Correction of Disfluencies in Spontaneous Speech using a Noisy-Channel Approach

Matthias Honal

Studienarbeit
April 2003

Carnegie Mellon University,
Pittsburgh, PA 15213,
USA

University of Karlsruhe,
76128 Karlsruhe,
Germany

Supervisor:
Tanja Schultz

Acknowledgements

I would like to thank my supervisor Tanja Schultz for all the discussions about my project and the valuable feedback to the preliminary versions of this report. With her advice and her guidance Tanja helped me to progress quickly with my project and to learn a lot about Natural Language Processing and scientific work in general.
I would like to thank Alex Waibel for giving me the opportunity to do the research for this report at the Interactive Systems Lab at Carnegie Mellon University in Pittsburgh.
Many thanks go to Stephan Vogel for the permission to use some components of the statistical machine translation toolkit which is developed at Carnegie Mellon University. Stephan always had time to explain me important details of the code which helped me a lot to adapt the components of this toolkit to the requirements of the problems treated in this work.
Many thanks also go to Fei Huang who provided annotated dialogs from the Mandarin Chinese CallHome corpus.
Finally I would like to thank my fianceé Laura for all the love and encouragement which helped me a lot in times when I found it difficult to progress with my project.

Abstract

In this work we present a system which automatically corrects disfluencies in spontaneous speech dialogues. Disfluencies are all kinds of repairs and filler words which occur frequently in spontaneous speech.
We adopt some ideas of statistical machine translation for the disfluency correction task. Thus the disfluent speech can be seen as source language which has to be translated into a target language. The target language is fluent speech, containing no disfluencies. The statistical models of our system are trained on texts in which disfluencies are annotated manually. We extract information about the structure of disfluencies and about the context in which they occur out of these annotated texts. Then our system can perform the task of disfluency correction on arbitrary sentences.
Advantages of this approach are that (1) no extensive linguistic knowledge is required to develop the system, (2) rapid adaptation to new target languages is possible and (3) models for properties of disfluencies which are not considered in the current system can easily incorporated.
We test our system on two different spontaneous speech corpora: Besides American English dialogues we conduct experiments on Mandarin Chinese speech. On the English data the system achieves a recall of 77.2% and a precision of 90.2%. On the Mandarin Chinese data the recall is 49.4% and the precision 76.8%. The latter results seem to confirm that our approach is applicable to multiple languages.
Nevertheless, our system must still be improved. A tighter coupling with speech recognition engines is desirable, so that word lattices from a speech recognition system can be used as input rather than manually transcribed speech. Furthermore the use of acoustic features for disfluency correction remains to be examined.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

In Natural Language Processing (NLP) it becomes more and more important to deal with spontaneous speech, such as dialogs between two people or even multi-party meetings. The goal of this processing can be the translation, the summarization or simply the archiving of a dialog or a meeting in a written form.
In contrast to texts containing more formal language such as articles in a newspaper or broadcast news texts, spontaneous speech includes a huge number of sentences which are not fluent. The elements which make a sentence not fluent are commonly referred to as "disfluencies". Repetitions or corrections of words uttered previously or complete restarts of sentences are common disfluencies. Further more words like "well", "alright" etc. which do not contribute to the semantic content of the text are considered to be disfluent, when they are only used to fill pauses or to start a turn.
By deleting the disfluent elements in a given sentence, a "clean", i.e. fluent sentence is generated which represents the utterance originally intended by the speaker.
The following example shows a disfluent sentence and the "clean" sentence which is the result of deleting the disfluent parts from the original sentence:

Original sentence:	`Well, the it has it has got a you know breakfast buffet.`
Clean sentence:	`It has got a breakfast buffet.`

As can be seen from the example above, the "clean" sentence is much easier to read, the information the speaker wanted to convey by uttering the sentence is easier to understand and the sentence is more grammatical and much shorter than the disfluent sentence. These aspects may be important to consider for various applications in NLP (section 1.3). Thus it might be desirable to have a system which automatically corrects disfluent sentences and produces a more well formed text without disfluencies as output which can then serve as input for NLP applications.

1.2 Scope and Goal of this work

The overall goal of this work is to develop a system which automatically corrects disfluencies in a text which is a transcription of spontaneous speech. We assume that disfluencies occur on sentence level and do not span over multiple sentences.

Nevertheless, the structure of the discourse in a spontaneous speech dialog may also contain phenomena which make it more difficult to understand. These phenomena could be called discourse related disfluencies. An example for that would be a misunderstanding where one speaker misinterprets the contents of an utterance previously uttered by his dialog partner:

A: `We could take a flight at six o'clock.`
B: `Six o'clock in the morning is too early for me.`
A: `I mean six o'clock in the evening.`

A detection or even a correction of a discourse related disfluency would involve deep semantic understanding of the discourse and the application of many sophisticated techniques involving syntax and semantics which is far beyond the scope of this work.
Furthermore we only consider textual phenomena and no errors like mispronunciation, wrong intonation or stuttering. These errors can only be corrected when taking the acoustics into account. However, the system we propose works on manually transcribed speech, where acoustic errors play no role. If on the other hand automatically transcribed speech from a speech recognizer was considered as input, we would have to deal with errors like this as well. In contrast to manual transcriptions, automatically generated transcripts are very likely to contain a lot of recognizer errors. That means that the words uttered by the speaker are not transcribed correctly, they are omitted or words which have not been uttered are inserted in the transcription. Furthermore these transcripts do not contain any sentence boundaries which complicates readability or further processing even more.
Although the system we propose here does not yet work together with a speech recognizer, its design principals allow an extension in this direction. For spontaneous speech applications manually transcribed speech can not assumed to be available. Thus a coupling between disfluency correction and speech recognition where recognizer errors and acoustic errors are treated must be implemented in further work.
One important goal for the system proposed in this work is, that it should acquire its disfluency correction abilities by training on a disfluent and the corresponding corrected text. After training, the system should be able to perform disfluency correction on an arbitrary sentence using statistical models. That means that on the one hand no extensive expert knowledge about syntax and semantics of a language should be necessary to train the system. On the other hand the system should not use static pattern matching rules (i.e. if some words match a certain pattern, they are disfluent) for disfluency correction, but make decisions which are based on statistical models considering many features. These features are drawn from the given training corpus where disfluencies are manually annotated. Corpora annotated in this way are not largely available, hence we are confronted with a sparse data problem, using this corpus-based approach.
Out of the annotated corpora text-based properties of disfluencies can be extracted. This can be the length or the position of a disfluency, the number of disfluencies occurring in a sentence or the context in which a disfluency occurs. All the features which are used in our system are described in more detail in section 4.2 and in section 3.3 where the feature-based statistical models are explained.
Although some authors have shown that acoustic and prosodic features such as the duration of pauses and words or intonational boundaries might be helpful for disfluency correction [Bear et al., 1992], [Nakatani and Hirschberg, 1994], we do not make use of them within this work. The only prosodic feature we use is the information about word fragments (words interrupted prematurely) which however have been manually annotated in the transcriptions.

Besides the idea of having a system which can operate without deep linguistic knowledge another goal of this work is to examine whether our approach of correcting disfluencies in spontaneous English speech can be generalized to other languages. Therefore experiments on Mandarin Chinese spontaneous speech have been conducted which seem to prove that our system can be easily adapted to other languages as English.

1.3 Potential Applications

When dealing with spontaneous speech recorded from dialogs or meetings, one goal of speech recognition and NLP is to represent the speech in another appropriate form. An appropriate form can be a written transcription, a summary or even the translation of the spontaneous speech into another language, which may be essential to enable people speaking different languages to communicate.
Typical systems that perform these tasks may have a structure as illustrated in figure 1.1. Speech is recorded and then automatically transcribed by a speech recognizer. The output from the speech recognizer is then either passed to a NLP system which for example translates or summarizes the speech recognizer output or it is directly archived on a hard disk. The output of the NLP system may then be passed to a speech synthesizer or it may be archived on a hard disk as well.

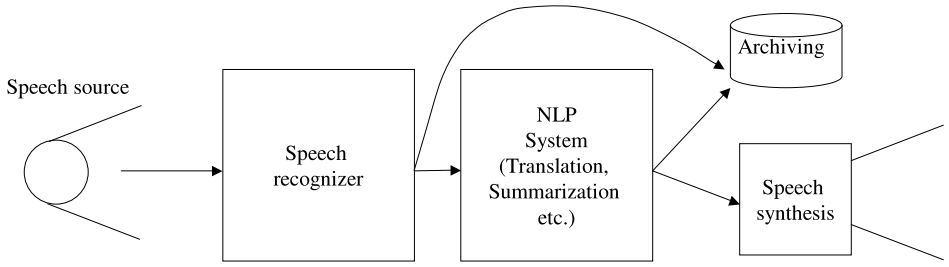

Figure 1.1: Typical system for processing spontaneous speech.

The idea behind archiving spontaneous speech in a written form is to provide easy access to the contents of a dialogs or a meeting which took place in the past. Easy access means that one wants to have texts which are easy to read and one may want to provide (automatically generated) summaries which allow quick information retrieval. In the first case correcting disfluencies is important to improve readability what has been shown in the example in section 1.1. The automatic summarization task may be facilitated a lot when summarization can be done on a fluent text because some non relevant or redundant information, which was present in the disfluent text because of filler words, repetitions and restarts of sentences is removed in the corrected text. Furthermore the input for the summarization system is more well formed and grammatical when disfluencies have been removed which might be important when some parsing is required.
For the purpose of speech-to-speech translation one can pass the automatically transcribed speech through a machine translation system and then synthesize the translated text to speech again. However, one would rather like a fluent text than a text packed with disfluencies as output of the translation system. The generation of a fluent translation will be much easier

for the system when the input is already fluent. Furthermore the assumptions and models in translation systems are usually based on fluent sentences and training corpora for these systems are mostly available as fluent texts.
In order to provide fluent input to the NLP system in figure 1.1 the scenario can be extended by using the disfluency correction system proposed in this work as illustrated in figure 1.2. Here a disfluency correction system is put between the speech recognizer and the NLP system in order to correct the disfluent speech recognizer output and remove disfluencies from the input of the NLP system.

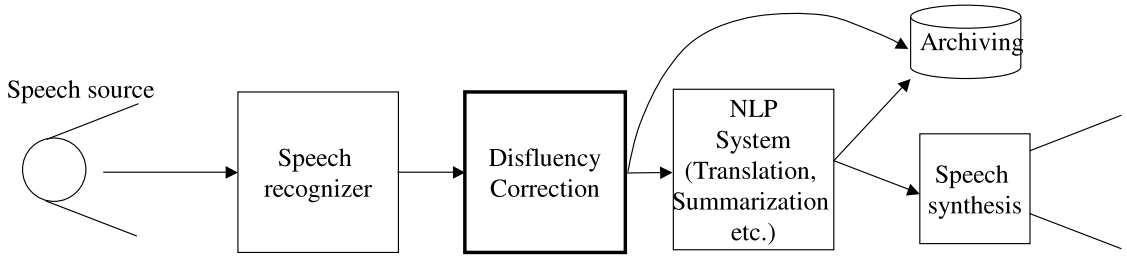

Figure 1.2: System for processing spontaneous speech using disfluency correction system.

As we explained in the previous section, the present system works only on manually transcribed speech. Modifications are necessary in order to use it in a scenario described above. However the architecture of our system allows that it only has to be extended and not to be rebuilt in order to work in such a scenario. Hence the system can be seen as a first step toward a disfluency correction system in a spontaneous speech processing environment.

1.4 Approach

In section 1.2 we stated that we deal only with disfluencies which occur at sentence level and which can be determined by inspection of manually transcribed words of a speech act. In this case, the process of correcting disfluencies can be seen as a process of deletions: A fluent sentence is generated from a disfluent one by deleting those words which do not belong to the utterance which was originally intended by the speaker. The following examples illustrate that:

```
original sentence:  Alright well , we got to plan this trip to Hannover.
clean sentence:                   We got to plan this trip to Hannover.

original sentence:  I have some trip some flights.
clean sentence:     I have           some flights.
```

To model this process, we adopt some ideas from statistical machine translation (SMT) which are then simplified and modified to meet the requirements of the disfluency correction task. The task of SMT is to translate a given sentence from a source language into a target language. To derive statistical models for this translation process, the idea of the noisy channel is very commonly used [Wang and Waibel, 1997]: The sentence in the target language can be seen as

input to a noisy channel. The noisy channel adds noise to its input sentence and creates the sentence in the source language as output. The goal of the translation is now to recover the channel input which is the sentence in the target language given the channel output which is the sentence in the source language.
For the problem of disfluency correction we can make use of this model by defining fluent speech to be the target language and disfluent speech to be the source language. Then the model of the noisy channel can be reformulated as follows: A fluent sentence is the input of the noisy channel. The noisy channel adds noise to the fluent sentence in form of disfluencies and produces as output a disfluent sentence (figure 1.3). The goal of disfluency correction is then to recover the fluent sentence given the disfluent output of the noisy channel.

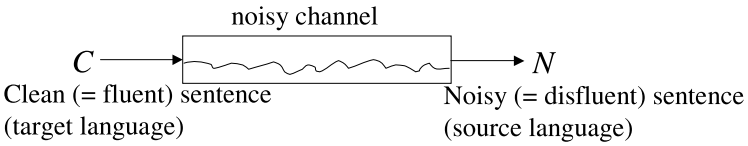

Figure 1.3: The model of the noisy channel for disfluency correction

Speaking in mathematical terms, we have a string C which represents a fluent sentence and a string N which is a disfluent or noisy sentence produced by the noisy channel. We want to find the string $\hat{C}$ that is most likely to be the input of the noisy channel given the channel output N:

$$\hat{C} = \arg\max_{C} P(C|N) \tag{1.1}$$

Similar to SMT the probability $P(C|N)$ in equation (1.1) can be modeled by decomposing it into a translation model and a language model for the target language. By application of Bayes rule one obtains:

$$\hat{C} = \arg\max_{C} P(C|N) = \arg\max_{C} P(N|C) \cdot P(C) \tag{1.2}$$

In equation 1.2 the translation model is represented by the probability $P(N|C)$ and the language model by $P(C)$.
As language models we use n-gram models which are very common in speech recognition and SMT (section 3.2). In contrast, the translation model we use is very different from the models which are commonly used in SMT. Our translation model is designed to meet the special requirements of disfluency correction. This is explained in detail in section 3.3.
To train both models, a "parallel" corpus is required which consists of the same text in the source and target language. In our case this is the transcription of spontaneous speech with annotated disfluencies. From this text the fluent text can be computed deterministically. The parallel corpus is then used to extract all the properties of disfluencies on which our translation model is based. The language model is trained on the fluent text which represents the target language in SMT terms.
In the search phase the string $\hat{C}$ is finally determined by first generating each possible string C from N which can be generated by deletions. Then a probability is assigned to each

hypothesis produced in this way according to equation 1.2. The search over all hypotheses to find the best one is finally done by building a lattice in which each path represents a possible string C. A dynamic programming based search algorithm (similar to the Viterbi search algorithm) is then used to find the best hypothesis $\hat{C}$ in the lattice.
The algorithms to train the system are not very complex and do not require a lot of computational effort. This is however often the case for other NLP applications like SMT. Furthermore, the search space which is built during decoding to find the best hypothesis is very small compared to SMT. Thus even without any pruning the search problem remains tractable in terms of memory and time.
The reason that disfluency correction using our approach does not require as much computational effort as SMT becomes evident, when considering the complexity of the two problems. In SMT a word in a source sentence can be mapped to an arbitrary position in a target sentence, and the corresponding target word can have a number of translations. As we define disfluency correction simply as a sequence of deletions of words, the number of possible target sentences here is much smaller than in SMT (although it still grows exponentially with the sentence length). Hence training and search for disfluency correction is more tractable than for SMT.

1.5 Overview

After this introduction chapter 2 describes in detail the phenomenon of disfluencies in spontaneous speech. First we describe all types of disfluencies we consider in our system. Then a classification scheme for disfluencies which is used by many authors is presented. Finally some related work is reviewed which has been done in analysis, automatic detection and correction of disfluencies.
The statistical models used in our approach are presented in detail in chapter 3. In this chapter we review some basic concepts of SMT and present some principals of statistical language modeling which are used in the language model component of our system. Then we describe our translation model which is based on some SMT concepts but still differs essentially from the translation models known from SMT. Finally we explain how exactly we model the different properties of disfluencies which are used in the translation model.
In chapter 4 the corpora we used for our experiments are described. These are the American English Verbmobil corpus [Wahlster, 2000] and the Mandarin Chinese CallHome corpus [Wheatley, 1997] which we used to demonstrate the potential of our system for rapid adaptation to different languages. In section 4.2 we analyze and compare several properties of disfluencies which can be found in the different corpora. These properties are encoded by the statistical models which are described in chapter 3.
Chapter 5 deals with some issues concerning the training and the implementation of our system. Some ideas for decoding and the structure of the lattice we use are discussed. Then the exact implementation of the models introduced in chapter 3 is explained.
The results from our experiments on the English Verbmobil and the Mandarin CallHome corpus are presented in chapter 6. Furthermore some examples of the output of the disfluency correction system for the English Verbmobil corpus are given.
In chapter 7 we draw some conclusions from our work and point out some possibilities for further work.

Chapter 2

Disfluencies

This chapter deals in more detail with the phenomenon of disfluencies in spontaneous speech. In section 2.1 we show that disfluencies can be grouped by different types. Then a pattern is presented with which the structure of disfluencies (the "surface structure") can be described (section 2.2). In section 2.3 we finally review some work which has been done about disfluencies up to now. We distinguish between theoretic studies about disfluencies (section 2.3.1) and systems for automatic detection and correction of disfluencies (section 2.3.2).

2.1 Types of Disfluencies

In this section we describe types of disfluencies which can be used to classify all disfluencies we consider for our system. In the corpora we use, disfluencies are annotated according to this classification. However the annotations for the VERBMOBIL corpus slightly differ from the annotations for the Mandarin CallHome corpus. Details are given below after the explanation of the different disfluency types.
In the VERBMOBIL corpus disfluencies are partly annotated manually by human transcribers following the transcription conventions in [Burger, 1997] and partly annotated automatically using the DIASUMM-System proposed in [Zechner, 2001]. Thus we follow the definitions of these two works for the definition of our disfluency types.
From the manual transcriptions we drew mainly information about complex disfluencies repetitions or corrections of phrases or restarts of whole sentences. Simple filler words are not annotated in the transcriptions of the VERBMOBIL corpus. Therefore we used the DIASUMM-System which provides information about filler words.
The fact that part of the disfluency annotation of the corpus is created automatically by a system which of course does not perform this task without errors is a clear deficiency. However the performance of the DIASUMM-System for identifying those disfluencies for which we use automatic annotation is reported to be very good in [Zechner, 2001] and for the lack of time we were not able to provide manual disfluency annotations. Filler words are annotated correctly in about 90% of all cases by the DIASUMM-System on a test set from the SWITCHBOARD corpus. Manual inspection of the automatically annotated transcripts from the VERBMOBIL corpus indicates, that reasonable results are produced here as well.
The different types of disfluencies we consider are given by the sources of information (i.e. manual transcriptions and the DIASUMM-System) which are available. This does not however restrict our definition of a disfluency, i.e. there are no phenomena which we would like to consider as disfluency but which are not annotated in our corpus.

