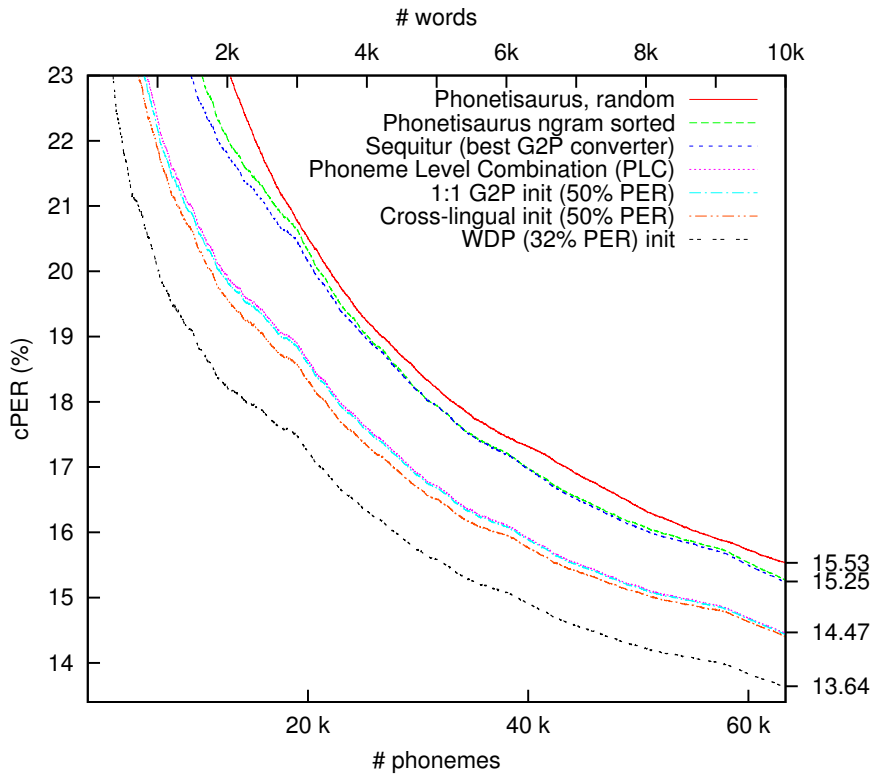
#### Web-driven G2P Converters’ Output

Since Web-derived pronunciations proved to support the dictionary generation process [GJK<sup>+</sup>09, SOS10, SQS14, SOS14], we investigated if they can be used to obtain initial training data for our grapheme-to-phoneme converters and outperform the conventional *1:1 G2P Mapping*. For our analysis we used *Sequitur* to build additional grapheme-to-phoneme converters for *en*, *de* and *es* with word-pronunciation pairs from *Wiktionary*. How the Web-derived pronunciations approach our reference pronunciations in terms of PER is illustrated in Table 5.15. Including the Web-derived pronunciations in the *PLC* benefited for the first 210 *en* words, the first 230 *es* words, and even the first

4,750 *de* words (*Quality cross-over*) since the word-pronunciation pairs from *Wiktionary* reflect a very consistent quality as shown in Figure 5.9. Instead of omitting them, we gained more cPER reduction by putting the Web-derived pronunciations from the first to the last position in the *PLC* after the average cross-over of all tested languages at 500 words. As demonstrated in Table 5.17, using the Web-driven grapheme-to-phoneme converters' output to reduce the initial effort performed better than the *1:1 G2P Mapping* with a relative improvement in cPER of 6% compared to the *PLC*. Applying our automatic filter methods which had further improved the quality of Web-driven grapheme-to-phoneme converters in [SOS12b] and [SOS12a] did not lower the cPER. The reason is that the filtering skips irregular pronunciations from the output, which have supplied valuable additional information to the *PLC*.

### Cross-lingual Pronunciations

In Chapter 6 we will show that using grapheme-to-phoneme models derived from existing dictionaries of other languages can severely reduce the necessary manual effort in the dictionary production, even more than with the *1:1 G2P Mapping*. According to the cross-lingual dictionary generation strategy in [SVYS13], we (1) mapped the target language graphemes to the graphemes of the related language, (2) applied a grapheme-to-phoneme model of the related language to the mapped target language words, and (3) mapped the resulting phonemes of the related language to the target language phonemes. With this strategy, we generated *en* pronunciations with a *de* grapheme-to-phoneme model and *de* pronunciations with an *en* grapheme-to-phoneme model. How close the *cross-lingual* pronunciations come to our reference pronunciations in terms of PER is illustrated in Table 5.15. Including the *cross-lingual* pronunciations in the *PLC* with the single grapheme-to-phoneme converter outputs helped slightly for the first 42 *en* words and the first 52 *de* words (*Quality cross-over*). Therefore, in our strategy we use those pronunciations in the first place in the *PLC* up to the average cross-over of all tested languages at 45 words and omit them afterwards. While we observe a small relative cPER reduction of 0.34% on top of the *PLC* for *en*, we obtain a relative increase of 0.56% for *de* as shown in Table 5.17. However, Figure 5.24 demonstrates that *cross-lingual* outperforms *1:1 G2P Mapping* in the beginning of the process, when less training data for the grapheme-to-phoneme models are available. In Chapter 6 we will discuss in more detail the challenge and potential of pronunciation generation across languages.



**Figure 5.24** – Comparison of grapheme-to-phoneme converter strategies on English.

	en	de	es	vi	sw	ht
PER 1:1	50.01	37.83	14.20	40.49	10.52	14.92
Quality cross-over	100	170	360	80	230	120
PER Wikt	32.55	13.47	11.40			
Quality cross-over	210	4,750	230			
PER x-lingual	50.42	46.64				
Quality cross-over	42	52				

**Table 5.15** – Phoneme error rate (%) and optimal cross-over for initial pronunciations.

Including the pronunciations generated with the *1:1 G2P Mapping* in the *PLC* with the single grapheme-to-phoneme converter outputs helps to reduce the cPER for the first 100 *en* words, the first 170 *de* words, the first 360 *es* words, the first 80 *vi* words, the first 230 *sw* words, and the first 120 *ht* words (*Quality cross-over*). Using the pronunciations from the *1:1 G2P Mapping* after these cross-overs reduces the pronunciation quality in the *PLC*. There-

	en	de	es	vi	sw	ht	average
Best single G2P	9.7k	10.0k	1.8k	841	191	220	
PLC	9.2k	9.9k	1.6k	718	118	168	
Relative to single	5.05 <sup>s</sup>	1.91 <sup>s</sup>	8.68 <sup>s</sup>	14.48 <sup>s</sup>	38.10 <sup>s</sup>	22.22 <sup>s</sup>	+15.07
1:1 G2P mapping + PLC	9.2k	9.9k	1.6k	710	106	155	
Relative to PLC	0.14	0.00	1.50	1.32 <sup>s</sup>	7.69 <sup>s</sup>	7.14 <sup>s</sup>	+2.97
WDP + PLC	8.6k	9.3k	1.5k				
Relative to PLC	5.74 <sup>s</sup>	5.46 <sup>s</sup>	6.50 <sup>s</sup>				+5.90
Cross-lingual + PLC	9.1k	9.9k					
Relative to PLC	0.34	-0.56					-0.11

**Table 5.16** – Reductions in total number of human edits.

	en	de	es	vi	sw	ht	average
Best single G2P	15.24	11.02	2.19	4.42	0.21	0.36	
PLC	14.47	10.81	2.00	3.78	0.13	0.28	
Relative to single	5.05 <sup>s</sup>	1.91 <sup>s</sup>	8.68 <sup>s</sup>	14.48 <sup>s</sup>	38.10 <sup>s</sup>	22.22 <sup>s</sup>	+15.07
1:1 G2P mapping + PLC	14.45	10.81	1.97	3.73	0.12	0.26	
Relative to PLC	0.14	0.00	1.50	1.32 <sup>s</sup>	7.69 <sup>s</sup>	7.14 <sup>s</sup>	+2.97
WDP + PLC	13.64	10.22	1.87				
Relative to PLC	5.74 <sup>s</sup>	5.46 <sup>s</sup>	6.50 <sup>s</sup>				+5.90
Cross-lingual + PLC	14.42	10.87					
Relative to PLC	0.34	-0.56					-0.11

**Table 5.17** – Reductions in cumulated phoneme error rate (cPER) (%).

fore, in our strategy we use the pronunciations from the *1:1 G2P Mapping* in the first place in the *PLC* up to the average cross-over of all tested languages at 180 words and omit them afterwards. Despite the high phoneme error rates in the pronunciations from the *1:1 G2P Mapping*, we obtain on average a relative cPER reduction of 3% on top of the *PLC* as shown in Table 5.17.

### 5.3.8 Summary

Table 5.16 and 5.17 summarize the cPERs and the necessary edits for 10k *en*, *de*, *es*, *sw*, and *ht* and 6.5k *vi* words. <sup>s</sup> marks results with statistical significance. While for the languages with a strong grapheme-to-phoneme relationship only a few hundred edits are required for all words, and for Spanish between 1.5k and 1.8k, we observe almost 10k required edits for *de* and *en*. In Figure 5.17 we have plotted the cPER reduction over the number of processed pronunciations for *en*, the language with the highest grapheme-to-phoneme complexity. Our word selection strategy *ngram sorted* outperforms *random*. Updating the grapheme-to-phoneme model according



to logistic growing intervals enables between 7% and 60% CPU time savings with performances comparable to other approaches. Our *PLC* of the output of multiple grapheme-to-phoneme converters reduces the editing effort by on average 15% relative to the best single converter, even 38% for *sw*. The traditional *1:1 G2P Mapping* helps *de* and *en* with more complex grapheme-to-phoneme relationships only slightly to reduce the editing effort. *cross-lingual* only outperforms *1:1 G2P Mapping* in the beginning of the process, when less training data for the grapheme-to-phoneme models are available. However, we recommend to use Web-derived pronunciations on top of *PLC* if available because they give us consistent improvements for different vocabulary sizes in the whole process and on average 6% relative for 10k words. Our new Rapid Language Adaptation Toolkit function, which is publicly available, allows to bootstrap a dictionary with the proposed methods supported with the possibility to listen to a synthesized wavefile of the pronunciation.



# Cross-lingual G2P Model-based Pronunciation Generation

---

Motivated by the fact that languages which are related to each other can share some pronunciation rules, we developed a strategy to use grapheme-to-phoneme models derived from existing dictionaries of other languages for the production of pronunciations. This strategy can contribute to a rapid and economic semi-automatic pronunciation dictionary development, thereby reducing the necessary manual effort, as shown in Section 5.3. In particular, it can be an alternative to the initialization with the substitution of graphemes with the most commonly associated phonemes and can be applied if word-pronunciation pairs cannot be found on the Web.

## 6.1 Cross-lingual Pronunciation Generation Strategy

To cross-lingually generate pronunciations for a target language, we elaborated the following strategy:

1. *Grapheme Mapping*: Mapping target language graphemes to the graphemes of a related language (*G2G Mapping*)
2. Applying grapheme-to-phoneme model of the related language to the mapped target language words (*G2P Conversion*)

3. *Phoneme Mapping*: Mapping resulting phonemes of the related language to the target language phonemes (*P2P Mapping*)
4. *Optional*: Post-processing rules to revise shortcomings (*Post-rules*)

Output of Step	ru	bg	de	en
1	биг	биг	bih	bih
2	ru_b ru_i ru_g	bg_b bg_i bg_g	de_b de_i	en_b en_ih
3	ua_b ua_i ua_h	ua_b ua_i ua_h	ua_b ua_i	ua_b ua_y
4	ua_bj ua_i ua_h	ua_bj ua_i ua_h	ua_bj ua_i	ua_b ua_y

**Table 6.1** – Cross-lingual pronunciation production for биг.

Table 6.1 shows the output for the Ukrainian word биг (running) after each step of our cross-lingual dictionary generation strategy with Russian (*ru*), Bulgarian (*bg*), German (*de*) and English (*en*) as related languages. The correct pronunciation in our handcrafted *GlobalPhone* Ukrainian dictionary is *ua\_bj ua\_i ua\_h*.

Step 1 (*Grapheme Mapping*) of our strategy contains the mapping of the target language graphemes to the graphemes of the related language. If existing, for language pairs with different alphabets we use official transliteration rules which define how letters should be mapped according to their similar pronunciation (e.g. the Ukrainian-to-English official transliteration system, defined by the Cabinet of Ministers of Ukraine [Rom11]). For our strategy we select only language pairs, which mainly use the same alphabet or official transliteration rules exist to map the graphemes of the target languages to the ones of the related language. Graphemes existing in both languages are kept in the target language. Graphemes with diacritic marks, which do not exist in the related language, are mapped to the same graphemes without diacritics (e.g. “ö” is mapped to “o”, if the related language is English) as we assumed only common available information like an official transliteration to be present in our experiments and no further linguistic in-house knowledge. Assuming more linguistic knowledge, they can be mapped to the graphemes, which usually have similar pronunciations. If some special characters (e.g. “-”) are not in the vocabulary list of the related language’s dictionary or can not be handled by the grapheme-to-phoneme model, we delete them.

After the *Grapheme Mapping*, the modified target language vocabulary list consists exclusively of graphemes of the related language. In step 2, we apply a grapheme-to-phoneme model to this list, which was trained on existing word-pronunciation pairs of the related language. We receive pronunciations consisting of phonemes of the related language.

In step 3 (*Phoneme Mapping*) we map the phonemes of the pronunciations which we receive from the previous step to the target language phonemes. The mapping is based on the closest distance in the IPA chart [IPA99]. If diphthongs and triphthongs are in the related language but not in the target language, they are mapped to two (or three) vowel phonemes which together represent their components. If a target language contains diphthongs (or triphthongs) not existing in the related language, two (or three) phonemes analogous to these vowels' components, are mapped to one diphthong (or triphthong). Such one-to-many mappings are also possible for consonants. Following these rules, we intend to achieve an optimal mapping without any in-house knowledge of the target language. However, we are aware that by applying this procedure information, such as the the duration of the diphthongs and triphthongs, can get lost.

Step 4 (*Post-rules*) contains automatic phoneme mappings to revise shortcomings in the pronunciations obtained from step 3. To define *Post-rules*, the help of native speakers or linguists is necessary. We stopped to include *Post-rules* once obviously no further improvement was possible due to the quality of the underlying grapheme-to-phoneme model of the related language.

Word	enact	balls	moans
Reference	EH N AE K T	B AO L ZH	M OW N ZH
Grapheme-based	e n a c t	b a l l s	m o a n s
Grapheme-based (mapped)	EH N AX K T	B AX L L S	M AW AX N S

**Table 6.2** – Comparison of pronunciations from grapheme-based vs. phoneme-based dictionary.

We used our cross-lingual pronunciation generation strategy to bootstrap a dictionary for Ukrainian and checked how close we can get to our *Global-Phone* reference pronunciations in terms of phoneme error rate. For comparison, the phoneme error rate (PER) of the *grapheme-based* dictionary is also computed. We evaluated the pronunciations of grapheme-based dictionaries after assigning the most likely phoneme to each grapheme, as the English example illustrates in Table 6.2. Our goal is to have a reduced manual effort with our cross-lingual strategy. Since the substitution of each grapheme based only on the most frequently uttered phoneme is cheap, we regard our approach as successful if the resulting pronunciations have lower phoneme error rates than the *grapheme-based* dictionaries. Furthermore, we evaluated the impact on automatic speech recognition performance.

Table 6.3 indicates that we can generate qualified dictionaries using *ru* and *bg* grapheme-to-phoneme models. Comparing the new pronunciations derived

from the two languages to those of the handcrafted Ukrainian dictionary results in lower phoneme error rates than *grapheme-based*.

	# Rules <i>G2G Mapping</i>	# Rules <i>P2P Mapping</i>	PER (%)	WER (%)	# <i>Post-</i> <i>rules</i>	PER (%)	WER (%)
ru	43	56	12.4	<b>22.8</b>	57	1.7	<b>21.6</b>
bg	40	79	10.3	<b>23.7</b>	65	2.8	<b>22.1</b>
de	(68)*	66	32.7	27.1	39	28.6	26.4
en	(68)*	63	46.8	34.9	21	36.6	34.0
grapheme			14.8	23.8			
manual				22.4			

**Table 6.3** – Effort (#rules) and performance (in terms of PER, WER) for a Ukrainian dictionary using cross-lingual rules.

Using the new dictionaries without *Post-rules* for training and decoding automatic speech recognition systems leads to word error rates (WERs) on the Ukrainian development set which outperform a *grapheme-based* dictionary (23.8% word error rate). Using the dictionary generated with the *bg* grapheme-to-phoneme model even performed better than the handcrafted one (22.4% word error rate). We need only 18% of the number of the 882 search-and-replace rules to generate a qualified Ukrainian dictionary using *ru* grapheme-to-phoneme models and 21% using *bg* grapheme-to-phoneme models. For *en* and *de*, we used the existing official standardized Ukrainian transliterations at the grapheme level (\*) [Rom11]. *de* and *en* grapheme-to-phoneme models did not outperform *grapheme-based*. The dictionaries generated with *bg* and *ru* grapheme-to-phoneme models outperform our handcrafted dictionary after applying *Post-rules* since due to the properties of *bg* and *ru* some semi-palatalized phonemes get lost which are less important for Ukrainian automatic speech recognition, as we show in [SVYS13].

In another set of experiments [Yur13], we investigated how close we approach validated pronunciations of other languages with our strategy. For that, we performed Step 1–3 with additional 40 different language pairs, as shown in Table 6.5. For comparison the phoneme error rate (PER) of the *grapheme-based* dictionaries is also given. The pronunciations of the *grapheme-based* dictionaries of languages with a looser grapheme-to-phoneme relationship (e.g. English (*en*), German (*de*)) have considerably higher phoneme error rates to the pronunciations of the reference dictionaries than language with a closer relationship (e.g. Bulgarian (*bg*), Spanish (*es*), Hausa (*ha*), Russian (*ru*)). We used grapheme-to-phoneme models trained with the *GlobalPhone* dictionaries of the eight languages namely, Bulgarian (*bg*), Ger-

man (*de*), Spanish (*es*), Swedish (*sv*), Russian (*ru*), Hausa (*ha*), French (*fr*) and Portuguese (*pt*) using Sequitur G2P [BN08]. For English (*en*) we employed the CMU dictionary. Pronunciations from the *GlobalPhone* and the CMU dictionaries served as references.

The phoneme error rates of the resulting dictionaries, which are lower than those of their *grapheme-based* dictionaries, are marked in bold. In Table 6.5 we observe that for only two out of nine target languages we could cross-lingually create dictionaries which are closer to our reference pronunciations than the *grapheme-based* dictionaries. For the other seven target languages our cross-lingual dictionary generation strategy performed worse.

Languages		Resulting dictionary	Grapheme-based dictionary
Target	Related	PER (%)	PER (%)
<i>pt</i>	<i>es</i>	56.00	42.89
	<i>fr</i>	49.96	
	<i>de</i>	54.41	
	<i>en</i>	67.23	
	<i>ha</i>	56.08	
<i>sv</i>	<i>en</i>	48.00	26.44
	<i>de</i>	46.43	
	<i>es</i>	55.36	
	<i>fr</i>	51.51	
<i>ha</i>	<i>en</i>	35.15	7.69
	<i>es</i>	25.88	
	<i>fr</i>	32.35	
	<i>pt</i>	31.27	
<i>bg</i>	<i>ru</i>	11.70	6.20
	<i>de</i>	15.78	
	<i>en</i>	32.07	
	<i>ua</i>	15.26	
<i>en</i>	<i>de</i>	<b>50.33</b>	63.57
	<i>fr</i>	<b>50.75</b>	
	<i>sv</i>	<b>53.92</b>	
	<i>es</i>	<b>58.43</b>	
	<i>ua</i>	73.77	
	<i>pt</i>	70.54	
	<i>ha</i>	63.58	
<i>bg</i>	70.21		
<i>ru</i>	<i>ua</i>	32.18	9.83
	<i>bg</i>	17.47	
<i>de</i>	<i>ua</i>	55.97	49.59
	<i>pt</i>	61.50	
	<i>sv</i>	<b>48.53</b>	
	<i>bg</i>	55.02	
	<i>en</i>	<b>47.65</b>	

Languages		Resulting dictionary	Grapheme-based dictionary
Target	Related	PER (%)	PER (%)
<i>es</i>	<i>pt</i>	37.04	13.71
	<i>sv</i>	28.37	
	<i>ha</i>	34.8	
	<i>en</i>	45.63	
<i>fr</i>	<i>pt</i>	53.01	43.08
	<i>sv</i>	52.19	
	<i>ha</i>	63.63	
	<i>en</i>	60.94	

**Table 6.5** – Comparison of cross-lingual generated dictionaries and grapheme-based dictionaries.

## 6.2 Which Languages are Appropriate?

To figure out which language is appropriate as related language for our cross-lingual strategy, we investigated correspondences in the characteristics between source and target language and their impact on cross-lingual pronunciation generation performance.

### 6.2.1 Language Characteristics

According to [KS12], who investigate the prediction of phoneme subsets for a language’s graphemes based on several features, we analyzed the language family information, text and phoneme level information plus their context information as features for our cross-lingual pronunciation generation.

#### Language Family

From Table 6.5 we learn that the language family does not necessarily have much influence on the similarity of the pronunciations. This was also reported for the task in [KS12]. All languages which we have chosen, except for Hausa, belong to the Indo-European family. Hausa belongs to the Afro-Asiatic family. However, we generated dictionaries for Indo-European languages (Portuguese, English and Spanish), using a Hausa grapheme-to-phoneme model. The resulting pronunciations obtain lower phoneme error rates than some other related languages belonging to the Indo-European family.



### Grapheme Information

We calculated grapheme coverages for each pair of target and related languages according to the following equation:

$$\text{Grapheme Coverage [\%]} = \frac{\text{Number of same graphemes in related and target language}}{\text{Number of target language graphemes}} \cdot 100\%$$

The grapheme coverage reflects which share of target language graphemes is contained in the related language's alphabet. If target and related languages use different alphabets, e.g. Ukrainian (Cyrillic script) and English (Latin alphabet), the grapheme coverage was computed after the target language's graphemes were mapped to the related language graphemes following official transliteration rules. In all other cases it was evaluated before the grapheme mapping. For example, for English as the target language and German as the related one the grapheme coverage is 100%. However, for German as the target language and English as the related one the grapheme coverage is 89.66% (ä, ö, ü, ß is missing in English). Since the official Ukrainian-to-English transliteration system maps all Ukrainian graphemes to English ones, the grapheme coverage for this pair is 100%.

Additionally, we evaluated all digraph and trigraph coverages as follows:

$$\text{Digraph (Trigraph) Coverage [\%]} = \frac{\text{Number of same digraphs (trigraphs) in related and target language}}{\text{Number of target language digraphs (trigraphs)}} * 100\%$$

If target and related languages use different alphabets, the digraph and trigraph coverages were evaluated after the target language graphemes in the vocabulary list were mapped to the related language graphemes.

### Phoneme Information

First, we mapped the language-specific phonemes to IPA characters. Then, we calculated phoneme coverages for each target and related languages according to the following equation:

$$\text{Phoneme Coverage [\%]} = \frac{\text{Number of same phonemes in related and target language}}{\text{Number of target language phonemes}} \cdot 100\%$$

The phoneme coverage describes which share of target language phonemes is contained in the related language’s alphabet. For example, for Swedish as the target language and Spanish as the related one, the phoneme coverage is 39.58%. However, for Spanish as the target language and Swedish as the related one, the phoneme coverage is 47.5%.

For most of the languages, which we have chosen, the phoneme sets mainly differ in their vowels. Therefore, we calculated separate vowel and consonant coverages:

$$\text{Vowel Coverage [\%]} = \frac{\text{Number of same vowels in related and target language}}{\text{Number of target language vowels}} \cdot 100\%$$

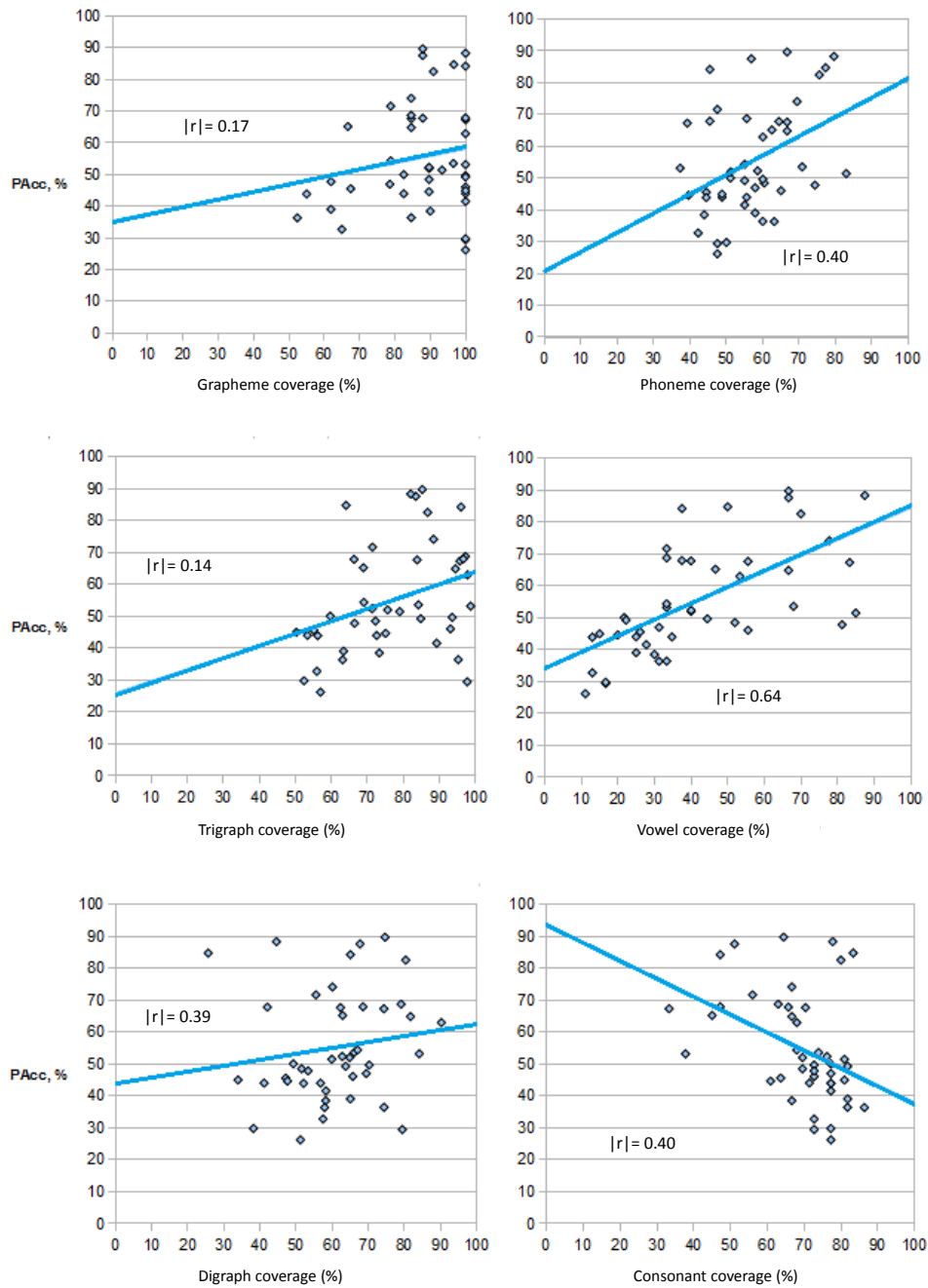
$$\begin{aligned} \text{Consonant Coverage [\%]} = \\ \frac{\text{Number of same consonants in related and target language}}{\text{Number of target language consonants}} \cdot 100\% \end{aligned}$$

Since the pronunciations for the target language are not available if we bootstrap a dictionary, it is impossible to evaluate diphone and triphone coverages for target and related languages. However, we additionally evaluated diphone and triphone coverages to investigate if these criteria help to predict the final pronunciation quality more precisely.

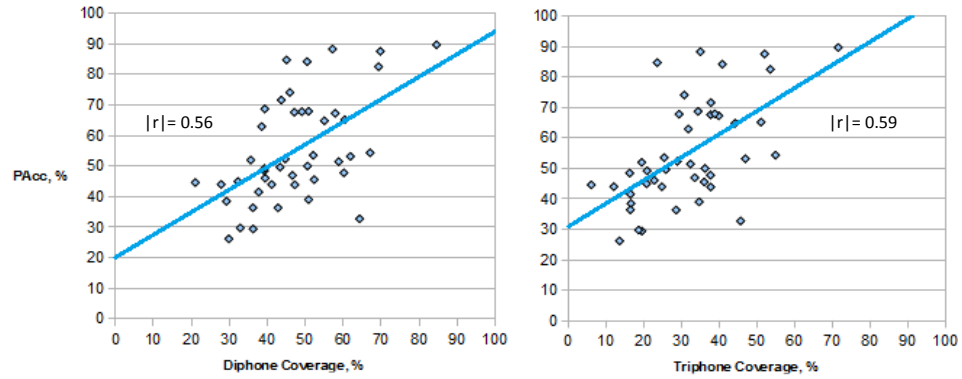
## 6.2.2 Predicting the Accuracy

To analyze the impact of the characteristics between source and target language on the quality of the pronunciations after *step 3* of our cross-lingual strategy, we computed linear regressions between the values of the characteristics and the phoneme accuracy (1 - phoneme error rate) of the resulting pronunciations compared to the reference pronunciations for all 44 language pairs from Table 6.3 and 6.5, as illustrated in Figure 6.1 and 6.2.

Figure 6.1 illustrates the values (grapheme, digraph, trigraph, phoneme, vowel, consonant coverage) together with the linear regression lines and Pearson’s correlation coefficients  $|r|$  [RN88]. We achieve the strongest correlation with the vowel coverage as well as with the phoneme and digraph coverages, while grapheme and trigraph coverages appear to play only a little role. The consonant coverage has a negative correlation with the phoneme accuracy (PAcc). We observe that for most of the languages the phoneme sets mainly differ in vowels, while the consonants are usually similar. Therefore, the consonant coverage seem not to be helpful for our prediction.



**Figure 6.1** – Correlation between characteristics (grapheme, digraph, tri-graph, phoneme, vowel, consonant coverage) and the phoneme accuracy.