In table 2.1 all types of disfluencies considered in the VERBMOBIL corpus are presented, examples are given and the information whether the annotation is obtained manually (TRL) or automatically (DIASUMM) is provided.
Note that the DIASUMM-System marks words like "yeah" or "okay" as filled pauses when annotating the VERBMOBIL corpus. Thus one can not anymore distinguish filled pauses and discourse markers by considering whether they are lexicalized or not. Furthermore, in a sentence like "Yeah, we would arrive eight o'clock." the word "Yeah" which is classified as filled pause helps the speaker to start his turn and thus has the function of a discourse marker. Even single utterances like "Yeah.", "Okay." or "Oh" have a discourse function in so far, as they indicate the speaker that one is listening and in the first two cases that one agrees with him/her.
We distinguish these two disfluency types only because this distinction is made in the annotation of the corpus (performed by the DIASUMM-System). However, this distinction plays no role when features for disfluency detection and correction are extracted. In all analysis of disfluencies and results we make, we will explicitly consider the case that discourse markers and filled pauses can be merged into one category.
For the distinction of the categories "Repetition or Correction" and "False Start" it is important to consider whether the phrase which has been abandoned is repeated with only slight or even no changes in the syntactical structure (substitutions, insertions or deletions of words). That identifies it as a correction or even an exact repetition. If a completely different syntactical structure with different semantics is chosen for the repair, the observed disfluency is a false start.
As already mentioned the disfluency annotation for the Mandarin Chinese CallHome corpus is slightly different from the annotation for the VERBMOBIL corpus. False starts and repetitions are marked using exactly the same rules as for the annotation of the VERBMOBIL corpus. The disfluency type filled pause comprises all disfluencies which would marked either as filled pause or as discourse marker according to the annotation convention for the VERBMOBIL corpus. As we explained above, it is more natural to consider these two disfluency types as one single type. This disfluency type will be abbreviated with "FP" in the following to distinguish it from the filled pause type used in the VERBMOBIL annotation. Interjections and editing terms are not annotated at all in the Mandarin Chinese CallHome corpus. The data was annotated manually by native speakers of Chinese [Huang, 2003].
The disfluency classification just presented is important to define what we consider as disfluent text and it will be used when reporting results, in order to illustrate the performance of our system for single types. However, the disfluency correction system we propose does not distinguish at all different disfluency types for correcting them. It might be helpful to do so as different disfluency types differ largely in their structure and their properties, however much more annotated text would be required in order to have enough data. The current algorithm for disfluency correction and the statistical models the algorithm uses do not rely on type specific properties of disfluencies. The advantage of this approach is, that slight changes of the definition of disfluency types do not affect the correction algorithms or the formulation of the statistical models. Therefore it was easily possible to adapt our system to the Mandarin Chinese corpus with its slightly different disfluency type classification.

2.2 Surface Structure of Disfluencies

In the following a pattern is presented which can be used to describe the structure of disfluencies. In [Shriberg, 1994] this pattern is called "surface structure" of disfluencies as only characteristics of disfluencies are considered which are observable from the text. The underlying explanatory factors concerning the processes going on in the speaker's mind during disfluency production are not taken into account for description of the surface structure of a disfluency. As well studies about disfluencies and works about automatic correction of these phenomena make use of the existence of such a surface structure. However the terminology which is used differs. In our description we follow [Shriberg, 1994].
As shown in figure 2.1 each disfluency can be divided into three sections: The reparandum, then - after the interruption point - the interregnum and finally the repair. One or more of these components may be empty. The reparandum contains those words, which are originally not intended to be in the utterance. Thus it consists of one or more words which will be repeated or corrected ultimately (in case of a repetition/correction) or abandoned completely (in case of a false start). The interruption point marks the offset of the reparandum. It is not connected with any kind of pause or audible phenomenon. The interregnum can consist of an editing term, a non lexicalized pause like "uh" or "uhm" or simply of an empty pause, i.e. a short moment of silence. In many cases however, the interregnum of a disfluency is empty and the repair follows directly after the reparandum. In the repair the words from the reparandum are finally corrected or repeated (repetition/correction) or a complete new sentence is started (false start). Note that in the latter case the extend of the repair can not be determined.

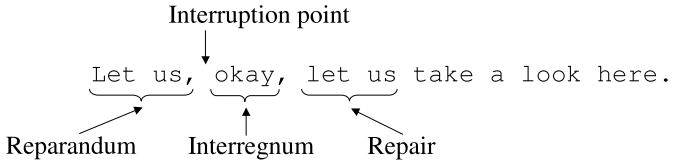

Figure 2.1: Surface structure of disfluencies

Obviously, the terms just introduced can be used to explain repetitions/corrections, false starts and editing terms. When assuming that the reparandum and the repair can be empty and only the interregnum is present, the presented surface structure fits as well for discourse markers/filled pauses and interjections, which can be seen as disfluencies only consisting of an interregnum. The examples in figure 2.2 show two examples with either empty interregnum or empty reparandum and repair.
In section 1.4 we explained that disfluency correction means the deletion of those words originally not intended by the speaker. Considering the surface structure above, it is easy to see that these are the words in the reparandum and interregnum. The deletion of these words produces that fluent sentence the speaker wanted to utter originally. Note that the term disfluency comprises reparandum, interregnum *and* repair. Disfluent words or the deletion region of a disfluency are defined to be only the words of the the reparandum and the interregnum. Note however that "deletion region of a disfluency" and "disfluency" are often used as synonyms. Thus the length of the deletion region of a disfluency is meant when speaking of the length of a disfluency.

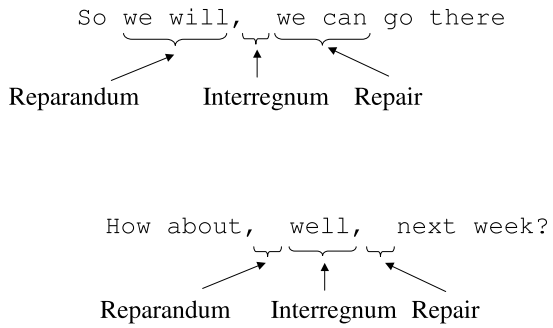

Figure 2.2: Disfluencies with either empty interregnum or empty repair and reparandum respectively

2.3 Related Work

In this section we review some work which has been done in the past about disfluencies. First we summarize some interesting findings about several properties of disfluencies presented in studies dealing with spontaneous speech. Then some approaches for automatic disfluency correction will be discussed.

2.3.1 Studies on Disfluencies

Spontaneous speech has been a research topic for many years. Thus there are a number of studies which investigate the phenomenon of disfluencies occurring frequently in spontaneous speech. Several studies investigate the processes going on the the speaker's mind during disfluency production. We are however much more interested in observable cues which can be useful for disfluency identification in spontaneous speech. Therefore we focus in this section on the studies in [Shriberg, 1994] and [Nakatani and Hirschberg, 1994] where characteristics of disfluencies are described which can be extracted out of the text or the speech signal.
The goal of Shriberg's work is to find regularities in disfluency production using a data-driven approach. Therefore manual transcriptions and digitized waveforms of three large speech corpora are examined. The corpora used are

- the ATIS corpus: human-computer dialogs in the air travel planning domain,
- the American Express/SRI (AMEX) corpus: human-human air travel planning dialogs,
- the SWITCHBOARD corpus: human-human telephone conversations on various topics

These three corpora are analyzed with respect to different features, where features are defined to be observable characteristics in the data. Five different feature dimensions are examined:

- Domain features which include all important aspects of the speech setting like discourse subject, communication mode etc.
- Speaker features associated with the individual producing disfluencies, such as age or gender of the speaker.

- Sentence features which refer to the sentence in which a disfluency occurs without referring to the actual words in the sentence. Examples are the position of a disfluency in a sentence or the number of disfluencies per sentence.
- Acoustic features which are concrete measurements of properties of the speech signal, e.g. the duration of words or pauses.
- Pattern features taking properties of single disfluencies into account such as their length or their type.

Shriberg classifies disfluencies according to their pattern and sentence features and develops a very detailed classification system. Disfluency analysis are then conducted for particular types and for all disfluencies as a whole. We summarize only the results of the latter analysis, since they are more important for our work. Disfluencies are analyzed for the three corpora according to the different feature dimensions. We present some of Shriberg's findings about sentence, pattern, acoustic and domain features in greater detail since we found them helpful for the disfluency correction task.

Sentence features

The rate of fluent sentences as a function of the sentence length can be predicted by an exponential model for the SWITCHBOARD and AMEX corpora, i.e the number of fluent sentences decreases exponentially over the sentence length. For the ATIS corpus a linear decrease is reported. Note that the sentence length is measured in efficient words, where Shriberg defines the efficient words of a sentence to be only those which are not considered as disfluent. However she reports a correlation between efficient and total sentence length.
Furthermore the number of disfluencies per sentence is shown to be growing roughly linearly with the sentence length (again measured in efficient words). This trend is stronger for the SWITCHBOARD and the AMEX corpus and weaker for the ATIS corpus. However, for the disfluency per word rate (which is obtained by dividing the number of disfluencies per sentence by the number of efficient words per sentence) some irregularities can be observed for SWITCHBOARD and AMEX. For short sentence lengths this rate is lower than for longer sentences where it is on a constant level. Shriberg suggests that this anomaly may be due to many short sentences consisting only of filler words without semantic content which are not considered as disfluencies in this analysis.
Finally the position of disfluencies within a sentence is analyzed for the three corpora. The result is that disfluencies are much more likely to occur at sentence-initial positions than at sentence medial positions. Note that sentence-initial filler words like "well", "alright" etc. are not considered as disfluencies.

Pattern features

The length of the deletion region of a disfluency is a pattern feature which is analyzed. Excluding filler words (which have a deletion region of length one), the number of disfluencies over the length of their deletion regions is described with an exponential model, i.e. the number of disfluencies with a deletion region of length k decreases exponentially with k. This trend is reported for all three corpora.
Word fragments are words which are aborted prematurely. An analysis is conducted, where the presence of fragments within disfluencies is examined. As a result it is reported that 23%

of the disfluencies in the SWITCHBOARD corpus, 27% in the AMEX corpus and even 58% of the disfluencies in the ATIS corpus contain a fragment. Shriberg remarks that most of the fragments occurring within disfluencies mark the offset of the reparandum.
Finally disfluencies with words in the interregnum are examined. The most important finding here is that only between 10% and 15% of all the disfluencies in the three corpora have an interregnum which contains one or more words. The rate for the ATIS corpus is reported to be significantly lower than the rates for the SWITCHBOARD and AMEX corpus.

Acoustic Features

Acoustic features are analyzed for filled pauses which Shriberg defines to be only the sounds "uh" and "uhm" and for one-word repetitions without words in the interregnum. For filled pauses Shriberg finds that the vowel duration is significantly longer than the duration of the same vowel in other words. The most interesting finding for one-word repetitions is that the first word (i.e. the reparandum) is lengthened compared to the same word occurring elsewhere. The second word (i.e. the repair) is slightly shortened. Furthermore a special characteristic of the fundamental frequency for filled pauses and the first word of a one-word repetition is reported.

Domain Features

From her analysis Shriberg concludes that there are a number of similarities between the AMEX and the SWITCHBOARD corpus concerning the different feature domains. The ATIS corpus seems to differ from both of these corpora concerning disfluency characteristics. Note that the SWITCHBOARD and the AMEX corpus consist of human-to-human telephone dialogs while the ATIS corpus consists of computer-to-human dialogs. Nevertheless Shriberg notes that there are also some similarities between all three corpora. Some effects concerning pattern and sentence features for example can be described with the same (exponential) models, sometimes even using the same parameters.

Nakatani and Hirschberg examine acoustic and prosodic properties of disfluencies which may be used for modeling them in correction systems or speech recognizers [Nakatani and Hirschberg, 1994]. They focus on speech "repairs" which correspond to our disfluency types repetition/correction and false start. Analysis are conducted on a subset of the ATIS corpus. While no cues are found identifying the reparandum onset, 73% of all repairs are reported to contain a fragment at the reparandum offset. Note that the corresponding figure reported in [Shriberg, 1994] is only 58% for the ATIS corpus. Some properties of fragments are presented which help to detect them during speech recognition. 91% of all fragments contain one syllable or less and vowel initial fragments are not very likely while fricative initial fragments are very likely to be observed. Furthermore the authors report that whenever the actual word associated with a fragment could be identified by the human transcribers, it was much more likely to be a content word than a function word.
Interruption glottalization and coarticulation are two other phenomena which occur at reparandum offsets. For 30.2% of all disfluencies interruption glottalization is found at the reparandum offset, usually associated with a fragment. Coarticulatory effects of acoustically missing subsequent phonemes are reported to occur at the reparandum offset as well. The authors

suggest to make use of these phenomena to distinguish between reparandum offsets and fluent phrase offsets.
Similar to the findings in [Shriberg, 1994] disfluencies with words in the interregnum occur very rarely (9.4% of all repairs) in the data examined by Nakatani and Hirschberg. Therefore they don't consider these cue phrases to be helpful for disfluency detection. For the disfluencies containing no words but only silence in the interregnum they report however that pausal duration is significantly shorter than for silences occurring in a fluent context. Nevertheless their experiments show that pausal duration alone can not be used to reliably identify disfluencies as too many false positives occur. Finally a slight but reliable difference in the fundamental frequency between the reparandum offset and the onset of the actual repair is found.
In the analysis of the repair part of a disfluency the authors focus on prosodic properties of the repair offset. They suggest that the identification of the repair offset may be helpful for disfluency correction. Analysis of all disfluencies shows that 83% of the repair parts are marked by intonational phrase boundaries.

2.3.2 Automatic Processing

In this section we will review some work which has been done in the past about automatic processing of disfluencies. Where ever it is possible we will report results in terms of recall and precision as defined for example in [Heeman, 1997]:
The recall rate is the number of disfluencies processed correctly ("hits") over the total number of disfluencies which actually occurred in a test text:

$$\text{recall} = \frac{\text{hits}}{\text{hits} + \text{misses}} = \frac{\text{hits}}{\text{total disfluencies}} \tag{2.1}$$

The precision rate is the number of disfluencies processed correctly over the number of total items which have been considered as disfluency by the algorithm which include hits and false positives:

$$\text{precision} = \frac{\text{hits}}{\text{hits} + \text{false positives}} \tag{2.2}$$

Recall and precision are commonly used to measure the performance of systems for automatic disfluency detection and correction. Comparison of these figures for different systems proposed by different authors is difficult however. This is because different corpora are used for testing the systems and different types of disfluencies are considered. Furthermore the annotation and the choice of the test data varies from case to case, which makes the task of disfluency processing sometimes more and sometimes less difficult. In some cases for example hand labeled interruption points are given which already identify the site of a disfluency in a sentence and sometimes only sentences are considered which contain at least one disfluency. All this must be noted, when we report in the following the results achieved by different systems.
In the past 20 years a lot of systems for automatic disfluency detection and correction have been proposed. In our presentation of some of these systems we distinguish

- rule based approaches, which rely on syntactic information and simple matching rules to find word correspondences between reparandum and repair,
- statistical approaches, which make use of statistical models for disfluency processing,

- hybrid approaches, combining the two other approaches and
- systems which model disfluencies for speech recognition.

Rule-based approaches

The first attempt toward automatic processing of disfluencies is described in [Hindle, 1983]. In this work correction strategies for disfluencies are investigated which are implemented as extension to a deterministic parser. In a corpus of narrative speech with about 1500 sentences interruption points of disfluencies (i.e. repetitions/corrections and false starts) are hand labeled. Hindle assumes however that they can be reliably identified by an acoustic edit signal. Applying editing and restart rules deterministically from left to right, Hindle's system is able to correct about 97% of the disfluencies occurring in the text. Note that the identification of the interruption point of a disfluency using acoustic cues is still an open problem. [Nakatani and Hirschberg, 1994] found some cues such as pausal duration which might be used for this task, however they are not reliable enough to account for all disfluencies (section 2.3.1).
Parsing techniques combined with pattern matching approaches are investigated as well in [Carbonell and Hayes, 1983] in order to process extra grammatical input for natural language processing systems. However no spoken language is assumed as input for such a system but a sequence of words typed by a user. In order to identify and to correct repairs the system looks for explicit corrective phrases (i.e. editing terms like "I mean") and takes the constituent found after this phrase as the correction for the constituent before the corrective phrase. Another correction rule is to take the first phrase, when two syntactically and semantically identical phrases appear consecutively in a sentence.
A system for repair detection and correction for the ATIS corpus is proposed in [Bear et al., 1992] where pattern matching rules are combined with syntactic and semantic knowledge. The disfluency types considered in this work correspond to repetitions/corrections according to our definition, which may of course include editing terms in the interregnum. Discourse markers/filled pauses, interjections and false starts are not considered.
The system uses a two-stage model in order to incorporate information from multiple knowledge sources. In the first stage pattern matching rules are applied to identify potential disfluencies in the test corpus consisting of 10517 sentences. The pattern matching component uses repeated words and syntactic anomalies like "a the" or "to from" as cues for repairs. Results for this pattern matching approach are reported in terms of correction rates or detection rates for complete sentences which contain at least one disfluency. How many disfluencies actually occur in a sentence is not considered. In terms of successfully detected disfluent sentences the system achieves a recall of of 76% and a precision of 62% using only the pattern matching component. For disfluency correction using pattern matching (again in terms of completely corrected sentences) a recall of 44% and a precision of 35% is reported. Note that the pattern matching component does not require any annotated texts for training, but rules have to be defined manually which reflect the typical patterns of disfluencies.
In the second stage of the system a natural language processing component is used to distinguish actual repairs from false positives in the set of sentences identified as potentially disfluent in the first stage. Each of these sentences is parsed by the system. If parsing succeeds a fluent sentence is hypothesized. Otherwise some repair patterns are applied and the sentence is parsed again. If parsing is successful, the new sentence is the hypothesized correction, otherwise it is marked with "no opinion". The parsing component is reported to be

quite accurate in the detection of repairs in the given set of sentences and a bit less accurate for the detection of false positives. Out of 68 correctly identified repairs in the second stage, 62 can be appropriately corrected. Explicit results in terms of recall and precision measuring the performance of the combination of the two stages of the system are not reported.
Certain acoustical cues for disfluency detection are investigated as well, but no experimental results using acoustic knowledge are reported.

Statistical approaches

In [Nakatani and Hirschberg, 1994] prosodic and acoustic cues for disfluency detection and correction are examined. The results of the analysis of the ATIS corpus for such cues are already presented in section 2.3.1. Here we describe briefly a system for disfluency detection in the ATIS corpus proposed in [Nakatani and Hirschberg, 1994]. This system combines the prosodic and acoustic cues which the authors found to be useful for disfluency detection with cues which are observable from the text. The system uses decision trees generated by the CART-Algorithm (described for example in [Huang et al., 2001]) in order to detect the interruption points of "repairs". The "repair" types which are considered correspond to repetitions/corrections and false starts according to our definition. The algorithm generates decision trees in a way that the prediction error rate on cross-validation data is minimized.
The decision trees are trained on 202 utterances and tested on 148 utterances with both training and test set containing at least one repair per utterance. They apply questions about the following acoustic/prosodic and textual features, which are manually annotated.
Acoustic prosodic features:

- duration of pauses (= silences)
- occurrence of word fragments
- occurrence of filled pauses
- energy peaks in the speech signal
- amplitude of the speech signal
- fundamental frequency
- accentuation

Textual features:

- position in a utterance
- number of words of an utterance
- word-matchings in a window of three words after the current word
- (Part-of-Speech) POS-tags in a window of four words around a potential interruption point
- repetitions of function words sharing the same POS-tag

For the best decision tree generated by the algorithm using these features a recall of 86.1% and a precision of 91.2% on the test set is obtained. This tree uses particularly information about silences, fragments, filled pauses, word matchings, function word repetitions, POS-tags and fundamental frequency. Note that the results apply for disfluency *detection* (not for correction) and that each sentence of the test set at least contains one disfluency.
The goal in [Heeman, 1997] is to model speakers' utterances in spontaneous speech. In order to do so the author claims that it is necessary to segment the utterance into intonational units, to detect and correct speech repairs and to identify discourse markers. These tasks are combined with POS based statistical language model which assigns a POS category[1] to each word and predicts the next word, given the history of the previous words.
This system is trained and tested on the corpus "Trains" which consists of 98 dialogues (6163 speaker turns) dealing with manufacturing and shipment of goods in a railroad freight system. The disfluencies Heeman considers in his work are "repairs" and discourse markers, which are both subdivided into several subtypes. According to our definition Heeman's "repairs" are repetitions/corrections and false starts, discourse markers correspond to our discourse markers/filled pauses and interjections.
For the identification of discourse markers a new POS category for such words is defined. POS categories are simply assigned using the knowledge about previous words as POS tags. More complex disfluencies (repetitions/corrections and false starts) are identified by tagging the interruption points, i.e. tags *between* words are used. These tags are assigned by decision trees applying questions about previous words, POS tags, intonational boundaries and silences, which is the only acoustic feature used to generate the trees. In order to correct complex disfluencies it is necessary to identify, i.e. to tag, the reparandum onset. The tag for the reparandum onset is placed in a way that the hypothesized reparandum onset or the words preceeding the onset predict optimally the repair. In case of repetitions/corrections word correspondences in reparandum and repair are used as predictors. Words preceeding the hypothesized reparandum onset are used for the prediction of the repair in case of false starts.
Heeman reports that his system improves POS-tagging by 8.1% and language model perplexity by 7.1%. For the detection of complex disfluencies (repetitions/corrections and false starts) the system achieves a recall of 76.8% and a precision of 86.7%. The results for correction are a recall of 65.9% and a precision of 74.3%. For discourse marker identification the system performs even better with a recall of 97.3% and a precision of 96.3%. For this disfluency type detection and correction results are the same, as only the words which have been identified as discourse marker have to be deleted for correction.
A disfluency correction component for spoken language systems is proposed in [Spilker et al., 2000]. In this work the authors partly address the problem that speech recognizer output must be assumed as input for a disfluency correction component rather than manually transcribed speech. Thus the system is given a lattice as input. In this lattice paths containing repairs are hypothesized and paths containing possible corrections are inserted. The system works on the German Verbmobil corpus, the German counterpart to the corpus we are using in this work. The only type of disfluencies which are considered are "modification repairs" which correspond to our repetitions/corrections.
Disfluency processing is divided in two steps. First acoustic and prosodic cues and word fragments are used to identify potential interruption points. Then a statistical model using principals form statistical machine translation is used to segment disfluencies into reparandum,

[1] POS categories are usually simple grammatical terms like adjective, preposition, modal verb etc.

interregnum and repair. Correction is seen as the deletion of reparandum and interregnum. The model determines the segmentation of a disfluency by translating a possible reparandum into the corresponding repair. In order to reduce search space reparanda and repairs may not exceed the length of four words each. Besides the actual words, the translation model takes POS-tags and semantic classes into account.
Several problems of lattice processing are addressed in this work. POS tags and their probabilities are represented in a separate tag lattice. Given and interruption point, the algorithm takes all possible paths of the lattice into account going through the word before the potential interruption point. Finally possible corrections are inserted as new paths into the existing lattice.
Although a framework for disfluency processing using speech recognizer output is proposed in this work, results are reported on tests with manual transcriptions only. For disfluency detection a recall of 71% and a precision of 85% is reported. The system corrects disfluencies with a recall of 64% and a precision of 84%.

Hybrid approaches

In [Zechner, 2001] the DiaSumm-system is presented which is a system for creating summaries of spontaneous speech texts. An essential component of this system, which we partly use to obtain the disfluency annotations for the Verbmobil corpus (section 2.1), performs the detection and correction of disfluencies. This component is trained using the Penn Treebank, a treebank of the Switchboard corpus provided by the Linguistic Data Consortium. Disfluency annotations are added to the treebank by human annotators. For testing Zechner uses manually transcribed excerpts of different corpora:

- the CallHome and CallFriend corpora which consist of telephone conversations between family members or friends
- recordings of group meetings at the Interactive Systems Lab at Carnegie Mellon University
- recordings of the TV shows NewsHour and CrossFire which Zechner characterizes to consist of more formal speech than the other corpora

The disfluency types considered in this work correspond almost exactly to the types we defined in section 2.1. (Note that we simply adopted Zechner's definitions for those disfluency types which are annotated by his system in our corpus.) Only "empty coordinating conjunctions" serving no important connective role in the discourse are not considered by our system but by the DiaSumm-System. The reason for this is that the words marked as empty coordinating conjunctions in our corpus by the DiaSumm-System have an important discourse function in our opinion.
Different types of disfluencies are treated with different methods by Zechner's system. A rule based POS tagger [Brill, 1994] tags filled pauses, discourse markers, editing terms and empty coordinating conjunctions. The tagger is trained on 1.4 million of annotated words from the Penn Treebank in order to develop context based rules for assigning POS tags to words. Simple word repetitions are detected by a repetition detection script, which marks repetitions of word or POS sequences with a length of up to four words. More complex corrections are not considered as they appear very rarely according to the author. Finally a decision tree is trained on 8000 sentences in order to identify false starts. As features the decision tree

encodes words, POS tags and chunks of a context free POS-grammar based chunk parser. The chunk parser takes the POS sequences for the words in a sentence as input and parses them into chunks which are simple grammatical concepts like noun phrases or prepositional phrases.
The results in disfluency correction and detection for the DiaSumm-system differ across the different test corpora. Generally the system performs better on the corpora involving more spontaneous speech than on the formal speech corpora. For the disfluency types which the DiaSumm-system annotates in our corpus (discourse markers, filled pauses and editing terms), recall and precision rates over 90% on a Switchboard test set are reported. As no over all results for disfluency correction are reported in Zechner's work, we evaluated his system ourselves on his test set from the CallHome and CallFriend corpora. We obtained a recall of 51.9% and a precision of 59.4%.

Disfluencies and speech recognition

In [Stolcke and Shriberg, 1996] a standard n-gram language model is extended to model explicitely disfluencies in spontaneous speech. The authors assume that predictions of the language model are more accurate, when they are based on the fluent context of the word to be predicted. Therefore their language model marks disfluencies which are then omitted for the predictions of following words. In order to identify disfluencies, they are treated like words. They are predicted given the context in which they occur and assigned a probability. The language model is trained on 1.4 million words from the Switchboard corpus, annotated with disfluencies. Testing is performed on 17500 words from the same corpus. The disfluency types which are considered are filled pauses like "uh", "uhm" etc., exact word repetitions and false starts with not more than two words in the reparandum.
Experiments are performed where the proposed language model is compared to a standard language model. Only very slight improvements in perplexity can be observed. Furthermore the word error rate does not improve much, when the proposed language model is used in a speech recognizer instead of a standard model. However in experiments where only repetitions and false starts are modeled an improvement of the perplexity for the local context of these phenomena is achieved. For filled pauses the contrary is the case. Thus the authors conclude that filled pauses themselves are the best predictor for following words. Interestingly however, the modeling of filled pauses improves the perplexity of the model, when the utterances are segmented according to linguistic criteria instead of acoustic criteria. This is explained with the observation that filled pauses tend to occur at linguistic boundaries. Therefore they are important to indicate linguistic boundaries in acoustically segmented utterances.
The goal of [Stolcke et al., 1998] is to identify disfluencies and sentence boundaries in automatically transcribed speech. A prosodic model and a modified n-gram language model are incorporated in a speech recognition system to perform this task.
1.2 million words from the Switchboard corpus are used for training, 231K words for development and testing on manually transcribed speech and finally 18K words for testing on automatically transcribed speech. For experiments on automatic transcriptions the best 100 hypotheses of the speech recognizer are used. Filled pauses, repetitions/corrections and false starts are the considered disfluency types.
The prosodic model uses decision trees generated with the CART-algorithm encoding features about the duration of pauses, final vowels and final rhymes, fundamental frequency patterns and sound-to-noise ratio. The language model predicts "event"/word pairs instead of single words, where events are defined to be disfluencies or sentence boundaries.

Experiments with both models separately show that they detect up to 90.0% of the events correctly in manually transcribed speech and up to 76.1% in automatically transcribed speech. Furthermore up to 3.7% improvement of word accuracy for the recognition engine is reported when the disfluencies are modeled. For the combination of both models even better results are reported: up to 93.0% of all events are detected in the manually transcriptions, 78.1% in the automatic transcriptions and the word accuracy improvement is 5.7%

Name	Abbrev.	Description	Example	Source of annotation
Repetition or Correction	REP	Exact repetition or correction of words previously uttered. A correction may involve substitutions, deletions or insertions of words. However, the correction continues with the same idea or train of thought started previously.	If I **can't find** don't know the answer myself, I will find it. **If** if nobody wants to influence President Clinton.	TRL
False Start	FS	An utterance is aborted and restarted with a new idea or train of thought.	**We'll never find a day** what about next month?	TRL
Editing Term	ET	Phrases of words which occur after that part of a disfluency which is repeated or corrected afterwards (REP) or even abandoned completely (FS). They refer explicitely to the words which just previously have been said [Shriberg, 1994], indicating that they are not intended to belong to the utterance.	We need two tickets, **I'm sorry**, three tickets for the flight to Boston.	DiaSumm
Discourse Marker	DM	Words that are related to the structure of the discourse in so far that they help beginning or keeping a turn or serve as acknowledgment [Heeman, 1997]. They do not contribute to the semantic content of the discourse.	**Well**, this is a good idea. This is, **you know**, a pretty good solution to our problem.	DiaSumm
Filled Pause	UH	Non lexicalized sounds with no semantic content	**uh**, **uhm** etc.	DiaSumm
Interjection	IN	A restricted group of non lexicalized sounds indicating affirmation or negation. In our corpus they were mainly used for back-channeling and have thus no semantic content.	**uh-huh, mhm, mm, uh-uh**	TRL

Table 2.1: Types of disfluencies; The bold parts in the example sentences refer to the parts to be deleted in order to obtain a fluent sentence

Chapter 3

Statistical Models

This chapter deals with the statistical models we are using in our system. As we adopt some ideas from statistical machine translation (SMT), first a very brief introduction in the basic models of SMT (section 3.1) is given. In section 3.2 the general ideas of statistical language modeling are explained and some techniques are introduced which are applied to the language model we are using for our system. Section 3.3 deals with the "translation" model we use in our system to "translate" disfluent speech into fluent speech. After the derivation and the mathematical formulation of this model, we explain in that section how the different properties of disfluencies are encoded in the translation model.

3.1 Statistical Machine Translation

In this section some basic ideas of SMT are summarized briefly. First we review the "noisy channel"- approach on which the mathematical models of SMT are based. Then three different types of translation models are explained which are commonly used in SMT. In our description we mainly follow [Vogel et al., 2000]. Details about the models we present can be found in [Stephan Vogel and Hermann Ney and Christoph Tillmann, 1996] and [Brown et al., 1993].
The goal of SMT is the translation of a string of words $S = s_1^J = s_1 \dots s_J$ of length J in a source language into a string of words $T = t_1^I = t_1 \dots t_I$ of length I in a target language. Thus the translation problem is to find the unknown words $t_1, \dots, t_I$ and the unknown length I of the target string given the source string S.
In [Wang and Waibel, 1997] this translation problem is explained using the model of the noisy channel which receives the string T in the target language as input, adds noise to it and produces the string S in the source language as output. The translation problem then is to recover the channel input T given the channel output S. This is done by determining that string $\hat{T}$ for which the probability of being the channel input given the channel output S is maximal.

$$\hat{T} = \arg\max_T P(T|S) \tag{3.1}$$

The argmax-operation denotes the search over all possible strings T in order to determine the "best" string $\hat{T}$.

By applying Bayes-Rule to the probability $P(T|S)$ one obtains:

$$P(T|S) = \frac{P(S|T) \cdot P(T)}{P(S)} \tag{3.2}$$

As the denominator does not depend on T, on can determine $\hat{T}$ as follows:

$$\hat{T} = \arg\max_{T} P(S|T) \cdot P(T) \tag{3.3}$$

In equation 3.3 the probability $P(T)$ denotes the language model based on the target language and the probability $P(S|T)$ denotes the translation model which specifies how words from the source language are translated into the target language. In the following we are explaining the translation model in larger detail.
A key issue of the translation model is to define the correspondence between the words s_j $(1 \leq j \leq J)$ and t_i $(1 \leq i \leq I)$ of a sentence pair (s_1^J, t_1^I). Very often the assumption is made that there is a pairwise correspondence between the source and the target words so that only all pairs (s_j, t_i) must be considered to find the best translation. An even more restrictive assumption is that each source word is assigned to *exactly* one target word.
Based on these assumptions one can define the alignment mapping:

$$j \rightarrow i = a_j \tag{3.4}$$

which assigns a source word s_j in position j to a target word t_i in position $i = a_j$.
For a number of Indo-European language pairs (Spanish-English, French-English, Italian-German etc.) one can observe strong localization effects for the alignments. That means that if position j in the source sentence is mapped to position i in the target sentence, it is very likely that position $j + 1$ in the source sentence will be mapped to a position very close to i in the target sentence, in many cases even to position $i + 1$. That is why it might be desirable to make the alignment a_j dependent on the previous alignment a_{j-1} for these languages.
This relationship is covered by the Hidden-Markov-Model (HMM) alignment proposed in [Stephan Vogel and Hermann Ney and Christoph Tillmann, 1996]. In this model for the source string s_1^J a sequence of 'hidden' alignments $a_1^J = a_1 \ldots a_J$ is assumed, so that the probability $P(S|T)$ can be written as follows:

$$P(S|T) = P(s_1^J|t_1^I) = P(J|I) \cdot \sum_{a_1^J} P(s_1^J, a_1^J|t_1^I) \tag{3.5}$$

The probability $P(J|I)$ is the probability for the sentence length which has to be included for normalization reasons. In the models presented here and in most models which are used in current machine translation systems this probability is however set to be uniform. The probability $P(s_1^J, a_1^J|t_1^I)$ can be decomposed into the product over the probabilities for each position in the source sentence j to have the word s_j and the alignment a_j given the words s_1^{j-1}, the alignments a_1^{j-1} which have been computed already and the target string t_1^I:

$$P(s_1^J|t_1^I) = P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} P(s_j, a_j|s_1^{j-1}, a_1^{j-1}, t_1^I) \tag{3.6}$$

By applying the Markov assumption that there is only a first order dependency for each alignment a_j (i.e. a_j depends only on a_{j-1}) one obtains:

$$\begin{aligned} P(s_1^J|t_1^I) = & \quad P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} P(s_j, a_j|a_{j-1}, t_1^I) \\ = & \quad P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} \Big(P(a_j|a_{j-1}, I, J) \cdot P(s_j|t_{a_j}) \Big) \end{aligned} \tag{3.7}$$

In the second part of equation 3.7 the probability $P(s_j, a_j|a_{j-1}, t_1^I)$ is decomposed into an alignment probability $P(a_j|a_{j-1}, I, J)$ and a lexicon probability $P(s_j|t_{a_j})$. The alignment probability models the assignment of position j in the source sentence to position a_j in the target sentence. Given this assignment the lexicon probability determines the actual word t_{a_j} which will be put at position $a_j = i$ in the target sentence. For the formulation of the lexicon probability the assumption is used, that there are word-to-word correspondences between target and source words. This is why s_j depends on t_{a_j} rather then on the whole target string t_1^I.
Because of mathematical requirements the lexicon model represents the probability of s_j *given* t_{a_j} although it is our goal to determine the unknown word t_{a_j}. This goal is achieved by considering the probability $P(s_j|t_{a_j})$ as function which has to be maximized depending only on t_{a_j}.
When the first order dependency in the HMM model is reduced to a zero order dependency where the alignment a_j only depends on the current position j in the source sentence, a model is obtained which is similar to the model 2 proposed in [Brown et al., 1993], commonly referred to as IBM2 model. For this model the following identity can be shown which is important for computational reasons as the problem of considering all possible sequences a_1^J is solved:

$$\begin{aligned} P(s_1^J|t_1^I) = & \quad P(J|I) \cdot \sum_{a_1^J} \prod_{j=1}^{J} \Big(P(a_j|j, t_1^I) \cdot P(s_j|t_{a_j}) \Big) \\ & = P(J|I) \cdot \prod_{j=1}^{J} \sum_{i=1}^{I} \Big(P(i|j, I, J) \cdot P(s_j|t_i) \Big) \end{aligned} \tag{3.8}$$

This model can be further simplified even more by setting the alignment probability $P(i|j, I, J)$ uniform:

$$P(i|j, I, J) = \frac{1}{I} \tag{3.9}$$

The model using this uniform alignment probability is commonly referred to as IBM1 model [Brown et al., 1993].
The models introduced above are commonly used in SMT. Note that these models allow that a word in the source sentence is aligned anywhere in the target sentence. For the system proposed in this work such a flexible alignment is not necessary. As we explained in section 1.4 the correction of disfluencies can be seen as a sequence of deletions in a given sentence requiring no word reorderings, insertions or substitutions. Thus, in our terms "alignment" means that a word is either aligned to itself or it is deleted which can be interpreted as an alignment to the empty word ϵ. This is the reason why the model we present in section 3.3 is much simpler and computationally more tractable than the models which are used in SMT.