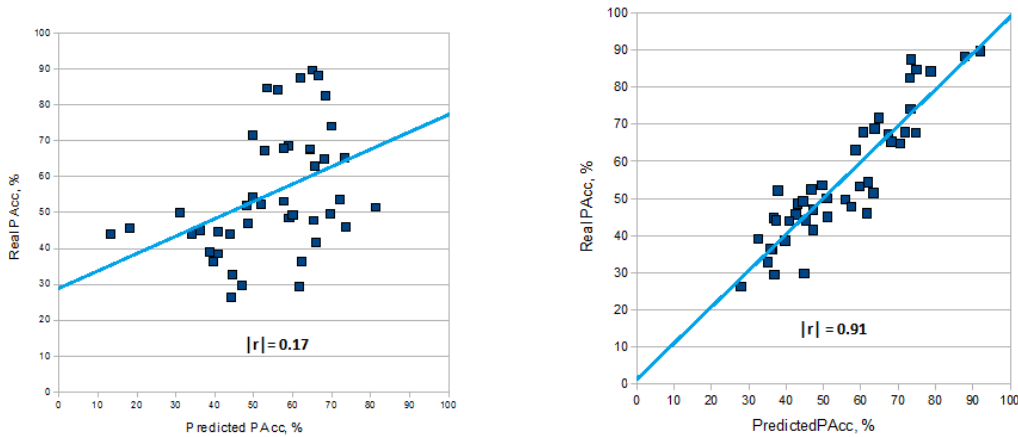
**Figure 6.2** – Correlation between characteristics (diphone, triphone coverage) and the phoneme accuracy.

If we assume to possess sample word-pronunciation pairs in the target language, we can also use diphone and triphone coverage information. Figure 6.2 illustrates the values together with linear regression lines. The correlation coefficients are 0.56 for the diphone and 0.59 for the triphone coverages.

With the help of all six criteria (phoneme, vowel, consonant, grapheme, digraph and trigraph coverages), we computed a predictor function for the phoneme accuracy. In a cross-validation we skipped the real phoneme accuracies of the target language and predicted them with a function that was computed using the information of the remaining language pairs.

Figure 6.3(a) demonstrates the predicted and the real phoneme accuracy. On average the predicted values have an absolute standard deviation of 13.89% to the real one. The coefficient of determination is 0.17.

Finally, we simulated a scenario where we possess enough sample word-pronunciation pairs in the target language which give reliable information about the diphone and triphone coverages and all language pairs as training data for the predictor function. Using this function, we predicted the phoneme accuracy with an absolute standard deviation of 5.6%, as depicted in Figure 6.3(b). The coefficient of determination is 0.91. This shows that in addition to more training data, the phoneme coverages would include supplementary information for our predictor function. [KS12] report a method whereby “characters are noisily converted to phones using their IPA notation”. This methods may be used to roughly predict the diphone and triphone coverages.



(a) Phoneme, vowel, consonant, grapheme, digraph and trigraph information.

(b) Oracle.

Figure 6.3 – Correlation between predicted and the real phoneme accuracy.

### 6.3 Towards Universal Grapheme-to-Phoneme Conversion

Finally, we investigated an approach to incorporate the rules of more than one language. With the help of a universal grapheme-to-phoneme converter, which was trained with word-pronunciation pairs from several languages, we restrict the output using information of the characteristics of the target language. Figure 6.4 illustrates this scenario.

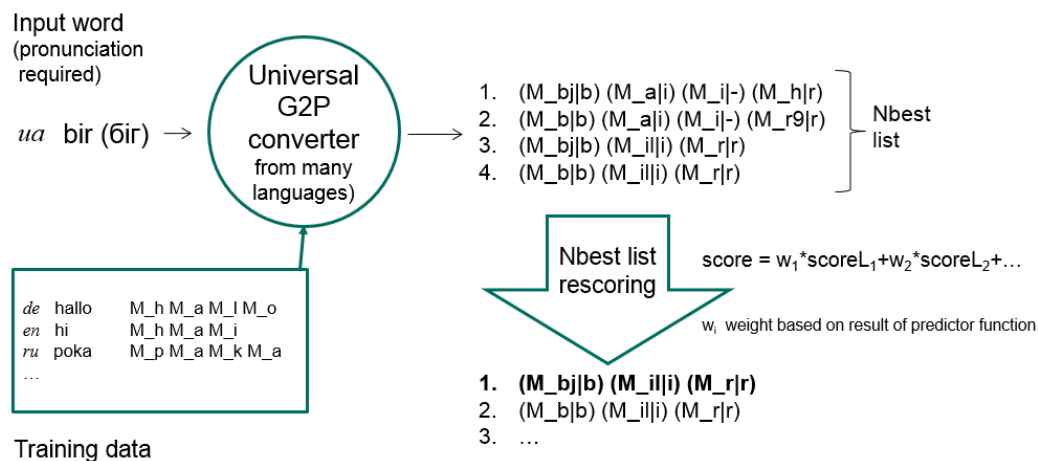


Figure 6.4 – Strategy for universal grapheme-to-phoneme conversion.

To train a universal grapheme-to-phoneme converter, we applied the following steps:

1. Take equal number of word-pronunciation pairs from each source language dictionary.
2. Map phonemes and graphemes to global phoneme/grapheme set (see Chapter 6.1).
3. Train universal grapheme-to-phoneme converter.

Then, we generated multiple pronunciations and selected the most appropriate pronunciation for each target language word:

1. Apply universal grapheme-to-phoneme converter to target language word and generate multiple possible pronunciations (1000-best list output).
2. Rescore each pronunciation (new scores based on a multilingual language model score, interpolation weights result from predictor function).
3. Output new 1st-best pronunciation after reordering.

For rescoreing each pronunciation in the 1000-best list with a language model score, we built a multilingual language model as follows: Since they outperformed phoneme-based ones, we trained a 3-gram grapheme-level language model from each source dictionary using the SRI Language Modeling Toolkit [Sto02]. The graphemes were derived from a 1:1 alignment between the graphemes and the phonemes of each word. We transferred the pronunciations in our 1000-best lists to grapheme-level the same way. The next step was to generate an interpolated language model. We defined the interpolation weight for each language model based on the similarity of source and target language. To investigate if the strategy works with optimal weights, we optimized the interpolation weights on the graphemes from the target language dictionary in an oracle experiment using the SRI Language Modeling Toolkit. In a realistic scenario, the interpolation weights for each language model can be determined based on the output of the predictor function after computing the coverages as described in Section 6.2.

We trained our universal grapheme-to-phoneme converter with Russian (*ru*), Bulgarian (*bg*), Polish (*pl*), Croatian (*hr*) and Czech (*cs*). 23k word-pronunciation pairs were randomly extracted from the *GlobalPhone* dictionaries for each language. The target language was again Ukrainian. Table 6.6 demonstrates that despite optimal language model weights, the phoneme error rate with the universal grapheme-to-phoneme converter is with 19.7% (*With rescoring and reordering*) almost 25% relative worse than with our grapheme-based dictionary.

	PER (%)
With rescoring and reordering	19.7
No rescoring and reordering	23.8
Best pronunciations in 1000-best lists	9.2
grapheme-based	14.8

**Table 6.6** – Performance of universal grapheme-to-phoneme conversion with Ukrainian as target language.

However, if we selected for all words always the pronunciation with the lowest phoneme error rate from each 1000-best list, the pronunciations of our final dictionary would have a phoneme error rate of 9.2%. This shows that there is 53.3% relative room for improvement in the rescoring and reordering process. To investigate the impact of our rescoring and reordering method, we exclusively evaluated the 1-best pronunciation from the universal grapheme-to-phoneme converter output and obtained a phoneme error rate of 23.8% (*No rescoring and reordering*). This indicates that our language model rescoring and reordering leads to a 17.2% relative lower phoneme error rate. The fact, that it does not reach the 53.3% relative improvement, indicates that the pure grapheme-level language model perplexity is not the best choice. Features which give better constraints for the selection of the optimal pronunciation from the n-best list need to be investigated.

## 6.4 Summary

We have elaborated a generic strategy for the rapid cross-lingual creation of pronunciation dictionaries, using grapheme-to-phoneme models derived from existing dictionaries of other languages. This strategy can support the semi-automatic pronunciation generation, as we have shown in Section 5.3.7.

To select the most appropriate related languages for a given target language, we investigated the relevance of grapheme, digraph, trigraph, phoneme, vowels phoneme and consonants phoneme coverages between related and target language.

Finally, we studied a method to combine grapheme-to-phoneme models from several languages. In our experiment our universal grapheme-to-phoneme converter generated pronunciations in the 1000-best lists which ideally lead to only 9.2% phoneme error rate. However, finding the correct pronunciation in the 1000-best lists is still a challenge and even with limited oracle information

the quality of the universal grapheme-to-phoneme converter's 1st-best output is still worse than the one of the grapheme-based dictionary. There is still room for improvement in the rescoring and reordering process.



## Pronunciation Generation for Foreign Words and Accents

---

In the previous chapters, we have presented methods to retrieve pronunciations for a new language. In this chapter, we approach the two following problems for the pronunciation generation: First, to economically build up lexical resources with automatic or semi-automatic methods, it is important to detect and treat words for foreign languages separately. Second, to improve the automatic speech recognition performance of speech with accent, we rapidly adapt the pronunciations in the dictionary.

With the globalization, more and more words from other languages come into a language without assimilation to the phonetic system of the new language. For example, due to the strong increase of Anglicisms, especially from the IT domain, features for their automatic detection are helpful. If foreign words are written in a writing system which differs from the target language, treating them separately can be done without a special detection. However, often the writing system is adapted.

Pronunciations in existing dictionaries are traditionally those of native speakers. Therefore, the use of the pronunciation dictionary in domains where speech of speakers with accent is to be transcribed, is a challenge. Accented speech occurs amplified where non-native speakers operate automatic speech recognition systems, e.g. at the airport, in radio communication of aircrafts or where parliamentary speeches in multilingual communities are transcribed, e.g. the cantonal parliament of Valais [IBC<sup>+</sup>12]. In addition to

acoustic model adaptation techniques such as MAP (Maximum A Posteriori) or MLLR (Maximum Likelihood Linear Regression), lexical adaptation can also give performance improvements [Mih11, VLW<sup>+</sup>12]. In Section 7.2, we analyze a parallel corpus of phoneme sequences from phonetic transcriptions of native US English and accented English in the speech accent archive of the George Mason University (GMU) [Wei10] to rapidly and economically adapt the pronunciation dictionary and improve the automatic speech recognition performance of accented English.

## 7.1 Pronunciation Generation for Foreign Words

Due to the globalization more and more words from other languages come into a language without assimilation to the phonetic system of the new language. As English is the prime tongue of international communication, English terms are widespread in many languages. This is particularly true for the IT sector but not limited to that domain. For example, African and Indian languages [GBP12, KC12], of which many are still under-resourced, use a lot of borrowed English words. Anglicisms – i.e. words borrowed from English into another language (the so called *matrix language*) – nowadays come naturally to most people but this mix of languages poses a challenge to speech communication systems.

Automatic speech recognition and speech synthesis systems need correct pronunciations for these words of English origin. However, a grapheme-to-phoneme model of the matrix language, which is usually employed to rapidly and economically generate pronunciations, does often give inappropriate pronunciations for these words. Nowadays many people are fluent in English and pronounce Anglicisms according to their original pronunciation. An automatic detection of Anglicisms enables us to use more adequate English pronunciation rules to generate pronunciations for them. Adding pronunciation variants for foreign words to the dictionary can reduce the word error rate of automatic speech recognition systems, as shown in [MK12]. Therefore, we developed new methods to automatically detect Anglicisms from word lists of different matrix languages and advance existing approaches. The term *matrix language* designates the main language of a text from which we try to distinguish the inclusions of English origin.

Development and analysis of our features were first performed with German as matrix language. Thus, in the following figures German reflects the matrix language. Later we additionally evaluated our features on the matrix language Afrikaans. However, our methods can easily be adapted to new languages.

### 7.1.1 Traditional Methods

Specific treatment of foreign inclusions for the pronunciation dictionary generation improves text-to-speech system [Ahm05] and automatic speech recognition system performance. [MK12] add pronunciation variants for automatically detected foreign words and reduce the word error rate of a Finnish automatic speech recognition system by up to 8.8% relative. [GBP12] lower the word error rate of a Swahili automatic speech recognition system from 26.9% to 26.5% by adding English pronunciations variants. These English words amount to almost 9% of the total words in their Swahili dictionary. Moreover, a foreign word detection improves part-of-speech parsing, as reported in [Ale08a]. A simple approach to detect foreign words in word lists and generate different pronunciations for them has already been patented in [AJSS11].

There have been many approaches for the detection of foreign words based on grapheme-level methods, mostly based on grapheme n-gram likelihoods. [MK12] focus on the effects of pronunciation variants on automatic speech recognition and use a simple grapheme perplexity threshold, treating the 30% of words with the highest perplexity as foreign word candidates. [JMLC99] and [KC02] compare syllable probabilities between a Korean and a foreign model and extracted the foreign word stem. [ACT05] developed a “Cumulative Frequency Addition” which distinguishes between a number of different languages. Thereby grapheme n-gram frequencies to classify a word. While [BMSW97] work with word-based Hidden-Markov-Models (HMMs), [KSNM03] switch to character-level HMMs thereby achieving high error reduction. We compare grapheme n-gram probabilities after converting them to perplexities for our *Grapheme Perplexity Feature*.

Another common approach to foreign word detection is a dictionary lookup. Even if grapheme n-grams performed better than dictionary lookup, their combination gives the best results in [And05]. [Ale05] use a dictionary lookup to reduce the number of English word candidates before applying more costly features. [OWS08, Och09] use the open source spell-checker and morphologi-

cal analyzer Hunspell<sup>1</sup>. Our *Hunspell Lookup Feature* uses a similar approach, also basing its classification on Hunspell lookups.

An innovative method is the comparison of the number of search engine results found for different languages [Ale05], which we reimplemented for our *Google Hit Count Feature*.

[KC12] interpolated probabilities of grapheme and phoneme language models for English and Bangla. Their classification is based on a comparison between those probabilities. The phoneme sequences are generated with a grapheme-to-phoneme converter producing the pronunciations for Bangla and English transliterated words. Our *G2P Confidence Feature* uses a similar approach, also basing its classification on a combination of phoneme- and grapheme-level information. For our feature we compare probabilities of grapheme-level models.

For named entity recognition, often the local context or specific trigger words are used. Part-of-speech tags, capitalization and punctuation are also common features as shown in [MLP03] and [WVD95]. The detection performance is usually evaluated in terms of F-score with equal weight for precision and recall [Pow11]. Results vary for the different methods and setups in related work. [KC02] achieve 88.0% F-score detecting foreign transliterations in Korean. [ACT05] reach 79.9% distinguishing between several language pairs. Detecting English inclusions in German text, [Ale05]’s experiments are very similar to ours and give comparable results of up to 77.2% F-score.

### 7.1.2 Experimental Setup

For evaluation, we built two German test sets [LSS14]. One from the IT domain and one from general news articles. Furthermore, we applied our methods to words from the Afrikaans NCHLT corpus [HDB12].

#### German IT Corpus

Our German IT corpus *Microsoft-de* contains about 4.6k word types crawled from the German website of Microsoft *www.microsoft.de*. To reduce the effort of hand-annotating, this word list only contains frequent types that occurred more than once in the crawled text. Before extracting the types for our word list, some normalization and cleanup was performed on the

---

<sup>1</sup>hunspell.sourceforge.net

crawled text. We removed all HTML tags, sentences containing more than 80% capital letters and replaced punctuation marks including hyphens with spaces.

In our German word lists English words and some additional word categories for further analyses were annotated. Like [Ale05], we base our annotation on the agreement of the annotators. In case of disagreement we consulted the well-known German dictionary Duden ([www.duden.de](http://www.duden.de)) and checked the context in which the word occurred in the text. The annotation of the German word lists follows the guidelines described in Table 7.1.

Category	
<i>English</i>	All English words were tagged as “English”. This comprises all types of words including proper names and also pseudo-Anglicisms. Words which could be German as well as English (homomorph words) were not tagged as English (e.g. <i>Admiral</i> , <i>Evolution</i> , . . .). Words containing an English part (see <i>Hybrid foreign word</i> ) were tagged as English since a monolingual German generate correct grapheme-to-phoneme model cannot pronunciations for those words.
<i>abbreviation</i>	Abbreviations were tagged as “abbreviation”. We did not distinguish between English and German abbreviations as our focus is to detect whole words with English part. Therefore no abbreviations were tagged as English.
<i>other foreign word</i>	Foreign words that are neither German nor English were tagged as “foreign”. As we limit our algorithms to classify exclusively between the categories <i>English</i> and <i>non-English</i> , these words fall into the category <i>non-English</i> .
<i>hybrid foreign word</i>	Words containing an English plus a German part were tagged as “hybrid” in addition to “English”. This covers for example compound words with a German and an English part (e.g. “Schadsoftware”) and grammatically conjugated forms of English verbs (e.g. “downloaden”).

**Table 7.1** – Annotation guidelines for the German test sets.

We selected 824 sentences, containing 2,276 unique words (types) from the *Microsoft-de* corpus. For this vocabulary we created a reference pronunciation dictionary. The pronunciations for words annotated as English were generated with an English grapheme-to-phoneme model which was built from the *CMU dictionary (CMUdict)* [cmu]. The pronunciations for non-English words were generated with a German grapheme-to-phoneme model which was generated from the German *GlobalPhone* dictionary (*GP-de*) [SS14]. The pronunciations for hybrid English words were created manually. To avoid ambiguous pronunciations for abbreviations, we only selected sentences not containing any abbreviation. For the whole dictionary we use the German pho-

neme set from *GP-de*. Pronunciations generated with the English grapheme-to-phoneme model were mapped to this German phoneme set based on the IPA scheme [IPA99].

## Other Domains and Languages

To compare the detection performance on different domains and languages, we use two more annotated word lists. The general news domain word list *Spiegel-de* contains about 6.6k types from 35 articles covering the domain of German political and business news. The texts were manually taken from the website of the German news journal Spiegel *www.spiegel.de*. The texts have not been crawled automatically to keep the word list clean of advertisements, user comments and other unwanted content. The punctuation marks were removed. The *NCHLT-af* word list contains about 9.4k types taken from the Afrikaans part of the *NCHLT* corpus [HDB12], which contains a collection in the eleven official languages of South Africa. In our Afrikaans test set English, foreign words and abbreviations have been annotated by [BD13]. The authors kindly provided this annotated word list for our experiments<sup>2</sup>.

## Distribution of the Word Categories

Figure 7.1 demonstrates the distribution of the word categories in our four word lists. Especially, in the IT domain we find many *foreign words* and *abbreviations*. Those are more than 21% of the *Microsoft-de* word list, where 15% of all words are *English*. In the general news domain (*Spiegel-de*) we find only approximately 4% *English* words. About 10% of the English words in our German word lists from each domain are *hybrid* words, consisting of German and English parts (e.g. “Schadsoftware”). The Afrikaans *NCHLT* corpus contains only 2% *English* words and 1% *abbreviations*.

### 7.1.3 Detection of Foreign Words

In our scenario we receive single words from texts in the matrix language as input [Lei14, LSS14]. As output we produce a classification between the

---

<sup>2</sup>We would like to thank Willem D. Basson for useful comments and the other members of the speech group from the North-West University, South Africa, for providing the *NCHLT* corpus.

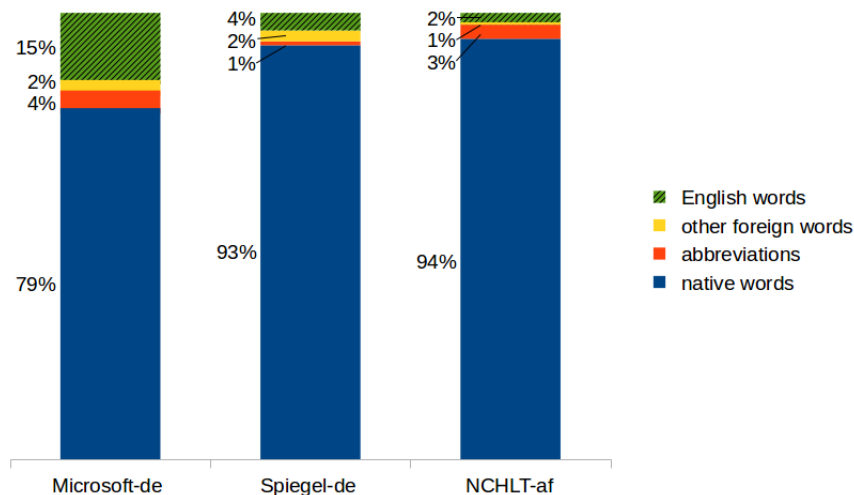


Figure 7.1 – Foreign words in different word lists

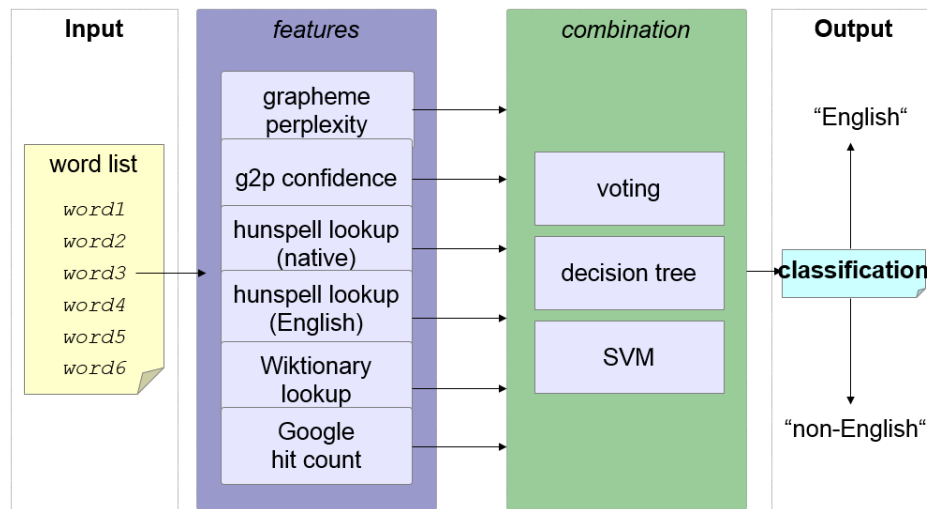
classes *English* and *non-English*. While some related work relies on information about the word context (e. g. part-of-speech), we concentrate on context-independent features of the examined word itself to classify it. This flexibility is useful as dictionaries are often based on lists of most frequent words [SVYS13]. Other features which use information about the word context can still be integrated in future work. As shown in Figure 7.2, we developed and evaluated a set of different features to detect English words in word lists:

- Grapheme perplexity
- G2P confidence
- Hunspell spell-checker dictionary lookup
- Wiktionary lookup
- Google hit count

Those features were separately tuned and evaluated before we proceeded to combine them. For the combination we experimented with different methods:

- Voting
- Decision Tree
- Support Vector Machine (SVM)

We leverage different sources of expert knowledge and unannotated training text to create features that are mostly language-independent and cheap to set



**Figure 7.2** – Anglicism detection system.

up. By developing features based on commonly available training data like unannotated word lists or spell-checker dictionaries, we avoid the expensive step of hand-annotating Anglicisms directly in lots of training data. This also enables to use available frameworks for the implementation of our approaches (e.g. the SRI Language Modeling Toolkit [Sto02] for the *Grapheme Perplexity Feature* or Phonetisaurus [Nov11, NMH12] for the *G2P Confidence Feature*). A more expensive resource which boosts our *G2P Confidence Feature* may be a pronunciation dictionary of the matrix language. For English and many other languages, dictionaries are available. To account for scenarios where a pronunciation dictionary is not available or of poor quality, we also evaluated our *G2P Confidence Feature* in simulated situations with pronunciation dictionaries containing only a small number of entries.

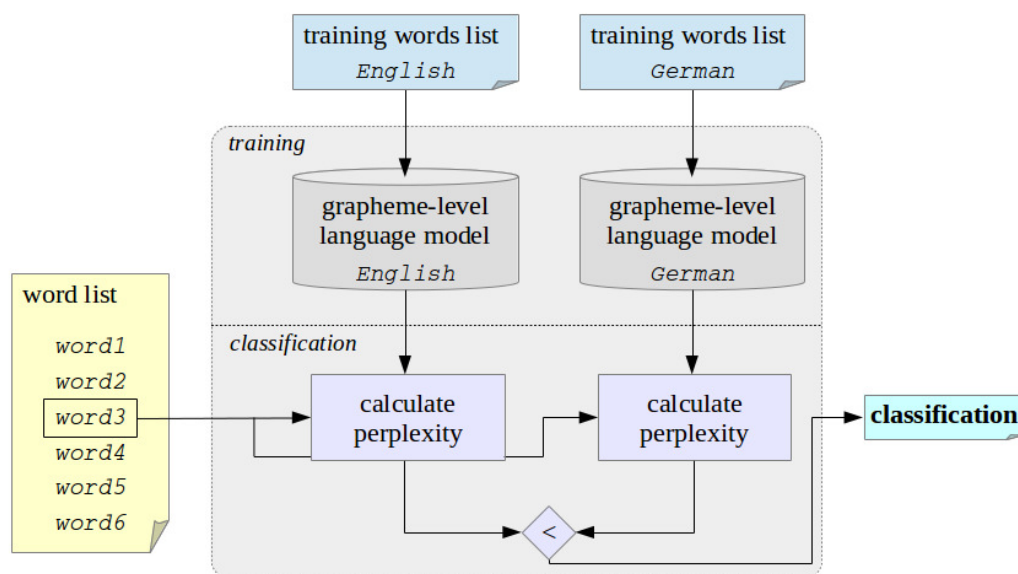
To detect Anglicisms, we advanced existing methods and developed entirely new features. In addition to the evaluation of new approaches, an important goal was to develop features which are inexpensive and portable to new languages. In contrast to standard supervised machine learning, our features do not rely on training data that is annotated specifically for the task of Anglicism detection. In contrast to other approaches which train and test features on disjunctive sets from the same word list, our test sets are only used for evaluation of our single features and never for their training. Instead we use common independent resources such as word lists and pronunciation or spell-checker dictionaries. Exceptions are only our feature combinations which are trained in a cross-validation on the test sets. To avoid supervised training of thresholds, for most features we base our classification on the dif-



ference between results calculated on an English model and a model of the matrix language.

### Grapheme Perplexity Feature

The grapheme-level detection of foreign words is based on the assumption that grapheme sequences depend on the language. For example, in our German word list 25.8% of the words end with “en” while in the English word list they only account for 1.7%. A grapheme (or character)  $n$ -gram is a sequence of  $n$  graphemes. Grapheme-level language models are trained from lists of training words. These models are a statistical representation of grapheme  $n$ -grams over all training words. In addition to the graphemes, word boundary symbols are included to specifically identify the grapheme  $n$ -grams at the beginning and end of words. We used the SRI Language Modeling Toolkit to build the  $n$ -gram grapheme models. The detection based on grapheme  $n$ -gram models deals well with conjugations and small variations of words. Unknown forms of a word can still be recognized because the overall grapheme sequences stay similar. Therefore, many works in the field of Named Entity Recognition and Foreign Entity Recognition are based on grapheme  $n$ -grams ([KSNM03], [MK12], [JMLC99], [KC02], [ACT05]).



**Figure 7.3** – Classification with the *Grapheme Perplexity Feature*.

We experimented with different training word lists and parameters to build grapheme  $n$ -gram models. The best Anglicism detection performance was

achieved using case-insensitive 5-gram models built from lists of unique training words. To train the grapheme language models, we used 116k word types from the CMU dictionary for English and 37k from the German *GlobalPhone* dictionary for German. The Afrikaans model was trained with 27k word types crawled on the Afrikaans news website `www.rapport.co.za`. To port this feature to another language, an unannotated word list from that language is sufficient as long as grapheme sequences of that language are more likely in this word list than in the English one. Our approach of using the perplexity difference between two models allows us to have an unsupervised classification based on a direct comparison of perplexities for an English model and a model of the matrix language. Figure 7.3 depicts the steps of our *Grapheme Perplexity Feature*:

1. Preparation: Training of grapheme-level language models from training word lists for English and the matrix language.
2. Calculation of the perplexity on the English model and the model of the matrix language for a word from the test set.
3. Comparison of the two perplexities and classification towards the language model whose perplexity is lower for the word.

The feature uses the difference of the English and matrix language perplexities. We calculate

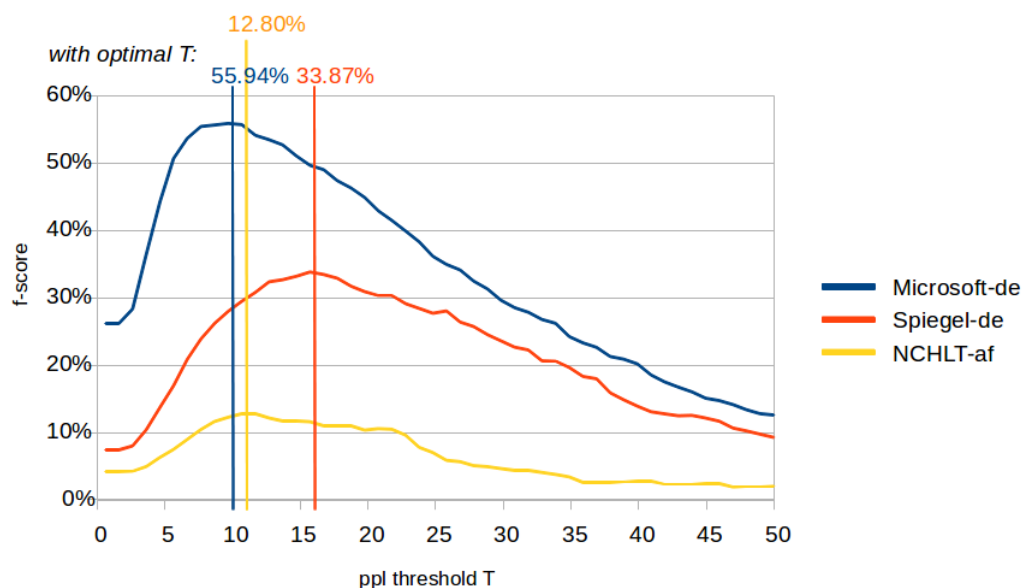
$$d = \text{ppl}_{\text{matrixlang.}}(w) - \text{ppl}_{\text{English}}(w)$$

and classify a word  $w$  as English if the difference  $d$  is greater than zero. We generically assume a threshold of zero, which leads to a simple comparison of which perplexity is smaller.

Test set	Threshold = 0	Optimal threshold	
	F-score	F-score	Threshold
Microsoft-de	67.17%	68.56%	0.5
Spiegel-de	36.00%	45.61%	6.5
NCHLT-af	25.75%	29.87%	2.5

**Table 7.2** – Detection performance (F-score) with different thresholds for the grapheme perplexity difference.

This is not an optimal choice as shown in Table 7.2 and Figure 7.4. We still make this trade-off to refrain from supervised training of a better threshold. The different optimal thresholds seem to be related to the portion of



**Figure 7.4** – Detection performance in relation to an absolute perplexity threshold.

Anglicisms in the test set. *Microsoft-de* contains almost four times as many Anglicisms as *Spiegel-de*. Further analyses of the performance gap between the test sets are summarized in Section 7.1.3. A further normalization by standard score (z-score) over the perplexities of all words of the test set led to worse results.

The results of the final version of our *Grapheme Perplexity Feature* are shown in Table 7.3 for the different test sets. We achieve a good recall, for which the *Grapheme Perplexity Feature* is one of our best features. This means most Anglicisms in the test sets are detected. Precision is considerably lower indicating that the feature is wrongly detecting a lot of words which are not Anglicisms.

Test set	<b>F-score</b>	Precision	Recall
Microsoft-de	<b>67.17%</b>	55.85%	84.26%
Spiegel-de	<b>36.00%</b>	22.73%	86.54%
NCHLT-af	<b>25.75%</b>	15.22%	83.50%

**Table 7.3** – Performance of the Grapheme Perplexity Feature.

## G2P Confidence Feature

We use *Phonetisaurus* [Nov11, NMH12] for our experiments. *Phonetisaurus* takes the following steps to predict pronunciations:

1. Alignment of graphemes and phonemes in the training dictionary (creating graphones).
2. Training of a graphone-level language model.
3. Prediction of pronunciations for novel words.

In the alignment step, graphemes are combined with the phonemes from the corresponding pronunciation. In literature, the resulting grapheme-phoneme clusters are usually named *graphones* [BN08]. Then, a 7-gram graphone-level language model is trained from all graphone sequences of the training dictionary. To predict pronunciations, *Phonetisaurus* searches the shortest path in the grapheme-to-phoneme model which corresponds to the input grapheme sequence. As path costs, the graphones' negative log probabilities are summed up. This value can be interpreted as a confidence measure: It is used to rank different pronunciation variants. In our experiments, we use this grapheme-to-phoneme confidence to measure the “sureness” between a word's pronunciation variants generated from different grapheme-to-phoneme models. To train grapheme-to-phoneme models, we use 133k word-pronunciation pairs from the CMU dictionary for English, 38k from the German *GlobalPhone* dictionary (*GP-de*) for German and the Afrikaans pronunciation dictionary (*dict-af*) created by [ES05] (42k word-pronunciation pairs). Our *G2P Confidence Feature* is conceptually similar to our *Grapheme Perplexity Feature*. The only difference is that we only compare scores for a word at graphone-level instead of grapheme-level. The steps to detect Anglicisms based on G2P confidence are like in Figure 7.3 for the *Grapheme Perplexity Feature*, but replacing the grapheme-level model with a grapheme-to-phoneme model and perplexity with the graphone log probability (*Phonetisaurus* G2P confidence):

1. Preparation: Training of grapheme-to-phoneme (graphone) models from English and matrix language pronunciation dictionaries
2. Prediction of pronunciation for a word from the test set
3. Comparison of G2P confidence and classification of the class for which the confidence is better

As described, we use *Phonetisaurus* for pronunciation prediction and rely on its confidence measure, the negative log probability of the grapheme sequence which represents the path costs. The G2P confidence of the first-best pronunciation for a word is used, while the generated pronunciation itself is discarded. The feature uses the difference of the G2P confidence for English and the matrix language. We calculate

$$d = \text{G2Pconf}_{\text{matrixlang.}}(w) - \text{G2Pconf}_{\text{English}}(w)$$

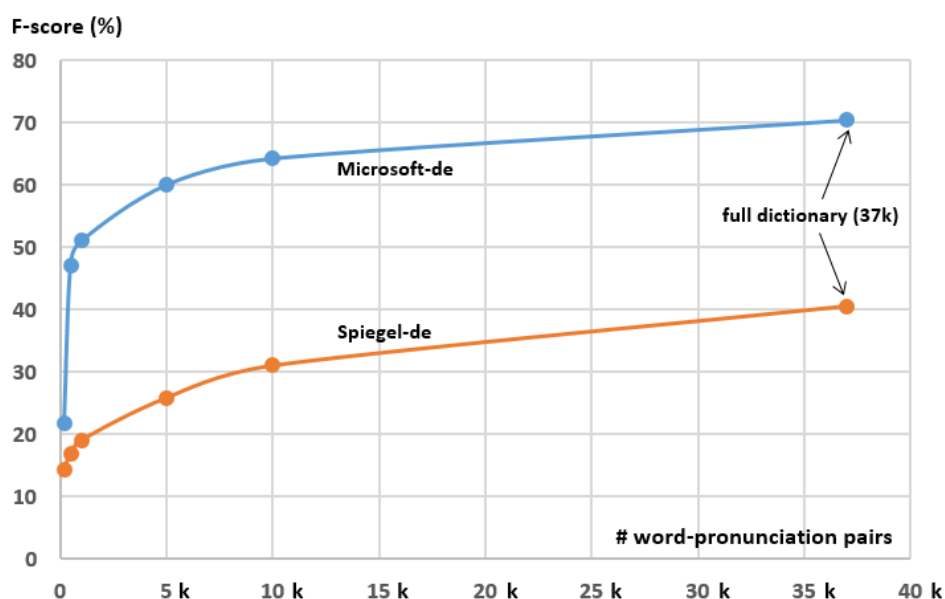
and classify a word  $w$  as English if the difference  $d$  is greater than zero since the *G2Pconf* reflects path costs. We generically assume a threshold of zero, which leads to a simple comparison of which grapheme-to-phoneme confidence is smaller. Like for the grapheme perplexity difference, this is not an optimal choice as shown in Table 7.4. Again we make this trade-off to refrain from supervised training of a better threshold.

Test set	Threshold = 0	Optimal threshold	
	F-score	F-score	Threshold
Microsoft-de	70.39%	71.40%	1.0
Spiegel-de	40.56%	45.00%	1.0
NCHLT-af	23.94%	40.23%	10.0

**Table 7.4** – Detection performance (F-score) with different thresholds for the grapheme-to-phoneme model confidence difference.

For both German test sets *Microsoft-de* and *Spiegel-de*, the optimal threshold is 1. This value seems to be depending on the dictionary used for the grapheme-to-phoneme model training as we only reach a good detection performance from much higher thresholds for the Afrikaans test set. To account for scenarios with low lexical resources in the matrix language, we also evaluated this feature in simulated situations with pronunciation dictionaries containing only a small number of entries. Figure 7.5 illustrates the detection performance with different amounts of word-pronunciation pairs to train the grapheme-to-phoneme model of the matrix language.

The results of our *G2P Confidence Feature* are shown in Table 7.5. They are similar to the performance of our *Grapheme Perplexity Feature* with some improvements for the German test sets. An in-depth comparison of all features is done in Section 7.1.3.



**Figure 7.5** – Detection performance (F-score) with different dictionary sizes for default threshold of zero.

We achieve a good recall, for which the *G2P Confidence Feature* together with the *Grapheme Perplexity Feature* is our best feature. This means most Anglicisms in the test sets are indeed detected. The precision is considerably lower indicating that the feature is wrongly detecting a lot of words which are not Anglicisms.

Test set	<b>F-score</b>	Precision	Recall
Microsoft-de	<b>70.39%</b>	59.44%	86.30%
Spiegel-de	<b>40.56%</b>	29.74%	83.91%
NCHLT-af	<b>23.94%</b>	14.21%	75.86%

**Table 7.5** – Performance of the G2P Confidence Feature.

### Hunspell Lookup Features

*Hunspell* is an open source spell-checker and morphological analyzer used in software like OpenOffice. It supports complex compounding and morphological analysis and stemming. The word forms are recognized based on rules defined in the spell-checker dictionary of a language. *Hunspell* spell-checker dictionaries are freely available for more than 60 languages, including

English, German and Afrikaans. For our features we used those *Hunspell* resources: The American English dictionary (*en\_US*) with 62k basic entries, the “frami” version of the German dictionary (*de\_DE-frami*) with 220k basic entries and the Afrikaans dictionary (*af\_ZA*) with 125k basic entries. Our *Hunspell Lookup Features* simply check whether a word is found in the dictionary of the language. The lookup includes an automatic check if the word in question can be derived by the morphological or compound rules in the dictionary. We use two independent features with this concept:

- *English Hunspell Lookup*  
If the word is found or derived from the English dictionary, it is classified as *English*, otherwise as *non-English*. This feature is language independent and can be used without modification for any matrix language.
- *Matrix language Hunspell Lookup*  
The *Matrix Language Hunspell Lookup Feature* does a lookup in the spell-checker dictionary of the matrix language. In the case that a word is found or derived from the matrix language dictionary it is classified as *non-English*, otherwise as *English*.

The *Matrix Language* and the *English Hunspell Lookup Feature* are independently evaluated. Their classifications can disagree if a word is found in both dictionaries or in neither dictionary. We also experimented with combinations of both features: We only classified a word as *non-English* if it was in the matrix language dictionary, while not being in the English dictionary. All other words were classified as *English*. However, this did not lead to better results.

Table 7.6 shows the performance of our *English Hunspell Lookup* and *Matrix Language Hunspell Lookup Features*.

Test set	English Hunspell Lookup			Matrix Language Hunspell Lookup		
	F-score	Precision	Recall	F-score	Precision	Recall
Microsoft-de	<b>63.65%</b>	57.13%	71.87%	<b>61.29%</b>	58.09%	64.87%
Spiegel-de	<b>39.17%</b>	26.41%	75.77%	<b>41.65%</b>	35.50%	50.38%
NCHLT-af	<b>31.90%</b>	19.27%	92.50%	<b>12.48%</b>	6.71%	89.50%

**Table 7.6** – Detection performance of English and Matrix Language Hunspell Lookup.

The Afrikaans spell-checker dictionary results in a weak detection performance. More than 25% of Afrikaans words were not found in the dictionary, as shown in Figure 7.6. On the other hand, the *English Hunspell Lookup Feature* is our best feature for the *NCHLT-af* Afrikaans test set.

In Figure 7.6 the portion of English and non-English words from the test sets found in each dictionary are illustrated. A significant amount of English words are already commonly used in German and included in the German dictionary – these lead to false positives. In particular, non-English foreign words and abbreviations, which are classified as *non-English* in our reference, make up the portion of “non-English” words not found in the German dictionary (false negatives).

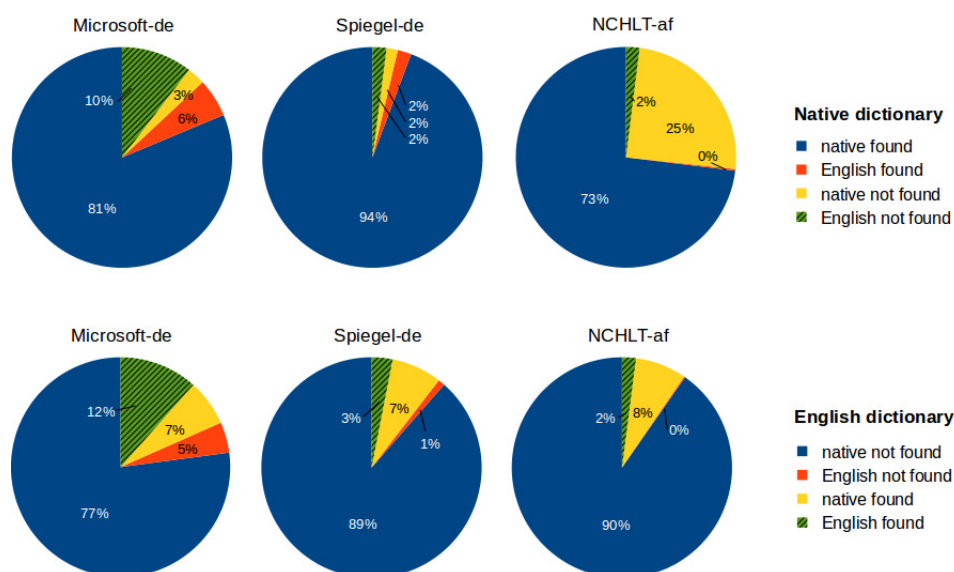


Figure 7.6 – Portion of words found in each dictionary.

In the English dictionary the false negatives (Anglicisms which were not detected) are mainly *hybrid words* containing an English and a German part. They are part of the *English* class in our reference annotation. There is also a sizable amount of words that are spelled exactly the same in both English and German. Together with names this makes up most of the false positives of the *English Hunspell Lookup Feature*.

## Wiktionary Lookup

*Wiktionary* ([www.wiktionary.org](http://www.wiktionary.org)) is a community-driven online dictionary. Like Wikipedia, the content is written by volunteers. *Wiktionary* is available for over 150 languages but scope and quality in the different languages vary [SOS14]. While the English and French *Wiktionary* each contain more than a million entries, the German *Wiktionary* currently has approximately 355,000 entries and the Afrikaans *Wiktionary* less than 16,000. However, the





**Figure 7.7** – Entry of German *Wiktionary* containing a paragraph about the word’s origin and language.

*Wiktionary* project is growing rapidly which is an advantage for our approach because information about recently introduced words is likely to be added in the future. *Wiktionary* provides a wide range of information. For example, in Section 5.1 we show how we extract pronunciations from *Wiktionary*. For most words, *Wiktionary* contains a paragraph about the word’s origin. The *Wiktionary* edition of one language does not only contain words from that language. Foreign words including the name of the source language are also added. The snapshot in Figure 7.7 shows the Anglicism “downloaden” as a German word (“Deutsch” meaning German) which is originating from English (explained in the section “Herkunft” meaning origin). To detect Anglicisms, we only use the information from the matrix language’s *Wiktionary* version. A word is classified as *English* if:

- There is an entry for this word belonging to the matrix language and the origin section contains a keyword indicating English origin.
- There is no entry belonging to the matrix language but an entry marked as “English” in the matrix language’s *Wiktionary*.

Unfortunately, entries of the *Wiktionary* versions from different languages do not have a common style and structure. Therefore, some language-dependent fine-tuning is necessary. In the German *Wiktionary* we check for the keywords “englisch”, “engl.”, “Anglizismus” and special *Wiktionary* markups indicating that the word is from the English language. To avoid false positives for loan translations or ancient common origins, we exclude words containing keywords like “Übersetzung” (translation) and “altenglisch” (Old English) in the origin section. The German *Wiktionary* also contains many conjugations and word forms that are linked to their principal form. We follow such links

and classify a word based on the *Wiktionary* entry of its principal form. The Afrikaans *Wiktionary* is not as comprehensive. A section about word origin is not available. Thus, we can only rely on the *Wiktionary* markup indicating that an entry describes an English word. Words which are not found at all in *Wiktionary* are treated as *non-English* words in our evaluation. When we combine all features, we give those words a neutral value.

To speed up the procedure and reduce the load on the *Wiktionary* servers, we used a *Wiktionary* dump, which is available to download all content of a language's *Wiktionary*. First, we extracted the relevant parts about the words' language and origin from *Wiktionary*. From this smaller file the actual *Wiktionary Lookup* of the words from our test sets can be done faster. Table 7.7 shows the portion of words found in Wiktionary. We found between 71% and 75% of all words from our German test sets in the German *Wiktionary* edition. More than half of the Anglicisms annotated in our test sets also have entries in the German *Wiktionary* edition indicating that the words are from the English language. In contrast, the Afrikaans *Wiktionary* edition has very few entries and we could find only 3.45% of the words from our test set, most without indicating the word's origin.

Test set	Words found	Anglicisms found
Microsoft-de	71.15%	57.58%
Spiegel-de	74.35%	59.23%
NCHLT-af	3.45%	0.09%

**Table 7.7** – Percentage of words found in Wiktionary.

Test set	F-score	Precision	Recall
Microsoft-de	<b>52.44%</b>	71.07%	41.55%
Spiegel-de	<b>36.85%</b>	36.78%	36.85%
NCHLT-af	<b>9.02%</b>	25.00%	5.50%

**Table 7.8** – Performance of the Wiktionary Lookup Feature.

Table 7.8 shows the Anglicism detection performance of our *Wiktionary Lookup Feature*. On the German test sets the *Wiktionary Lookup Feature*'s f-scores are lower than the f-scores of most of our other features. Detecting a much smaller portion of the Anglicism in the test sets, its recall is at least 13% absolute lower than any other feature's. However, in terms of precision - indicating how many non-Anglicisms are wrongly detected - the *Wiktionary Lookup* is one of our best features. Hence it contributes to a good performance in the combination of our features as described in Chapter 7.1.3.

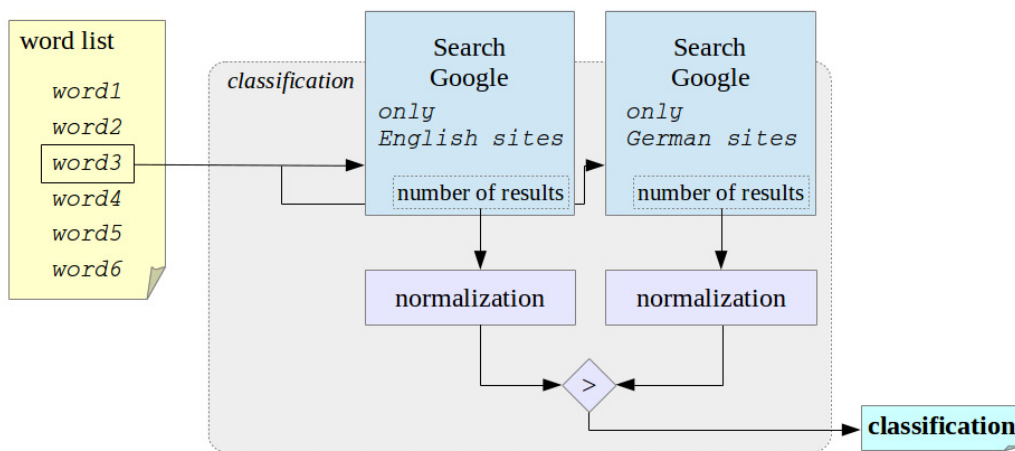


Figure 7.8 – Classification with the *Google Hits Count Feature*.

### Google Hits Count Feature

Our *Google Hits Count Feature* is an implementation of the *Search Engine Module* developed by [Ale08a]. They use the method to detect English words in a two step approach, first filtering potential English words with a dictionary lookup. Many search engines offer the advanced option to exclusively search on websites of a specific language. Given a correct language identification by the search engine, the assumption is that an English word is more frequently used in English, while a German or Afrikaans word is more frequently used in its language [Ale08a]. [Ale08b] notes that since current information is dynamically added, this Web-based approach also deals well with unknown words like recent borrowings which have not yet been entered into dictionaries. Figure 7.8 illustrates the process of the *Google Hits Count Feature*:

1. Search of a word from the test set with search results restricted to English.
2. Search of a word from the test set with search results restricted to the matrix language.
3. Normalization of the number of search results from (1.) and (2.) with the estimated size of the Web in each language.
4. Comparison and classification towards the language for which the normalized number of search results is higher.

As there is much more English than German content on the Web – and for Afrikaans it is just a fraction –, the raw number of search results has to be normalized before comparison. The normalized number of hits of a word  $w$  returned for the search in each language  $L$  is calculated as:

$$\text{hits}_{\text{norm}}(w, L) = \frac{\text{hits}_{\text{absolute}}(w, L)}{\text{Web-size}(L)}$$

$\text{hits}_{\text{norm}}(w, \text{'English'})$  and  $\text{hits}_{\text{norm}}(w, \text{'matrix language'})$  are compared to classify the word  $w$  depending on which normalized score is higher. Following [Ale08a], we need the estimated size of the Web corpus which is accessible through the search engine in a specific language. This number,  $\text{Web-size}(L)$ , is used to normalize the search hits, as shown above. The estimation method was developed by [GN00]:

1. The frequencies of the 20 most common words are calculated within a large text corpus of the language.
2. The search engine limited to pages of the language is queried for each of these most common words.
3. The number of search hits for each word is divided by its frequency in the training text. The resulting number is an estimate of the total number of search results in that language.

Like [GN00], we then remove the highest and the lowest estimates as potential outliers. The average of the rest of the estimates is the final estimation of the Web corpus size in the language. For English and German we used the most common words and frequencies from [GN00] and calculated the Web corpus sizes based on new *Google* hits counts for these words. While [GN00] report an estimated number of 47 billion English words, we estimated 3 trillion words in 2013, as shown in Table 7.9. For Afrikaans information of the most common words and frequencies was not provided by [GN00]. Thus, we calculated the 20 most common words and their frequencies from the Afrikaans Bible. The normalization based on the Bible text resulted in better detection performance than with a normalization on current news articles from [www.rapport.co.za](http://www.rapport.co.za). Table 7.9 shows our estimations of the total number of accessible search results in each language.

Table 7.10 shows the Anglicism detection performance of our *Google Hit Counts Feature*. On the *Spiegel-de* test set this is our best feature. The *Google Hit Counts Feature* also has the highest precision among our features

Language	Estimated size of Web	Ratio to English
English	3,121,434,523,810	
German	184,085,953,431	1 : 17
Afrikaans	6,941,357,100	1 : 450

**Table 7.9** – Estimated size of the Web in different languages.

Test set	F-score	Precision	Recall
Microsoft-de	<b>66.30%</b>	71.31%	61.95%
Spiegel-de	<b>49.03%</b>	40.10%	63.08%
NCHLT-af	<b>26.85%</b>	16.48%	72.50%

**Table 7.10** – Performance of the Google Hit Counts Feature.

on both German test sets. Among the false positives (*non-English* wrongly detected) are a lot of abbreviations, proper names and homomorphs<sup>3</sup>. The false negatives (*non-English* words which are not detected) consist of English words very commonly used in German – like “Computer”, “online” or “Facebook”. The normalization gives those words a higher score for the German search than for the English search. Many hybrid words containing English and German characteristics are also among the false negatives.

## Results of the Single Features

Figure 7.9 gives an overview of the Anglicism performance for all features. In particular, our *G2P Confidence Feature* performs well. For all of our features, the Anglicism detection performance is consistently better on the *Microsoft-de* test set than the *Spiegel-de* test set. This large difference in case of different domains was also observed by [Ale05]. As both test sets are German, our features use identical resources for their classification. The vast F-score difference of up to 30% absolute for the same feature is therefore somewhat surprising.

Figure 7.10 gives a more detailed view of the detection performance. It compares all feature performances on *Microsoft-de* and *Spiegel-de* with respect to precision and recall, which make up the F-score.

For most of our features, the recall is similar on both test sets. A comparable portion of the Anglicisms contained in each test set is correctly detected.

<sup>3</sup>Words that are spelled the same in English and German.

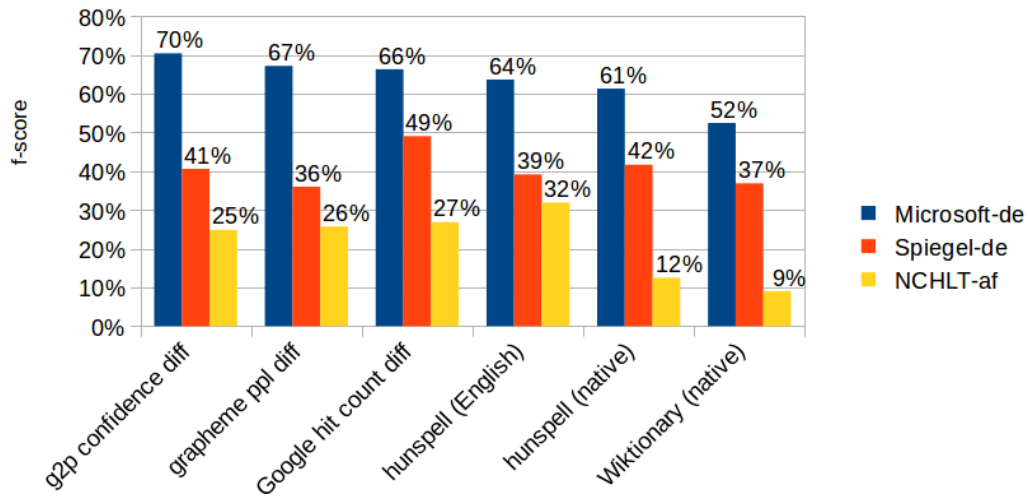


Figure 7.9 – Anglicism detection performance (F-score) of all features.

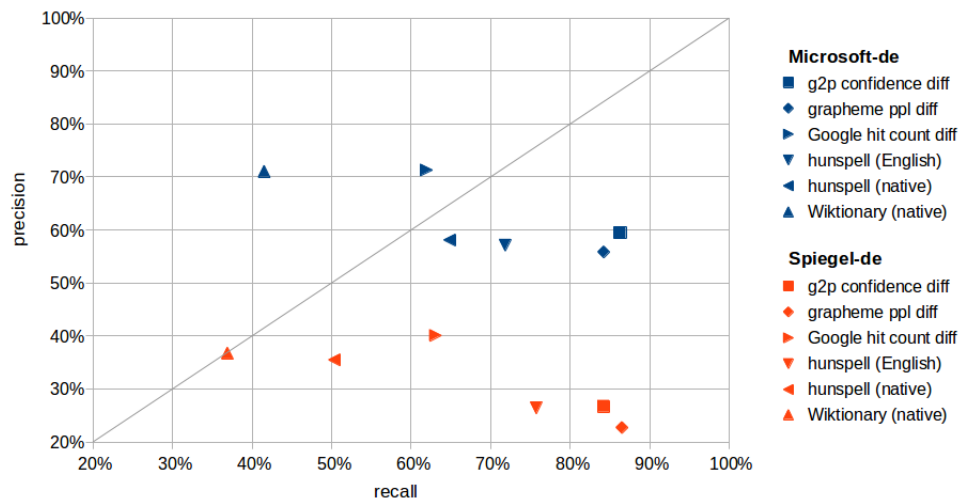


Figure 7.10 – Precision-recall chart of all features on the Microsoft-de and Spiegel-de test sets.

A principle difference between the test sets is visible in terms of precision. All our features consistently have lower precision on *Spiegel-de* with a gap of up to 30% absolute for the same feature.

As presented in Table 7.11, the portion of false positives is very similar between both test sets. This indicates that depending on the feature between 2.6% and 12.07% of *non-English* words are wrongly detected as Anglicisms. The different test sets do *not* have a big influence on this false positive rate.

Feature	Microsoft-de	Spiegel-de
G2P Confidence	10.43%	9.47%
Grapheme Perplexity	11.80%	12.07%
Hunspell (non-English)	8.29%	3.76%
Hunspell (English)	9.56%	8.66%
Wiktionary	3.00%	2.60%
Google Hit Counts	4.42%	3.87%

**Table 7.11** – Portion false positives (Non-Anglicisms wrongly detected)

Precision is defined as  $\frac{\sum \text{true positive}}{\sum \text{tested positive}}$ . Consequently, on *Microsoft-de*, with almost four times as many Anglicisms, the portion of false positives is weighted much less since the features detect a higher absolute number of Anglicisms (true positives). *Microsoft-de*, from the IT domain, contains approximately 15% Anglicisms, *Spiegel-de*, from the general news domain, contains only 4% Anglicisms.

The different F-scores between *Microsoft-de* and *Spiegel-de* are due to the different portion of Anglicisms in the test sets. The domain of the test set and the resulting amount of Anglicisms turn out to play a big role.

### Combined Features

For the combination of the above described features we experimented with *Voting*, *Decision Tree* and *Support Vector Machine* (SVM) methods.

To reach a classification based on all features, all Boolean detection hypotheses of the separate features are summed up in a *Voting*:

1. Separate classification by all features of a word from test set.
2. Calculation of the sum of all separate classification results.
3. Final classification by comparing the vote count to a threshold.

We consider a feature classifying the word as *English* with +1 and a feature classifying the word  $w$  as *non-English* as -1. An exception is the *Wiktionary Lookup Feature* (see Section 7.1.3): Its contribution in the *Voting* can also be 0 in case the word is not found in the native *Wiktionary*. The vote( $w$ ) for a word  $w$  is calculated as:

$$\text{vote}(w) = \text{Classified}_{\text{English}}(w) - \text{Classified}_{\text{non-English}}(w)$$

The final hypothesis of the *Voting* is based on a threshold  $T$  for this vote. With  $T > 0$  more than half the features need to vote for a word to be *English*. The threshold was chosen through a 10-fold cross-validation on each test set. Table 7.12 compares the optimal thresholds, which vary for our different test sets. Particularly, the detection performance on *NCHLT-af* is significantly improved if some features are not included in the vote. For the final method for Afrikaans, we therefore use a *Voting* without the *Google Hit Counts Feature*.

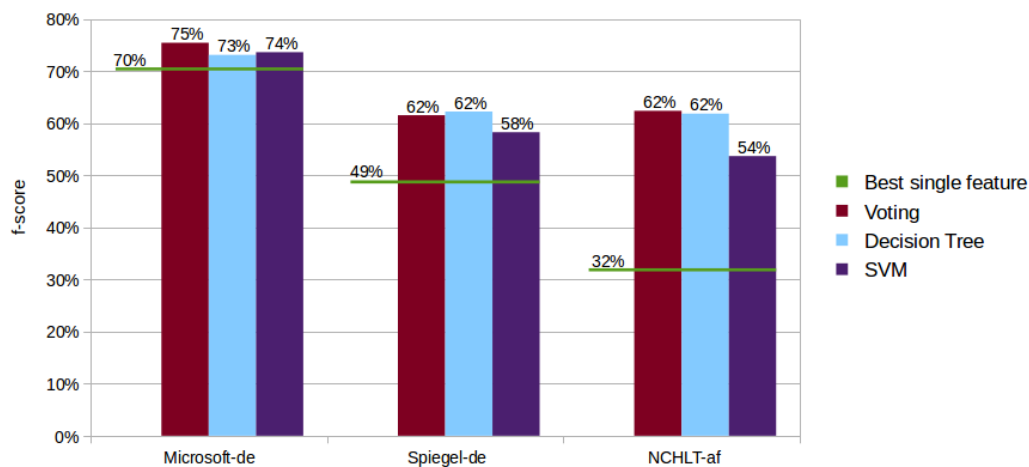
Test set	Threshold = 0	Optimal threshold		Best single feature
	F-score	F-score	Threshold	
Microsoft-de	75.44%	75.44%	0	70.39%
Spiegel-de	56.78%	61.54%	1	49.03%
NCHLT-af	35.33%	51.66%	4	31.90%

**Table 7.12** – Detection performance of *Voting* with different thresholds.

We also experimented with *Support Vector Machines* (SVM) with a linear kernel as a powerful state-of-the-art classification method [SC08] and *Decision Trees* as a common classification method [Qui86]. For those algorithms we used the default parameters of Matlab and trained each test set separately in a 10-fold cross-validation. The information from the single features is given as Boolean input. Like for the *Voting*, the input from a feature classifying the word as *English* is +1 and from a feature classifying the word as *non-English* is -1. The *Wiktionary Lookup Feature* can also be 0 if the word is not found in the native *Wiktionary*. With Boolean features as input for the decision tree, we discard some information. It may help the Anglicism detection if the decision tree receives the “confidence” of each feature’s hypothesis. Thus, we additionally experimented with continuous feature input. This improves the detection performance on *Microsoft-de* but deteriorates it on *Spiegel-de*. Consequently, we do not use continuous features in our final combination method.