3.2 Statistical Language Modeling

Language modeling is a crucial part of SMT systems. However it is very important as well for applications in speech recognition and finally for our disfluency correction system. Although

we start our explanation here from the SMT point of view, the theory we present is applicable everywhere where language models are necessary. Note that we concentrate here on statistical language modeling. Many other language modeling techniques exist as well and are commonly used in different applications.
In the following the basic ideas of statistical language modeling are described mainly following [Jelinek, 1985] and [Huang et al., 2001]. Then some techniques used in the language model of our disfluency correction system will be introduced.
When considering again the search problem to find this string $\hat{T}$ with

$$\hat{T} = \arg\max_{T} P(S|T) \cdot P(T),$$

the probability $P(T)$ denotes the language model on the target language. Remember that for our application the language model models fluent speech containing no disfluencies anymore. The idea of the language model is to predict the next word of the target string given the words which are already known, i.e. the words preceeding the current word. Given the string $T = t_1 \dots t_I$ this idea can be expressed formally as follows:

$$P(T) = P(t_1 \dots t_I) = \prod_{i=1}^{I} P(t_i|t_1 \dots t_{i-1}) \tag{3.10}$$

However the probability of having t_i given all the history $t_1 \dots t_{i-1}$ would be very hard to estimate using a given training text, as for a vocabulary size of V there are V^{i-1} different histories. (For $V = 100$ and a history of 5 words there are $100^5 = 2^{10}$ different histories, of which the major part will never be observed in a given training text, even if it is tremendously large.) That is why histories are usually put into a smaller number of equivalence classes in order to estimate probabilities. One way of doing this is to define two histories to be equivalent iff the last $n-1$ words preceding the current words coincide:

$$\begin{aligned} P(t_i|t_{i-1} \dots t_1) = &\quad P(t_i|\hat{t}_{i-1} \dots \hat{t}_1) \Leftrightarrow \\ t_{i-1} \dots t_{i-n+1} = &\quad \hat{t}_{i-1} \dots \hat{t}_{i-n+1} \end{aligned} \tag{3.11}$$

That means that only the $n-1$ last words are considered for the prediction of the current word. These kinds of language models are called n-gram language models and they are often used in statistical machine translation and speech recognition. An example is the trigram language model, where equation 3.10 is simplified as follows:

$$P(T) = \prod_{i=1}^{I} P(t_i|t_{i-2}t_{i-1}) \tag{3.12}$$

Generally, trigram probabilities can be estimated from a given training text by calculating the relative frequencies that the word t_i occurs after the history $t_{i-2}t_{i-1}$. That means that the count $C(t_{i-2}t_{i-1}t_i)$ for observing the string $t_{i-2}t_{i-1}t_i$ is divided by the count $C(t_{i-2}t_{i-1})$ for observing the history $t_{i-2}t_{i-1}$ no matter which word occurs at position i:

$$P(t_i|t_{i-2}t_{i-1}) \approx \frac{C(t_{i-2}t_{i-1}t_i)}{C(t_{i-2}t_{i-1})} \tag{3.13}$$

However when using this method, most probability estimates would be zero or very small as even in a very large training corpus most of the possible trigrams $t_{i-2}t_{i-1}t_i$ do not occur at

all or only once as explained above. That means that the well formed sentence "Cleda knows Kenny" will be assigned a zero probability, if the name "Kenny" does not occur with the history "Cleda knows" in the training data. In that case the count C(Cleda knows Kenny) would be zero and so the probability estimate for P(Kenny|Cleda knows) would be zero.
In order to overcome this problem, different smoothing techniques have been proposed which make use of a more general history when the more specific history is not observed in the training data. Very commonly shorter histories are used if longer histories can not be found. In our example the word "Kenny" would then be predicted by the bigram probability P(Kenny|knows), the unigram probability P(Kenny) or simply the uniform probability $\frac{1}{V}$ where V is the size of the vocabulary. Which probability is finally chosen depends on the information which is available from the training corpus.
Smoothing is a technique which combines higher order n-grams with lower order n-grams (i.e. n-grams with larger values of n with n-grams with smaller values of n) in order to produce more robust probability estimates for unseen events and to avoid zero probabilities. Smoothing can be done by interpolating lower and higher order n-gram models or by backing off to a lower order model, when a probability can not be estimated using a higher order model.
The following equation gives a very general definition of the backing-off method (V is the size of the vocabulary):

$$P_{smooth}(t_i|t_{i-n+1}\dots t_{i-1}) = \begin{cases} \alpha(t_i|t_{i-n+1}\dots t_{i-1}) & \text{if } C(t_{i-n+1}\dots t_i) > 0 \\ \gamma(t_i|t_{i-n+1}\dots t_{i-1}) \cdot P_{smooth}(t_i|t_{i-n+2}\dots t_{i-1}) & \text{if } C(t_{i-n+1}\dots t_i) = 0, n > 1 \\ \gamma(t_i) \cdot \frac{1}{V} & \text{if } C(t_i) = 0 \end{cases} \tag{3.14}$$

That means whenever a count for a given n-gram has not been observed in the training data, the model backs off to (n-1)-grams. In order to assure that the probability estimates for all possible words t_i given a specific history $t_{i-n+1}\dots t_{i-1}$ still sum up to 1, it is important that some of the probability mass of the higher order model is taken away to distribute it among the estimates using the lower order models. $\alpha(t_i|t_{i-n+1}\dots t_{i-1})$ symbolizes this modified probability for the higher order model. The factor $\gamma(t_i|t_{i-n+1}\dots t_{i-1})$ assures that only the probability mass which has been subtracted from the higher order model is distributed among the estimates using lower order models.
A very simple backing off method is absolute discounting which is used as one smoothing method in our system. For absolute discounting a fixed discount $D \leq 1$ is subtracted from each non-zero n-gram count. The probability mass gained in this way is then distributed among those predictions which have an n-gram count of zero. Hence equation 3.14 must be modified as follows as follows:

$$P_{abs}(t_i|t_{i-n+1}\dots t_{i-1}) = \begin{cases} \frac{\max\{C(t_{i-n+1}\dots t_i)-D,0\}}{C^*(t_{i-n+1}\dots t_{i-1},t_i)} & \text{if } C(t_{i-n+1}\dots t_i) > 0 \\ \frac{D}{C^*(t_{i-n+1}\dots t_{i-1},t_i)} \cdot C(t_{i-n+1}\dots t_{i-1}\bullet) \cdot P_{abs}(t_i|t_{i-n+1}\dots t_{i-1}) & \text{if } C(t_{i-1+1}\dots t_i) = 0, n > 1 \\ \frac{D}{C^*(t_{i-n+1}\dots t_{i-1},t_i)} \cdot C(t_{i-n+1}\dots t_{i-1}\bullet) \cdot \frac{1}{V} & \text{if } C(t_i) = 0 \end{cases} \tag{3.15}$$

$C(t_{i-n+1}\dots t_{i-1}\bullet)$ counts the number of unique words occurring after the history $t_{i-n+1}\dots t_{i-1}$ in the training data. (This count is *not* equal to $C(t_{i-n+1}\dots t_{i-1})$ which counts the occur-

rences of the history $t_{i-n+1} \dots t_i$ in the training data.) The factor

$$\gamma(t_i|t_{i-n+1} \dots t_{i-1}) = \frac{D}{C^*(t_{i-n+1} \dots t_{i-1}, t_i)} \cdot C(t_{i-n+1} \dots t_{i-1}\bullet)$$

has been adjusted in a way that the absolute discounting probabilities for all possible words t_i given a particular history sum up to one.
The count $C^*(t_{i-n+1}, \dots t_{i-1}, t_i)$ is different for the highest order n-grams used in a language model and for all other n-grams:

$$C^*(t_{i-n+1} \dots t_{i-1}, t_i) = \begin{cases} C(t_{i-n+1} \dots t_{i-1}) & \text{for the highest order n-grams} \\ \sum_{t_i: C(t_{i-n} \dots t_i)=0} C(t_{i-n+1} \dots t_i) =: \hat{C}(t_{i-n+1} \dots t_i) & \text{for all other n-grams} \end{cases} \tag{3.16}$$

The reason for this choice of C^* is again that the probability estimates for all words t_i given a particular history sum up to one. We give an example for the probability estimates for a bigram language model using absolute discounting. We for simplicity we assume that all words have been seen during training, i.e. $C(t_i) > 0$ for all t_i in the vocabulary. Thus no discounting factor is subtracted form the unigram probabilities. It can be easily verified for this example that

$$\sum_{\text{all words } t_i} P_{abs}(t_i|t_{i-1}) = 1.$$

$$P_{abs}(t_i|t_{i-1}) = \begin{cases} \frac{\max\{C(t_{i-1}t_i)-D,0\}}{C(t_{i-1})} & \text{if } C(t_i, t_{i-1}) > 0 \\ \frac{D}{C(t_{i-1})} \cdot C(t_{i-1}\bullet) \cdot \frac{C(t_i)}{\tilde{C}(t_i)} & \text{if } C(t_i, t_{i-1}) = 0 \end{cases} \tag{3.17}$$

In equation 3.17 the variable n from equation 3.15 equals to 2 for the higher order estimates and to one for the lower order estimates.
Linear interpolation is another way to overcome the problem of zero counts for many longer n-grams. An interpolated language model uses for *every* probability estimate higher and lower order n-gram models, no matter if n-gram counts for the higher order models are zero or not, while back-off models do not. In mathematical terms, an interpolated n-gram model can be expressed as follows:

$$\begin{aligned} &P_{interp}(t_i|t_{i-n+1} \dots t_{i-1}) \\ &= q_n \cdot P(t_i|t_{i-n+1} \dots t_{i-1}) + q_{n-1} \cdot P(t_i|t_{i-n+2} \dots t_{i-1}) + \dots + q_1 \cdot P(t_i) \end{aligned} \tag{3.18}$$

The non-negative weights q_i must sum up to 1:

$$\sum_{i=1}^{n} q_i = 1$$

Equation 3.18 can be written as a recursive equation as follows:

$$\begin{aligned} &P_{interp}(t_i|t_{i-n+1} \dots t_{i-1}) \\ &= p_n \cdot P(t_i|t_{i-n+1} \dots t_{i-1}) + (1 - p_n) \cdot P_{interp}(t_i|t_{i-n+2} \dots t_{i-1}) \end{aligned} \tag{3.19}$$

The disfluency correction system proposed in this work uses a language model which combines absolute discounting with interpolation. The implementation of the language model component was not done within the context of this work. We rather use the language modeling toolkit which is part of the current SMT system at Carnegie Mellon University. The combination of interpolation and absolute discounting proofed to perform well in machine translation applications. The following mathematical formulation represents the combination of both techniques:

$$\begin{aligned} &P_{combined}(t_i|t_{i-n+1}\dots t_{i-1}) = \\ &\quad \frac{\max\{C(t_{i-n+1}\dots t_i) - D_n, 0\}}{C(t_{i-n+1}\dots t_{i-1})} + \\ &\quad \frac{D_n}{C(t_{i-n+1}\dots t_{i-1})} \cdot C(t_{i-n+1}\dots t_{i-1}\bullet) \cdot P_{combined}(t_i|t_{i-n+2}\dots t_{i-1}) \end{aligned} \tag{3.20}$$

$P_{combined} = \frac{1}{V}$ for a word which has not been seen during training.
As for absolute discounting we give an example for a bigram language model using the smoothing technique presented above. Again we assume $C(t_i) > 0$ for all words in the vocabulary and don't discount the unigram probability. Again in can be easily verified that the probabilities for all words given a specific history sum up to one.

$$P_{combined}(t_i|t_{i-1}) = \frac{\max\{C(t_{i-1}t_i) - D_2, 0\}}{C(t_{i-1})} + \frac{D_2}{C(t_{i-1})} \cdot C(t_{i-1}\bullet) \cdot \frac{C(t_i)}{V} \tag{3.21}$$

The discounting factors D_n for this language model are determined iteratively: For each model of order n the discounting factor is adjusted that the quality for this n^{th}-order model is optimized. An n^{th}-order model is defined here as this model which considers only m-grams with $m \leq n$.
The optimization criteria is to minimize perplexity on a cross validation set for the particular model. The "optimal" discounting factor is found by iteratively trying a small set of different discounting factors between 0 and 1.
Perplexity is often used to measure the quality of a language model. It can be roughly interpreted as the average number of possible candidates for the next word given a certain history. Another way to explain perplexity is to see it as the average branching factor of the language model when a text is presented to it.
Formally perplexity for a given text T with length $|T|$ is defined as

$$2^{H(T)}$$

where

$$H(T) = -\frac{1}{|T|}\log_2 P(T)$$

is the average amount of information (in bits) which is necessary to determine or specify a word in the text T. $P(T)$ is simply the probability the language model would assign to the text T.
Some implementation details of the language modeling component we use in our system work are described in section 5.1, since it uses a sophisticated method to determine the counts which are required for probability estimations.

3.3 "Translation" Model

In this section the statistical models used for the disfluency correction system proposed in this work are described. First we describe the overall translation model which "translates" disfluent speech into fluent speech (section 3.3.1). In section 3.3.2 we present the properties of disfluencies our translation model uses. Each property is encoded by one of five statistical models. All five models compose together the overall translation model.

3.3.1 Overall Model

As already mentioned in section 1.4 the idea of the noisy channel which we already explained for SMT is used for disfluency correction as well. While in statistical machine translation a string S in a source language should be translated into a string T in a target language, the goal of disfluency correction is to "translate" a noisy string N which may contain some disfluencies into a clean string C where all disfluencies are removed. Hence the disfluent string can be considered as source language and the fluent string as target language. Similar to translation machine translation the goal is to find this clean string $\hat{C}$ among all possible clean strings, for which the probability of observing it, given the noisy string N, is maximal:

$$\hat{C} = \arg\max P(C|N) = \arg\max_{C} P(N|C) \cdot P(C) \tag{3.22}$$

The probability $P(C)$ denotes the language model of fluent speech without disfluencies. In section 3.2 we explained already how to model this probability. This section concentrates on modeling the probability for observing a disfluent string N given the fluent string C: $P(N|C)$. Remember our assumption that the correction of disfluencies is a sequence of deletions to be made in the disfluent string in order to obtain the corrected fluent string. The fluent string represents then the originally intended utterance (section 2.2). Thus insertions or substitutions or even reorderings of words are not necessary to correct a disfluency. The following example illustrates that:

Disfluent string N:	`Let's meet on Monday, no, on Tuesday.`
Fluent string C:	`Let's meet on Tuesday.`

To model this process of deletions, we adopt the idea of alignments between the words of the source and the target sentence from SMT which we modify for our requirements. For the rest of this section, we set N to be a disfluent string of length J with the words n_j and C to be a fluent string of length I with the words c_i:

$$N = n_1^J = n_1 \dots n_J$$
$$C = c_1^I = c_1 \dots c_I$$

As in SMT we introduce for the string n_1^J a sequence of hidden alignments a_1^J and rewrite the translation model probability $P(N|C)$ as follows:

$$P(N|C) = P(J|I) \cdot P(m|J,C) \cdot \sum_{a_1^J} P(n_j^J, a_1^J | c_1^I, I, J, m) \tag{3.23}$$

Equation 3.23 shows, that we now have to calculate the sum over all possible alignment sequences in order to determine the probability $P(n_1^J|c_1^J)$. $P(J|I)$ denotes the probability for

having a disfluent sentence of length J given the length of the fluent sentence I. We don't use a sentence length model in our system, so we set $P(J|I)$ to be uniform. Nevertheless, there is a correlation between the length of the disfluent sentence and the fluent sentence, as shown in [Shriberg, 1994] for example. We model this correlation indirectly using the probability $P(m|J,C)$, which is probability for having m disfluencies in a potentially disfluent sentence with the length J. Thus m is the number of deletions per sentence where a deletion is defined as a deletion of a contiguous word sequence. Although the length of deletions may vary as we will show later, it is obvious that I decreases with growing m. Therefore we decided to model the correlation between m and J rather then the correlation between I and J or even both correlations in order to keep the number of parameters we have to learn low. Given a particular J, there are much less possible values for m than for I.
Alignments in SMT usually allow that position j in the source sentence is mapped to an arbitrary position i in the target sentence (section 3.1). Our definition of alignment is more restrictive: A position j in the source sentence is either aligned to position 0 in the target sentence or it is appended to the target sentence, i.e. it is aligned to its last position, when we assume that the target sentence is generated from left to right. The alignment of the word n_j to position zero means that it does not occur in the target sentence as we hypothesize it to be disfluent, hence it is deleted. Appending n_j means that we hypothesize it to be a fluent word which belongs to the utterance originally indented by the speaker. Formally our alignment mapping is defined as follows:

$$a_j : \begin{cases} j \rightarrow 0 & \text{if the word at position } j \text{ is deleted} \\ j \rightarrow i = \max_{k<j}\{a_k\} + 1 & \text{otherwise} \end{cases} \tag{3.24}$$

Note that in our case for a given pair of strings (n_1^J, c_1^I) the number of possible alignments for which the probability $P(n_j^J, a_1^J | c_1^I, I, J, m)$ is non-zero is very small. In most cases there is even only one reasonable alignment, between *given* clean and disfluent strings. The example below shows all possible alignments between a disfluent sentence containing a correction and the corresponding fluent sentence. The reason for this small number of possible alignments is, that - as explained above - no word reorderings occur in disfluency correction.

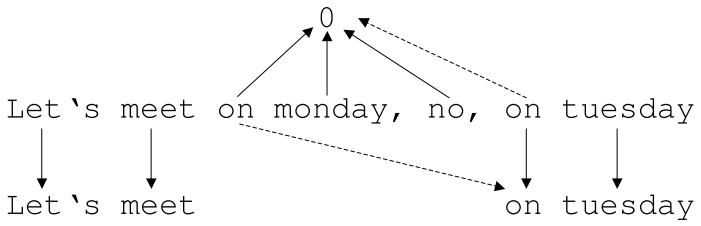

Figure 3.1: Example for alignment between n_1^J and c_1^I. The dotted lines indicate the possible second alignment, which is however very unlikely.

In order to eliminate the sum over all possible alignment sequences in equation 3.23, we use of the so-called maximum approximation. The sum is replaced by the maximum operator, which determines this alignment, which contributes most to the overall probability (equation 3.25).

The maximum approximation is sometimes used in SMT as well (for example in [Stephan Vogel and Hermann Ney and Christoph Tillmann, 1996]).

$$P(N|C) \approx P(J|I) \cdot P(m|J,C) \cdot \max_{a_1^J} P(n_j^J, a_1^J | c_1^I, I, J, m) \tag{3.25}$$

Two properties of the set of possible alignments between a disfluent and a fluent sentence allow us to make use of this approximation:

- As illustrated in figure 3.1 there are only very few possible alignment sequences for a sentence pair, i.e. there are only few terms in the sum of equation 3.23.
- There is one alignment sequence which is assigned a very high probability. All other sequences have very low probabilities.

As in SMT we can now decompose the probability $P(n_j^J, a_1^J | c_1^I, I, J, m)$ into the product over all positions j of the probabilities to have the word n_j and the alignment a_j, given the words n_1^{j-1} and the alignments a_1^{j-1} which have already been processed:

$$P(n_1^J, a_1^J | c_1^I, I, J, m) = \prod_{j=1}^{J} P(n_j, a_j | n_1^{j-1}, a_1^{j-1}, c_1^I, I, J, m) \tag{3.26}$$

Assuming that the probability for having n_j and a_j depends only on n_1^{j-1}, a_1^{j-1} and c_i with $i = a_j = \max_{k<j}\{a_k\} + 1$ or $i = 0$ we obtain equation 3.27. The last line of this equation is obtained by decomposition of the conditional probability.

$$\begin{aligned} & P(n_1^J, a_1^J | c_1^I, I, J, m) \\ &= \prod_{j=1}^{J} P(n_j, a_j | n_1^{j-1}, a_1^{j-1}, c_i) \\ &= \prod_{j=1}^{J} P(a_j | n_1^{j-1}, a_1^{j-1}, c_i) \cdot P(n_j | n_1^{j-1}, a_1^{j-1}, a_j, c_i) \end{aligned} \tag{3.27}$$

Now we consider the probability $P(n_j | n_1^{j-1}, a_1^{j-1}, a_j, c_i)$ from the last line of equation 3.27 and show that it can be ignored during the search for the best hypothesis $\hat{C}$ because it does not depend on C according to some assumptions we make. We claim that this probability is evaluated as follows:

$$P(n_j | n_1^{j-1}, a_1^{j-1}, a_j, c_i) = \begin{cases} 1 & \text{if } c_i = n_j \text{ and } a_j \neq 0 \\ 0 & \text{if } c_i \neq n_j \text{ and } a_j \neq 0 \\ P(n_j) & \text{if } a_j = 0 \end{cases} \tag{3.28}$$

If $a_j = 0$, then the word n_j is deleted because it is a disfluent word. In this case we make no special assumptions about n_j and claim that it is determined by the a-priori probability $P(n_j)$. If $a_j \neq 0$, then $a_j = i = \max_{k<j}\{a_k\} + 1$. In this case the word n_j is determined by c_i which is given. Thus there are only two choices for the probability of n_j. The probability for observing this n_j with $n_j = c_i = c_{a_j}$ is one, since it is the only word which fits when a_j and c_i are given. Therefore the probability for observing another word n_j with $n_j \neq c_i$ has to be zero.

As one can see from equation 3.28, the probability for observing n_j does only depend on c_1^I in so far, that we distinguish whether n_j equals to c_i or not. We take this into account by simply not considering those pairs (n_j, a_j, c_{a_j}) in our search, for which the probability evaluates to zero. For the case $a_j = 0$ the probability does not depend at all on c_1^I so we can ignore it during search because of the argmax-operation.
Note that we would have to consider this probability, if the words n_j and c_i were different with a certain probability. This would be the case, if our input was automatically transcribed speech by a speech recognizer, where the translation from n_j to c_i included the correction of recognizer errors.
Hence only the "alignment" probability $P(a_j|n_j, n_1^{j-1}, a_1^{j-1}, c_i)$ remains to be explained. This probability is composed of five different models, each modeling one property of disfluencies we consider to be important for disfluency correction. We chose these five properties as we found evidence in our data that they are useful for modeling disfluencies (section 4.2). Findings in several studies about disfluencies (e.g. [Shriberg, 1994]) confirm these choices. However there are certainly some more properties of disfluencies which could be modeled in order to improve the performance of our system, in particular acoustic properties. This will be left to further work.
We obtain the total alignment probability by summing up the probabilities contributed by each of the five models we use and dividing the sum by the total number of models we used. We will explain that in some cases not all of the five models contribute to the over all probability. Therefore we define the assignment function α which evaluates to one for the probability P_u, if P_u contributes to the overall probability and to zero otherwise. Thus we can express the "alignment" probability as follows:

$$P(a_j|n_1^{j-1}, a_1^{j-1}, c_i) = P_{alignment} = \frac{\sum_{u=1}^{5} P_u}{\sum_{u=1}^{5} \alpha(P_u)} \tag{3.29}$$

Note that because of this formulation of the total alignment probability, it is comparatively easy to add further models. The probability for a new model is incorporated by simply adding a term in the sums in the enumerator and the denominator.

3.3.2 Models for Properties of Disfluencies

In this section we explain the five different models which contribute to the alignment probability. For each model we give first a verbal description and then present a mathematical formulation which expresses the model using our definition of alignments presented in the previous section. The following five properties of disfluencies are modeled:

- position of a disfluency in a sentence
- length of the deletion region of a disfluency
- length of the deletion region of a disfluency with a fragment at the end of the reparandum
- context of words in which a potential deletion occurs
- context of deletions in which a potential deletion occurs

The probability estimates for the models are based on the relative frequencies for the parameter values of each feature. Some statistics about these estimates will be presented in section 4.2.

Position of a disfluency in a sentence

As presented in section 2.3.1 one of the findings in [Shriberg, 1994] is that disfluencies are much more likely to occur at sentence initial positions than at sentence medial positions. The analysis of our data confirms these findings. Thus we decided to use a model which assigns different probabilities for deletions/no deletions of words at a sentence initial or a sentence medial position. We do not distinguish between different medial positions, since the distribution of deletions/no deletions at medial positions is almost uniform. The position of a disfluency is modeled by the probability P_1. We define the position of the first word of a disfluency as the position of this disfluency.
Thus P_1 decides whether the word at position n_j in the source sentence is deleted given only the information about its position j. Formally this can be expressed as follows:

$$P_1(a_j|a_1^{j-1}, n_1^{j-1}, c_i) = P(a_j|j, c_i) \tag{3.30}$$

Length of the deletion region of a disfluency

This model takes into account that the number of disfluencies decreases rapidly over length of their deletion regions. In [Shriberg, 1994] this behavior is described with an exponential model, which fits pretty well for the corpora examined in this work. We do not use such a model, because the distributions of our data differ from an exponential model for short deletion regions. This may be due to the fact that we consider all types of disfluencies, while Shriberg considers only repetitions/corrections and false starts for her model, and no filler words.
The probability P_2 uses the information about the rapidly decreasing number of disfluencies with increasing length of their deletion region as follows. It models the probability that the word at position j is deleted or not given the information about the deletions of the words *directly* preceeding position j. Thus P_2 depends only on the previous alignments which contain the information about previous deletions.

$$P_2(a_j|a_1^{j-1}, n_1^{j-1}, c_i) = P(a_j|a_1^{j-1}, c_i) \tag{3.31}$$

P_2 is only part of the overall alignment probability, when position j is in a hypothesized deletion region. That means that it models the length of a deletion, given that there is a deletion. Thus P_2 does not contribute to the decision whether a deletion will start at position j or not. Therefore it is one of the probabilities for which the function α evaluates to zero in some cases: $\alpha(P_2) = 0$ whenever there is no deletion region hypothesized at the current position.

Length of the deletion region of a disfluency containing a fragment

In most of the works we reviewed in section 2.3 fragments are found to be very useful for disfluency detection an correction. They can be used to detect the offset of the reparandum. Thus the task which remains in order to be able to delete the reparandum is to determine its onset. We analyzed the length of deletion regions of disfluencies with a fragment at the end of the reparandum in order to create a probabilistic model which helps to determine the reparandum onset.

P_3 uses the resulting distribution of this analysis (length of deletion regions over number of disfluencies containing a fragment) to model the probability that a word at position j is deleted, given that a fragment occurs at position $j + d$.
As we know the complete source sentence when we start correcting it, we also have the information about all fragments in the sentence. However our mathematical modeling restricts us in so far, that we can't assume the source sentence to be given completely, but only the words n_1^{j-1} when processing n_j. Therefore we use the a-priori probability $P(f_{j+d})$ to predict a fragment at position $j + d$, which can however be ignored during search as the probability does not depend on c_1^I. Thus we obtain the following formulation for P_3:

$$P_3(a_j|a_1^{j-1}, n_1^{j-1}, c_i) = P(a_j|f_{j+d}, c_i) \cdot P(f_{j+d}) \tag{3.32}$$

P_3 is another example for a probability which sometimes does not contribute to the overall probability for the alignment model. This is the case, whenever no fragment occurs in the current sentence *after* the position j which is currently processed.

Word context in which a potential deletion occurs

Very often disfluencies are phenomena which can be detected and corrected when taking the local context into account. This is evident for discourse makers like the word "well" which is disfluent in the sentence "Alright, well, let's take the flight tomorrow" but fluent in the sentence "Well done!". The left context "alright" in the first sentence indicates that "well" is a real discourse marker, the right context "done" in the second sentence indicates that "well" can not be deleted as a disfluency in this sentence. Short repetitions and corrections can also be identified using the local context. One word repetitions like "I I" or syntactic anomalies like "he she" are a good example for this.
We use a model which decides whether a word is deleted or not given the local word context. The probability P_4 denotes this context model. Formally the context model predicts the alignment a_j for position j, given the word n_j at this position, the previous word n_{j-1} and the following word n_{j+1}.
Again we have to use of a-priori probabilities, this time $P(n_{j+1})$ and $P(n_j)$, in order to determine n_{j+1} and n_j which we can't assume to be given according to our modeling. However these probabilities do not depend on c_1^I and can therefore be ignored during search. Thus we can express P_4 as follows:

$$P_4(a_j|a_1^{j-1}, n_1^{j-1}, c_i) = P(a_j|n_{j-1}, n_j, n_{j+1}, c_i) \cdot P(n_{j+1}) \cdot P(n_j) \tag{3.33}$$

Deletion history of a potential deletion

Finally we predict the alignment a_j for position j given the word n_j at this position and the two previous alignments a_{j-1} and a_{j-2}. We call the model performing this prediction "binary context model" since the deletion history (encoded by the previous alignments) can be described with a binary sequence: 0 for deletion, 1 for no deletion. The goal of implementing this model is to examine the hypothesis that the knowledge about the previous deletions and the current word is sufficient to predict the deletion of the current word. An advantage of such a model compared to the context model would be that much less parameters have to be learned. This is beneficial especially in situations of data sparseness. As we will show in chapter 6 however this hypothesis seems to be wrong.

The probability P_5 denote the binary context model. Its mathematical formulation looks as follows:

$$P_5(a_j|a_1^{j-1}, n_1^{j-1}, c_i) = P(a_j|n_j, a_{j-1}, a_{j-2}, c_i) \cdot P(n_j) \tag{3.34}$$

Some more details about how the different models make use of the data acquired from the training corpus will be given in section 5.2.3.
We conclude this section by presenting another way to combine the five models which proved to be more efficient in our experiments than just taking the average of all probabilities.
Remember that the model represented by the probability P_3 predicts a deletion at position j given that there is a fragment at position $j+d$. Instead of adding this probability to the others and taking the average as over all probability, we tried to use P_3 in order to explicitely favor those hypotheses in which some words preceeding a fragment are deleted and to penalize those hypotheses where these words are not deleted. That means that we *add* P_3 to the other probabilities when a deletion is hypothesized and we subtract it when no deletion is hypothesized at position j.
For clarity we skip now the given information for each probability (i.e. a_1^{j-1}, n_1^{j-1} and c_i) and denote the over all alignment probability for a deletion at an arbitrary position as $P_{alignment}(0)$ and for no deletion as $P_{alignment}(i)$ respectively. In the same fashion we write for the probabilities of the models composing the overall probability $P_u(0)$ and $P_u(i)$ respectively.
Now we can express the new way for combining the different models as follows:

$$\begin{aligned} P_{alignment}(0) &= \frac{\left(\sum_{u=1,u\neq 3}^{5} P_u(0)\right) + P_3}{\sum_{u=1,u\neq 3}^{5} \alpha(P_u(0))} \\ P_{alignment}(i) &= \frac{\left(\sum_{u=1,u\neq 3}^{5} P_u(i)\right) - P_3}{\sum_{u=1,u\neq 3}^{5} \alpha(P_u(i))} \end{aligned} \tag{3.35}$$

Note that $\alpha(P_3)$ does not occur in the denominator in order to make sure that $P_{alignment}(0) + P_{alignment}(i)$ sums up to one for each position j.

Chapter 4

Corpora

This chapter describes the two corpora we deal with in this work: The English VERBMOBIL corpus on which most analysis and experiments are done and the Mandarin Chinese CallHome corpus which is used to investigate the portability of our system to other languages than English.
In section 4.1 we give some general information about the two corpora. The setup for the recordings and the contents of the dialogs is explained and some details about the transcriptions are given. Furthermore we present important statistics about the two corpora. Finally we compare some characteristics of both corpora which we consider to be important for our work.
In the next section the properties of disfluencies are analyzed and compared for the two corpora. We present distributions for each disfluency property which is used in one of the models we explained in section 3.3. Some of these distributions are directly used in the corresponding model.

4.1 Corpus Characteristics

4.1.1 English Verbmobil Corpus

The goal of the VERBMOBIL project was to build a speech-to-speech translation system being able to cope with spontaneous speech. Besides the English corpus, German and Japanese corpora exist. The information we present about the data in the following is drawn from [Kurematsu et al., 2000] . A detailed description of the VERBMOBIL project can be found in [Wahlster, 2000].
The transcriptions which we are using are taken from the English corpus of the VERBMOBIL 2 project and consist of of 127 face-to-face dialogs of American English speech. In the dialogs two people who play the role of business partners make a travel arrangement for a trip to Hannover. They have to schedule a date which is suitable for both, find a flight, agree on a hotel and optionally dicuss possibilities for going out in the evening or visiting sights. The following documents were provided previously to the recordings: calender sheets, a flight timetable and hotel information.
Both subjects were native speakers of America in English and they were instructed to speak freely without any restrictions concerning pronunciation. They sat face-to-face on one desk and interferences by the dialog partner (i.e. cross-talking) were allowed. The domain was restricted by instructing the speakers to stay with the topics of organizing their business trip.

The goal of these role playing scenarios was to achieve a realistic dialog situation for the recordings.
For each of the dialogs an orthographic transliteration is provided which we use in this work. Words are transcribed according to orthographic rules of the specific language and syntactical structures such as periods, question marks and commas are marked. Word categories like names, number or foreign names are indicated by special characters. To meet the special requirements for annotating spontaneous speech, phenomena like disruption of sentences, false starts, repetitions, corrections,interjections, hesitations, word fragments, pauses etc. are annotated as well. Out of these annotations the information about repetitions/corrections, false starts, interjections and fragments is extracted for our work (section 2.1). Periods and question marks are used to separate sentences in one turn. All other annotations in the transcripts are ignored. Note that in the VERBMOBIL project a turn is defined as one contribution of a dialog participant (for example [Alexandersson et al., 1997]). Thus one turn can consist of multiple sentences.
The criteria when to mark words as repetition/correction, false start or fragment (in VERBMOBIL terminology "aborted articulation") can be found in [Burger, 1997]. Remember that we adopted these criteria for our definition of repetition/correction, false start and interjection (section 2.1).
Table 4.1 presents important figures about the English corpus from the VERBMOBIL 2 project. The sentence count which we determined differs from the count reported in [Kurematsu et al., 2000] which is 16525. This may be because we count a turn which consists only of a single word fragment as one sentence. Furthermore we have only a vocabulary of 2290 words while in [Kurematsu et al., 2000] a vocabulary size of 2557 is reported. The reason for that might be that we don't consider fragments to be part of the vocabulary.

Dialogs	127
Turns	10566
Sentences	16583
Words	118356
Vocabulary	2290
Speakers	60

Table 4.1: Corpus statistics for the VERBMOBIL corpus

4.1.2 Mandarin CallHome Corpus

The Mandarin CallHome corpus was collected and transcribed by the Linguistic Data Consortium (LDC) in 1997. It consists of 120 telephone conversations between native speakers of Mandarin. For our experiments we used the manual transcriptions provided by the LDC [Wheatley, 1997]. As for the VERBMOBIL corpus, we first outline the setup for the recordings of the data and then present important corpus statistics.
The data which is available consists of transcripts which cover contiguous 5 or 10 minute segments taken from recorded conversations lasting up to 30 minutes. The participants which were recruited for the recordings were given a free choice of whom to call and there were no guidelines concerning what they should talk about. Most participants called close friends or family members overseas. All speakers were aware that they are recorded.

The corpus is split into a training set which consists of 80 dialog fragments each lasting 10 minutes, a development test set consisting of 20 dialog fragments each lasting 10 minutes and finally an evaluation test set consisting of 20 dialog fragments where each lasts only 5 minutes.
Mandarin Chinese texts usually contain no spaces between words. For the transcriptions of the Mandarin CallHome corpus however, a segmentation was performed in order to provide the actual word sequence for each utterance. The "Dragon Mandarin Segmenter" was used to create this segmentation automatically. More detailed information and further references about the segmentation procedure and segmentation principals are given in the documentation of the transcription of the corpus [Wheatley, 1997]. We describe here only the key idea of the automatic segmentation.
The automatic segmenter uses a lexicon in order to break down a given string of Chinese characters into known and unknown words. The latter ones do not appear in the lexicon and are usually single-character words. Each hypothesized word is assigned a cost, which is computed from its frequency. The word frequencies are determined from another text on which the lexicon was build. The cost of a completely segmented string is the sum of the costs of its words. Thus a dynamic programming approach can be used in order to find the lowest cost segmentation for each string.
For our experiments we used the training set to extract all the features from the corpus we need for our system and to train the language model. Testing was performed on the evaluation test set. As no parameter tuning using a cross-validation set is performed for our system, we didn't use the development test set.
In table 4.2 important corpus statistics about the training set and the evaluation test set are presented. While there are more sentences than turns in the VERBMOBIL corpus, the Mandarin CallHome corpus has more turns than sentences. The reason for this is the different definition of turn for the two corpora. As explained above, a turn can consist of multiple sentences in the VERBMOBIL corpus. For the Mandarin CallHome corpus a turn is defined to be a continuous stretch of speech which starts and ends with a pause [Wheatley, 1997]. Thus one sentence can consist of multiple turns.