The performance of the different combination approaches are shown in Figure 7.11. For two out of our three test sets our simple *Voting* gives the best overall results – although we only use training data to fine-tune the vote threshold, whereas *Decision Tree* and *SVM* learned more complex relations from the input features. As we did not spend much time on fine-tuning the





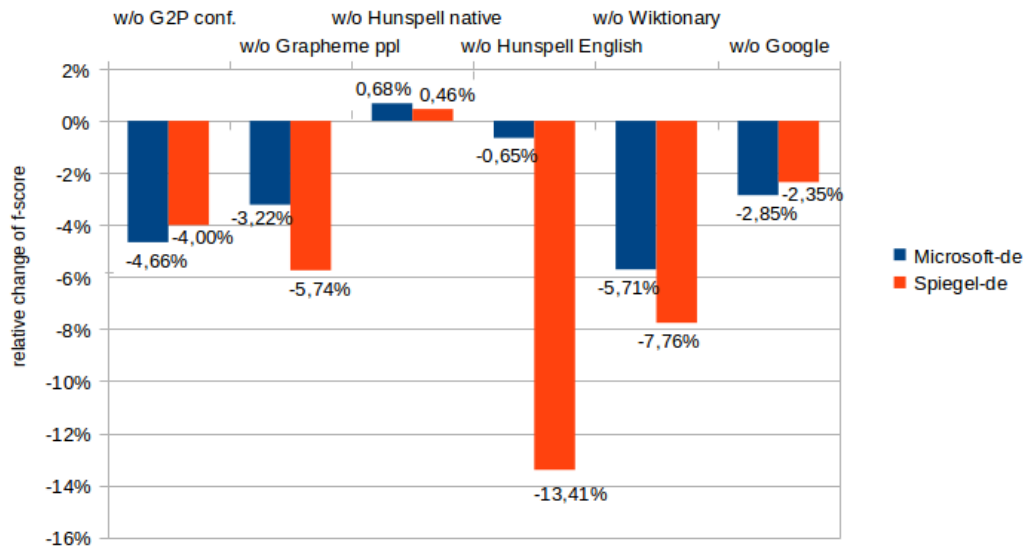
**Figure 7.11** – F-scores (%) of the feature combinations and the best single features.

parameters of the *SVM*, some further improvements may be possible. The improvements compared to the best single feature are striking, almost doubling the F-score on the *NCHLT-af* test set. In particular, on the Afrikaans test set, for which all our single features had poor detection performances, the combination gives a massive relative improvement of 44.74%.

We further analyzed the contribution of the single features for our two German tests. As shown in Figure 7.12, the *Wiktionary Lookup* provides important additional information and supports the detection in feature combinations. For *Spiegel-de*, the *German Hunspell Lookup Feature* gives most improvement. On both German test sets the *Wiktionary Lookup Features* is an important part of the *Voting*. Apart from the *German Hunspell Lookup*, which gives a very minor improvement if left out, all features contribute to good performance of the *Voting*.

### Challenges: Abbreviations, Hybrid, Other Foreign Words

We have annotated *hybrid English words*, *other foreign words* and *abbreviations* in our German test sets. The classification of these words is somewhat ambiguous because they are either both German and English (*hybrid English words*) or not clearly any of the two (*abbreviations*, *other foreign words*). In oracle experiments we removed these types of words from the test sets before evaluating the F-score. The results show the performance of our features only on the unambiguous test words. Figure 7.13 compares the results of

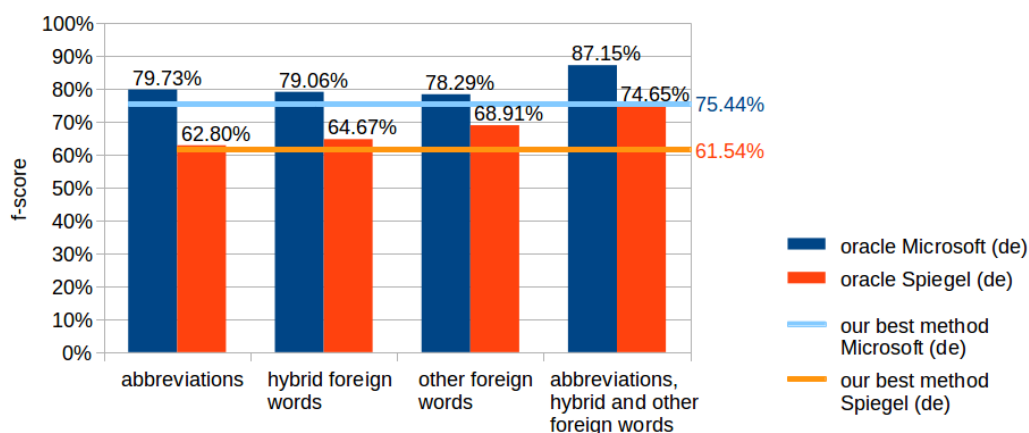


**Figure 7.12** – Relative F-score change (%) of *Voting* if one feature is left out.

our *Voting* when one or all of those word categories are removed from the test set. After manually removing those words, we achieve a relative improvement of up to 47.70%. The varying contribution of the different word categories depends on the composition of the test set. *Spiegel-de* has – relative to the whole test set – more *other foreign words* and less *abbreviations* and *hybrids*. A lot of potential improvement remains in handling these special word categories. We did not experiment with this, but word stemming and compound splitting algorithms seem a good way to deal with *hybrid English words*. *abbreviations* might be filtered using regular expressions or an absolute grapheme perplexity threshold.

## Pronunciation Dictionary Generation

Based on our Anglicism detection, we produced pronunciations for the 2,276 words in our German IT corpus and compared their phoneme error rate to the reference pronunciations. Then, we compared the resulting quality to the quality of pronunciations which have been exclusively produced with our German grapheme-to-phoneme model (*German G2P Model*) to reflect the case of not dealing with Anglicisms at all. Furthermore, we compared it to the quality of pronunciations having been generated with mixed language grapheme-to-phoneme models trained with different fractions of word-pronunciation pairs from the English CMU dictionary and German *GP-de* dictionary. For



**Figure 7.13** – Performance (F-score) of *Voting* after removing difficult word categories from the test sets.

that, all pronunciations were mapped to the German phoneme set of *GP-de* before training the grapheme-to-phoneme model, based on the closest distance in the IPA chart. The *Mixed Language 50:50* model was built with all 40k word-pronunciation pairs from *GP-de* and additional randomly selected 40k word-pronunciation pairs from the CMU dictionary. The *Mixed Language 80:20* model was built from all 40k word-pronunciation pairs from *GP-de* and additional randomly selected 10k word-pronunciation pairs from the CMU dictionary. This ratio is close to the actual portion of Anglicisms contained in the corpus. Finally, we applied English our grapheme-to-phoneme model to all words in the test set for comparison (*English G2P Model*).

Our approach to generate pronunciations based on automatic Anglicism detection selects the appropriate grapheme-to-phoneme model for each word:

- Pronunciations of words detected as *English* are generated by the English grapheme-to-phoneme model based on the dictionary.
- Pronunciations of words detected as *not English* are generated by the German grapheme-to-phoneme model based on *GP-de*.

As shown in Table 7.13, our approach to use automatic Anglicism detection prior to the pronunciation generation produces a far lower phoneme error rate than generically using a German grapheme-to-phoneme model. The phoneme error rate is reduced from 4.95% to 1.61%. The mixed language grapheme-to-phoneme models produce a phoneme error rate even slightly higher than the single German grapheme-to-phoneme model.

	PER
Automatic Anglicism Detection	1.61%
German grapheme-to-phoneme model	4.95%
Mixed Language 80:20	4.97%
Mixed Language 50:50	5.46%
English grapheme-to-phoneme model	39.66%

**Table 7.13** – Phoneme error rate (%) of different approaches to generate pronunciations for the German IT corpus.

### 7.1.4 Summary

To detect Anglicisms in text of a matrix language, we developed a set of features and combined those to further improve the performance. Our features are based on grapheme perplexity, G2P confidence, native *Hunspell* lookup, English *Hunspell* lookup, *Wiktionary* lookup, and *Google* hits count. With the *G2P Confidence Feature* we developed an approach which incorporates information from a pronunciation dictionary. This was our most successful single feature. The *Wiktionary Lookup Feature*, leveraging web-derived information, is also a new approach that especially supported the performance of feature combinations. None of our single features rely on text with Anglicisms annotated for training. The features are instead based on other resources like unannotated word lists or dictionaries and are portable to other languages. The combination of the diverse set of features boosted detection performance considerably, especially for the test sets on which the separate features did not bring satisfactory results. A separate handling of the detected Anglicisms from the matrix language words based on our results enhanced our automatic pronunciation dictionary generation process.

## 7.2 Lexical Adaptation for Non-Native Speakers

Recognizing low proficiency non-native speech is still a challenge with error rates two or three times higher than those for native speech [Tom00a]. The reasons for the high error rates are: In addition to the direct translations of grammar constructs from the first language, the non-native speakers tend to pronounce the foreign language phonemes differently from the native speakers. Neuroscience research shows that learning to discriminate between

native and non-native sounds happens during the infant's first year [CK10]. As a consequence, people acquiring their second language at the age of seven and later cannot perfectly discriminate the non-native sounds.

Many obstacles contribute to the low performance when automatic speech recognition is applied to foreign speech. Some of the speaker-related factors having negative impact on automatic speech recognition performance for non-native speech are [TW01, TB99]:

- High intra- and inter-speaker inconsistency of the phonetic realizations.
- Different second language acquisition methods and backgrounds, thus different acoustic or grammatical realizations and proficiency levels.
- The speakers' perception of the non-native phones.
- Higher cognitive load due to the non-nativeness.
- Reading errors in read speech.
- Slower reading with more pauses in read speech.
- Grammatically incorrect phrases in spontaneous speech.

Another general difficulty in building accent-specific automatic speech recognition systems is the lack of accented speech data for training. There are few databases with accented speech and often the speakers have different mother tongues, which makes it difficult for the researchers to draw conclusions and compare the methods they experiment with. A widely used approach is to adapt native automatic speech recognition systems to the non-native condition as it does not require the amount of data which is usually necessary to build automatic speech recognition systems from scratch.

English is the language with the highest amount and percentage of non-native speakers in the world (approximately 500 million to 1 billion second language compared to 340 million first language speakers [Mey09, Cry04]). Due to this fact and the free availability of the speech accent archive of the George Mason University (GMU) [Wei10] containing phonetic transcriptions of native US English and accented English, we investigated if this database is helpful to rapidly and economically adapt the pronunciation dictionary and improve the automatic speech recognition performance of accented English. With almost 2,000 speakers from approximately 341 accents to date, this speech accent archive has a potential to generate pronunciation variants for a variety of accents without any required linguistic knowledge. The archive is continually growing. We observed an increase of more than 40 accents and 1,000 speakers since October 2010.

To accommodate for non-native pronunciations, we follow two directions: In addition to the modification of the dictionary to better reflect the non-native pronunciations – often denoted as “lexical adaptation” in literature, we investigated its impact to state-of-the-art techniques for acoustic model adaptation.

To rapidly and economically adapt the dictionary, our proposed methods for the dictionary modification are data-driven. Consequently, no language-specific rules are necessary: The idea is to extract a parallel corpus of phoneme sequences from phonetic transcriptions of native US English and accented English from the speech accent archive of the George Mason University. With this corpus, statistical machine translation models are generated to translate the US English pronunciations in the US English CMU dictionary into accented pronunciations which are then used as new pronunciation variants. Compared to other approaches using phoneme recognizers to retrieve the phonetic transcriptions of native US English containing phoneme errors, we leverage a source for qualified phonetic transcriptions of more than 340 accents which have been carefully generated by experienced native English transcribers [Wei10].

As described in the next section, pronunciation variants are usually generated with linguistic rules or with the help of phoneme recognition. The manual production of rules is precise but can be time-consuming and costly, while phoneme recognition can be erroneous, especially for accented speech. To the best of our knowledge, we are the first who use an online source of manual transcribed transcriptions for the lexical adaptation.

## 7.2.1 Related Work

### Lexical Modeling for Accented Speech

The task is to adapt the native target language’s pronunciations in a given dictionary so that they reflect the pronunciations of non-native speech. Usually, the dictionary is enriched with pronunciation variants fitting to the accented speech.

Information about the phonological structure of the speakers’ native language and common mistakes made when speaking the target language can be used to create a set of context-sensitive variant generation rules [Tom00b, GE03]. However, data-driven methods are suggested in literature since manual pron-

unciation production and defining rules are time-consuming and mostly native speakers or linguists need to be hired for this task.

In related works, pronunciation variants are produced in a data-driven way as follows: A phoneme recognizer is applied to the target speech data. Then, the resulting phoneme sequence is aligned to native pronunciations from existing dictionaries [HWP96b, HW98, AKS00, Tom00b]. This way the generated alignment is used to extract rules further applied to convert existing pronunciations into new ones. The challenge is to find systematic phoneme modifications (substitutions, deletions, insertions) which originate from the accent and abstain from additional alignment and phoneme errors [GE03, RBF<sup>+</sup>99, HWP96a, AKS00, F199].

### Acoustic Modeling for Accented Speech

Modifying the acoustic models results in greater word error rate improvement compared to lexical modeling methods [Tom00b, GE03].

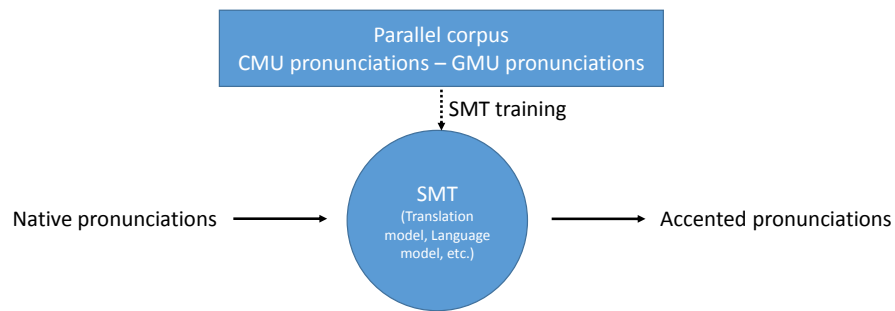
Several techniques to adapt an existing acoustic model have been analyzed, e.g. polyphone decision tree specialization [WSW03, SW00, WS03], multilingual weighted codebooks [RGN08], HMM modification [BFIH06]. However, two adaptation techniques proved to be very successful for the recognition of accented speech [BKKB98, VLG10] and are used by most research groups: Maximum a Posteriori (MAP) [GL92] and Maximum Likelihood Linear Regression (MLLR) [Gal98]. Experiments show that MAP adaptation outperforms MLLR when bigger amounts of adaptation data is available [TW01]. The reason is that MAP adaptation collects statistics and then adjusts only parameters of the distributions corresponding to phonemes which are observed in the calibration data. In contrast, the MLLR approach defines a general model which shifts all distribution parameters mean and standard deviation towards the observed data according to some pre-calculated model. Since first adapting the acoustic model with MLLR and then on top with MAP proved to outperform the single adaptations (*MAP over MLLR*) [GK99], we applied both adaptation techniques in that order in our experiments.

Although experiments have shown significant improvements with modifications in the acoustic model, deletions and insertions and other perturbation of the phonological structure are inappropriately modeled just by increasing the complexity of the acoustic model and acoustic model adaptation [GE03]. To reflect this, [GE03], [Woo99] and [HCZL00] recommend to extend the dictionary to also account for such non-native pronunciations. This extended dic-

tionary should then be combined with the adaptation of the acoustic model to account for both – phonemes pronounced in a non-native way as well as non-native phoneme sequences.

## 7.2.2 Pronunciation Variants Generation

As outlined in Figure 7.14, our method to adapt the pronunciation dictionary in a low-cost and rapid way is to translate existing native pronunciations into accented pronunciations. For that, we automatically modified pronunciations of the native English CMU dictionary using statistical machine translation (SMT) models derived from parallel human-made IPA-based transcriptions from the online GMU speech accent archive [Wei10].



**Figure 7.14** – Statistical machine translation based pronunciation variant generation.

For our experiments, we collected the *GlobalPhone* accented speech database comprising English with Bulgarian (*bg*), Chinese (Mandarin or Cantonese) (*zh*), German (*de*) and Indian (*in*) accents [Mih11]. The Indian part was spoken by native speakers of Hindi, Marathi, Bengali, Telugu, and Tamil. Although English is an official language in India, our study described in [Mih11] showed that the speakers from India have a specific accent. Bulgarian is from the Slavic language family, Mandarin and Cantonese are members of the Sino-Tibetan family, German is a Germanic language and all Indian languages descend from the Indo Iranian language family [Mey09]. In total, we recorded 60 accented speakers reading between 30 and 40 sentences from the Wall Street Journal (WSJ) [GGPP93] which are also available from native English speakers. The majority of topics are economy related news. The read sentences are the same across the accents. The speakers from India



have spent on average two years as residents in the USA, the Chinese speakers approximately 2.5 years. The German and the Bulgarian speakers spent 4.5 months and less than a month in the US. More information about our accented speech corpus including an accent level analysis is given in [Mih11].

For each accent, the utterances of five speakers define the *test set*, five speakers are in the development set (*dev set*) and additional five speakers are used for the acoustic model adaptation experiments.

Our *baseline* system is trained with 15.9 hours from 103 native speakers of American English reading WSJ articles [GGPP93]. No adaptation techniques are used in the acoustic model training. The vocabulary size is approximately 60k. The pronunciations for the words are covered with 85k US English pronunciations from the CMU dictionary [cmu] (*baseline dictionary*).

### GMU Speech Accent Archive

The GMU speech accent archive [Wei10] uniformly presents a large set of speech samples from a variety of language backgrounds. Native and non-native speakers of English read the same paragraph and are carefully transcribed. The archive is used by people who wish to compare and analyze the accents of different English speakers. The archive is freely available online. We used the phonetic transcriptions of the accented archive to build the statistical machine translation system which translates the native to accented pronunciations [Mih11]. Each speaker in the database reads the same phonetically rich paragraph:

```
Please call Stella. Ask her to bring these things with
her from the store: Six spoons of fresh snow peas, five
thick slabs of blue cheese, and maybe a snack for her
brother Bob. We also need a small plastic snake and a
big toy frog for the kids. She can scoop these things
into three red bags, and we will go meet her Wednesday
at the train station.
```

The read text contains 69 words, of which 55 are unique. To build individual statistical machine translation systems for our accents, we used the phonetic transcriptions of 30 speakers with the same Indian accents as in our audio data, 43 speakers with Mandarin and Cantonese accent, 22 speakers with German accent and 11 speakers with Bulgarian accent from the database. The phonetic transcriptions were produced by two to four experienced native English transcribers, following the 1996 version of the IPA chart.

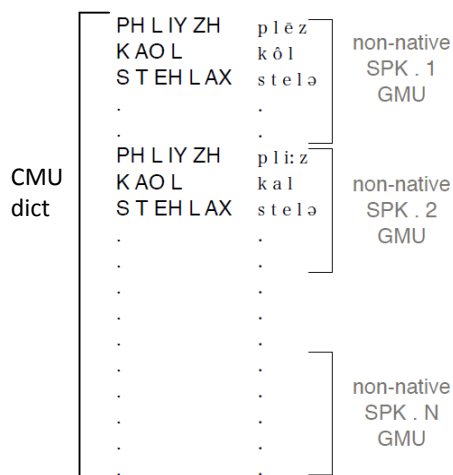
### Statistical Machine Translation Based Pronunciation Variant Generation

The word error rates of the baseline system confronted with the accents (*bg*, *zh*, *de*, *in*) and native English (*en*) are given in Table 7.14. The baseline system which is trained with native English speech has 13.85% word error rate on the native English development set and 14.52% on the corresponding test set. Its word error rates on the accented test sets are relatively close to each other and about 50% absolute worse than on native English. For *de*, one speaker in the development set has a much lighter accent than the others which results with 45.19% in a much lower word error rate.

	<i>en</i>	<i>bg</i>	<i>zh</i>	<i>de</i>	<i>in</i>
dev set	13.85	62.98	76.63	45.19	74.13
test set	14.52	64.24	65.97	64.78	64.45

**Table 7.14** – Word error rate (%) of the baseline system tested with accented speech.

As outlined in Figure 7.15, the pronunciations in the GMU accent archive and the native English pronunciations from the CMU dictionary are used to build the parallel corpus.



**Figure 7.15** – Parallel corpus for the phoneme-level statistical machine translation-based pronunciation variant generation.

We applied the following steps to generate accented pronunciations for each accent as follows [Mih11]:

1. For each word in the phonetic transcriptions of the GMU speech accent archive, assign the accented pronunciation to the CMU dictionary pronunciation.
2. Map the GMU accented pronunciations to the phoneme set of the CMU dictionary based on the closest distance in the IPA chart.
3. Compute a 3-gram phoneme-level language model from the GMU accented pronunciations (with the help of the SRI Language Modeling Toolkit [Sto02]).
4. Train further models for the statistical machine translation system with the parallel corpus (using Moses Package [KHB<sup>+</sup>07] and GIZA++ [ON03]).
5. Translate all pronunciations from the baseline dictionary into their accented version.

Since the complete dictionaries are generated from pronunciations of only 55 unique words, they contain “poorly estimated” variants, resulting from the lack of information about all possible phoneme combinations. To remove the inconsistent and faulty pronunciation variants, different filter methods, described in [Mih11], were implemented and analyzed. The accented pronunciations were filtered by using the Levenshtein distance metric, forced-alignment information or a combination of both filter methods. The remaining new pronunciation variants were then included in the native English dictionary. A simple filtering based on the Levenshtein distance as a similarity measure between new and existing variants performed best. Finally, from 45-55k initial variants, 31-33k were added to the baseline dictionary.

	<i>bg</i>	<i>zh</i>	<i>de</i>	<i>in</i>
baseline (test)	64.24	65.97	64.78	64.45
lexical adaptation	64.79	65.34	62.83	62.61
rel. change	-0.86	+0.95	+3.01	+2.85

**Table 7.15** – Word error rate (%) on test sets with lexical adaptation.

As demonstrated in Table 7.15, the lexical adaptation gives small but not significant improvements for *zh*, *de*, and *in* except for *bg*, compared to the baseline system. Additionally, we investigated if the modified dictionaries have a positive impact in the combination of lexical and acoustic model adaptation.

### 7.2.3 Combination of Lexical and Acoustic Model Adaptation

The biggest word error rate reduction is expected to be the result of a combined acoustic model and lexical accent adaptation. For this purpose, the most successful methods were combined.

	<i>bg</i>	<i>zh</i>	<i>de</i>	<i>in</i>
baseline (dev)	62.98	76.63	45.19	74.13
MAP over MLLR	54.44	49.03	35.12	38.74
rel. change to baseline	+13.56	+36.02	+22.28	+47.74
+lexical adaptation1	54.38	48.36	<b>34.32</b>	<b>38.08</b>
rel. change to baseline	+13.66	+36.89	+24.05	+48.63
rel. change to MAP over MLLR	+0.11	+1.37	+2.28	+1.70
+lexical+adaptation2	<b>54.14</b>	<b>48.00</b>	35.12	38.56
rel. change to baseline	+14.04	+37.36	+22.28	+47.98
rel. change to MAP over MLLR	+0.55	+1.91	+0.00	+0.46

**Table 7.16** – Word error rate (%) on development sets with lexical and acoustic model adaptation.

Table 7.16 illustrates the word error rates on our development sets with lexical and acoustic model adaptation. With the acoustic model adaptation, the parameters originally trained with native English are shifted. Therefore, phonemes which are substituted by accented speakers are covered by the acoustic model adaptation and do not have to be additionally modeled in the dictionary. Since it is even possible, that the lexical adaptation of substituted phonemes can have a negative impact on the acoustic model adaptation, we kept in the dictionary only new variants containing at least one insertion or deletion compared to the original native English pronunciation (*+Lexical Adaptation2*) [Mih11]. As indicated in Table 7.16, this leads to improvement for *de* and *in*. Consequently, we used *+Lexical Adaptation2* to decode the test set for those two accents.

Table 7.17 demonstrates the results on our test sets. We note that the positive effect of our lexical adaptation can be weakened in combination with an acoustic model adaptation. Apart from that, although the lexical adaptation alone deteriorated the baseline system for *bg*, in combination with the acoustic model adaptation we observe slight improvements. With the combination of the lexical and acoustic model approaches, relative improvements of 26.9% for Bulgarian, 33.2% for Chinese, 30.9% for German, and 53.2% for Indian accents are achieved.

	<i>bg</i>	<i>zh</i>	<i>de</i>	<i>in</i>
baseline (test)	64.24	65.97	64.78	64.45
lexical adaptation	64.79	65.34	62.83	62.61
rel. change	-0.86	+0.95	+3.01	+2.85
MAP over MLLR	47.33	44.79	44.89	30.77
rel. change to baseline	+26.32	+32.11	+30.70	+52.26
+lexical adaptation	46.95	44.10	44.74	30.19
rel. change to baseline	+26.91	+33.15	+30.94	+53.16
rel. change to MAP over MLLR	+0.80	+1.54	+0.33	+1.88

**Table 7.17** – Word error rate (%) on test sets with lexical and acoustic model adaptation.

### 7.2.4 Summary

The analyses prove that the GMU speech accent archive can be a helpful source for the rapid and low-cost generation of English automatic speech recognition systems for non-native speakers. The pronunciation variants which we generated based on information from the phonetic transcriptions using our statistical machine translation approach give information not covered with acoustic model adaptation. However, the acoustic model adaptation gives a lot more improvement than the lexical adaptation. With our focus on lexical adaptation, we did not use the recordings available at the GMU speech accent archive. Future work may include an acoustic model adaptation with these recordings.



# Dictionary Generation for Non-Written Languages

---

For non-written languages, our goal is to extract word pronunciations of a target language with the help of cross-lingual word-to-phoneme alignments between the target language phonemes and the words of a written translation in another source language. We present two different methods (source word dependent and independent clustering) that extract word pronunciations from word-to-phoneme alignments and compare them. We show that both methods compensate for phoneme recognition and alignment errors. Finally, we use the extracted pronunciations in an automatic speech recognition system for the target language and report promising word error rates – given that the pronunciation dictionary and language model is learned completely unsupervised and no written form for the target language is required for our approach.

## 8.1 Word Pronunciation Extraction Algorithm

Let  $V_{src}$  be the vocabulary of the source language plus the NULL token and  $PhonemeSet_{trgt}$  the phoneme set of the target language. The data source which we explore in our scenario is a set  $DB \subset V_{src}^+ \times PhonemeSet_{trgt}^+$  of pairs containing a written sentence and its spoken translation. The sentence

is represented as a sequence of source language words and the translation is represented as a sequence of target language phonemes. As described in Section 3.3.3, we use the PISA Alignment Tool to find word-to-phoneme alignments for each sentence-phoneme sequence pair in  $DB$ . An alignment  $A_{s,t}$  of a pair  $(s, t) \in DB$  consists of a mapping between tokens in  $s$  and phoneme subsequences in  $t$ . We formalize  $A_{s,t}$  as a word over an alphabet containing pairs of source language words and target language phoneme sequences:

$$A_{s,t} \in (V_{src} \times PhonemeSet_{trgt}^+)^+$$

Each element in  $A_{s,t}$  contains a putative target language *word* represented by its phonemes and the source language word aligned to it. We postulate that the source language words are elements in  $s$ , and that concatenating all phonemic representations of the target language words results in the complete phoneme sequence  $t$ .

The difference between source word dependent and independent clustering is discussed in the next sections. Our general algorithm for extracting word pronunciations from word-to-phoneme alignments is illustrated in Figure 8.1<sup>1</sup>. It consists of five steps:

1. **Word-to-phoneme Alignment:** Use the PISA Alignment Tool to generate word-to-phoneme alignments  $A_{s,t}$  for each pair  $(s, t) \in DB$ .
2. **Alignment Pair Collection:** Collect all cross-lingual mappings in the alignments in the set  $P$ :<sup>2</sup>

$$P \leftarrow \{(w_s, w_t) \in V_{src} \times PhonemeSet_{trgt}^+ \mid \exists (s, t) \in DB : (w_s, w_t) \in A_{s,t}\}$$

3. **Phoneme Sequence Clustering:** Conceptually,  $P$  contains lexical translations and pronunciation variants of them. In reality, the set is affected by phoneme recognition and alignment errors. Therefore, group elements in  $P$  into clusters  $C_i$ :

$$C = \{C_1, C_2, \dots, C_n\} \subset 2^P \text{ with } P = \bigsqcup_{i \in [1, n]} C_i.$$

<sup>1</sup>“Sprache zu Sprache Übersetzung” → “Speech to speech translation”, “Sprache die für dich dichtet und denkt” → “Language verses and thinks for you”, “Erkennung von Sprache” → “Speech recognition”)

<sup>2</sup>For technical reasons, we define the  $\in$  sign for a symbol  $x \in \Sigma$  and a word  $w \in \Sigma^+$  as  $x \in w :\Leftrightarrow \exists i \in \mathbb{N} : x = w_i$



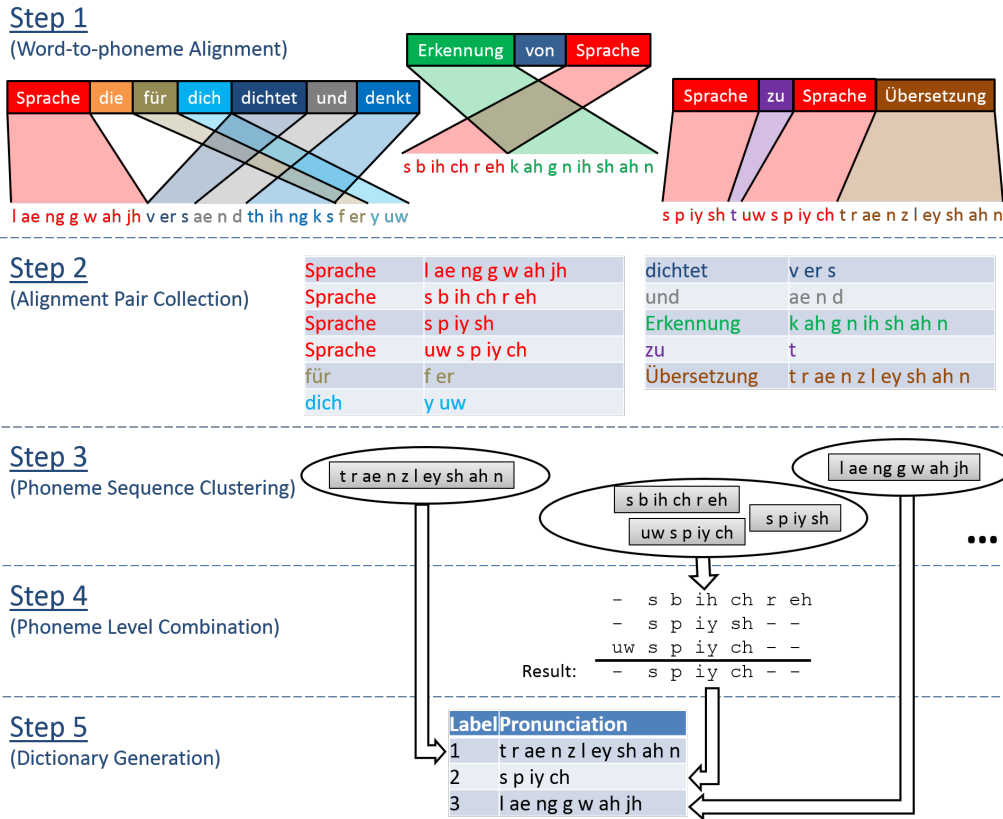


Figure 8.1 – Pronunciation extraction with German-English.

- Phoneme-Level Combination:** At this point, the clusters should contain phoneme sequences representing the same target language word, but differing due to alignment and phoneme recognition errors. Thus we try to reconstruct the correct phoneme sequence for each cluster by merging its elements with our *PLC* method, which we introduced in Section 4.3.2. *PLC* can extract the correct pronunciation even if all elements in the cluster contain errors. We obtain a set  $H \subset PhonemeSet_{trgt}^+$  of  $n$  phoneme sequences ( $|H| = n$ ), which are now assumed to correspond to real target language words.
- Dictionary Generation:** For each pronunciation  $h \in H$ , we choose a word label  $id_h \in \mathbb{N}$  and add both to the pronunciation dictionary *Dict*.

### 8.1.1 Source Word Dependent Clustering

Source word dependent and independent clustering differ in *step 3* (phoneme sequence clustering). Source word dependent clustering extracts pronunciations based on the assumption that phoneme sequences which are aligned to the same source language word are likely to represent the same target language word. They only differ in phoneme recognition and alignment errors. Source word dependent clustering applies a two-level approach (Figure 8.2): First, we form coarse-grained clusters by grouping all elements in  $P$  that have the same source language word. However, this cannot separate different translations of the same source language word from each other: In our example from Figure 8.1, the German word *Sprache* has two different English translations (*Speech* and *Language*). Therefore, a fine-grained clustering algorithm post-processes the source word dependent clusters. We apply the density-based clustering algorithm DBSCAN [EKSX96] ( $\epsilon = 1$ ,  $minPts = 3$ ) implemented in the ELKI [AGK<sup>+</sup>12] environment with the Levenshtein distance metric. Roughly speaking, DBSCAN finds clusters with mutually density-reachable elements. An element is density-reachable if it is within  $\epsilon$ -distance and surrounded by sufficiently many other elements ( $minPts$ ). We use DBSCAN since it does not require the desired number of clusters as input parameter.

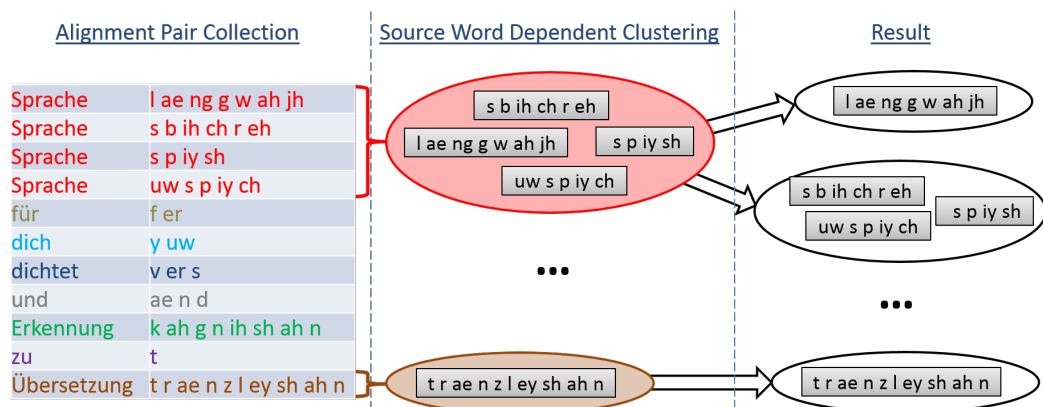
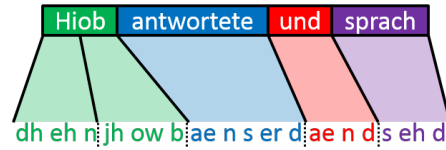


Figure 8.2 – Source word dependent clustering.

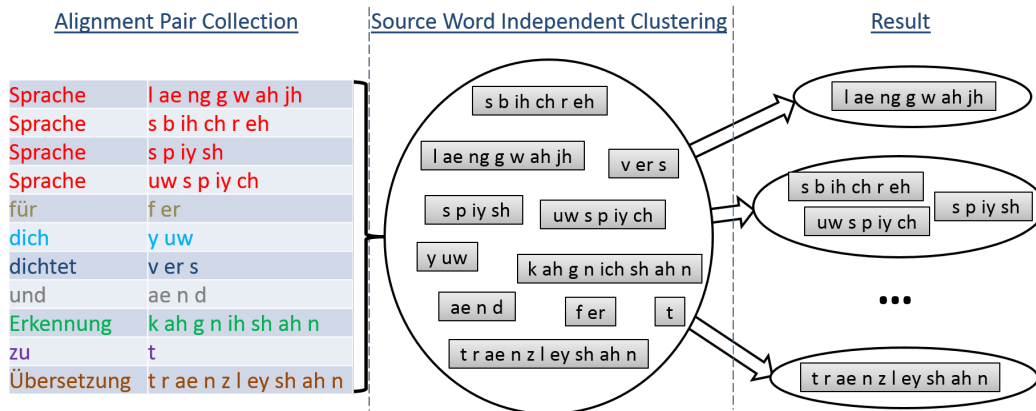
### 8.1.2 Source Word Independent Clustering

Source word dependent clustering in the previous section assumes that phoneme sequences which are aligned to the same source language word are



**Figure 8.3** – German-English example with correct word boundaries but incorrect alignment.

likely to represent the same target language word. Source word independent clustering drops this assumption completely since we observed that the alignments sometimes induce the correct word segmentation, but map the wrong source language words to the segments. For example, the phoneme sequence *dh eh n* (*then*) in Figure 8.3 (Job 19:1, “Then Job answered and said”) is incorrectly aligned to “Hiob” (*Job*), but all the word boundaries are set correctly. Therefore, when extracting pronunciations with the source word “Hiob”, *dh eh n* pollutes the cluster containing *Job* pronunciations, and it cannot be used to extract a pronunciation for *then*. Since there is no equivalent for *then* in the German counterpart, it should be aligned to NULL. However, the NULL cluster in source word dependent clustering is usually very large and widespread: All kinds of words can be modeled as “spurious” and aligned to NULL (see the generative story of *Model 3P* in Section 3.3.3). Therefore, it is hard to extract correct pronunciations from the NULL cluster with source word dependent clustering.



**Figure 8.4** – Source word independent clustering.

Consequently, source word independent clustering [SSVS14a] groups pronunciations in  $P$  not regarding the source language word. Therefore, it replaces the two-level approach of source word dependent clustering by a single-level

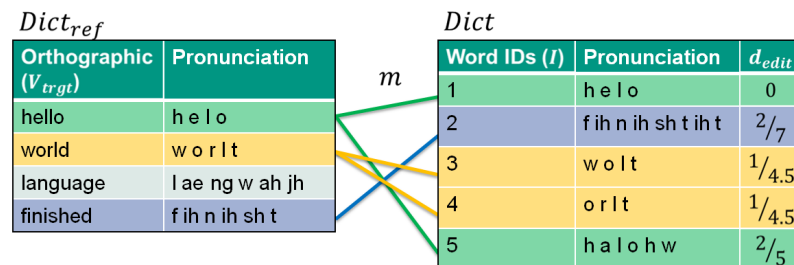
clustering solely based on the Levenshtein distance. Combining DBSCAN with source word independent clustering performs poorly in our experiments, so we use the  $k$ -means [Mac03] algorithm to provide additional hints for the clustering process like the desired number of clusters. The means are initialized with the  $k$  most frequent elements. However,  $k$  should be initialized close to the actual vocabulary size of the target language which can be estimated using the vocabulary sizes of the source languages. In the latter experiments we assume to know the target language vocabulary size. Section 8.3 discusses the performance degradations when  $k$  is set too low or too high in detail.

## 8.2 Evaluation Metrics

Let  $I$  be the set of all word labels in the extracted dictionary  $Dict : I \rightarrow PhonemeSet_{trgt}^+$ . We measure the structural quality of  $Dict$  by the **out-of-vocabulary rate (OOV)** on running words. The out-of-vocabulary rate cannot be calculated directly since  $Dict$  contains word labels instead of written words consisting of graphemes. Therefore, a mapping between the word labels and the written words is required. Let  $V_{trgt}$  be the target language vocabulary (written words) and  $Dict_{ref} : V_{trgt} \rightarrow PhonemeSet_{trgt}^+$  the reference dictionary with the correct pronunciations. The mapping  $m : I \rightarrow V_{trgt}$  assigns each word label to the written word with the most similar pronunciation (illustrated in Figure 8.5):

$$m(n) = \arg \min_{v \in V_{trgt}} d_{edit}(Dict(n), Dict_{ref}(v)) \quad (8.1)$$

where  $d_{edit}$  denotes the edit distance. The set  $m(I)$  of matched vocabulary entries in  $Dict_{ref}$  is then used to calculate the out-of-vocabulary rate on running words (word tokens).



**Figure 8.5** – Mapping between word labels and written words for evaluation.

While the out-of-vocabulary rate indicates the coverage of *Dict* on a Bible text, the **dictionary phoneme error rate (dictPER)** reflects the quality of the extracted pronunciations at the phoneme level. It is defined as the average edit distance between the entries in *Dict* and the closest entry in the reference dictionary *Dict<sub>ref</sub>*:

$$\text{dictPER} = \frac{\sum_{n \in I} d_{\text{edit}}(\text{Dict}(n), \text{Dict}_{\text{ref}}(m(n)))}{|I|} \quad (8.2)$$

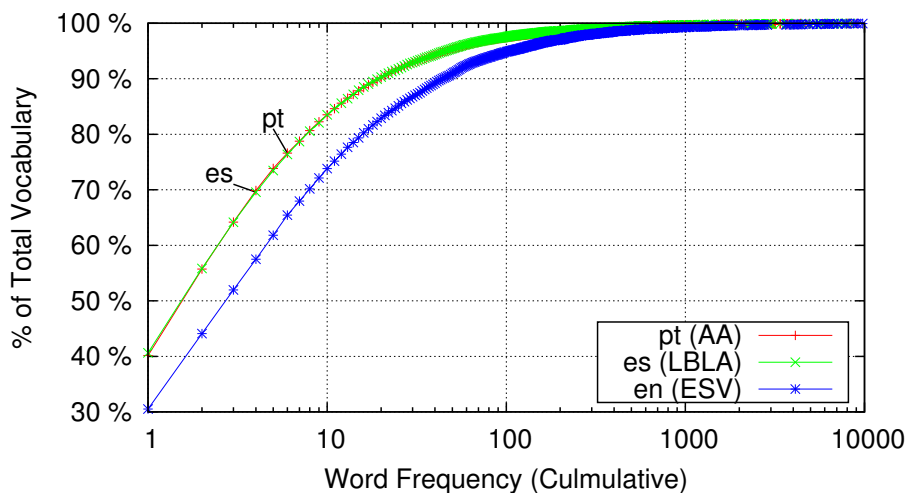
The dictPER can be greater than zero even when we operate on error-free phonetic transcriptions since it does not capture only the remaining phoneme recognition errors: The dictPER is also heavily affected by segmentation errors (inserted or dropped blank between two words or morphemes). Finally, the **Hypo/Ref ratio** indicates how many hypothesis entries in *Dict* are mapped by *m* to a single reference entry in *Dict<sub>ref</sub>* on average ( $|I|$  divided by  $|m(I)|$ ). Ideally, this ratio is 1 as we do not have pronunciation variants in our corpus. The higher the Hypo/Ref ratio, the more pronunciations are extracted unnecessarily.

## 8.3 Evaluation of Extracted Pronunciations

### Corpus

We test our pronunciation extraction algorithms on parallel data from the Christian Bible since it is available in many languages in written form<sup>3</sup> and in some languages also as audio recordings. A variety of linguistic approaches to Bible translation (Dynamic equivalence, formal equivalence, and idiomatic translation [Tho90]) enables us to compare different translations within the same source language. In our experiments, English takes the role of the under-resourced target language. English is by no means under-resourced and comprehensive pronunciation dictionaries are readily available. However, we feel that understanding the target language gives a deeper insight in the strengths and weaknesses of our algorithms. We select an English Bible translation with predominant use of the formal equivalence translation method (i.e. syntactical equivalent and literal)[Bor03]: English Standard Version (ESV) [Cro01]. Figure 8.6 shows that the words in the ESV Bible are distributed approximately according to the Zipfian distribution [MS99]

<sup>3</sup>Extracted from <http://www.biblegateway.com/> (Accessed on 15th May 2014)



**Figure 8.6** – Log-log plot of the word distribution in the ESV Bible.

( $s = 0.98$ , asymptotic standard error of 0.04%): The frequency of any word is inversely proportional to its rank in the frequency table. A large portion of the words has only one occurrence (30.5%), while only 26% occur ten times or more in the text. Some of the function words (*in*, *the*, *and*) are repeated very often, but also content words like *lord*, *god*, *said*, *people*, *Israel* occur frequently. In the Spanish LBLA Bible and the Portuguese AA Bible, words with a frequency of 1 and words with a frequency of 10 or more appear about 10% more than in the ESV Bible. High word frequencies are suitable for our extraction algorithm as we merge more phoneme sequences leading to better error correction as shown in Section 8.5.

Verses in the Christian Bible are identified by unique verse numbers (such as *Galatians 5:22*), which are consistent with verse numbers across all investigated Bible translations. Based on these numbers, we extract a parallel and verse-aligned corpus consisting of 30.6k verses in 15 different written translations in 10 languages (Table 8.1). We refer to the English portion as  $EN_{all}$ .

In initial experiments, we replace the words with their canonical pronunciations and removed word boundary markers to generate the target language phoneme sequences [SSVS13]. Thereby, we simulate a perfect phoneme recognizer (0% phoneme error rate). The pronunciations are taken directly from the CMU pronunciation dictionary (39 phonemes) or generated with a grapheme-to-phoneme model trained on it.

ID	Language	Full Bible Version Name	# running words	Vocab. Size
bg	Bulgarian	Bulgarian Bible	643k	38k
cs	Czech	Bible 21	547k	48k
da	Danish	Dette er Biblen på dansk	653k	24k
de1	German	Schlachter 2000	729k	26k
de2	German	Luther Bibel	698k	21k
en	<i>English</i>	<i>English Standard Version</i>	<i>758k</i>	<i>14k</i>
es1	Spanish	Nueva Versión Internacional	704k	28k
es2	Spanish	Reina-Valera 1960	706k	26k
es3	Spanish	La Biblia de las Américas	723k	26k
fr1	French	Segond 21	756k	26k
fr2	French	Louis Segond	735k	23k
it	Italian	Nuova Riveduta 2006	714k	28k
pt1	Portuguese	Nova Versão Internacional	683k	25k
pt2	Portuguese	João Ferreira de Almeida Atualizada	702k	26k
se	Swedish	Levande Bibeln	595k	21k

**Table 8.1** – Overview of the used Bible translations.

Word-to-phoneme alignments are the basis of our pronunciation extraction. As described in [SSVS12], GIZA++ first calculates initial alignments that are then further refined by the PISA Alignment Tool applying *Model 3P*. However, GIZA++ has restrictions regarding the maximum number of target tokens (100 in our GIZA++ version) or the maximum ratio between the number of source language words and target language phonemes in a sentence pair (*maxfertility* was set to 12 in our experiments). We remove all sentence pairs which violate these restrictions and end up with 23k verses. We refer to this subcorpus as  $EN_{filt}$ .

## Phoneme Recognition

For English, Crossway (the publisher of the ESV Bible) provides high quality verse-level audio recordings of a single male speaker for the entire Bible text<sup>4</sup>. This enables us to test our methods using real phoneme recognizers instead of merely error-free phonetic transcriptions. All recordings ( $EN_{all}$ ) have a total length of 67:16h from which the filtered subcorpus  $EN_{filt}$  consists of 40:28h speech.

<sup>4</sup>Available for purchase at <http://www.crossway.org/bibles/esv-hear-the-word-audio-bible-610-d1/> (Accessed on 15th May 2014)

Acoustic Model	Train+Dev Set	Training Set Size	Dev Set Size	PER on Dev Set	PER on $EN_{filt}$
$AM_{13.1}$	$EN_{all} \setminus EN_{filt}$	24:07h	02:41h	14.0%	<b>13.1%</b>
$AM_{45.1}$	WSJ	15:28h	00:24h	35.9%	<b>45.1%</b>

**Table 8.2** – Acoustic models  $AM_{13.1}$  and  $AM_{45.1}$  applied in a phoneme recognizer on  $EN_{filt}$ .

We trained context-dependent acoustic models for English on two different training sets with different performances on  $EN_{filt}$  to investigate the influence of phoneme recognition errors to our methods.  $AM_{13.1}$  was trained on  $EN_{all}$  without  $EN_{filt}$  ( $EN_{all} \setminus EN_{filt}$ ) and represents a speaker-dependent system without overlapping training and test set.  $AM_{45.1}$  was trained on the Wall-Street-Journal (WSJ) corpus [PB92] and therefore contains speaker independent, corpus mismatched acoustic models. All experiments use the CMUdict phoneme set consisting of 39 phonemes. The preprocessing consists of feature extraction applying a Hamming window of 16ms length with a window shift of 10ms. Each feature vector has 143 dimensions by stacking 11 adjacent frames of 13 Melscale Frequency Cepstral Coefficients (MFCC) frames. A Linear Discriminant Analysis (LDA) transformation is computed to reduce the feature vector size to 42 dimensions. The acoustic model uses a fully-continuous 3-state left-to-right HMM with emission probabilities modeled by Gaussian Mixtures with diagonal covariances (64 Gaussians per state). For our context-dependent acoustic models with different context sizes, we stopped the decision tree splitting process at 2,500 triphones. After context clustering, a merge-and-split training was applied, which selects the number of Gaussians according to the amount of data (50 on average, 125k in total).

For phoneme recognition, a trigram phoneme-level language model was trained on the phonetic transcriptions of the corresponding training sets. Table 8.2 summarizes the phoneme error rates on  $EN_{filt}$  of phoneme recognizers with the acoustic models  $AM_{13.1}$  and  $AM_{45.1}$ .

Both  $AM_{13.1}$  and  $AM_{45.1}$  represent oracle experiments since they were trained using target language transcriptions that are not supposed to be available in our scenario. Acoustic modeling in the target language is a research topic for its own and not our present concern. The interested reader can consult the literature for details, such as [JDG<sup>+</sup>13]. However, we feel that for some possible target languages a phoneme error rate in the magnitude of 45.1% (as we obtain with  $AM_{45.1}$ ) is realistic even without or only little training transcriptions. For example, [VKD08] achieve a phoneme error rate of under 25% for Japanese speech from only one speaker by unsupervised learning of HMMs



and only very light supervision for their HMM label-to-phone transducer (5 minutes of transcribed speech). Recent developments in using Deep Neural Networks in zero acoustic model resource systems (as reported in [GKRR14] in the context of the Babel program [Har14]) are also promising.

SOURCE WORD DEPENDENT CLUSTERING									
Source Lang.	dictPER (in %)			Hypo/Ref Ratio			OOV rate (in %)		
	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$
es3	35.36	40.81	46.26	1.53	1.45	1.29	2.57	3.78	11.37
es2	37.07	41.72	46.59	1.50	1.42	1.28	2.96	4.52	12.44
pt2	37.34	42.06	46.15	1.53	1.43	1.30	2.75	4.07	12.40
fr2	37.09	41.14	45.83	1.45	1.41	1.25	3.59	4.99	13.00
de1	38.84	43.63	53.55	1.50	1.41	1.27	3.20	4.68	13.58
it	38.96	43.08	47.07	1.54	1.45	1.31	2.84	4.10	14.79
de2	36.70	41.09	51.25	1.44	1.37	1.21	4.37	5.81	15.64
fr1	40.93	44.13	47.74	1.48	1.40	1.27	3.64	5.24	13.53
da	38.97	42.69	51.96	1.46	1.40	1.24	3.74	5.85	16.72
pt1	40.11	42.83	46.26	1.50	1.41	1.27	3.79	5.42	13.03
bg	44.26	47.57	57.74	1.71	1.59	1.40	1.96	2.87	8.32
es1	42.25	45.55	48.43	1.53	1.42	1.28	3.75	5.15	13.18
cs	49.00	51.56	60.53	1.68	1.57	1.44	2.58	3.48	8.76
se	50.93	47.86	51.51	1.33	1.27	1.18	8.31	10.25	19.96

Table 8.3 – Dictionary evaluation with source word dependent clustering.

SOURCE WORD INDEPENDENT CLUSTERING									
Source Lang.	dictPER (in %)			Hypo/Ref Ratio			OOV rate (in %)		
	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$
es3	29.63	30.57	32.55	1.58	1.77	1.81	2.49	4.04	6.94
es2	31.20	31.75	33.55	1.60	1.75	1.79	2.70	4.31	7.12
pt2	31.05	31.52	33.15	1.62	1.77	1.83	2.76	4.26	6.99
fr2	31.20	31.91	34.17	1.61	1.76	1.75	3.03	4.45	6.95
de1	30.98	31.82	34.44	1.64	1.77	1.72	2.98	4.50	6.87
it	31.60	32.02	33.33	1.61	1.77	1.82	2.89	4.53	7.26
de2	31.69	32.50	34.42	1.64	1.76	1.73	3.11	4.56	6.83
fr1	32.93	33.20	35.67	1.64	1.76	1.70	3.47	4.75	6.29
da	33.15	33.58	34.31	1.65	1.80	1.74	3.25	4.65	6.84
pt1	32.95	33.45	34.14	1.66	1.77	1.79	3.41	4.72	7.00
bg	33.99	34.21	34.01	1.67	1.80	1.82	3.39	4.83	7.27
es1	33.70	34.21	35.89	1.66	1.78	1.72	3.65	5.03	7.05
cs	36.92	36.33	36.53	1.74	1.83	1.79	4.53	5.66	7.72
se	37.57	39.27	44.23	1.69	1.66	1.50	4.34	4.98	5.78

Table 8.4 – Dictionary evaluation with source word independent clustering.

In addition to experiments on error-free phonetic transcriptions (designated as  $R_0$  in this section), we also use phoneme recognizers based on the acoustic models  $AM_{13.1}$  and  $AM_{45.1}$  to produce the target language phoneme sequences (designated as  $R_{13.1}$  and  $R_{45.1}$ ). Tables 8.3 and 8.4 show the performance of both source word dependent and independent clustering using the evaluation metrics for all investigated language pairs. The best results are highlighted. The source languages are ordered descending by segmentation accuracy.

There are often big differences between translations within the same source language: For example, with source word dependent clustering, *es3* has 16.3% lower dictPER and 31.5% lower out-of-vocabulary rate than *es1* (relative improvement) since *es3 La Biblia de las Américas* is a very literal translation [Loc86]. Similar gains can be observed within other source languages (Portuguese and French) and source word independent clustering. This suggests that selecting a translation which is as literal as possible (e.g. following the formal equivalence translation method) is a crucial factor for a high overall quality.

Source word independent clustering outperforms source word dependent clustering in terms of dictPER: The values in the first three columns in Table 8.4 are significantly lower than the corresponding values in Table 8.3. On the other hand, the Hypo/Ref ratio is usually better when source word dependent clustering is used. There is no such clear tendency regarding the out-of-vocabulary rate. For example, using *es3* (first row), source word dependent clustering does better than source independent clustering with  $R_0$  ( $2.57 > 2.49$ ), but worse with  $R_{13.1}$  ( $3.78 < 4.04$ ). The opposite is true for *bg* and  $R_0$  ( $1.96 < 3.39$ ) or *es2* and  $R_{13.1}$  ( $4.52 > 4.31$ ). On very noisy phoneme sequences (last column), source word independent clustering always leads to significantly better out-of-vocabulary rates than source word dependent clustering. Therefore, whether source word dependent or independent clustering should be used depends on the application: If the dictionary is required to contain a minimum of unnecessary entries (and thus needs to have a low Hypo/Ref Ratio), source word dependent clustering is the right choice. If the phonetic correctness of the pronunciations is more important (low dictPER) or the input sequences are very noisy, source word independent clustering should be used. In Section 8.6 we elaborate the case when the dictionary is used in an automatic speech recognition system.

## Compensating for Recognition and Alignment Errors

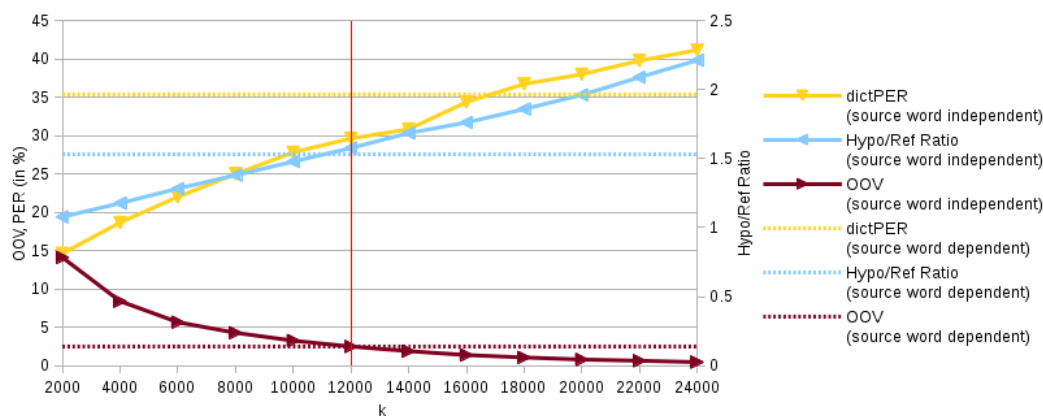
For all source languages and both clustering methods, dictPERs and out-of-vocabulary rates increase for noisy phoneme sequences: The  $R_0$  columns contain lower values than the corresponding  $R_{13.1}$  and  $R_{45.1}$  columns in the Tables 8.3 and 8.4 for both the dictPER and the out-of-vocabulary rate. This performance drop is less severe with source word independent clustering: The dictPER is usually only slightly affected by recognition errors. For example, *es3* achieves a dictPER of 32.55% on phoneme sequences with 45.1% errors ( $R_{45.1}$ ) which is still better than most other source languages on perfect phoneme sequences ( $R_0$ ). Consequently, the influence of the translation is more important than the influence of phoneme recognition errors. This indicates that we are able to compensate for phoneme recognition errors. The increase of the out-of-vocabulary rates with noisier phoneme sequences is usually larger (about factor 2 on average for source word independent clustering) but stays on a surprisingly low level given that nearly every second phoneme in the input sequences is wrong ( $R_{45.1}$ ). This again supports the claim that both pronunciation extraction methods and the alignment model *Model 3P* are robust against recognition errors. Table 8.5 illustrates the effectiveness of both source word dependent and independent clustering to compensate for alignment and phoneme recognition errors and reduce the noise in the dictionaries. When no clustering is applied, the phoneme sequence segments induced by the word-to-phoneme alignments are directly used as pronunciations. In this case, the dictionaries are huge and thus have a low out-of-vocabulary rate – it is a matter of chance that the mapping  $m$  hits a reference entry. Consequently, the out-of-vocabulary rate is only comparable when the dictPER is at the same level. The Hypo/Ref ratio gives a better picture of the improvements due to clustering. Without clustering, 5 of 6 pronunciations are extracted unnecessarily (Hypo/Ref ratio  $\geq 6$ ) on noisy phoneme sequences ( $R_{13.1}$  and  $R_{45.1}$ ). With clustering, the Hypo/Ref ratio is 3.6 to 5.3 times lower. Furthermore, the dictPERs with both clustering methods are significantly lower than without clustering. This indicates that we are able to compensate for errors even at the phoneme level.

## Number Of Clusters In Source Word Independent Clustering

Our source word independent clustering algorithm is based on the  $k$ -means algorithm. The parameter  $k$  (number of desired clusters) should be initial-

Clustering Method	dictPER (in %)			Hypo/Ref Ratio			OOV (in %)		
	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$
No Clustering	48.61	50.44	54.03	4.74	6.43	6.87	0.04	0.04	0.04
Source Word Dependent	35.36	40.81	46.26	1.53	1.45	1.29	2.57	3.78	11.37
Source Word Independent	29.63	30.57	32.55	1.58	1.77	1.81	2.49	4.04	6.94

**Table 8.5** – Error recovery through source word dependent and independent clustering (source language: Spanish (*es3*)).