	Training set	Evaluation test set
Dialogues	80	20
Turns	26305	3335
Sentences	22089	2178
Words	178781	23318
Vocabulary	6995	2131
Speakers[1]	160	40

Table 4.2: Corpus statistics for the Mandarin Chinese CallHome Corpus

4.1.3 Comparison

An important difference between the VERBMOBIL corpus and the Mandarin CallHome corpus is the variety of topics. While the topic is restricted to travel arrangements in the VERBMOBIL

[1] In each conversation usually the participant recruited for the recordings and the dialog partner he called were involved. However sometimes there were more than one dialog partners, when the telephone was passed around.

corpus, the speakers in the Mandarin CallHome dialogs can chose their topics freely. This is certainly one reason why the vocabulary of the Mandarin CallHome corpus is more than three times larger than the vocabulary of the VERBMOBIL corpus, although the Mandarin CallHome corpus comprises only little more words than the VERBMOBIL corpus ($\approx$ 179000 words vs. $\approx$ 107000 words).
Furthermore we measured the percentage of words in the test set vocabulary which have not been seen during training (i.e. the OOV-Rate). Remember that we used the predefined splitting of training and test set for the Mandarin CallHome corpus. For the VERBMOBIL corpus we used 10 disjoint test sets each consisting of 10% of the whole corpus in order to avoid biased results. For each test set the remaining 90% of the corpus were used for training (section 6.2). For the VERBMOBIL corpus we found an OOV-Rate of 7.8% while we have 23.8% for the Mandarin CallHome corpus.
We believe that because of the much larger OOV-Rate for the Mandarin CallHome corpus the task of disfluency correction for this corpus is much harder then for the VERBMOBIL corpus. Especially the language model and the two models relying on the context of an potentially disfluent word (context model and binary context model, section 3.3.2) have to cope much more often with unseen events in case of the Mandarin CallHome corpus than in case of the VERBMOBIL corpus.

4.2 Disfluency distributions

In this section we analyze the disfluencies in the two corpora. First we present the statistics of disfluencies by type. Then each property of disfluencies is analyzed which is modeled by one of the models introduced in section 3.3.2. Interesting differences between the two corpora concerning the disfluency properties are compared. Further details about how our system actually uses the results of the analysis are presented in section 5.2.3. Note that the figures we report for the Mandarin CallHome corpus always apply to this part of the corpus we use, i.e. the training set and the evaluation test set.

4.2.1 Disfluencies by type

In section 2.1 we explained that we consider slightly different disfluency types for the two corpora. For the English VERBMOBIL corpus we consider false starts (FS), repetitions/corrections (REP), editing terms (ET), discourse markers (DM), filled pauses (UH) and interjections (IN). For the Mandarin CallHome corpus only false starts (FS), repetitions/corrections (REP) and filled pauses (FP) are considered, where the latter type corresponds to the two types DM and UH in the English VERBMOBIL corpus. (See section 2.1 for the exact explanation of these terms.)

English Verbmobil corpus

Table 4.3 shows the the absolute and relative counts for each disfluency type in the corpus. In the last two rows the percentage of sentences containing one or more disfluencies is displayed. First we use all sentences in the corpus for the total number of sentences. Then only those sentences which contain at least one disfluency are taken into account. These are 8790 sentences which is about 53% of the total number of sentences in the corpus. However most of these sentences (7251) contain only one disfluency.

Note that all filler words with no semantic content, i.e. filled pauses and discourse markers make up together 63.7% of all disfluencies. In 41% of all sentences and in 78.5% of all disfluent sentences at least one of these filler words occurs. That means that this is a very important category of disfluencies to be treated by the correction system. However filler words are mostly easy to identify, as the word itself and the local context provide strong evidence for their identification.

	REP	FS	ET	DM	UH	IN	Total
Absolute counts	2303	1209	155	1248	5776	336	11027
Relative counts	20.9%	11.0%	1.4%	11.3%	52.4%	3.0%	100%
Percentage of sentences containing at least one DF	10.7%	6.2%	0.9%	7.3%	34.3%	2%	53%
Percentage of sentences containing at least one DF considering only disfluent sentences[2]	20.3%	11.8%	1.8%	13.7%	64.7%	3.8%	100%

Table 4.3: Disfluencies (DFs) in the English VERBMOBIL corpus

Mandarin CallHome corpus

Table 4.4 illustrates the distribution of the three disfluency types which are annotated in the Mandarin CallHome corpus. As in table 4.3 the percentage of each disfluency type of the total number of disfluencies and the percentage of sentences containing at least one disfluency considering all sentences and disfluent sentences only is shown.

	REP	FS	FP	Total
Absolute counts	9603	241	11745	21589
Relative counts	44.5%	1.1%	54.4%	100%
Percentage of sentences containing at least one DF	30.2%	1.0%	34.7%	50.4%
Percentage of sentences containing at least one DF considering only disfluent sentences[2]	60.9%	1.9%	69.9%	100%

Table 4.4: Disfluencies (DFs) in the Mandarin Chinese CallHome corpus

We note that the percentage repetitions/corrections in the Mandarin CallHome corpus is much higher than in the English VERBMOBIL corpus. However filled pauses (i.e. discourse markers/filled pauses for the English VERBMOBIL corpus) are still the most frequent phenomenon in both corpora, although the percentage is higher for the English VERBMOBIL corpus than for the Mandarin CallHome corpus. This again suggests that disfluency correction may be harder for the Mandarin CallHome corpus than for the English VERBMOBIL

[2]The values in this row sum up to more than 100% as it is possible that disfluencies of different types occur in one sentence.

corpus, as filler words are easier to correct than repetitions or especially corrections of phrases. For filler words it is mostly sufficient to consider the word itself and the local context to identify it. For repetitions/corrections the onset and offset of the reparandum must be identified which is a harder task.
The rates of total sentences which are disfluent are in the same region for both corpora (53% in the English VERBMOBIL corpus and 50.4% in the Mandarin CallHome corpus). When considering the rate of disfluencies per word, one can find however that the Mandarin CallHome corpus (0.119 DFs per word) is more disfluent than the English VERBMOBIL corpus (0.093 DFs per word).

4.2.2 Number of deletions per sentence

The number of deletions per sentence which has to be made for disfluency correction is modeled in our system by the probability $P(m|J,C)$. In this probability m is the number of deletions which have to be made in a source sentence with length J in order to obtain the corresponding corrected sentence C. This probability is included in the overall translation model (section 3.3.1). In the following we present some statistics on which our modeling of the probability $P(m|J,C)$ is based. As there are a lot of parameters to estimate when for each sentence length J each possible number of deletions m is considered, we introduce equivalence classes for different sentence lengths and numbers of deletions per sentence.
Note that the number of disfluencies per sentence is exactly equal to the number of deletions which has to be made to correct a sentence perfectly. This is because we assume that each disfluency can be corrected by deleting a sequence of words. In case of filler words these are just the words themselves. For repetitions/corrections and false starts, the reparandum has to be deleted (section 2.2). Thus exactly one deletion of a contiguous word string is required to correct a disfluency. Therefore we use the number of disfluencies per sentence to model the number of deletions which have to be made for correction.

English Verbmobil corpus

First we analyze number of disfluencies per sentence for the English VERBMOBIL corpus without considering the sentence length. Table 4.5 shows the distribution for the number of disfluencies per sentence, grouped by disfluency type and in total. According to this table there are 8790 sentences which are non-fluent. The major part of these sentences contains only one disfluency (82.5%).
When considering the number of disfluencies per sentence grouped by type one can see that particularly complex disfluencies (i.e. repetitions/corrections and false starts) tend to occur multiply in one sentence. The reason for this might be that sometimes multiple corrections or restarts are necessary until the speaker finally expresses what he/she intended to say. Note that the total number of sentences with m disfluencies (last column of table 4.5) is *not* equal to the sum over all sentences containing m disfluencies of a particular type (i.e. the sum over all entries in a row of the table). This is because disfluencies of different types can occur in one sentence.
Figure 4.1 shows the rate of disfluencies per sentence length. For sentences with length of more than 20 words only very little data exists. Therefore we think that the resulting estimates for sentence lengths over 20 words are too unreliable to plot them in this graph. For the other sentence lengths one can observe a roughly linear growth of this rate except

Number of DFs per sentence	REP	FS	ET	DM	UH	IN	Total
1	1408	896	153	1168	5612	336	7251
2	287	117	1	37	79		1074
3	62	21		2	2		310
4	28	4					107
5	2						31
6	1						10
7							4
8							1
9							1
10							1

Table 4.5: Number of disfluencies (DFs) per sentence for the English VERBMOBIL corpus

for sentence lengths one and two. For sentence length two however, the deviation from the linear trend is not so large.

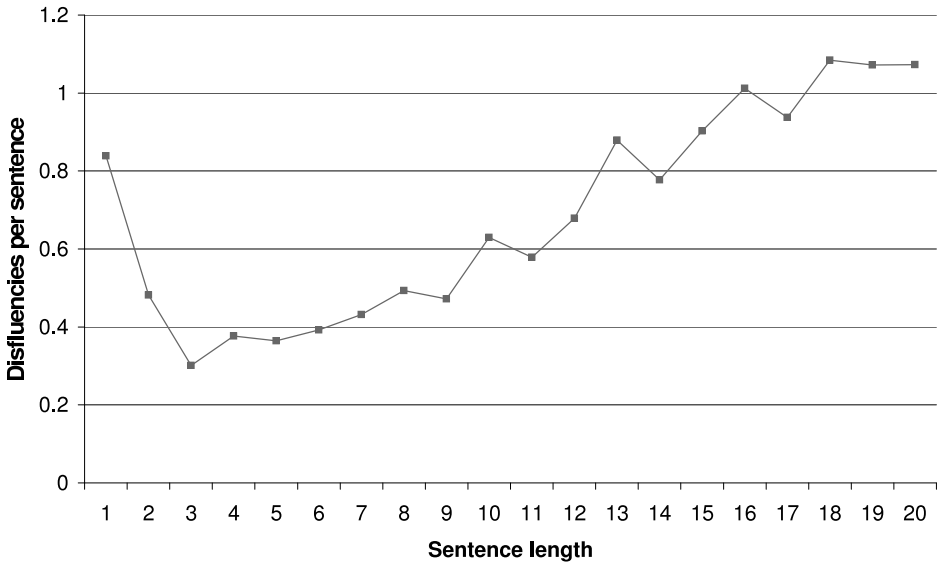

Figure 4.1: Rate of disfluencies per sentence for the English VERBMOBIL corpus

The distribution of disfluencies per sentence motivated us to distinguish only between sentences with zero, one or more than one disfluencies: 7793 (47.0%) sentences do not contain a disfluency at all, 7251 (43.7%) sentences contain exactly one disfluency and only 1539 (9.3%) sentences contain more than one disfluency.
According to the trend shown in figure 4.1 we chose equivalence classes for sentence lengths. Since there are 4100 sentences with consist only of one word (24.7% of the whole number of sentences), and an anomaly for the rate of disfluencies for this sentence length can be observed in figure 4.1, we use an extra class for this sentence length. Furthermore we put sentences with 2 to 10 words in one class and sentences with 11 to 20 words in another class. Finally sentences with more than 20 words are put in one class as very little data is available here. Table 4.6 shows the choice of equivalence classes for sentence lengths and number of deletions

per sentence. In parentheses relative frequencies are shown, which we use as probability estimates to predict the number of deletions per sentence given the sentence length.

Sentence length (from - to)	0 DF	1 DF	>1 DF	Total sentences
1	661 (0.16)	3439 (0.84)	0 (0.0)	4100
2 - 10	5586 (0.65)	2539 (0.29)	524 (0.06)	8649
11 - 20	1291 (0.45)	1001 (0.35)	585 (0.2)	2877
≥ 21	255 (0.27)	272 (0.28)	430 (0.45)	957

Table 4.6: Equivalence classes for sentence lengths and number of disfluencies (DFs) for the English VERBMOBIL corpus

Mandarin CallHome Corpus

For the English VERBMOBIL corpus we observed that most disfluent sentences contain only one disfluency. This is also the case for the Mandarin CallHome corpus, however there are much more sentences containing more than one disfluency than in the English VERBMOBIL corpus. 12044 (49.6%) sentences do not contain a disfluency, 7063 (29.1%) sentences contain exactly one disfluency and 5160 (21.3%) sentences contain more than one disfluencies. Nevertheless we use the same equivalence classes for the numbers of disfluencies per sentences which we used for the English VERBMOBIL corpus.
For the rate of disfluencies per sentence we found an interesting difference between the English VERBMOBIL corpus and the Mandarin CallHome corpus. While in the first one there is an anomaly for sentence length one, because the rate of disfluencies is significantly *higher* than for the surrounding sentence lengths, in the latter corpus this rate is significantly *lower* than for the surrounding sentence lengths. For the other sentence lengths we observe a roughly linear growth in both corpora. The rate of disfluencies over the sentence length is compared for both corpora in figure 4.2.
Although there are different kinds of anomalies for sentence length one, we use the same equivalence classes for sentence lengths for both corpora. This can be done, as there is only an anomaly for sentence length one, which is put in an extra class.
Table 4.7 shows for the different sentence length equivalence classes the number of sentences with zero, one and more than one disfluencies for the Mandarin CallHome corpus. In parentheses the relative frequencies for having a sentence in one of the different classes are shown. From this table the difference for sentence length one for the two corpora can be seen as well. While in the English VERBMOBIL corpus the relative frequency for one disfluency in a sentence with length one is 0.84, it is only 0.04 for the Mandarin CallHome corpus. That means that most one-word sentences are fluent in this corpus.

4.2.3 Position of a disfluency in a sentence

The position of a disfluency in a sentence is modeled by the probability P_1 which is one of the five probabilities which contribute to the overall translation model. We explained already that we only distinguish between initial and medial positions of disfluencies in a sentence. The exact position of a disfluency within all possible medial positions plays no role (section 3.3.2).

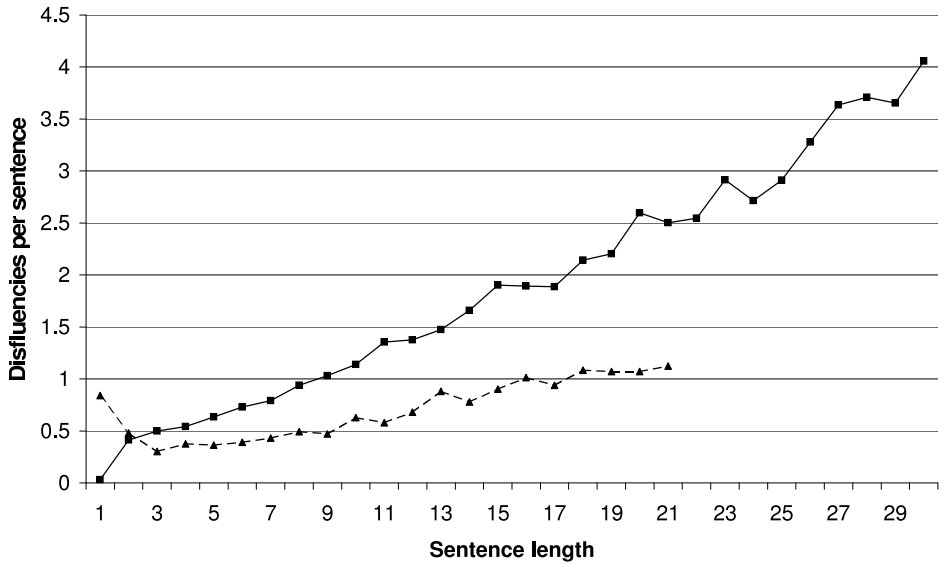

Figure 4.2: Rate of disfluencies per sentence for the Mandarin CallHome corpus (solid line) compared to the English VERBMOBIL corpus (dotted line)

Sentence length (from - to)	0 DFs	1 DF	> 1 DF	Total sentences
1	4037 (0.96)	133 (0.04)	0 (0.0)	4170
2 - 10	7077 (0.49)	5443 (0.37)	2004 (0.14)	14524
11 - 20	865 (0.20)	1329 (0.30)	2208 (0.50)	4402
≥ 21	65 (0.06)	158 (0.13)	948 (0.81)	1171

Table 4.7: Equivalence classes for sentence lengths and number of disfluencies (DFs) for the Mandarin CallHome corpus

English Verbmobil corpus

Table 4.8 shows the number of disfluencies at initial and medial positions, the number of potential locations for disfluencies at these positions and finally the relative frequencies for disfluencies at initial and medial positions which are obtained by dividing the number of disfluencies at a particular position by the number of potential locations at this position. The number of potential locations for disfluencies at an initial position is simply the number of sentences in the corpus. The number of potential locations at a medial position is the number of all potential locations minus the number of potential locations at an initial position which is equal to the number of words in the whole corpus minus the number of sentences. We define the position of a disfluency to be the position of its first word.
In table 4.9 disfluencies are grouped by position and by type. Except repetitions/corrections and editing terms all disfluencies occur more often at initial than at medial positions. This is most prominent for discourse markers/filled pauses which can be explained by their function of helping the speaker to start a turn. Analyzing the data showed that this was mostly the case. Note that the relative frequency of 0.46 for a disfluency at an initial position is much higher than the corresponding figure reported in [Shriberg, 1994]. This is probably because Shriberg does not consider filler words (i.e. discourse markers/filled pauses) for her analysis

	Initial Position	Medial Position
Number of DFs	7664	3363
Number of potential sites	16583	101769
Relative frequencies	0.46	0.03

Table 4.8: Disfluencies by position for the English VERBMOBIL corpus

of disfluencies by position.

	REP	FS	ET	DM	UH	IN
Initial position	498	685	82	808	5259	332
Medial position	1805	524	73	440	517	4

Table 4.9: Disfluencies by position and by type for the English VERBMOBIL corpus

Mandarin CallHome corpus

The distribution of disfluencies at sentence initial and sentence medial positions for the Mandarin CallHome corpus is shown in table 4.10 in the same style as in table 4.8 for the English VERBMOBIL corpus. Although the relative frequency for having a disfluency at a sentence initial position is higher than for having a disfluency at a sentence medial position as in the English VERBMOBIL corpus, disfluencies at sentence initial position are less likely to occur than in the English VERBMOBIL corpus, and disfluencies at medial positions are more likely to occur.

	Initial Position	Medial Position
Number of DFs	7663	13926
Number of potential sites	24267	177832
Relative frequencies	0.32	0.08

Table 4.10: Disfluencies by position for the Mandarin CallHome corpus

Because of the differences between the relative frequencies for disfluencies at initial and medial positions between the two corpora, it is interesting to compare the percentage of disfluencies which occur at initial and medial positions. This is illustrated in table 4.11. We see that in the English VERBMOBIL corpus much more disfluencies occur at initial positions while in the Mandarin CallHome corpus the major part of disfluencies occurs at medial positions. This agrees with the differences in relative frequencies we observe.