**Figure 8.7** – Evaluation metrics over  $k$  for source word independent clustering (source language: Spanish (*es3*)).

ized with a value as close to the target language vocabulary size as possible. However, the vocabulary size of the target language is unknown in our scenario. Therefore, we propose to derive it from the vocabulary size of a very similar source language, as we did in [SSVS14a]. Figure 8.7 shows the impact to our evaluation metrics when  $k$  is set too low or too high for the target language English on error-free phoneme sequences. The out-of-vocabulary rate decreases with  $k$  since the larger the extracted dictionary the more reference entries can be mapped by  $m$ . This drop of the out-of-vocabulary rate comes at the expense of the dictPER and the Hypo/Ref ratio – the dictionary becomes noisier. If  $k$  is initialized with the correct value (12k for our subcorpus of the ESV Bible), the out-of-vocabulary rate and Hypo/Ref Ratio is at the same level as source word dependent clustering, but the dictPER is lower. Small variations of  $k$  around 12k cause only minor changes of the error rates indicating that the estimated value for the target language vocabulary size

	Target Language English (ESV)	Target Language Spanish (LBLA)	Target Language Portuguese (AA)	
$ r $				Sent. Accuracy
Vocab. size	.46	.36	.32	.6
				.38
$\Delta$ Vocab. size	.46	.37	.45	.6

Figure 8.8 – Layout for Table 8.6 and 8.7.

SOURCE WORD DEPENDENT CLUSTERING									
$ r $	dictPER			Hypo/Ref Ratio			OOV		
	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$
Vocabulary size	.48	.75	.75	.88	.87	.94	.50	.57	.77
	.66			.90			.61		
Avg. number of words per verse	.52	.69	.80	.63	.64	.67	.24	.28	.46
	.67			.65			.33		
Avg. word frequency	.59	.81	.77	.82	.80	.88	.39	.44	.67
	.72			.83			.50		
IBM-4 perplexity	.76	.51	.21	.50	.54	.39	.91	.87	.66
	.49			.48			.81		

Table 8.6 – Absolute correlation coefficients between our evaluation metrics and different influencing factors for source word dependent clustering.

does not have to be very accurate. In our other experiments we always set  $k$  to the correct value.

## 8.4 Which Source Language Is Favourable?

We investigate the impact of four factors to our evaluation metrics:

- **Vocabulary size.** The vocabulary size of the source language.
- **Average number of words per verse.** The average verse length in the source language.
- **Average word frequency.** The average number of word repetitions in the source language (inverse of the lexical density [Ure71]).
- **IBM-4 PPL.** To measure the general correspondence of the translation to IBM Model based alignment models, we train IBM Model 4 [BPPM93] on the source and target language transcriptions (written word level) using GIZA++ [ON03] with default configuration and measure its perplexity.

SOURCE WORD INDEPENDENT CLUSTERING									
$ r $	dictPER			Hypo/Ref Ratio			OOV		
	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$	$R_0$	$R_{13.1}$	$R_{45.1}$
Vocabulary size	.42	.25	.07	.60	.67	.37	.45	.64	.67
	<b>.25</b>			<b>.55</b>			<b>.59</b>		
Avg. number of words per verse	.56	.42	.07	.73	.57	.23	.56	.69	.58
	<b>.35</b>			<b>.51</b>			<b>.61</b>		
Avg. word frequency	.52	.38	.05	.65	.56	.27	.51	.66	.58
	<b>.32</b>			<b>.49</b>			<b>.58</b>		
IBM-4 perplexity	.79	.89	.96	.56	.68	.87	.73	.52	.61
	<b>.88</b>			<b>.70</b>			<b>.62</b>		

**Table 8.7** – Absolute correlation coefficients between our evaluation metrics and different influencing factors for source word independent clustering.

Tables 8.6 and 8.7 show Pearson’s correlation coefficients  $|r|$  [RN88] between those four factors and our evaluation metrics from Section 8.2. Figure 8.8 describes the table layout. High correlations are highlighted.

As shown in Figure 8.8, each cell is subdivided into four cells. Three of these cells (upper row) contain the correlation coefficients for error-free phoneme sequences ( $R_0$ ) and noisy phoneme sequences ( $R_{13.1}$  and  $R_{45.1}$ ). The lower row (bold font) contains the mean of these specific coefficients.

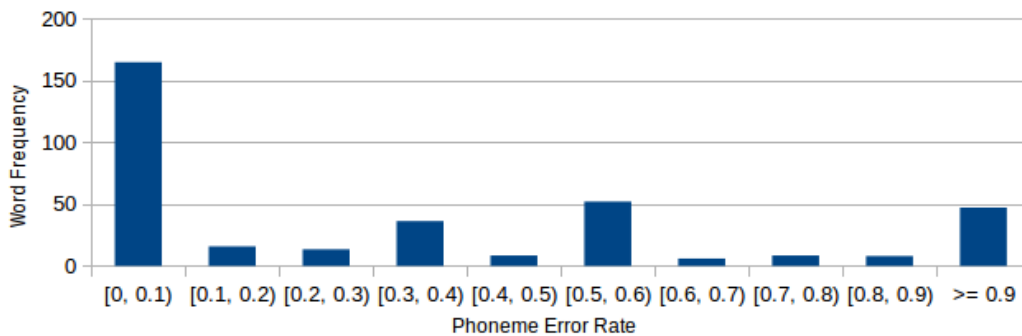
Regardless of how noisy the phoneme sequences are, the vocabulary size and the average word frequency are good indicators to predict the Hypo/Ref Ratio for source word dependent clustering ( $|r| \geq .80$  in Table 8.6). The IBM-4 PPL has a high linear correlation with the dictPER on error-free phoneme sequences ( $|r| = .76$ ), but is less correlated when the phoneme sequences are noisy. For noisy phoneme sequences, the number of word repetitions in the source language becomes more important for the dictPER ( $|r| \geq .77$ ): Our error recovery methods work better when many realizations of the target language word can be merged. In source word dependent clustering, the number of occurrences of a source word is a rough upper bound for the number of elements in its clusters. The verse length also becomes more important with noisier phoneme sequences: With error-free phoneme sequences the alignment model is able to align even long verses. However, only short verses can be aligned reliably when the phoneme sequence is noisy. The dominant factor for the out-of-vocabulary rate is the IBM-4 PPL on error-free phoneme sequences, and the vocabulary size on noisy phoneme sequences. Generally, the vocabulary size of the source language is highly correlated with all evaluation metrics due to the dependency on the source word with this clustering method.

In contrast, the linear correlations of the vocabulary size are much smaller for source word independent clustering (Table 8.7). For this clustering method, the IBM-4 PPL is an important factor for the extraction quality. This suggests that selecting a translation which is as literal as possible (e.g. following the formal equivalence translation method) is crucial.

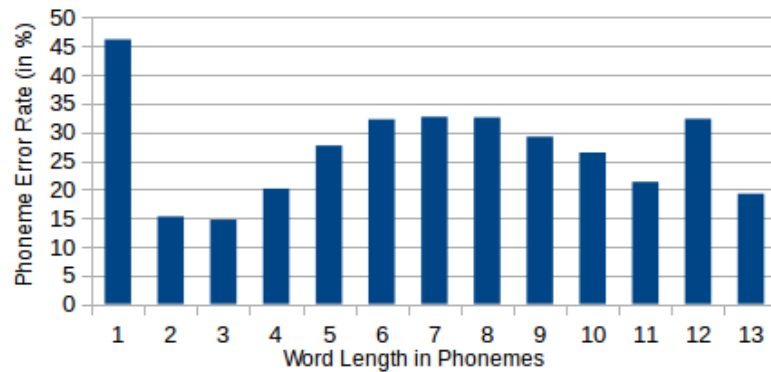
## 8.5 Which Words Are Extracted Correctly?

This section describes the characteristics of words which are likely to be extracted correctly when the source language *es3* and the target language English is used and pronunciations are extracted with source word dependent clustering. Experiments with other source languages and source word independent clustering show similar results. Figure 8.9 indicates that frequently occurring words tend to contain no or only minor errors at the phoneme level. The relation between the word length in terms of phonemes and the phoneme level pronunciation quality is illustrated in Figure 8.10. One phoneme words are often extracted incorrectly. This is due to *Model 3P*'s problems with aligning one phoneme words: A single phoneme usually occurs at different positions in the phoneme sequence, and it has to be inferred by the rather weak distortion model and the neighboring alignments which position is the correct one. For longer words, the learned translation probabilities give stronger hints of where the target word is likely to start in the phoneme sequence. Apart from that, very short and very long words are generally extracted more accurately than words with average length.

A look at some extracted pronunciations reveals two major sources of errors for words with only 1-2 phoneme errors:



**Figure 8.9** – Mean word frequency over phoneme error rate (Spanish-English).



**Figure 8.10** – Phoneme error rate over word length in phonemes (Spanish-English).

1. Single phonemes are added or dropped in the beginning or end of a word:

- z f i h s t s instead of f i h s t s (fists)
- i h k s t instead of f i h k s t (fixed)
- i h z r e y l a h instead of i h z r e y l (israel)

2. Different words with the same stem are merged together:

- s i h d u w s i h t instead of s i h d u w s t (seduced) or s i h d u w s i h n g (seducing)
- i h k n a a l i h j h m instead of i h k n a a l i h j h (acknowledge) or i h k n a a l i h j h m a h n t (acknowledgement)

Entries with two phoneme errors or more, often contain two consecutive words due to missing word boundaries between words often occurring in the same context:

- w e r i h n d i h g n a h n t (were indignant)
- f i h n i h s h t i h t (finished it)

We assume that this kind of errors would not be very critical when using the dictionary in a speech-to-speech translation system since those words are likely to be stuck together as a *phrase* later in the training process of the translation model anyway. Since this work is only concerned with the automatic speech recognition part, proving this claim by training a full speech-to-speech translation system remains future work.



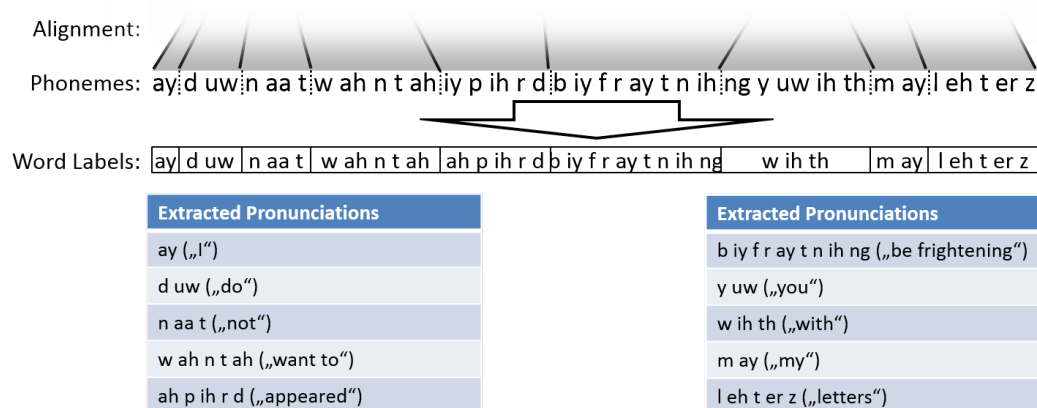


Figure 8.11 – Training set generation for language modeling.

## 8.6 Automatic Speech Recognition in the Target Language

Our final goal is to integrate the extracted pronunciations in a word recognizer for the target language. The three basic components of our automatic speech recognition system are acoustic model, pronunciation dictionary, and language model. We use the acoustic models  $AM_{13.1}$  and  $AM_{45.1}$  presented in Section 8.3 in our experiments. The pronunciation dictionaries are extracted as described in the previous Section 8.1. To train a language model, we replace the segments in the segmented phoneme sequences (Section 3.3.3) with the closest word labels in the extracted dictionary. Thereby we obtain sequences of word labels which serve as training data for trigram language models. The process is illustrated in Figure 8.11<sup>5</sup>. To reduce complexity, we restrict our experimental settings to the best, middle, and worst source language (*es3*, *de2*, and *se*).

<sup>5</sup>Example taken from 2 Corinthians 10:9 “I do not want to appear to be frightening you with my letters.” with  $R_{13.1}$  and *es3*.

Reference Pronun.	We w ih	are er	not n aa t	able ey b ah l	to t ah	go g ow	up ah p	against ah g ey n s t	the dh ih	people, p ih p ah l
WER (33.3%)	w ih er Sub	Del	n aa t	ey b ah l	t ah	g ow	ah p	ah g ih n s t Sub	dh ih	p ih p ah l
multiWER (11.1%)	w ih er		n aa t	ey b ah l	t ah	g ow	ah p	ah g ih n s t Sub	dh ih	p ih p ah l

Table 8.8 – Word error rate and multiWER.

The evaluation of the resulting automatic speech recognition systems is difficult as they output sequences of word labels that are to be compared with the written reference sentences. One conservative way is to modify the standard **word error rate (WER)**: We allow a word label to match with a written word in the reference if its pronunciation is equal to one possible pronunciation variant of the written reference word. However, this measure is highly sensitive to segmentation errors (inserted or missing word boundaries). Which words are written together and which separately is often due to conventions which are impossible to detect for our automatic algorithms. For example, our algorithms often extract the pronunciation **n ow h w ah n** (“no one”) in our experiments. The phrase “no one” is written separately, but “nobody” is written together. As both are grammatically and semantically interchangeable in English, our automatic methods are not able to learn the difference – it would be just as reasonable to treat “no one” as single pronoun and “no-body” as two separate words. To reduce the impact of these artefacts we propose the **multi-word error rate (multiWER)**. The multiWER allows 1:n mappings from words in the reference to a sequence of word labels and vice versa before calculating the word error rate. Mappings must be applied to all sentences consistently (not only to one sentence) and the concatenated pronunciations on both sides must be equal. Table 8.8 compares both measures with the help of an example. The example is taken from our experiments with *es3* as source language using the acoustic model  $AM_{13.1}$  (excerpt from Numbers 13:31). We feel that even the multiWER underestimates the usefulness in a speech-to-speech translation scenario: Minor spelling errors are not important as long as the word is always substituted with the same misspelled word label. For example, “against” is misspelled in Table 8.8, but since there are no similar entries in the dictionary, “against” is always misspelled the same way. Therefore, it would not affect speech-to-speech translation performance.

SPEAKER DEPENDENT ACOUSTIC MODEL ( $R_{13.1}, AM_{13.1}$ )						
Extraction Method	WER in %			multiWER in %		
	es3	de2	se	es3	de2	se
Source Word Dependent Clustering	<b>34.5</b>	37.9	54.5	<b>26.6</b>	29.8	40.7
Source Word Independent Clustering	<b>33.4</b>	36.3	46.3	<b>25.3</b>	27.8	33.8

**Table 8.9** – Automatic speech recognition performance with a speaker dependent acoustic model ( $R_{13.1}, AM_{13.1}$ ).

Table 8.9 summarizes the results with the acoustic model  $AM_{13.1}$ . In this setting, the pronunciations are extracted from phoneme sequences with 13.1% phoneme error rate ( $R_{13.1}$ ) and the automatic speech recognition system is

based on a speaker-dependent acoustic model trained on  $EN_{all} \setminus EN_{filt}$ . The gold standard (correct pronunciation dictionary and a language model trained on the  $AM_{13.1}$  training transcriptions  $EN_{all} \setminus EN_{filt}$ ) has a word error rate of 3.6% and a multiWER of 3.6%: There are no segmentation errors when the correct pronunciation dictionary is applied. In our experiments, the multiWER is usually significantly lower than the word error rate. This illustrates the effectiveness of the multiWER evaluation metrics to reduce artefacts due to segmentation errors. Source word independent clustering consistently outperforms source word dependent clustering for all three source languages. We report a multiWER of 25.3% using the best source language *es3*.

SPEAKER INDEPENDENT AND CORPUS-MISMATCHED ACOUSTIC MODEL ( $R_{45.1}, AM_{45.1}$ )						
Extraction Method	WER in %			multiWER in %		
	es3	de2	se	es3	de2	se
Source Word Dependent Clustering	<b>87.7</b>	88.6	95.4	<b>85.7</b>	86.5	92.5
Source Word Independent Clustering	<b>86.1</b>	86.6	90.7	<b>83.1</b>	84.0	88.3

**Table 8.10** – Automatic speech recognition performance with a speaker independent, corpus-mismatched acoustic model ( $R_{45.1}, AM_{45.1}$ ).

Table 8.10 shows the results when a worse acoustic model ( $AM_{45.1}$ ) is used in the automatic speech recognition system and the phoneme sequences for the pronunciation extraction contain 45.1% errors ( $R_{45.1}$ ). The gold standard based on  $AM_{45.1}$  has a word error rate and multiWER of 28.7% and thus is significantly worse than with the  $AM_{13.1}$  acoustic model. This is reflected by the general performance drop compared to Table 8.9. The differences between the word error rate and multiWER is smaller than with  $AM_{13.1}$  because spelling errors are more common. The best system (source word independent clustering, source language *es3*) achieves a multiWER of 83.1%. This indicates that the effectiveness of our methods strongly depends on the underlying acoustics.



## Conclusions and Future Directions

---

Over the last years, speech and language processing technology has become more and more present in daily life. Speech technology is expected to work for multiple languages. However, a challenge today is still to rapidly establish speech processing systems for new domains and languages with low human effort and at reasonable cost. Pronunciation dictionaries are a central component for both, automatic speech recognition and speech synthesis. This dissertation presents a wide range of research for the rapid generation of pronunciation dictionaries for new application domains and languages. The developed methods in this thesis are evaluated on many automatic speech recognition results. In this chapter, we review the mayor contributions of this thesis and suggest potential future research directions.

### 9.1 Contributions

The most important achievements of this thesis are structured in the following categories corresponding to the three challenges of the generation of pronunciation dictionaries for new domains and languages:

- Rapid vocabulary selection.
- Rapid generation of pronunciations.
- Dictionary generation for non-written languages.

Our contributions are published in 25 publications in two journals and on international conferences. The citations demonstrate that other researchers have already followed our ideas or algorithms. The tools, which we imbedded in our Rapid Language Adaptation Toolkit within this work, are publicly accessible and already used by another research group [MBDW13, MBdW14].

We believe that our research ideas presented in this thesis have the potential to influence both, practical applications and future research. The following sections summarize the most important results and show the importance of the thesis in the context of the generation of pronunciation dictionaries.

### 9.1.1 Strategies For Rapid Vocabulary Selection and Text Normalization

For written languages, our mayor contributions for vocabulary selection are:

**Rapid Vocabulary Selection:** We developed strategies for the collection of words from Web texts which are time- and topic-relevant for the application domain of the automatic speech recognition system. This includes the use of information from RSS Feeds and Twitter which is particularly helpful for the automatic transcription of broadcast news. Our methods, which are published in [VSKS10] and [SGVS13], can be easily adapted to new domains and languages existing on the World Wide Web.

**Text Normalization through Crowdsourcing:** For the rapid development of text normalization systems at low cost, we developed methods where Internet users generate training data for such systems by simple text editing. Our methods and analyses are published in [SZGS10] and [SZLS13]. The annotation process for English training data was realized fast and at low cost with the crowdsourcing platform Amazon Mechanical Turk. Due to the high ethnic diversity in the U.S. where most Turkers come from and Turkers from other countries [RIS<sup>+</sup>10], we believe that a collection of training data for other languages is also possible.

As no textual representation usually exists for non-written languages, we developed methods to extract word-like units from the phoneme sequence derived from audio data, which enables an automatic “invention” of a writing system for the target language.

Since exploiting the written translation of the spoken phrases has proven to outperform monolingual approaches, we use it to segment the phoneme

sequence into word units. Thus, we tackled the problem of “human translations guided language discovery”. The assumption is that a human translator produces utterances in the non-written target language from prompts in a resource-rich source language. We align source language words to target language phoneme sequences across languages, i.e. cross-lingually. Based on this alignment, we induce phoneme sequences forming words of the target language.

Our mayor contribution for the vocabulary selection given this scenario is:

**Model 3P:** We developed a new alignment model *Model 3P* especially for cross-lingual word-to-phoneme alignments, which extends the generative process of IBM Model 3 by a word length step and additional dependencies for the lexical translation probabilities. *Model 3P* is more robust against phoneme errors and achieves significantly better results than traditional statistical alignment models for word-to-word alignment, in particular with respect to word segmentation quality. A multi-threaded implementation of *Model 3P* is available for download at <http://pisa.googlecode.com/> and corresponding analyses are published in [SSVS12], [SSVS13], [SSVS14a], and [SSVS14b].

We have shown that monolingual segmentation can also help to retrieve word units from the phoneme sequence if no translation is available. However, the performance is worse than with the information of the translation.

### 9.1.2 Techniques For Rapid Generation Of Pronunciations

To decide which strategy for the pronunciation generation is optimal for a given scenario, one can traverse the binary tree illustrated in Figure 9.1.

For written languages, our mayor contributions for the rapid and low-cost generation of pronunciations are:

**Web-based Tools For The Rapid Pronunciation Dictionary Creation:** With our Automatic Dictionary Extraction Tool, part of the Rapid Language Adaptation Toolkit, we developed a system for automatically extracting phonetic notations in IPA from any Web source that has separate pages for individual words. We analyzed the quantity and quality of pronunciations in *Wiktionary* as it is available in many languages. Our analyzes and results are published in [SOS10], [SOS12a], [SOS12b], and [SOS14]. Due to the fast growth in language presence on *Wiktionary*, there is a future potential of harvesting pronunciations for under-resourced languages from this source.

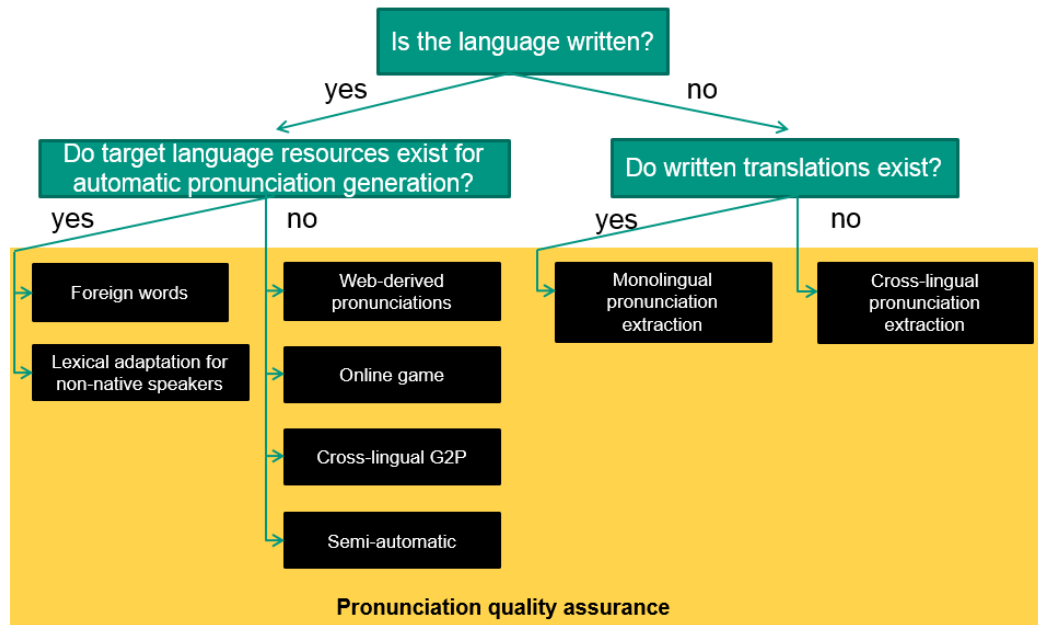


Figure 9.1 – Pronunciation Generation.

Furthermore, we investigated evaluation criteria for the quality of the developed dictionaries and presented solutions to raise the quality. [KSN<sup>+</sup>13] refer to our work employing pronunciations from the World Wide Web as training data for grapheme-to-phoneme models without a cross-check of language experts as a motivation for their robust grapheme-to-phoneme conversion with discriminative methods. Moreover, [Big13] reference our work to quickly and economically create dictionaries for new languages and domains.

Additionally, we are the first who implemented with *Keynounce* an online game for the pronunciation generation and analyzed an inexpensive collection with the crowdsourcing platform Amazon Mechanical Turk. While the Amazon Mechanical Turk experiment showed that micropayment workers in general are not overly fond of investing a lot of time in fine-tuning answers, we demonstrate that given the right kind of incentive in providing a gamelike experience, anonymous users will help for free. The interface is intuitive and simple so that there is no work involved in training new users and we observe a learning effect. *Keynounce* is publicly accessible online at <http://i19pc7.ira.uka.de/keynounce>. The quality of the results is high enough to calculate a good quality of pronunciations with our filtering methods.

**Cross-lingual Grapheme-to-Phoneme Model based Pronunciation Generation:** We developed a strategy to use grapheme-to-phoneme mod-



els derived from existing dictionaries of other languages for the production of pronunciations. This strategy can contribute to a rapid and economic semi-automatic pronunciation dictionary development, thereby reducing the necessary manual effort. Our strategy is published in [SVYS13].

**Semi-Automatic Pronunciation Generation:** We have proposed efficient methods for rapid and economic semi-automatic dictionary development. In addition to the traditional concatenation of single phonemes most commonly associated with each grapheme, we show that Web-derived pronunciations and cross-lingual grapheme-to-phoneme models can help to reduce the initial editing effort. Furthermore, we show that our phoneme-level combination of the output of multiple grapheme-to-phoneme converters reduces the editing effort more than the best single converters. Our new Rapid Language Adaptation Toolkit function which is publicly available allows to bootstrap a dictionary with the proposed methods supported with the possibility to listen to a synthesized wavefile of the pronunciation. Our methods are published in [SMS14].

Furthermore, we approached the two following problems for the pronunciation generation for new domains:

**Pronunciation Generation for Foreign Words:** We developed new methods to automatically detect foreign words from a word list of a matrix languages and advanced existing approaches to economically build up lexical resources for domains with foreign words. Our methods for the detection of Anglicisms, which can be adapted to new languages, are published in [LSS14].

**Lexical Adaptation for Non-Native Speakers:** Accented speech occurs amplified where non-native speakers operate automatic speech recognition systems. We analyze a parallel corpus of phoneme sequences from phonetic transcriptions of native US English and accented English in the speech accent archive of the George Mason University to rapidly and economically adapt the pronunciation dictionary and improve the automatic speech recognition performance of accented English.

For non-written languages, our mayor contributions for the rapid and economic generation of pronunciations are:

**Dictionary Generation for Non-Written Languages:** We presented two algorithms to deduce phonetic transcriptions of target language words from *Model 3P* alignments (source word dependent [SSVS13] and independent clustering [SSVS14a]). To the best of our knowledge, we are the first who used the extracted pronunciations for producing a pronunciation dic-

tionary and in an automatic speech recognition system. [SPC<sup>+</sup>13] report our phonetic-like representation of the target speech and the changes to statistical machine translation modeling methods which we have proposed to specifically deal with phoneme strings in the target language. Our methods are highly relevant to bootstrap dictionaries from audio data for automatic speech recognition and bypass the written form in speech-to-speech translation, particularly in the context of under-resourced languages, and those which are not written at all.

**Pronunciation Quality Assurance:** Since our approaches for the pronunciation generation have to deal with limited, inconsistent or erroneous resources, we developed and applied several methods for the pronunciation quality assurance. For written languages, we analyzed languages with differing grade in grapheme-to-phoneme relationship. We demonstrated that despite varying grapheme-to-phoneme correspondences 15k phonemes with corresponding graphemes are sufficient to obtain a stable consistency in the resulting pronunciations for most languages. Our analyses are published in [SOS12b], [SOS14], and [SS14]. Several researchers already use our lessons learned for their work [IBC<sup>+</sup>12, RMD13, Hof14, KSN<sup>+</sup>14]. Moreover, we have presented methods to recover from errors and inconsistencies automatically, which we published in [SOS12a] and [SDV<sup>+</sup>12]. The methods are based on the means and deviations of certain characteristics computed on the word-pronunciation pairs of the dictionaries. As our phoneme-level combination has potential to generate a better pronunciation out of different pronunciations of the same word, it improves crowdsourcing-based approaches where multiple annotators create pronunciations for the same word and is helpful in the grapheme-to-phoneme conversion with very limited training data as well as in our methods for the dictionary generation for non-written languages. Our analyses are published in [SQS14], [SMS14] and [SSVS14b].

## 9.2 Potential Future Research Directions

This section suggests two research directions which we account as most important. They are related to the grapheme-to-phoneme converter performance for written languages and the development of speech processing systems for non-written languages.

### 9.2.1 More Robust Grapheme-to-Phoneme Conversion

As we have shown in several experiments, grapheme-to-phoneme conversion is essential to support the rapid generation of dictionaries for written languages. We have proposed several methods to improve the quality of grapheme-to-phoneme converters and their training data which reduces the manual effort. However, a direct use of the retrieved pronunciations degrades the performance of the automatic speech recognition system in exchange for improvements of cost and time for dictionary construction.

[KSN<sup>+</sup>13] have already been motivated by our work with pronunciations from the World Wide Web to investigate robust discriminative methods for the grapheme-to-phoneme conversion. Even though successful for the signal processing, acoustic and language modeling, techniques with neural networks have not yet brought much improvement over grapheme-based methods [Nov11, NMH12]. Further investigations of neural network-based approaches for the grapheme-to-phoneme conversion may change this.

### 9.2.2 Language Technology for Non-Written Languages

According to [LSF14], there are currently listed 7,105 living languages. 3,570 have a developed writing system. Data indicates that 696 languages are unwritten. However, for the remaining 2,839 languages no information exists. Despite such a high number of non-written languages in the world, methods to produce speech processing systems for them have not yet received much attention. By simplifying the task, we provided first ideas and approaches, which serve as a good starting point for further research.

Future directions may be to enhance the pronunciation extraction from phoneme sequences. Enforcing a Zipfian cluster size distribution on source word independent clustering may improve clustering accuracy. Errors due to single phonemes dropped or added at the begin or end of a pronunciation may be reduced by reinforcing the alignments with the extracted pronunciations after each iteration of our algorithm. Monolingual word segmentation methods as in [Joh08, Gol10] may give additional hints. Additionally, promising acoustic modeling and phonetic discovery methods as in [LG12, VKD08, CHR11] should be investigated on the target language speech. The acoustic models could be further improved by iteratively recognizing the speech to provide

target language transcriptions, and then using the transcriptions to adapt the models. We simulated non-written languages in our experiments due to the limited financial resources. Interesting is to use the extracted dictionaries in an automatic speech recognition system for truly under-resourced and non-written languages. Moreover, it is interesting to build a speech-to-speech translation system or a dialog system without any linguistic knowledge or writing system of the target language.

Preserving non-written and under-resourced languages with speech technology would suspend the rapid endangerment and death of many under-resourced languages across the world described by David Crystal in [Cry02]. Then, cultural identity can also be protected.