4.2.4 Length of the deletion region of a disfluency

The length of the deletion region of a disfluency is modeled by the probability P_2 which contributes to the overall translation model (section 3.3.2). The statistics presented below confirm the intuition that the number of disfluencies with a deletion region of k words decreases rapidly with k.

	English VERBMOBIL	Mandarin CallHome
Initial Positions	69.5%	35.5%
Medial Positions	30.5%	64.5%

Table 4.11: Percentage of disfluencies at initial and medial positions for the English VERBMOBIL and the Mandarin CallHome corpus

English Verbmobil corpus

Table 4.12 shows the number of disfluencies grouped by type and by length of their deletion region. 8402 disfluencies (76.2% of all disfluencies) have a deletion region of length one. Out of these disfluencies 6327 are discourse markers/filled pauses (57.4% of all disfluencies). This is easy to understand as discourse markers/filled pauses are usually single words like "right", "well", "yeah" etc. Longer occurrences of these phenomena are sequences of filler words like "**Well, right**, we can do that". Furthermore we note that only repetitions/corrections and false starts have deletion regions which are longer than four words. However such deletion regions occur very rarely (in 2.2% of all disfluencies).

lod	REP	FS	ET	DM	UH	IN	Total
1	1371	356	12	946	5381	336	8402
2	580	358	143	294	372		1747
3	188	229		7	19		443
4	85	103		1	4		193
5	42	69					111
6	23	50					73
7	6	20					26
8	5	14					19
9	3	6					9
10		3					3
11		1					1

Table 4.12: Distribution of disfluencies over the length of their deletion regions (lod) for the English VERBMOBIL corpus

The distribution of the length of the deletion regions of all disfluencies is illustrated graphically in figure 4.3. As explained above, the number of disfluencies is high for one word deletion regions and decreases rapidly for the other lengths.

Mandarin CallHome corpus

Table 4.13 shows the distribution of disfluencies over the length of their deletion regions grouped by types and in total for the Mandarin CallHome corpus. Similar to this distribution for the English VERBMOBIL corpus, one can observe that, except for false starts, all disfluencies have mostly deletion regions of length one (72% of all disfluencies). As in the English VERBMOBIL corpus, filler words make up the biggest part of disfluencies with deletion regions of length one. Generally, in the Mandarin CallHome corpus longer deletion regions

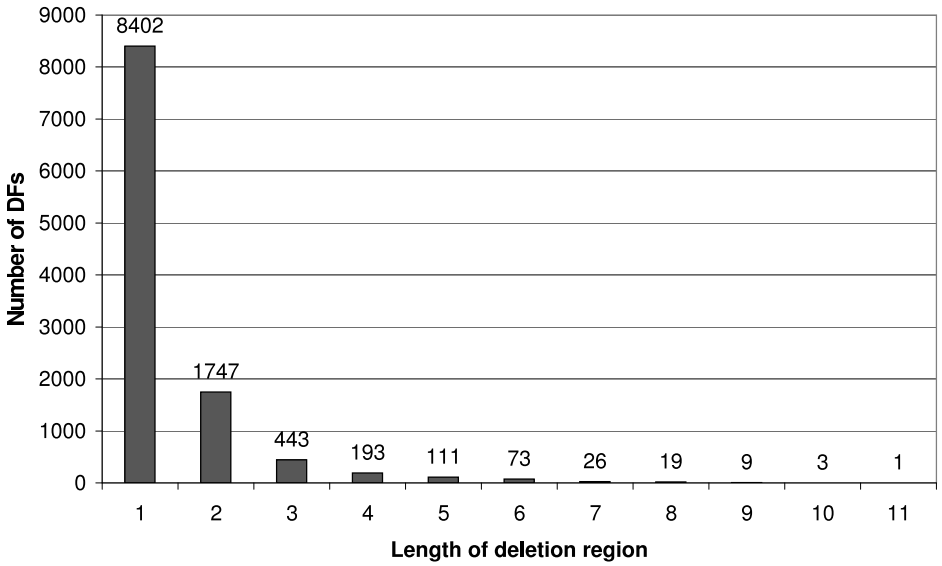

Figure 4.3: Length of deletion regions for the English VERBMOBIL corpus

(up to 17 words) occur in comparison to the English VERBMOBIL corpus with deletion regions up to 11 words.
Figure 4.4 shows the relative frequencies for disfluencies over the length of their deletion regions for the English VERBMOBIL corpus (black bars) and the Mandarin CallHome corpus (white bars). The figures are obtained by dividing the number of disfluencies with deletion regions of a particular length by the total number of disfluencies. One can see that only for deletion regions of length one disfluencies in the English VERBMOBIL corpus are more frequent than in the Mandarin CallHome corpus. In all other cases the disfluency frequency for the Mandarin CallHome corpus is higher. This anomaly might be partly due to the fact, that our disfluency type "interjection" (non-lexicalized sounds of affirmation and negation, mostly with back channeling function) which comprises only disfluencies with deletion regions of length one, is not considered in the Mandarin disfluency annotations. In all other cases the higher bars for the Mandarin Chinese corpus support our claim that the Mandarin CallHome corpus is more disfluent than the English VERBMOBIL corpus.

4.2.5 Length of the deletion region of a disfluency containing a fragment

The probability P_3 of the overall translation model, models deletion regions of disfluencies containing a fragment. Supported by the findings of many authors (section 2.3), we assume that fragments indicate the offset of the reparandum of a disfluency. We examined this assumption only for the English VERBMOBIL corpus, since no fragments are annotated in the Mandarin CallHome corpus.

English Verbmobil corpus

Table 4.14 shows the number of complex disfluencies (i.e. false starts and repetitions/corrections) which have a fragment at the end of the reparandum. In the last column the figures are shown for both disfluency types together. The percentage of the total number of disfluencies of a particular type is shown in the last row. Some conclusions can be drawn from the table. First we note that fragments indicate indeed reparandum offsets, i.e. it is very likely to have

Length of deletion region	REP	FS	FP	Total
1	5111	19	10573	15703
2	2529	61	910	3500
3	1040	57	183	1280
4	479	37	61	577
5	214	23	10	247
6	110	18	5	133
7	50	12	1	63
8	27	4	1	32
9	18	5	1	24
10	9	2	0	11
11	2	1	0	3
12	6	0	0	6
13	2	0	0	2
14	2	0	0	2
15	1	1	0	2
16	0	1	0	2
17	3	0	0	1

Table 4.13: Distribution of disfluencies over the length of their deletion regions for the Mandarin Chinese CallHome Corpus

a reparandum offset at the location of a fragment. On the other hand the percentages in the last row of the table show that fragments to not account for all complex disfluencies. Furthermore it is important to remark that of the 753 fragments which occur at the end of a reparandum of a disfluency 476 fragments (63%) occur in "one-word"-disfluencies where the reparandum comprises only the fragment. Thus we note that fragments are important to locate reparandum offsets but other means must be used as well for the fulfillment of that task.

	Total	in FS	in REP	in REP and FS
Number of fragments	914	221	532	753
Percentage of total DFs for these types		18.3%	23%	21.4%

Table 4.14: Fragments at the end of the reparanda of false starts and repetitions/corrections for the English VERBMOBIL corpus

Our statistical model uses the information about fragments in order to model the length of deletion regions of disfluencies which contain a fragment at the end of the reparandum. The corresponding distribution is shown in figure 4.5. Note that the trend that the number of disfluencies decreases rapidly with increasing length of the deletion region can be observed as well here and in figure 4.3, where the length of the deletion regions for all disfluencies is illustrated. However the difference between the bars for length one and length two is larger in 4.3 which is due to the huge number of filler words of length one.

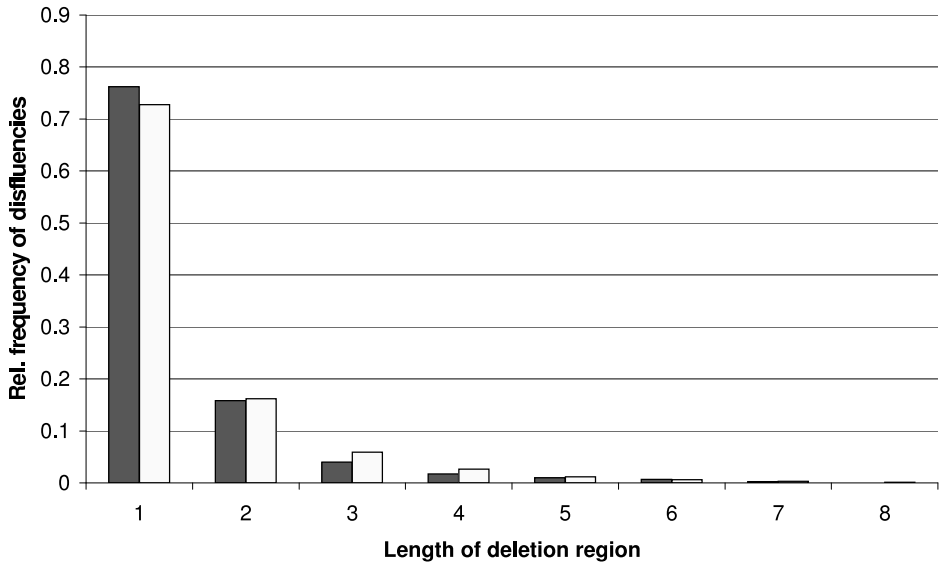

Figure 4.4: Relative frequencies of disfluencies for different lengths of deletion regions; English VERBMOBIL : black bars, Mandarin CallHome: white bars

4.2.6 Word context in which a potential deletion occurs

The context model is modeled by the probability P_4 of the overall translation model. It considers the two words surrounding the current word in order to predict the deletion of the current word (section 3.3.2). We present here some statistics which illustrate how reliable our estimates for the context model are. Thus we analyze how often the different words in a particular context occur. Since we are faced with a sparse data problem in our corpora, we never consider for a word w_i the left word w_{i-1} and the right word w_{i+1} together, but we use word "bigrams". Word unigrams are used when no reliable estimates for bigrams are available. Thus we distinguish the following events:

- left bigrams: $w_{i-1}w_i$
- right bigrams: w_iw_{i+1}
- unigrams: w_i

English Verbmobil corpus

Table 4.15 shows the occurrences of left and right bigrams and unigrams for the English VERBMOBIL corpus. We distinguish between the different events occurring only once, less than three times and at least three times. This is because we decided that an estimation is only reliable if an event has been seen at least three times. For each event type in parentheses the percentage of events is shown which occur once, less than three times or at least three times.
We can see from the table that for bigrams only about $\frac{1}{4}$ of all such events occur at least three times. That means that in most cases no bigrams can be used for predictions. Even for unigrams only about 50% of all events occur more than three times. Therefore we decided to construct equivalence classes for the words in our corpus. These classes are found by an

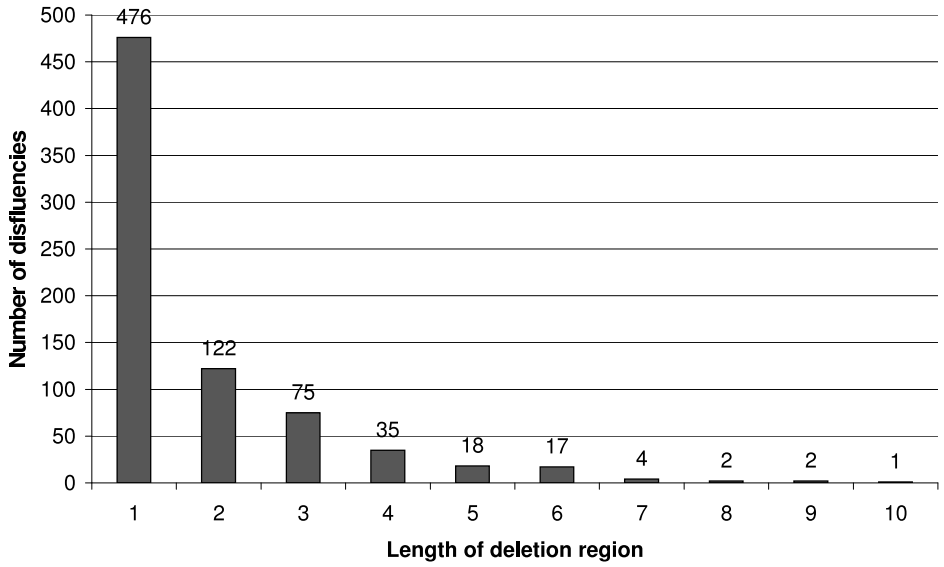

Figure 4.5: Length of the deletion region of disfluencies with a fragment at the end of the reparandum for the English VERBMOBIL corpus

Occurrences	Left bigrams	Right bigrams	Unigrams
1	13021 (60.0%)	13295 (59.6%)	860 (37.6%)
< 3	16134 (74.3%)	16489 (73.9%)	1149 (50.2%)
≥ 3	5584 (25.7%)	5827 (26.1%)	1141 (49.8%)
Total	21718	22316	2290

Table 4.15: Occurrences of left and right bigrams and unigrams for the English VERBMOBIL corpus

unsupervised learning technique proposed in [Kneser and Ney, 1993]. The proposed algorithm determines classes in that way that the probability for a language model based on these classes to predict the training text is maximized. The wordclasses are obtained by using a clustering tool from the CLAUSI toolkit [Ries et al., 1996].
Table 4.16 shows the occurrences of the different events when each word of the text is assigned to its class. 474 wordclasses are used here.
We see that the percentages for events occurring at least three times improve much. For unigrams all events are found more than three times in the corpus.

Mandarin CallHome corpus

Table 4.17 shows the occurrences left and right bigrams and unigrams for the Mandarin CallHome corpus in the same style as table 4.15 for the English VERBMOBIL corpus. The percentage of events occurring at least three times is even smaller than in the English VERBMOBIL corpus. The reason for this is that the vocabulary of the Mandarin CallHome corpus is more than three times larger than the vocabulary of the English VERBMOBIL corpus but the Mandarin CallHome corpus comprises only little more words than the English VERBMOBIL corpus (section 4.1). Nevertheless we did not construct word equivalence classes for the

Occurrences	Left bigrams	Right bigrams	Unigrams
1	6814 (44.2%)	6793 (43.8%)	0 (0.0%)
< 3	9402 (61.0%)	9363 (60.4%)	0 (0.0%)
≥ 3	6002 (39.0%)	6134 (39.6%)	474 (100.0%)
Total	15404	15497	474

Table 4.16: Occurrences of left and right bigrams and unigrams when each word is assigned its equivalence class for the English VERBMOBIL corpus

Mandarin CallHome corpus, as the use of classes for the English VERBMOBIL corpus did not improve our results (section 6.2).

Occurrences	Left Contexts	Right Contexts	No contexts
1	44789 (72.4%)	45214 (72.3%)	2906 (41.5%)
< 3	52778 (85.3%)	53318 (85.3%)	3979 (56.9%)
≥ 3	9079 (14.7%)	9216 (14.7%)	3016 (43.1%)
Total	61857 (100%)	62534 (100%)	6995 (100%)

Table 4.17: Occurrences of left and right bigrams and unigrams in the Mandarin CallHome corpus

4.2.7 Deletion history of a potential deletion

The probability P_5 of the overall translation model denotes the binary context model. This model considers the last two deletions in the current hypothesis in order to predict the deletion of the current word (section 3.3.2). The parameters for this model are learned from the data similar as for the context model. We simply analyze whether the words w_{i-2} and w_{i-1} preceeding w_i are disfluent or not. As for some words not enough data is available to make reliable estimates using a deletion history of two words, we are also interested in the deletion history of one word or even in the deletion history of zero words. Thus we distinguish the following events (the deletion history of a word w_j is symbolized by its alignment history $a_{j-2}a_{j-1}$):

- deletion history of two preceeding words: a_{j-2}, a_{j-1}, w_j
- deletion history of one preceeding word: a_{j-1}, w_j
- deletion history of zero preceeding words: w_j

Note that the last event type is identical with the unigrams in the context model. Since this model did not prove to be successful in our experiments with the English VERBMOBIL corpus (section 6.2), we did not implement it for the Mandarin CallHome corpus.

English Verbmobil corpus

Table 4.18 shows the different events occurring once, less than three times and at least three times in the same style as table 4.15. Similar to the context model, we assume that estimates

are reliable enough only, if they are based on events occurring at least three times. We see that the percentage of events occurring at least three times is generally higher as for the context model.

Occurrences	Two preceeding words	One preceeding word	No context
1	1440 (42.0%)	1106 (39.2%)	860 (37.6%)
< 3	1909 (55.7%)	1472 (52.2%)	1149 (50.2%)
≥ 3	1519 (44.3%)	1350 (47.8%)	1141 (49.8%)
Total	3428	2822	2290

Table 4.18: Occurrences of different events for binary context model for the English VERBMOBIL corpus

Chapter 5

Implementation

This chapter deals with the implementation of our disfluency correction system. In section 5.1 we describe some important implementation details of the language model component of our system. Section 5.2 deals with the search problem to find the best hypothesis. The search algorithm and the data structure on which the algorithm is executed are presented. Finally the exact implementation of the five models we introduced in section 3.3 is described.

5.1 Language Model

The equations for estimating the language model probabilities presented in section 3.2 show that obtaining counts for certain word sequences is essential to calculate these estimates. Many language model implementations calculate the necessary counts during a training phase and store for each n-gram the probabilities which are estimated using the counts from the training corpus. Thus the language model can be represented as a lookup table, which maps every n-gram to a certain probability.
The language model implementation used in our system works differently: during training a suffix array is created which allows to calculate counts efficiently at runtime. We use this language model implementation because programs for constructing suffix array based language models and an interface for using such language model in a decoder were already available. They are used in the SMT toolkit which is developed at the Carnegie Mellon University in Pittsburgh.
An advantage of suffix array based language models is that quite long histories can be easily used for predictions when they have been seen often enough during training. A lookup table based language model would have to store for example all 7-grams which have been seen during training in order to be able to use 7-grams for predictions. Most of these 7-grams would be stored in vain however, since they would never be used when an unknown text is presented to the language model. In contrast to that, the suffix array based language model can determine quickly at runtime whether a 7-gram for prediction is available, without having to store all 7-grams during training. In the following the suffix array based language model component we use is explained briefly.
The idea of a suffix array is illustrated in figure 5.1. Besides the training corpus an array of pointers (i.e. the suffix array) has to be stored. The pointers direct to the words in the corpus in a way that a lexicographic ordering is established: Let S be the suffix array and i and j two indices of this array. Then:

$$i < j \Rightarrow S[i] \leq_{lex} S[j] \tag{5.1}$$

The predicate $\leq_{lex}$ which symbolizes the lexicographic ordering remains to be explained. Let W and V be two sentences with

$$W = w_1 \dots w_l \dots w_n$$
$$V = v_1 \dots v_k \dots v_m$$

Let furthermore $S[i]$ (i.e. the i^{th} position in the suffix array) direct to w_l and $S[j]$ (i.e. the j^{th} position in the suffix array) direct to v_k. The word sequences $w_l^n = w_l \dots w_n$ and $v_k^m = v_k \dots v_m$ are defined as suffixes of the sentences W and V. Then $S[i] <_{lex} S[j]$ means that w_l^n is ordered lexicographically before v_k^m. Let for example w_l^n be "will be nice". Then v_k^m can be "will be now", as "nice" is ordered before "now", however v_k^m can't be "where it was yesterday" as "will" is ordered *after* "where". Now we can define that $S[i] \leq_{lex} S[j]$ holds, when either $S[i] <_{lex} S[j]$ holds or $S[i]$ and $S[j]$ direct to words w_l and v_k so that $w_l^n = v_k^m$.