## Bibliography

- [AADGL97] G. Adda, M. Adda-Decker, J.-L. Gauvain, and L. Lamel. Text Normalization And Speech Recognition In French. In *Proc. of Eurospeech*, 1997.
- [ABP09] G. Adam, C. Bouras, and V. Pouloupoulos. Utilizing RSS Feeds for Crawling the Web. In *Proc. ICIW*, 2009.
- [ACT05] B. Ahmed, S.-H. Cha, and C. Tappert. Detection of Foreign Entities in Native Text Using N-gram Based Cumulative Frequency Addition. In *Proc. of Student/Faculty Research Day, CSIS, Pace University*, 2005.
- [Ade09] T. Adegbola. Building Capacities in Human Language Technology for African Languages. In *Proc. of AfLaT*, 2009.
- [AGFF00] C. Auzanne, J. S. Garofolo, J. G. Fiscus, and W. M. Fisher. Automatic Language Model Adaptation for Spoken Document Retrieval. In *Proc. of RIAO 2000 Conference on Content-Based Multimedia Information Access*, 2000.
- [AGK<sup>+</sup>12] E. Achtert, S. Goldhofer, H.-P. Kriegel, E. Schubert, and A. Zimek. Evaluation of Clusterings – Metrics and Visual Support. In *Proc. of ICDE*, 2012.
- [AHHL<sup>+</sup>09] H. Al-Haj, R. Hsiao, I. R. Lane, A. W. Black, and A. Waibel. In *Proc. of ASRU*, 2009.
- [Ahm05] B. Ahmed. *Detection of Foreign Words and Names in Written Text*. PhD thesis, Pace University, 2005.
- [AJSS11] N. Alewine, E. Janke, R. Sicconi, and P. Sharp. Systems and Methods for Building a Native Language Phoneme Lexicon Having Native Pronunciations of the Non-Native Words Derived from Non-Native Pronunciations, 2011. US 7472061 B1.
- [AKS00] I. Amdal, F. Korkmazskiy, and A. C. Surendran. Data-Driven Pronunciation Modelling for Non-Native Speakers using Association Strength between Phones. In *Proc. of ASRU*, 2000.
- [AKT<sup>+</sup>14] H. Adel, K. Kirchhoff, D. Telaar, N. T. Vu, T. Schlippe, and T. Schultz. Features for Factored Language Models for Code-Switching Speech. In *Proc. of SLTU 2014*, 2014.

- [Ale05] B. Alex. An Unsupervised System for Identifying English Inclusions in German Text. In *Proc. of ACL Student Research Workshop*, 2005.
- [Ale08a] B. Alex. *Automatic Detection of English Inclusions in Mixed-lingual Data with an Application to Parsing*. PhD thesis, University of Edinburgh, 2008.
- [Ale08b] B. Alex. Comparing Corpus-based to Web-based Lookup Techniques for Automatic English Inclusion Detection. In *Proc. of LREC*, 2008.
- [And05] G. Andersen. Assessing Algorithms for Automatic Extraction of Anglicisms in Norwegian Texts. *Corpus Linguistics*, 2005.
- [ANX<sup>+</sup>05] M. Afify, L. Nguyen, B. Xiang, S. Abdou, and J. Makhoul. Recent Progress in Arabic Broadcast News Transcription at BBN. In *Proc. of Interspeech*, 2005.
- [arp] Arpabet: <http://en.wikipedia.org/wiki/Arpabet>.
- [ARS<sup>+</sup>07] C. Allauzen, M. Riley, J. Schalkwyk, W. Skut, and M. Mohri. OpenFst: A General and Efficient Weighted Finite-State Transducer Library. In *Proc. of CIAA*, 2007.
- [AV10] V. Ambati and S. Vogel. Can Crowds Build parallel corpora for Machine Translation Systems? In *Proc. of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010.
- [AYS04] Siu-Kei Au Yeung and Man-Hung Siu. Improved Performance of Aurora 4 Using HTK and Unsupervised MLLR Adaptation. In *Proc. of Interspeech*, 2004.
- [AZXS06] A. Aw, M. Zhang, J. Xiao, and J. Su. A Phrase-based Statistical Model for SMS Text Normalization. In *Proc. of COLING*, 2006.
- [BBKS14] L. Besacier, E. Barnard, A. Karpov, and T. Schultz. Automatic Speech Recognition for Under-Resourced Languages: A Survey. *Speech Communication*, 56:85–100, 2014.
- [BD12] W. D. Basson and M. Davel. Comparing Grapheme-based and Phoneme-based Speech Recognition for Afrikaans. In *Proc. of PRASA*, 2012.

- [BD13] W. Basson and M. Davel. Category-Based Phoneme-To-Grapheme Transliteration. In *Proc. of Interspeech*, 2013.
- [Bes94] S. Besling. Heuristical and Statistical Methods for Grapheme-to-Phoneme Conversion. In *Proc. of Konvens*, 1994.
- [BFIH06] G. Bouselmi, D. Fohr, I. Illina, and J. P. Haton. Fully Automated Non-Native Speech Recognition Using Confusion-Based Acoustic Model Integration and Graphemic Constraints. In *Proc. of ICASSP*, 2006.
- [BFOS84] L. Breiman, J. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA., 1984.
- [Big13] B. Bigi. A Phonetization Approach For The Forced-Alignment Task. In *Proc. of LTC*, 2013.
- [BKKB98] W. Byrne, E. Knodt, S. Khudanpur, and J. Bernstein. Is Automatic Speech Recognition Ready for Non-Native Speech? A Data Collection Effort and Initial Experiments in Modelling Conversational Hispanic English. In *Proc. of ESCA Conference on Speech Technology in Language Learning*, 1998.
- [BL00] A. W. Black and K. Lenzo. Building Voices in the Festival Speech Synthesis System. *Festvox*. Retrieved December 30 2013, from <http://festvox.org/bsv/>, 2000.
- [BLP98] A. W. Black, K. Lenzo, and V. Pagel. Issues in Building General Letter to Sound Rules. In *Proc. of ESCA Workshop on Speech Synthesis*, 1998.
- [BM00] E. Brill and R. C. Moore. An Improved Error Model for Noisy Channel Spelling Correction. In *Proc. of ACL*, 2000.
- [BMR08] S. N. Buk, J. Macutec, and A. A. Rovenchak. Some Properties of the Ukrainian Writing System. *CoRR*, 2008.
- [BMSW97] D. M. Bikel, S. Miller, R. Schwartz, and R. Weischedel. Nymble: a High-Performance Learning Name-finder. In *Proc. of Conference on Applied Natural Language Processing*, 1997.
- [BN08] M. Bisani and H. Ney. Joint-Sequence Models for Grapheme-to-Phoneme Conversion. *Speech Communication*, 2008.
- [Bor03] J. A. Borland. The English Standard Version – A Review Article. *Faculty Publications and Presentations*, page 162, 2003.

- [BOS03] I. Bulyko, M. Ostendorf, and A. Stolcke. Getting More Mileage from Web Text Sources for Conversational Speech Language Modeling using Class-Dependent Mixtures. In *Proc. of HLT-NAACL*, 2003.
- [BPPM93] P. F. Brown, V. J. Della Pietra, S. A. Della Pietra, and R. L. Mercer. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311, 1993.
- [Bre14] W. Breiter. Rapid Bootstrapping of Haitian Creole Large Vocabulary Continuous Speech Recognition. Bachelor’s thesis (studienarbeit), KIT, CSL, Germany, 2014.
- [BZG06] L. Besacier, B. Zhou, and Y. Gao. Towards Speech Translation of Non Written Languages. In *Proc. of SLT*, 2006.
- [CBD10] C. Callison-Burch and M. Dredze. Creating Speech and Language Data With Amazon’s Mechanical Turk. In *Proc. of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon’s Mechanical Turk*, 2010.
- [CCG+09] D. Can, E. Cooper, A. Ghoshal, M. Jansche, S. Khudanpur, B. Ramabhadran, M. Riley, M. Saraclar, A. Sethy, M. Ulinski, and C. White. Web Derived Pronunciations for Spoken Term Detection. In *Proc. of SIGIR*, 2009.
- [CG91] K. W. Church and W. A. Gale. Probability Scoring for Spelling Correction. *Statistics and Computing*, 1(2):93–103, 1991.
- [CG98] S. F. Chen and J. Goodman. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proc. of ACL*, 1998.
- [Che03] S. F. Chen. Conditional and Joint Models for Grapheme-to-Phoneme Conversion. In *Proc. of Eurospeech*, 2003.
- [CHR11] S. Chaudhuri, M. Harvilla, and B. Raj. Unsupervised Learning of Acoustic Unit Descriptors for Audio Content Representation and Classification. In *Proc. of Interspeech*, 2011.
- [CHS06] P. Charoenpornasawat, S. Hewavitharana, and T. Schultz. Thai Grapheme-Based Speech Recognition. In *Proc. of HLT-NAACL*, 2006.



- [CK10] B. T. Conboy and P. K. Kuhl. Impact of Second-Language Experience in Infancy: Brain Measures of First- and Second-Language Speech Perception. In *Developmental Science*, 2010.
- [cmu] cmudict: <http://www.speech.cs.cmu.edu/cgi-bin/cmudict>.
- [Cro01] Crossway. The Holy Bible: English Standard Version, 2001.
- [Cry02] D. Crystal. *Language Death*. Canto Refresh Your Series. Cambridge University Press, 2002.
- [Cry04] D. Crystal. The Past, Present and Future of World English. A. Gardt and B. Hüppauf (eds), *Globalization and the Future of German* (Berlin: Mouton de Gruyter), pages 27–46, 2004.
- [Dav05] M. Davel. The Default & Refine Algorithm, A Rule-based Learning Algorithm. <http://code.google.com/p/defaultrefine/>, 2005.
- [DB04a] M. Davel and E. Barnard. A Default-and-Refinement Approach to Pronunciation Prediction. In *Proc. of PRASA*, 2004.
- [DB04b] M. Davel and E. Barnard. The Efficient Creation of Pronunciation Dictionaries: Machine Learning Factors in Bootstrapping. In *Proc. of ICSLP*, 2004.
- [DB04c] M. Davel and E. Barnard. The Efficient Generation Of Pronunciation Dictionaries: Human Factors During Bootstrapping. In *Proc. of Interspeech*, 2004.
- [DB05] M. Davel and E. Barnard. Bootstrapping Pronunciation Dictionaries: Practical Issues. In *Proc. of Interspeech*, 2005.
- [DB06] M. Davel and E. Barnard. Developing Consistent Pronunciation Models for Phonemic Variants. In *Proc. of Interspeech*, 2006.
- [DB08] M. Davel and E. Barnard. Pronunciation Prediction with Default & Refine. *Computer Speech and Language*, 22(4):374–393, October 2008.
- [DdW10] M. Davel and F. de Wet. Verifying Pronunciation Dictionaries using Conflict Analysis. In *Proc. of Interspeech*, 2010.
- [DFG<sup>+</sup>05] S. L. Davis, S. Fettaers, B. Gustafson, L. Loney, and D. E. Schulz. System And Method For Preparing A Pronunciation

- Dictionary For A Text-to-Speech Voice. Technical Report US Patent 7630898 B1, AT&T, September 2005.
- [Djo11] E. Guevara Komgang Djomgang. Hausa Large Vocabulary Continuous Speech Recognition. Bachelor's thesis (studienarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2011.
- [DL10] M.I. Denkowski and A. Lavie. Exploring Normalization Techniques for Human Judgments of Machine Translation Adequacy Collected Using Amazon Mechanical Turk. In *Proc. of NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, 2010.
- [DLR77] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society, series B*, 39(1):1–38, 1977.
- [DM09] M. Davel and O. Martirosian. Pronunciation Dictionary Development in Resource-Scarce Environments. In *Proc. of Interspeech*, 2009.
- [EGMA09] M. Elmahdy, R. Gruhn, W. Minker, and S. Abdennadher. Effect of Gaussian Densities and Amount of Training Data on Grapheme-Based Acoustic Modeling for Arabic. In *Proc. of NLP-KE*, 2009.
- [EK SX96] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases With Noise. In *Proc. of KDD*, 1996.
- [ES05] H. Engelbrecht and T. Schultz. Rapid Development of an Afrikaans-English Speech-to-Speech Translator. In *Proc. of SLT*, 2005.
- [eSp] eSpeak text to speech synthesizer. <http://espeak.sourceforge.net/>.
- [FGH+97] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal. The Karlsruhe-Verbmobil Speech Recognition Engine. In *Proc. of ICASSP*, 1997.
- [Fis07] J. Fiscus. Speech Recognition Scoring Toolkit ver. 2.3 (sctk), 2007.

- [F199] E. Fosler-lussier. Multi-Level Decision Trees for Static and Dynamic Pronunciation Models. In *Proc. of Eurospeech*, 1999.
- [Fla] Adobe Flash. <http://www.adobe.com/de/products/flash.html>.
- [For73] Jr. Forney, G.D. The Viterbi Algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.
- [FR12] J. Feng and B. Renger. Language Modeling for Voice-Enabled Social TV Using Tweets. In *Proc. of Interspeech*, 2012.
- [GABP11] H. Gelas, S. T. Abate, L. Besacier, and F. Pellegrino. Quality Assessment of Crowdsourcing Transcriptions for African Languages. In *Proc. of Interspeech*, 2011.
- [Gal98] M. J. F. Gales. Maximum likelihood linear transformations for hmm-based speech recognition. In *Computer Speech and Language*, volume 12, 1998.
- [GBP12] H. Gelas, L. Besacier, and F. Pellegrino. Developments of Swahili Resources for an Automatic Speech Recognition System. In *Proc. of SLTU*, 2012.
- [GE03] S. Goronzy and K. Eisele. Automatic Pronunciation Modeling for Multiple Non-Native Accents. In *Proc. of ASRU*, 2003.
- [GF09] M. Gerosa and M. Federico. Coping with Out-of-vocabulary Words: Open versus Huge Vocabulary ASR. In *Proc. of ICASSP*, 2009.
- [GGPP93] J. Garofalo, D. Graff, D. Paul, and D. Pallett. Continuous Speech Recognition (CSR-I) Wall Street Journal (WSJ0) News, Complete. Technical report, Linguistic Data Consortium, Philadelphia, 1993.
- [GJK<sup>+</sup>09] A. Ghoshal, M. Jansche, S. Khudanpur, M. Riley, and M. Ulin-ski. Web-derived Pronunciations. In *Proc. of ICASSP*, 2009.
- [GJWW06] F. Gralinski, K. Jassem, A. Wagner, and M. Wypych. Text Normalization as a Special Case of Machine Translation. In *Proc. of IMCSIT*, 2006.
- [GK99] S. Goronzy and R. Kompe. A Combined MAP + MLLR Approach for Speaker Adaptation. In *In Proc. of the Sony Research Forum*, 1999.

- [GKRR14] M. J. F. Gales, K. M. Knill, A. Ragni, and S. P. Rath. Speech Recognition and Keyword Spotting for Low Resource Languages: Babel Project Research at Cued. In *Proc. of SLTU*, 2014.
- [GL92] J.-L. Gauvain and C.-H. Lee. MAP Estimation of Continuous Density HMM: Theory and Applications. In *Proc. of DARPA Speech and Natural Language Workshop*, 1992.
- [GN00] G. Grefenstette and J. Nioche. Estimation of English and non-English Language Use on the WWW. In *Proc. of RIAO*, 2000.
- [GN14] M. Goudi and P. Nocera. Sounds and Symbols: An Overview of Different Types of Methods Dealing With Letter-To-Sound Relationships In A Wide Range Of Languages In Automatic Speech Recognition. In *Proc. of SLTU*, 2014.
- [Gol06] J. Goldsmith. An Algorithm for the Unsupervised Learning of Morphology. *Natural Language Engineering*, 12(04):353–371, 2006.
- [Gol10] J.A. Goldsmith. *Segmentation and Morphology*, pages 364–393. Wiley-Blackwell, 2010.
- [Gre11] L. Gren. Enhancing Language Models for ASR using RSS Feeds. Bachelor’s thesis (studienarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2011.
- [Gre13] L. Gren. Unsupervised Language Model Adaptation for Automatic Speech Recognition of Broadcast News Using Web 2.0. Master’s thesis (diplomarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2013.
- [HAH01] X. Huang, A. Acero, and H.-W. Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall, 2001.
- [Har14] M. P. Harper. IARPA Babel Program. <http://www.iarpa.gov/index.php/research-programs/babel>, 2014. Accessed on 15th May 2014.
- [HCZL00] C. Huang, E. Chang, J. Zhou, and K.-F. Lee. Accent Modeling Based On Pronunciation Dictionary Adaptation For Large Vocabulary Mandarin Speech Recognition. In *Proc. of Interspeech*, 2000.

- [HDB12] C. V. Heerden, M. Davel, and E. Barnard. The Semi-Automated Creation of Stratified Speech Corpora. In *Proc. of PRASA*, 2012.
- [He09] Q. He. Automatic Pronunciation Dictionary Generation from Wiktionary and Wikipedia. Bachelor's thesis (studienarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2009.
- [He14] Q. He. RLAT Light - An Enhanced Version for Novices of the Rapid Language Adaptation Toolkit. Master's thesis (diplomarbeit), KIT, CSL, Germany, 2014.
- [HG08] B.-J. Hsu and J. Glass. Iterative Language Model Estimation: Efficient Data Structure & Algorithms. In *Proc. of Interspeech*, 2008.
- [HH09] C. A. Henriquez and A. Hernandez. A N-gram-based Statistical Machine Translation Approach for Text Normalization on Chat-speak Style Communications. CAW2 (Content Analysis in Web 2.0), April 2009.
- [HKD08] C. Haruechaiyasak, S. Kongyoung, and M. Dailey. A Comparative Study on Thai Word Segmentation Approaches. In *Proc. of ECTI-CON*, 2008.
- [HLMW08] A. M. Harrison, W. Y. Lau, H. M. Meng, and L. Wang. Improving Mispronunciation Detection and Diagnosis of Learners' Speech with Context-sensitive Phonological Rules based on Language Transfer. In *Proc. of Interspeech*, 2008.
- [HLQM09] A. M. Harrison, W. Lo, X. Qian, and H. Meng. Implementation of an Extended Recognition Network for Mispronunciation Detection and Diagnosis in Computer-Assisted Pronunciation Training. In *Proc. of SLaTE*, 2009.
- [HMA<sup>+</sup>99] D. Herron, W. Menzel, E. Atwell, R. Bisiani, F. Daneluzzi, R. Morton, and J. A. Schmidt. Automatic Localization and Diagnosis of Pronunciation Errors for Second-Language Learners of English. In *Proc. of Eurospeech*, 1999.
- [HN00] B. Heine and D. Nurse. *African Languages: An Introduction*. Cambridge University Press, 2000.

- [Hof14] S. Hoffmann. *A Data-driven Model for the Generation of Prosody from Syntactic Sentence Structures*. PhD thesis, ETH Zürich, 2014.
- [HVB12] S. Hahn, P. Vozila, and M. Bisani. Comparison of Grapheme-to-Phoneme Methods on Large Pronunciation Dictionaries and LVCSR Tasks. In *Proc. of Interspeech*, 2012.
- [HW98] J. J. Humphries and P. C. Woodland. The Use of Accent Specific Pronunciation Dictionaries in Acoustic Model Training. In *Proc. of ICASSP*, 1998.
- [HWP96a] J. J. Humphries, P. C. Woodland, and D. Pearce. Using Accent-Specific Pronunciation Modelling For Robust Speech Recognition. In *Proc. of ICSLP*, 1996.
- [HWP96b] J. J. Humphries, P.C. Woodland, and D. Pearce. Using Accent-Specific Pronunciation Modelling for Robust Speech Recognition. In *Proc. of ICSLP*, 1996.
- [IBC<sup>+</sup>12] D. Imseng, H. Bourlard, H. Caesar, P. N. Garner, G. Lecorvé, and A. Nanchen. MediaParl: Bilingual Mixed Language Accented Speech Database. In *Proc. of SLT*, 2012.
- [IO99] R. Iyer and M. Ostendorf. Relevance Weighting for Combining Multidomain Data for N-Gram Language Modeling. *Computer Speech & Language*, 13(3):267–282, 1999.
- [IPA99] *Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet*. Cambridge University Press, 1999.
- [Jav] Oracle, Java website. <http://www.java.com/de/>.
- [JDG<sup>+</sup>13] A. Jansen, E. Dupoux, S. Goldwater, M. Johnson, S. Khudanpur, K. Church, N. Feldman, H. Hermansky, F. Metze, R. Rose, et al. A Summary of the 2012 JHU CLSP Workshop on Zero Resource Speech Technologies and Models of Early Language Acquisition. In *Proc. of ICASSP*, 2013.
- [JFI12] D. Jouvét, D. Fohr, and I. Illina. Evaluating Grapheme-to-Phoneme Converters in Automatic Speech Recognition Context. In *Proc. of ICASSP*, 2012.
- [JG09] M. Johnson and S. Goldwater. Improving Non-Parameteric Bayesian Inference: Experiments on Unsupervised Word Seg-

- mentation with Adaptor Grammars. In *Proc. of HLT-NAACL*, 2009.
- [JKS07] S. Jiampoamarn, G. Kondrak, and T. Sherif. Applying Many-to-Many Alignments and Hidden Markov Models to Letter-to-Phoneme Conversion. In *Proc. of HLT*, 2007.
- [JM00] D. Jurafsky and J. H. Martin. *Speech and Language Processing*. Prentice Hall, 2000.
- [JMLC99] K. S. Jeong, S. H. Myaeng, J. S. Lee, and K.-S. Choi. Automatic Identification and Back-Transliteration of Foreign Words for Information Retrieval. *Information Processing and Management*, 35:523–540, 1999.
- [Joh08] M. Johnson. Using Adaptor Grammars to Identify Synergies in the Unsupervised Acquisition of Linguistic Structure. In *Proc. of ACL-HLT*, 2008.
- [KB06] J. Kominek and A. W. Black. Learning Pronunciation Dictionaries: Language Complexity and Word Selection Strategies. In *Proc. of ACL-HLT*, 2006.
- [KC02] B. Kang and K. Choi. Effective Foreign Word Extraction for Korean Information Retrieval. *Information Processing and Management*, 38, 2002.
- [KC12] B. Kundu and S. Chandra. Automatic Detection of English Words in Benglish Text. In *Proc. of IHCI*, 2012.
- [Kem99] T. Kemp. *Ein automatisches Indexierungssystem für Fernsehnachrichtensendungen*. PhD thesis, 1999.
- [KHB<sup>+</sup>07] P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proc. of ACL, Demonstration Session*, 2007.
- [Kit00] C. Kit. Unsupervised Lexical Learning as Inductive Inference. Technical report, CityU of HK Press, 2000.
- [KK94] R. M. Kaplan and M. Kay. Regular Models of Phonological Rule Systems. *Comput. Linguist.*, 20(3):331–378, September 1994.

- [KL10] P. Karanasou and L. Lamel. Comparing SMT Methods For Automatic Generation Of Pronunciation Variants. In *Proc. of IceTAL*, 2010.
- [KN02] S. Kanthak and H. Ney. Context-Dependent Acoustic Modeling Using Graphemes For Large Vocabulary Speech Recognition. In *Proc. of ICASSP*, 2002.
- [Kne00] R. Kneser. Grapheme-to-Phoneme Study. Technical Report WYT-P4091/00002, Philips Speech Processing, Germany, 2000.
- [Kni99] K. Knight. A Statistical MT Tutorial Workbook. In *Proc. of JHU Summer Workshop*, 1999.
- [Kom06] J. Kominek. *TTS From Zero: Building Synthetic Voices for New Languages*. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 2006.
- [KS12] Y.-B. Kim and B. Snyder. Universal Grapheme-to-Phoneme Prediction Over Latin Alphabets. *Proc. of EMNLP*, 2012.
- [KSN<sup>+</sup>13] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura. Grapheme-To-Phoneme Conversion Based On Adaptive Regularization Of Weight Vectors. In *Proc. of Interspeech*, 2013.
- [KSN<sup>+</sup>14] K. Kubo, S. Sakti, G. Neubig, T. Toda, and S. Nakamura. Narrow Adaptive Regularization of Weights for Grapheme-to-Phoneme Conversion. In *Proc. of ICASSP*, 2014.
- [KSNM03] D. Klein, J. Smarr, H. Nguyen, and C. Manning. Named Entity Recognition with Character-Level Models. In *Proc. of NAACL-HLT*, 2003.
- [KSS03] M. Killer, S. Stüker, and T. Schultz. Grapheme based Speech Recognition. In *Proc. of Eurospeech*, 2003.
- [KSTY03] G. Kikui, E. Sumita, T. Takezawa, and S. Yamamoto. Creating Corpora for Speech-to-Speech Translation. In *Proc. of Eurospeech*, 2003.
- [KYD08] C. Kobus, F. Yvon, and G. Damnati. Normalizing SMS: Are Two Metaphors Better Than One? In *Proc. of COLING*, 2008.
- [LB02] A. F. Llitjos and A. W Black. Evaluation and Collection of Proper Name Pronunciations Online. In *Proc. of LREC*, 2002.



- [LCD<sup>+</sup>11] L. Lamel, S. Courcinous, J. Despres, J.-L. Gauvain, Y. Josse, K. Kilgour, F. Kraft, L. V. Bac, H. Ney, M. Nußbaum-Thom, I. Oparin, T. Schlippe, R. Schlüter, T. Schultz, T. Fraga Da Silva, S. Stüker, M. Sundermeyer, B. Vieru, N. T. Vu, A. Waibel, and C. Woehrling. Speech Recognition for Machine Translation in Quaero. In *Proc. of IWSLT*, 2011.
- [LDHM12] G. Lecorve, J. Dines, T. Hain, and P. Motlicek. Supervised and Unsupervised Web-based Language Model Domain Adaptation. In *Proc. of Interspeech*, 2012.
- [LDM09] A. Laurent, P. Deléglise, and S. Meignier. Grapheme-to-Phoneme Conversion using an SMT System. In *Proc. of Interspeech*, 2009.
- [Lee88] K.-F. Lee. On Large-Vocabulary Speaker-Independent Continuous Speech Recognition. *Speech Communication*, 7(4):375 – 379, 1988. Word Recognition in Large Vocabularies.
- [Lei14] S. Leidig. Single and Combined Features for the Detection of Anglicisms in German and Afrikaans. Bachelor’s thesis, Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2014.
- [Lem13] D. Lemcke. Keynounce - A Game for Pronunciation Generation through Crowdsourcing. Bachelor’s thesis (studienarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2013.
- [Len97] K. Lenzo. t2p: Text-to-Phoneme Converter Builder. <http://www.cs.cmu.edu/afs/cs.cmu.edu/user/lenzo/html/areas/t2p/>, 1997.
- [Lev66] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions, and Reversals. *Soviet Physics-Doklady*, 1966. 10:707-710.
- [LG12] C. Lee and J. Glass. A Nonparametric Bayesian Approach to Acoustic Model Discovery. In *Proc. of ACL-HLT*, 2012.
- [LGP08] G. Lecorve, G. Gravier, and P. Sebillot. An Unsupervised Web-based Topic Language Model Adaptation Method. In *Proc. of ICASSP*, 2008.
- [LGS08] G. Lecorve, G. Gravier, and P. Sebillot. On the Use of Web Resources and Natural Language Processing Techniques to Im-

- prove Automatic Speech Recognition Systems. *Proc. of LREC*, 2008.
- [Loc86] Lockman. La Biblia de las Américas. <http://www.lockman.org/lblainfo/>, 1986. Accessed on 15th May 2014.
- [LSF14] M. Paul Lewis, Gary F. Simons, and Charles D. Fenni. *Ethnologue: Languages of the World, Seventeenth edition*. Dallas, Texas: SIL International. Online version: <http://www.ethnologue.com>, 2014.
- [LSS14] S. Leidig, T. Schlippe, and T. Schultz. Automatic Detection of Anglicisms for the Pronunciation Dictionary Generation: A Case Study on our German IT Corpus. In *Proc. of SLTU*, 2014.
- [Mac03] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.
- [Mar08] C. A. D. Martins. *Dynamic Language Modeling for European Portuguese*. dissertation, Universidade de Aveiro, 2008.
- [MBDW13] R. Molapo, E. Barnard, and F. De Wet. A Distributed Approach To Speech Resource Collection. In *Proc. of PRASA*, 2013.
- [MBdW14] R. Molapo, E. Barnard, and F. de Wet. Speech Data Collection In An Under-resourced Language Within A Multilingual Context. In *Proc. of SLTU*, 2014.
- [MBT04] S. R. Maskey, A. W. Black, and L. M. Tomokiyo. Bootstrapping Phonetic Lexicons for New Languages. In *Proc. of ICSLP*, 2004.
- [MD07] O. Martirosian and M. Davel. Error Analysis of a Public Domain Pronunciation Dictionary. In *Proc. of PRASA*, 2007.
- [Mer14] M. Merz. Different Methods for Efficient Semi-Automatic Pronunciation Generation. Master’s thesis (diplomarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2014.
- [Mey09] C. F. Meyer. *Introducing English Linguistics*. Cambridge University Press, 2009.
- [Mih11] Z. Mihaylova. Lexical and Acoustic Adaptation for Multiple Non-Native English Accents. Master’s thesis (diplomarbeit)

- beit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, May 2011.
- [MK06] T. Misu and T. Kawahara. A Bootstrapping Approach for Developing Language Model of New Spoken Dialogue Systems by Selecting Web Texts. In *Proc. of Interspeech*, 2006.
- [MK12] A. A. Mansikkaniemi and M. Kurimo. Unsupervised Vocabulary Adaptation for Morph-based Language Models. In *Proc. of NAACL-HLT*, 2012.
- [MLP03] R. Munro, D. Ler, and J. Patrick. Meta-learning Orthographic and Contextual Models for Language Independent Named Entity Recognition. In *Proc. of NAACL-HLT*, 2003.
- [MPR01] M. Mohri, F. Pereira, and M. Riley. Weighted Finite-State Transducers in Speech Recognition, 2001.
- [MS99] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
- [NMH12] J. Novak, N. Minematsu, and K. Hirose. WFST-based Grapheme-to-Phoneme Conversion: Open Source Tools for Alignment, Model-Building and Decoding. In *Proc. of FSMNLP*, 2012.
- [Nov11] J. Novak. *Phonetisaurus: A WFST-driven Phoneticizer*, 2011.
- [Och09] S. Ochs. Verbesserung der automatischen Transkription von englischen Wörtern in deutschen Vorlesungen. Bachelor's thesis (studienarbeit), KIT, ISL, Germany, 2009.
- [Okr08] Marc Okrand. *The Star Trek: The Klingon Dictionary*. Simon and Schuster, 2008.
- [omn14] Omniglot: the online encyclopedia of writing systems and languages, 2014. <http://www.omniglot.com>.
- [ON00] F. J. Och and H. Ney. Improved Statistical Alignment Models. In *Proc. in ACL*, 2000.
- [ON03] F. J. Och and H. Ney. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, 2003.

- [ON07] K. Ohtsuki and L. Nguyen. Incremental Language Modeling for Automatic Transcription of Broadcast News. *IEICE Transactions on Information and Systems*, 90(2):526–532, 2007.
- [O’S87] D. O’Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley series in electrical engineering. Addison-Wesley Pub. Co., 1987.
- [OWS08] S. Ochs, M. Wölfel, and S. Stüker. Verbesserung der automatischen Transkription von englischen Wörtern in deutschen Vorlesungen. In *Proc. of ESSV*, 2008.
- [PB92] D. B. Paul and J. M. Baker. The Design for the Wall Street Journal-based CSR Corpus. In *Proc. of Speech and Natural Language*, 1992.
- [PGMSPL11] G. Pauw, G.-M. Schryver, L. Pretorius, and L. Levini. Introduction to the Special Issue on African Language Technology. *Language Resources and Evaluation*, 45, 2011.
- [Pow11] D. M. W. Powers. Evaluation: From Precision, Recall and F-Measure to ROC, Informedness, Markedness Correlation. *Journal of Machine Learning Technologies*, 2(1):37–63, 2011.
- [PPPH04] N. Parihar, J. Picone, D. Pearce, and H.-G. Hirsch. Performance Analysis of the Aurora Large Vocabulary Baseline System. In *Proc. of EUSIPCO*, 2004.
- [Qua13] W. Quaschnigk. Analyzing single and combined g2p converter outputs over training data. Bachelor’s thesis (studienarbeit), Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2013.
- [Qui86] J. R. Quinlan. Induction of Decision Trees. *Machine Learning*, 1(1):81–106, 1986.
- [Rab89] L. R. Rabiner. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. In *Proc. of IEEE*, 1989.
- [RBF<sup>+</sup>99] M. Riley, W. Byrne, M. Finke, S. Khudanpur, A. Ljolje, J. W. McDonough, H. J. Nock, M. Saraclar, C. Wooters, and G. Zavaliagos. Stochastic Pronunciation Modelling From Hand-Labelled Phonetic Corpora. *Speech Communication*, 29(2-4):209–224, 1999.

- [RGN08] M. Raab, R. Gruhn, and E. Nöth. Multilingual Weighted Codebooks for Non-Native Speech Recognition. In *Proc. of TSD*, 2008.
- [RHL<sup>+</sup>11] D. Rybach, S. Hahn, P. Lehnen, D. Nolden, M. Sundermeyer, Z. Tüske, S. Wiesler, R. Schlüter, and H. Ney. RASR - The RWTH Aachen University Open Source Speech Recognition Toolkit. In *Proc. of ASRU*, 2011.
- [RIS<sup>+</sup>10] J. Ross, L. Irani, M. S. Silberman, A. Zaldivar, and B. Tomlinson. Who are the Crowdworkers?: Shifting Demographics in Mechanical Turk. In *Proc. of CHI*, 2010.
- [RMD13] R. Rasipuram and M. Magimai.-Doss. Probabilistic Lexical Modeling and Grapheme-based Automatic Speech Recognition. Technical report, 2013.
- [RN88] J. L. Rodgers and W. A. Nicewander. Thirteen Ways to Look at the Correlation Coefficient. *The American Statistician*, 42(1):59–66, 1988.
- [Rom11] Romanization System in Ukraine. 2011.
- [Ros95] R. Rosenfeld. Optimizing Lexical and N-Gram Coverage via Judicious Use of Linguistic Data. In *Proc. of Eurospeech*, 1995.
- [RSW99] J. Reichert, T. Schultz, and A. Waibel. Mandarin Large Vocabulary Speech Recognition Using The GlobalPhone Database. In *Proc. of Eurospeech*, 1999.
- [SBB<sup>+</sup>07] T. Schultz, A. W. Black, S. Badaskar, M. Hornyak, and J. Kominek. SPICE: Web-based Tools for Rapid Language Adaptation in Speech Processing Systems. *Proc. of Interspeech*, 2007.
- [SBW09] S. Stüker, L. Besacier, and A. Waibel. Human Translations Guided Language Discovery for ASR Systems. In *Proc. of Interspeech*, 2009.
- [SC08] I. Steinwart and A. Christmann. *Support Vector Machines*. Information Science and Statistics. 2008.
- [SCB<sup>+</sup>05] S. Suebvisai, P. Charoenpornasawat, A. W. Black, M. Woszczyna, and T. Schultz. Thai Automatic Speech Recognition. In *Proc. of ICASSP*, 2005.

- [Sch02] T. Schultz. GlobalPhone: A Multilingual Speech and Text Database Developed at Karlsruhe University. In *Proc. of IC-SLP*, 2002.
- [Sch04] T. Schultz. Towards Rapid Language Portability of Speech Processing Systems. In *Proc. of SPLASH*, 2004.
- [SCT04] S.-C-Tseng. Processing Mandarin Spoken Corpora. *Traitement Automatique des Langues, Special Issue: Spoken Corpus Processing*, 45(2):89–108, 2004.
- [SDV<sup>+</sup>12] T. Schlippe, E. Guevara Komgang Djomgang, N. T. Vu, S. Ochs, and T. Schultz. Hausa Large Vocabulary Continuous Speech Recognition. In *Proc. of SLTU*, 2012.
- [SFBW06] S. Stüker, C. Fügen, S. Burger, and M. Wölfel. Cross-System Adaptation and Combination for Continuous Speech Recognition: The Influence of Phoneme Set and Acoustic Front-End. In *Proc. of Interspeech - ICSLP*, 2006.
- [SGG05] R. Sarikaya, A. Gravano, and Y. Gao. Rapid Language Model Development using External Resources for New Spoken Dialog Domains. In *Proc. of ICASSP*, 2005.
- [SGN] A. Sethy, P. G. Georgiou, and S. Narayanan. Building Topic Specific Language Models from Webdata using Competitive Models. In *Proc. of Eurospeech*.
- [SGVS13] T. Schlippe, L. Gren, N. T. Vu, and T. Schultz. Unsupervised Language Model Adaptation for Automatic Speech Recognition of Broadcast News Using Web 2.0. In *Proc. of Interspeech*, 2013.
- [SK06] T. Schultz and K. Kirchhoff, editors. *Multilingual Speech Processing*. Academic Press, Amsterdam, 2006.
- [SMFW01] H. Soltau, F. Metze, C. Fügen, and Alex Waibel. A One-Pass Decoder Based On Polymorphic Linguistic Context Assignment. In *Proc. of ASRU*, 2001.
- [SMS14] T. Schlippe, M. Merz, and T. Schultz. Methods for Efficient Semi-Automatic Pronunciation Dictionary Bootstrapping. In *Proc. of Interspeech*, 2014.

- [SNV08] T. Schlippe, T. Nguyen, and S. Vogel. Diacritization as a Translation Problem and as a Sequence Labeling Problem. In *Proc. of AMTA*, 2008.
- [SOS10] T. Schlippe, S. Ochs, and Tanja Schultz. Wiktionary as a Source for Automatic Pronunciation Extraction. In *Proc. of Interspeech*, 2010.
- [SOS12a] T. Schlippe, S. Ochs, and T. Schultz. Automatic Error Recovery for Pronunciation Dictionaries. In *Proc. of Interspeech*, 2012.
- [SOS12b] T. Schlippe, S. Ochs, and T. Schultz. Grapheme-to-Phoneme Model Generation for Indo-European Languages. In *Proc of ICASSP*, 2012.
- [SOS14] T. Schlippe, S. Ochs, and T. Schultz. Web-based tools and methods for rapid pronunciation dictionary creation. *Speech Communication*, 56(0):101 – 118, 2014.
- [SPC<sup>+</sup>13] S. Sitaram, S. Palkar, Y.-N. Chen, Alok Parlikar, and Alan W Black. Bootstrapping Text-to-Speech for Speech Processing in Languages Without an Orthography. In *Proc. of ICASSP*, 2013.
- [SQS14] T. Schlippe, W. Quaschnigk, and T. Schultz. Combining Grapheme-to-Phoneme Converter Outputs for Enhanced Pronunciation Generation in Low-Resource Scenarios. In *Proc. of SLTU*, 2014.
- [SS04] S. Stüker and T. Schutz. A Grapheme-based Speech Recognition System for Russian. In *Proc. of SPECOM*, 2004.
- [SS14] T. Schultz and T. Schlippe. GlobalPhone: Pronunciation Dictionaries in 20 Languages. In *Proc. of LREC*, 2014.
- [SSB<sup>+</sup>08] S. Seng, S. Sam, L. Besacier, B. Bigi, and E. Castelli. First Broadcast News Transcription System for Khmer Language. In *Proc. of LREC*, 2008.
- [SSK10] Hagen Soltau, George Saon, and Brian Kingsbury. The IBM Attila Speech Recognition Toolkit. In *Proc. of SLT*, 2010.
- [SSVS12] F. Stahlberg, T. Schlippe, S. Vogel, and T. Schultz. Word Segmentation through Cross-Lingual Word-to-Phoneme Alignment. In *Proc. of SLT*, 2012.

- [SSVS13] F. Stahlberg, T. Schlippe, S. Vogel, and T. Schultz. Pronunciation Extraction from Phoneme Sequences through Cross-Lingual Word-to-Phoneme Alignment. In *Proc. of SLSP*, 2013.
- [SSVS14a] F. Stahlberg, T. Schlippe, S. Vogel, and T. Schultz. Towards Automatic Speech Recognition without Pronunciation Dictionary, Transcribed Speech and Text Resources in the Target Language using Cross-Lingual Word-to-Phoneme Alignment. In *Proc. of SLTU*, 2014.
- [SSVS14b] F. Stahlberg, T. Schlippe, S. Vogel, and T. Schultz. Word Segmentation and Pronunciation Extraction from Phoneme Sequences Through Cross-Lingual Word-to-Phoneme Alignment. *Computer Speech & Language*, (0), 2014.
- [Sta11] F. Stahlberg. Discovering Vocabulary of a Language through Cross-Lingual Alignment. Bachelor's thesis, Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2011.
- [Sta14] F. Stahlberg. Towards Automatic Speech Recognition for Non-Written Languages Using Translations From Other Languages. Bachelor's thesis, Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, 2014.
- [Sto02] A. Stolcke. SRILM – an Extensible Language Modeling Toolkit. In *Proc. of Interspeech 2006 - ICSLP*, 2002.
- [SVS13] T. Schultz, N. T. Vu, and T. Schlippe. GlobalPhone: A Multilingual Text & Speech Database in 20 Languages. In *Proc. of ICASSP*, 2013.
- [SVYS13] T. Schlippe, M. Volovyk, K. Yurchenko, and T. Schultz. Rapid Bootstrapping of a Ukrainian Large Vocabulary Continuous Speech Recognition System. In *Proc. of ICASSP*, 2013.
- [SW00] T. Schultz and A. Waibel. Polyphone Decision Tree Specialization for Language Adaptation. In *Proc. of ICASSP*, 2000.
- [SW01] T. Schultz and A. Waibel. Language-independent and Language-adaptive Acoustic Modeling for Speech Recognition. *Speech Communication*, 35(1):31–51, 2001.
- [SW08] S. Stüker and A. Waibel. Towards Human Translations Guided Language Discovery for ASR Systems. In *Proc. of SLTU*, 2008.



- [SZGS10] T. Schlippe, C. Zhu, J. Gebhardt, and T. Schultz. Text Normalization based on Statistical Machine Translation and Internet User Support. In *Proc. of Interspeech*, 2010.
- [SZLS13] T. Schlippe, C. Zhu, D. Lemcke, and T. Schultz. Statistical Machine Translation based Text Normalization with Crowdsourcing. In *Proc. of ICASSP*, 2013.
- [TB99] L. Mayfield Tomokiyo and S. Burger. Eliciting Natural Speech from Non-Native Users Collecting Speech Data for LVCSR. In *Proc. of ACL-IALL Joint Workshop on Computer Mediated Language Assessment and Evaluation in NLP*, 1999.
- [TDK04] Y. Tsubota, M. Dantsuji, and T. Kawahara. An English Pronunciation Learning System for Japanese Students Based on Diagnosis of Critical Pronunciation Errors. *ReCALL*, 16(1):173–188, May 2004.
- [Tho90] R. L. Thomas. Bible Translations: The Link Between Exegesis and Expository Preaching. *The Masters Seminary Journal*, 1:53–74, 1990.
- [TM02] K. Toutanova and R. C. Moore. Pronunciation Modeling for Improved Spelling Correction. In *Proc. of ACL*, 2002.
- [TMWW00] W. J. Teahan, Rodger McNab, Yingying Wen, and Ian H. Witten. A Compression-based Algorithm for Chinese Word Segmentation. *Comput. Linguist.*, 26(3):375–393, September 2000.
- [Tom00a] L. Mayfield Tomokiyo. Handling Non-native Speech in LVCSR: A Preliminary Study. In *Proc. of InSTIL*, 2000.
- [Tom00b] L. Mayfield Tomokiyo. Lexical and Acoustic Modeling of Non-native Speech in LVCSR. In *Proc. of ICSLP*, 2000.
- [TW01] L. Mayfield Tomokiyo and A. Waibel. Adaptation Methods for Non-Native Speech. In *Multilinguality in Spoken Language Processing*, 2001.
- [Ure71] J. Ure. Lexical Density and Register Differentiation. *Applications of linguistics*, pages 443–452, 1971.
- [vA06] L. von Ahn. Games with a Purpose. *Computer*, 39(6):92–94, June 2006.

- [vALB06] L. von Ahn, R. Liu, and M. Blum. Peekaboom: A Game for Locating Objects in Images. In *Proc. of SIGCHI*, 2006.
- [VALR03] P. Vozila, J. Adams, Y. Lobacheva, and T. Ryan. Grapheme to Phoneme Conversion and Dictionary Verification using Graphonemes. In *Proc. of Eurospeech*, 2003.
- [VIM04] VIM. International Vocabulary of Basic and General Terms in Metrology. *International Organization*, pages 09–14, 2004.
- [Vit67] A.J. Viterbi. Error Bounds For Convolutional Codes And An Asymptotically Optimum Decoding Algorithm. *Information Theory, IEEE Transactions on*, 13(2):260–269, April 1967.
- [VKD08] B. Varadarajan, S. Khudanpur, and E. Dupoux. Unsupervised Learning of Acoustic Sub-Word Units. In *Proc. of ACL-HLT*, 2008.
- [VKS11] N. T. Vu, F. Kraus, and T. Schultz. Rapid Building of an ASR System for Under-Resourced Languages Based on Multilingual Unsupervised Training. In *Proc. of Interspeech*, 2011.
- [VLG10] D. Vergyri, L. Lamel, and J.-L. Gauvain. Automatic Speech Recognition of Multiple Accented English Data. In *Proc. of Interspeech*, 2010.
- [VLW<sup>+</sup>12] N. T. Vu, D.-C. Lyu, J. Weiner, D. Telaar, T. Schlippe, F. Blaicher, E.-S. Chng, T. Schultz, and H. Li. A First Speech Recognition System For Mandarin-English Code-Switch Conversational Speech. In *Proc. of ICASSP*, 2012.
- [VNT96] S. Vogel, H. Ney, and C. Tillmann. HMM-based Word Alignment in Statistical Translation. In *Proc. in COLING*, 1996.
- [VS09] N. T. Vu and T. Schultz. Vietnamese Large Vocabulary Continuous Speech Recognition. In *Proc. of ASRU*, 2009.
- [VSKS10] N. T. Vu, T. Schlippe, F. Kraus, and T. Schultz. Rapid Bootstrapping of five Eastern European Languages using the Rapid Language Adaptation Toolkit. In *Proc. of Interspeech*, 2010.
- [WEH02] M. Wolff, M. Eichner, and R. Hoffmann. Measuring the Quality of Pronunciation Dictionaries. In *Proc. of PMLA*, 2002.
- [Wei91] C. J. Weinstein. Opportunities for Advanced Speech Processing in Military Computer-Based Systems. Technical report,

- Defense Technical Information Center, Massachusetts Institute of Tech Lexington Lincoln Lab, 1991.
- [Wei10] S. Weinberger. The Speech Accent Archive. [accent.gmu.edu](http://accent.gmu.edu). George Mason University. 2010.
- [Wel97] J.C. Wells. SAMPA Computer Readable Phonetic Alphabet. *Gibbon, D., Moore, R. and Winski, R. (eds.), 1997. Handbook of Standards and Resources for Spoken Language Systems. Berlin and New York: Mouton de Gruyter. Part IV, section B., 1997.*
- [Wik12] Wikimedia. List of Wiktionary editions, ranked by article count, 2012.
- [wik14] List of Wiktionaries, 2014. [http://meta.wikimedia.org/wiki/List\\_of\\_wiktionaries](http://meta.wikimedia.org/wiki/List_of_wiktionaries).
- [Woo99] P.C. Woodland. Speaker Adaptation: Techniques and Challenges. In *Proc. of ASRU*, 1999.
- [WS03] Z. Wang and T. Schultz. Non-Native Spontaneous Speech Recognition through Polyphone Decision Tree Specialization. In *Proc. of Eurospeech*, 2003.
- [WSS+00] A. Waibel, H. Soltau, T. Schultz, T. Schaaf, and F. Metze. Multilingual Speech Recognition. *Verbmobil: Foundations of Speech-to-Speech Translation, ed. Wolfgang Wahlster, Springer Verlag*, 2000.
- [WSW03] Z. Wang, T. Schultz, and A. Waibel. Comparison of Acoustic Model Adaptation Techniques on Non-Native Speech. In *Proc. of ICASSP*, 2003.
- [WVD95] F. Wolinski, F. Vichot, and B. Dillet. Automatic Processing of Proper Names in Texts. In *Proc. of EACL*, 1995.
- [You96] S. Young. Large Vocabulary Continuous Speech Recognition: A Review. In *Proc. of ASRU*, 1996.
- [YOW94] S. Young, J.J. Odell, and P. C. Woodland. Tree-Based State Tying for High Accuracy Acoustic Modelling. In *Proc. of HLT*, 1994.
- [YTTWW00] H. Yu, T. Tomokiyo, Z. Wang, and A. Waibel. New Developments In Automatic Meeting Transcription. In *Proc. of IC-SLP*, 2000.

- [Yur13] K. Yurchenko. Cross-Lingual Pronunciation Dictionary Production. Bachelor's thesis, Karlsruhe Institute of Technology, Cognitive Systems Lab, Germany, May 2013.
- [ZSS06] I. Zitouni, J. S. Sorensen, and R. Sarikaya. Maximum Entropy Based Restoration of Arabic Diacritics. In *Proc. of COLING*, 2006.

## Publications

### Journal Articles

- Tim Schlippe, Sebastian Ochs, and Tanja Schultz. Web-based Tools and Methods for Rapid Pronunciation Dictionary Creation. *Speech Communication*, 56(0):101 – 118, 2014.
- Felix Stahlberg, Tim Schlippe, and Tanja Schultz. Word Segmentation and Pronunciation Extraction from Phoneme Sequences Through Cross-Lingual Word-to-Phoneme Alignment. *Computer Speech & Language*, 2014, Available online: 18 October 2014.

### Peer-Reviewed Conference Publications

#### 2014

- Tim Schlippe, Matthias Merz, and Tanja Schultz. Methods for Efficient Semi-Automatic Pronunciation Dictionary Bootstrapping. In *Proceedings of The 15th Annual Conference of the International Speech Communication Association (Interspeech 2014)*, Singapore, 14-18 August 2014.
- Dominic Telaar, Michael Wand, Dirk Gehrig, Felix Putze, Christoph Amma, Dominic Heger, Ngoc Thang Vu, Mark Erhardt, Tim Schlippe, Matthias Janke, Christian Herff, and Tanja Schultz. BioKIT - Real-time Decoder For Biosignal Processing. In *Proceedings of The 15th Annual Conference of the International Speech Communication Association (Interspeech 2014)*, Singapore, 14-18 August 2014.
- Tanja Schultz and Tim Schlippe. GlobalPhone: Pronunciation Dictionaries in 20 Languages. In *Proceedings of The 9th edition of the Language Resources and Evaluation Conference (LREC 2014)*, Reykjavik, Iceland, 26-31 May 2014.
- Tim Schlippe, Wolf Quaschnigk, and Tanja Schultz. Combining Grapheme-to-Phoneme Converter Outputs for Enhanced Pronunciation Generation in Low-Resource Scenarios. In *Proceedings of The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2014)*, St. Petersburg, Russia, 14-16 May 2014.

- Sebastian Leidig, Tim Schlippe, and Tanja Schultz. Automatic Detection of Anglicisms for the Pronunciation Dictionary Generation: A Case Study on our German IT Corpus. In Proceedings of The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2014), St. Petersburg, Russia, 14-16 May 2014.
- Felix Stahlberg, Tim Schlippe, Stephan Vogel, and Tanja Schultz. Towards Automatic Speech Recognition without Pronunciation Dictionary, Transcribed Speech and Text Resources in the Target Language using Cross-Lingual Word-to-Phoneme Alignment. In Proceedings of The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2014), St. Petersburg, Russia, 14-16 May 2014.
- Heike Adel, Kartin Kirchhoff, Dominic Telaar, Ngoc Thang Vu, Tim Schlippe, and Tanja Schultz. Features for Factored Language Models for Code-Switching Speech. In Proceedings of The 4th Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2014), St. Petersburg, Russia, 14-16 May 2014.

## 2013

- Tim Schlippe, Lukasz Gren, Ngoc Thang Vu, and Tanja Schultz. Unsupervised Language Model Adaptation for Automatic Speech Recognition of Broadcast News Using Web 2.0. In Proceedings of The 14th Annual Conference of the International Speech Communication Association (Interspeech 2013), Lyon, France, 25-29 August 2013.
- Felix Stahlberg, Tim Schlippe, Stephan Vogel, and Tanja Schultz. Pronunciation Extraction from Phoneme Sequences through Cross-Lingual Word-to-Phoneme Alignment. In Proceedings of The 1st International Conference on Statistical Language and Speech Processing (SLSP 2013), Tarragona, Spain, 29-31 July 2013.
- Tim Schlippe, Mykola Volovyk, Kateryna Yurchenko, and Tanja Schultz. Rapid Bootstrapping of a Ukrainian Large Vocabulary Continuous Speech Recognition System. In Proceedings of The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), Vancouver, Canada, 26-31 May 2013.

- Tanja Schultz, Ngoc Thang Vu, and Tim Schlippe. GlobalPhone: A Multilingual Text & Speech Database in 20 Languages. In Proceedings of The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), Vancouver, Canada, 26-31 May 2013.
- Heike Adel, Ngoc Thang Vu, Franziska Kraus, Tim Schlippe, Haizhou Li, and Tanja Schultz. Recurrent Neural Network Language Modeling for Code Switching Conversational Speech. In Proceedings of The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), Vancouver, Canada, 26-31 May 2013.
- Tim Schlippe, Chenfei Zhu, Daniel Lemcke, and Tanja Schultz. Statistical Machine Translation based Text Normalization with Crowdsourcing. In Proceedings of The 38th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2013), Vancouver, Canada, 26-31 May 2013.

## 2012

- Felix Stahlberg, Tim Schlippe, Stephan Vogel, and Tanja Schultz. Word Segmentation through Cross-Lingual Word-to-Phoneme Alignment. In Proceedings of The Fourth IEEE Workshop on Spoken Language Technology (SLT 2012), Miami, Florida, 2-5 December 2012.
- Tim Schlippe, Sebastian Ochs, Ngoc Thang Vu, and Tanja Schultz. Automatic Error Recovery for Pronunciation Dictionaries. In Proceedings of The 13th Annual Conference of the International Speech Communication Association (Interspeech 2012), Portland, Oregon, 9-13 September 2012.
- Tim Schlippe, Edy Guevara Komgang Djomgang, Ngoc Thang Vu, Sebastian Ochs, and Tanja Schultz. Hausa Large Vocabulary Continuous Speech Recognition. In Proceedings of The 3rd Workshop on Spoken Language Technologies for Under-resourced Languages (SLTU 2012), Cape Town, South Africa, 7-9 May 2012.
- Tim Schlippe, Sebastian Ochs, and Tanja Schultz. Grapheme-to-Phoneme Model Generation for Indo-European Languages. In Proceedings of The 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012), Kyoto, Japan, 25-30 March 2012.

- Ngoc Thang Vu, Dau-Cheng Lyu, Jochen Weiner, Dominic Telaar, Tim Schlippe, Fabian Blaicher, Eng-Siong Chng, Tanja Schultz, and Haizhou Li. A First Speech Recognition System For Mandarin-English Code-Switch Conversational Speech. In Proceedings of The 37th International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2012), Kyoto, Japan, 25-30 March 2012.

## 2011

- Lori Lamel, Sandrine Courcinous, Julien Despres, Jean-Luc Gauvain, Yvan Josse, Kevin Kilgour, Florian Kraft, Le Viet Bac, Hermann Ney, Markus Nußbaum-Thom, Ilya Oparin, Tim Schlippe, Ralf Schlüter, Tanja Schultz, Thiago Fraga Da Silva, Sebastian Stüker, Martin Sundermeyer, Bianca Vieru, Ngoc Thang Vu, Alexander Waibel, and Cécile Woehrling. Speech Recognition for Machine Translation in Quaero. In Proceedings of The International Workshop on Spoken Language Translation (IWSLT 2011), San Francisco, USA, 8-9 December 2011.

## 2010

- Tim Schlippe, Sebastian Ochs, and Tanja Schultz. Wiktionary as a Source for Automatic Pronunciation Extraction. In Proceedings of The 11th Annual Conference of the International Speech Communication Association (Interspeech 2010), Makuhari, Japan, 26-30 September 2010.
- Ngoc Thang Vu, Tim Schlippe, Franziska Kraus, and Tanja Schultz. Rapid Bootstrapping of five Eastern European Languages using the Rapid Language Adaptation Toolkit. In Proceedings of The 11th Annual Conference of the International Speech Communication Association (Interspeech 2010), Makuhari, Japan, 26-30 September 2010.
- Tim Schlippe, Chenfei Zhu, Jan Gebhardt, and Tanja Schultz. Text Normalization based on Statistical Machine Translation and Internet User Support. In Proceedings of The 11th Annual Conference of the International Speech Communication Association (Interspeech 2010), Makuhari, Japan, 26-30 September 2010.



**2008**

- Tim Schlippe, ThuyLinh Nguyen, and Stephan Vogel. Diacritization as a Translation Problem and as a Sequence Labeling Problem. In Proceedings of The Eighth Conference of the Association for Machine Translation in the Americas (AMTA 2008), Waikiki, Hawai'i, USA, 21-25 October 2008.

**Co-advised Student Theses**

- Felix Stahlberg. Towards Automatic Speech Recognition for Non-Written Languages Using Translations from Other Languages. Master Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Tanja Schultz, in part carried out at Qatar Computing Research Institute (QCRI) in Doha under guidance of Dr. Stephan Vogel.
- Matthias Merz. Different Methods for Efficient Semi-Automatic Pronunciation Generation. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Tanja Schultz.
- Qingyue He. RLAT Light – An Enhanced Version for Novices of the Rapid Language Adaptation Toolkit. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Ngoc Thang Vu and Tanja Schultz.
- Mohamed Yassine Khelifi. Methods for the Analysis of Pronunciation Dictionaries. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Tanja Schultz.
- Sebastian Leidig. Single and Combined Features for the Detection of Anglicisms in German and Afrikaans. Bachelor Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Tanja Schultz.
- Kateryna Yurchenko. Cross-Lingual Pronunciation Dictionary Production. Bachelor Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2013, with Tanja Schultz.
- Wojtek Breiter. Rapid Bootstrapping of Haitian Creole Large Vocabulary Continuous Speech Recognition. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2014, with Ngoc Thang Vu and Tanja Schultz.

- Lukasz Gren. Unsupervised Language Model Adaptation for Automatic Speech Recognition of Broadcast News Using Web 2.0. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2013, with Ngoc Thang Vu and Tanja Schultz.
- Wolf Quaschnigk. Analyzing Single and Combined G2P Converter Outputs over Training Data. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2013, with Tanja Schultz.
- Daniel Lemcke. Keynounce - A Game for Pronunciation Generation through Crowdsourcing. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2013, with Tanja Schultz.
- Adnene Zairi. Management of the Text and Speech Resources in the Rapid Language Adaptation Toolkit. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2013, with Tanja Schultz.
- Dario Ernst. Bootstrapping Pronunciation Dictionaries with Multilingual Phoneme Recognition. Bachelor Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2012, with Ngoc Thang Vu and Tanja Schultz.
- Jan Gebhardt. Speech Recognition on English-Mandarin Code-Switching Data using Factored Language Models - with Port-of-Speech Tags, Language ID and Code-Switch Point Probability as Factors. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Tanja Schultz, in part carried out at Hong Kong University of Science and Technology under guidance of Prof. Pascale Fung.
- Fabian Blaicher. SMT-based Text Generation for Code-Switching Language Models. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Tanja Schultz, in part carried out at Nanyang Technological University, Singapore under guidance of Prof. Haizhou Li and Prof. Chng Eng Siong.
- Zlatka Mihaylova. Lexical and Acoustic Adaptation for Multiple Non-Native English Accents. Diploma Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Ngoc Thang Vu, Dominic Telaar and Tanja Schultz, in part carried out at Carnegie Mellon University, Pittsburgh under guidance of Prof. Florian Metze.

- Felix Stahlberg. Discovering Vocabulary of a Language through Cross-Lingual Alignment. Bachelor Thesis, Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Tanja Schultz, in part carried out at Qatar Computing Research Institute (QCRI) in Doha under guidance of Dr. Stephan Vogel.
- Edy Guevara Komgang Djomgang. Hausa Large Vocabulary Continuous Speech Recognition. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Tanja Schultz.
- Lukasz Gren. Enhancing Language Models for ASR using RSS Feeds. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2011, with Ngoc Thang Vu and Tanja Schultz.
- Zlatka Mihaylova. An Architecture of a Telephone-based System for Speech Data Collection. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2010, with Tanja Schultz.
- Qingyue He. Automatic Pronunciation Dictionary Generation from Wiktionary and Wikipedia. Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2009, with Tanja Schultz.
- Jan Gebhardt. Multilingual Acoustic Model Combination using Rapid Language Adaption Toolkit (RLAT). Student Research Thesis (Studienarbeit), Cognitive Systems Lab, Karlsruhe Institute of Technology, 2009, with Tanja Schultz.
- Ngoc Thang Vu. Entwicklung eines vietnamesischen Spracherkennungssystems für große Vokabulare. Diploma Thesis, Institut für Anthropomatik, Fakultät für Informatik, University of Karlsruhe, 2009, with Tanja Schultz.