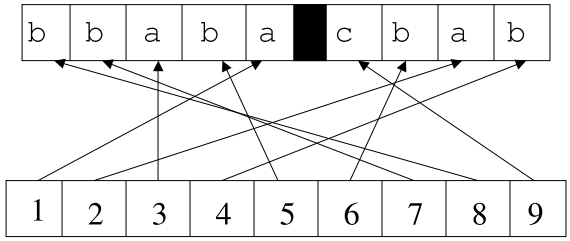

Figure 5.1: Suffix array and corpus with the vocabulary a,b,c consisting of the two sentences "bbaba" and "cbab"

Now assume we want to obtain the count $C(w_1 w_2 \dots w_h)$ for calculating a certain probability estimate. This is done by determining two indices i_{first} and i_{last} so that the following conditions hold:

- $S[i_{first}]$ and $S[i_{last}]$ direct to positions in the corpus where suffixes $w_1 w_2 \dots w_h$ begin.
- For all $j < i_{first}$ holds $S[j] <_{lex,h} S[i_{first}]$.
- For all $k > i_{last}$ holds $S[k] >_{lex,h} S[i_{last}]$.

Then $i_{last} - i_{first} + 1$ equals to the count $C(w_1 w_2 \dots w_h)$. Note that the predicate $<_{lex,h}$ means, that only suffixes of a maximum length h are considered for lexicographic comparison. Figure 5.2 shows an example how a count is determined.
The indices i_{first} and i_{last} can be determined by a binary search in $O(\log N)$ where N is the length of the suffix array.
This suffix array language model implementation is used in our system in order to build language models which combine absolute discounting with linear interpolation as described in equation 3.20. During training the suffix array is constructed and the discounting factors for the models of different order are determined. Probabilities for predictions are estimated at runtime using the suffix array.

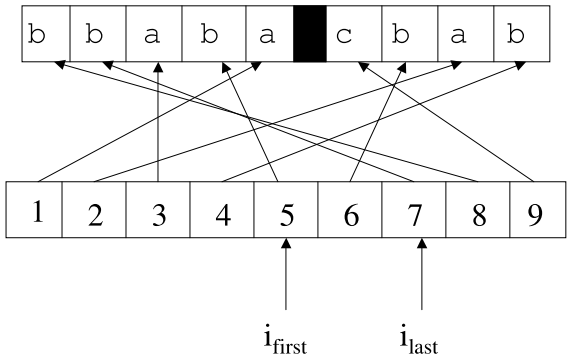

Figure 5.2: Finding the count $C(b, a)$ in a suffix array

5.2 Decoding

In this section, we explain the decoder as implemented in this work. Decoding means to find the best hypothesis for a fluent string among all hypotheses we consider.
Some parts of the code are taken from the SMT toolkit from which we adopted as well the language model implementation. These parts comprise classes for treating texts, vocabularies, strings and arrays efficiently and some basic classes which can be used to build lattices. However, the major part of the code, including all complex algorithms, has been implemented within the context of this work.
As already explained in section 1.4, we construct our search space by generating each possible hypothesis (i.e. each "clean" string) C which can be generated by deletions from a potentially disfluent source string (i.e. "noisy" string) N. For each deletion which is made, a score[1]is assigned to C. All generated strings are stored in a lattice in which we finally search for the best string $\hat{C}$.
Given a string $N = n_1 \dots n_J$ one can represent C uniquely by the alignment sequence which is observed during the creation of C from N. This is because the alignment for all words which do not appear in C are zero and the other alignments represent the position of the particular word in C. In this section and in our implementation we describe the idea of alignment differently: An alignment is zero for a deletion and one for no deletion. The original alignments can be obtained from these "simplified" alignments by counting the number of non-zero alignments preceeding the current alignment. Figure 5.3 shows a string N, an alignment for N and the resulting string C which is uniquely determined by this alignment.

5.2.1 Search space reduction

We explained above that it is sufficient to store the corresponding alignments in order to store all generated hypotheses. This might be done efficiently using a binary tree, as illustrated in figure 5.4.
In this tree each node represents a possible alignment a_j for the word n_j. The incoming edge of the node representing a_j gets assigned a score that this alignment is made. The scores are assigned to the incoming edges and not to the nodes themselves, what facilitates the

[1]We use scores instead of probabilities as we do all our calculations in log-space. This is explained in larger detail in section 5.2.1.

	n_1	n_2	n_3	n_4	n_5	n_6	n_7	n_8	n_9
n_1^J	So,	let	us	let	us	go	for	another	time.
a_1^J	0	0	0	1	1	1	0	1	1
c_1^I				Let	us	go		another	time.
				c_1	c_2	c_3		c_4	c_5

Figure 5.3: A source string string n_1^J, an alignment a_1^J and the corresponding hypothesis c_1^I

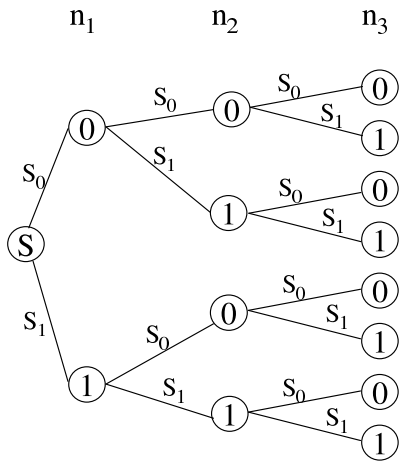

Figure 5.4: Binary search tree to store the alignments corresponding to hypotheses; An edge gets the score S_0, if its end node represents a zero alignment, and the score S_1 otherwise

generation of a lattice out of the tree in order to reduce the search space. In a lattice one node can have multiple incoming edges representing different alignment histories. Since the scores which are assigned depend very often on the alignment histories, assigning one score to each edge is easier than assigning a list of scores to a node where each incoming edge of this node must be distinguished.
In a binary tree the number of nodes grows exponentially with its depth. This property is usually not desirable in terms of memory consumption and time required for searching. Furthermore there are a lot of paths in the binary search tree which represent very similar hypotheses which are only different in a few alignments (figure 5.5). We decided to make use of these similarities in order to reduce the search space.
This reduction of the search space is done by defining some criteria under which two partial paths can be merged. These criteria are chosen in a way that no information gets lost. That means that only merges are performed which do not make the assignment of scores to an edge ambiguous.
Two partial paths can be merged iff[2]:

- the partial paths have the same length

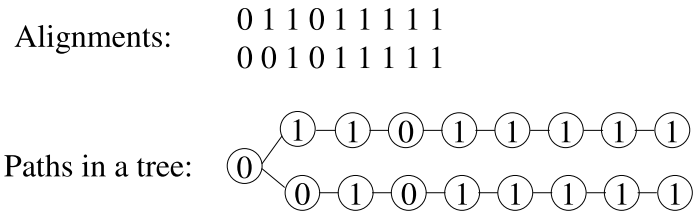

Figure 5.5: Two very similar paths in a binary search tree

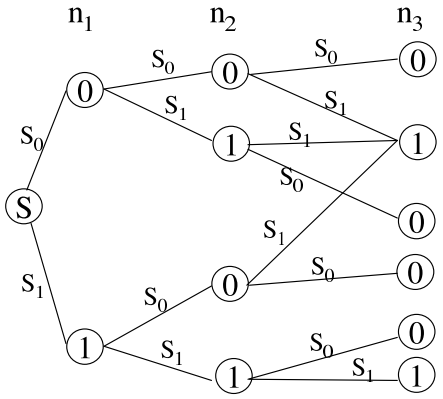

Figure 5.6: Lattice which is generated by applying the merge criteria on a binary search tree of depth three; The edges get the scores S_0 and S_1 assigned as in the binary search tree

- the partial paths have the same number of deletions (i.e. the same number of contiguous zero alignments)
- the length of the current deletion region in both partial paths is the same

Applying these criteria to a binary search tree, we obtain a lattice which is illustrated for a tree with depth three in figure 5.6.
Table 5.1 shows the number of nodes in a lattice compared to the corresponding binary search tree of depth n. While the number of nodes in the binary tree grows exponentially ($\text{depth}(n) = 2^{n+1} - 1$), we only observe a growth for the lattice which is a bit more than quadratic.
The lattice structure which is built for different strings N with equal length is always the same. This is because the structure is only determined by the possible alignments and not by the actual words. Furthermore, always the whole lattice which contains *all possible* partial

[2]One can show that because of these three conditions partial paths are only merged if the length of the current deletion region is zero.

n	1	2	3	4	5	6	7	8
Number of lattice nodes	3	7	13	22	34	50	70	95
Number of binary tree nodes	3	7	15	31	63	127	255	511

Table 5.1: Number of nodes of a lattice and of a binary search tree of depth n.

hypotheses is built. As the search space is small enough after the merges have been applied, no pruning is done. The lattices for two strings of equal length only differ in the scores which are assigned to the different edges.
In our implementation we do not use directly the search criterion presented in equation 3.22, but we take the negative logarithm of this equation. The advantage of doing so is that computationally expensive multiplications become summations in log space. Furthermore we are not confronted with the problem of very small numbers which arises when multiplying a lot of probabilities which are naturally < 1. Hence our new search criterion looks as follows:

$$\hat{C} = \arg\min_c \Big(-\log \left(P(N|C) \cdot P(C) \right) \Big) = \arg\min_c \left(-\log P(N|C) - \log P(C) \right) \tag{5.2}$$

The negative logarithm of the "translation" probability $-\log P(N|C)$ can be rewritten as follows using the maximum approximation from equation 3.25 and equation 3.26:

$$\begin{aligned}
& -\log P(N|C) \\
\approx\ & -\log \left(P(J|I) \cdot P(m|J,C) \cdot \max_{a_1^J} P(n_j, a_j | c_1^I, I, J, m) \right) \\
=\ & -\log \left(P(J|I) \cdot P(m|J,C) \cdot \max_{a_1^J} \prod_{j=1}^{J} P(n_j, a_j | n_1^{j-1}, a_1^{j-1}, c_1^I, I, J, m) \right) \\
=\ & -\log P(J|I) - \log P(m|J, c_1^I) - \max_{a_1^J} \sum_{j=1}^{J} \log P(n_j, a_j | n_1^{j-1}, a_1^{j-1}, c_1^I, I, J, m)
\end{aligned} \tag{5.3}$$

Note that the sentence length probability $P(J|I)$ in equation 5.3 is not considered during search, because we decided to set it uniform as explained in section 3.3.
When working with scores (i.e. negative logarithms of probabilities) the goal of our search is to find the path with the smallest score. The different scores are assigned to the paths of the lattice as follows: During construction of the lattice $-\log P(n_j, a_j | n_1^{j-1}, a_1^{j-1}, c_1^I, I, J, m)$ is assigned to each edge. This score consists of the contributions of the five models introduced in section 3.3.
The language model probability is assigned during search. Whenever a node is visited it gets a language model score assigned for observing the word associated with the node, given the history. For each node a list of different language model probabilities must be stored, as there can be different histories, i.e. different paths which lead to a node as illustrated in figure 5.7. Note that the language model scores are assigned to nodes instead of the incoming edges. This is however only an implementation detail, which does not change the basic ideas of our algorithm.
At the end of the search a number of paths are available, which are arranged in an N-best list. Each of these paths is now assigned a sentence end probability from the language model

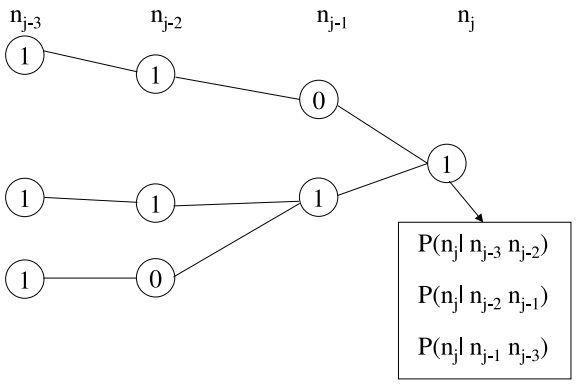

Figure 5.7: Different paths with different histories leading to one node; For this node a list of different language model probabilities must be stored

(P(sentence end|history)) and the score $-\log P(m|J,C)$ for having m deletions in this path. Then the best path (or the N best paths) can be extracted.

5.2.2 Dynamic Programming Search

It still remains to be explained how search is actually done. We use a dynamic programming approach with some similarities to a Viterbi search. In order to be able to explain the search algorithm we define $K_1 \dots K_J$ to be J disjoint subsets of nodes of a lattice. Subset K_j contains all nodes of the lattice which are associated with the word n_j, i.e. which represent the alignment a_j. Furthermore we define that each node in the whole lattice has a unique index. Thus the subsets K_j can be written as follows:

$$
\begin{aligned}
K_0 &= \{k_0\} \\
K_1 &= \{k_1, \dots, k_{|K_1|}\} \\
K_2 &= \{k_{|K_1|+1}, \dots k_{|K1|+|K2|}, \} \\
&\dots
\end{aligned}
$$

Note that subset K_0 only contains the start node of the lattice which is associated with no word but only with a begin-of-sentence marker.
Given these subsets our algorithm can be explained intuitively as follows: The best partial path to node $k_u \in K_j$ is determined by choosing the node $k_v \in K_{j-1}$ as previous node to k_u that the overall score to get from the start node k_0 to k_u is maximized. The overall score is determined by adding the score for the best path to get from the start node to k_v and the score to get from k_v to k_u. The best path from k_0 to k_v has already been determined, following the principle of dynamic programming.
The score which is assigned to a path $k_{j_1}, k_{j_2}, k_{j_3} \dots k_{j_m}$ is simply the sum over the scores for getting from $k_{j_{i-1}}$ to k_{j_i} for all $1 < i \leq m$. Thus one can show easily that the algorithm described above finds a global optimum, i.e. the overall best path.
Now we define the search algorithm formally. Let $S_{K_j}(k_u)$ be the score for the partial path from the start node k_0 to a node k_u in subset K_j. Let furthermore $B_{K_j}(k_u)$ be a back pointer

to the node k_v in K_{j-1} which precedes the node k_u on the best partial path from k_0 to k_u. Finally, let t_{vu} be the score for getting from k_v to k_u. Then we can describe our algorithm inductively:

Initialization:

$$S_{K_0}(k_0) = 0$$
$$B_{K_0}(k_0) = 0$$

Induction:

$$S_{K_j}(k_u) = \min_{k_l \in K_{j-1}} S_{K_{j-1}}(k_l) + t_{lu}$$
$$B_{K_j}(k_u) = \arg \min_{k_l \in K_{j-1}} S_{K_{j-1}}(k_l) + t_{lu}$$

Termination:

$$\text{Best score} = \min_{k_l \in K_J} S_{K_J}(k_l)$$
$$\text{Best end node} = k^* = \arg \min_{k_l \in K_J} B_{K_J}(k_l)$$

Backtracking:

$k_{K_{j-1}} = B(k_{K_j})$
$(k_{K_1} \dots k_{K_J})$ represents the best alignment sequence[3].

5.2.3 Translation Model

In this this section we give some implementation details about the five different models contributing to the overall "translation" model. To make the following explanations easier to understand we describe all models in terms of probabilities instead of scores.

Position of a disfluency in a sentence

The probability for having a deletion at an initial or medial position is simply estimated by using the relative frequencies shown in table 4.8. Thus the two edges leading to the two nodes in K_1 (they represent the alignments for n_1) are assigned the probability for having a deletion or no deletion at an initial position. All other edges are assigned the probability for a deletion or a no deletion at a medial position, according to the alignments their end nodes represent (figure 5.8).

[3]The node k_{K_j} is that node from the subset K_j which belongs to the best alignment sequence.

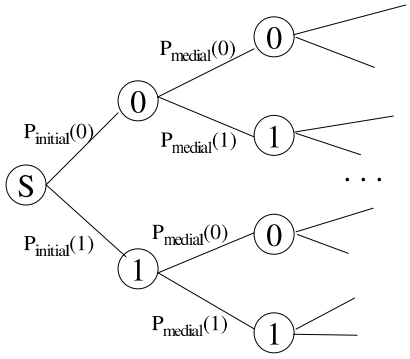

Figure 5.8: Assignment of the probabilities for having deletions at initial or medial positions

Length of the deletion region of a disfluency

The assignment of probabilities for different deletion lengths is a bit more complex. As illustrated in figure 5.9, the model which assigns this probability is only used *within* a deletion region. It does not contribute to the decision where a deletion region starts. In a deletion region each edge leading to a node representing a zero-alignment is assigned the probability $P(L > l)$. Here l is the length of the deletion region in the start node of the edge which is assigned $P(L > l)$. The random variable L is described by the relative frequencies for the lengths of the different deletion regions in the training corpus. Thus its distribution can be computed directly from the distribution of disfluencies over the length of their deletion region as presented in figure 4.3. Edges ending in nodes with a non-zero alignment are assigned the "deletion end" probability $\overline{P(L > l)}$ with the same l as above.

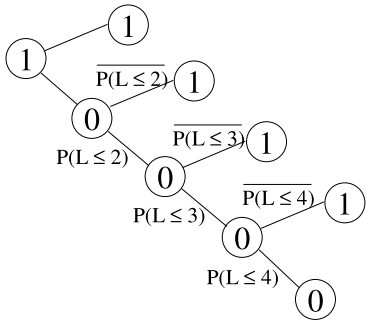

Figure 5.9: Assignment of probabilities for different lengths of deletion regions

Length of the deletion region of a disfluency containing a fragment

The probability for having a deletion at position j given that a fragment occurs at position $j + d$ is estimated by relative frequencies as well. They are obtained by considering the distribution of deletion lengths shown in figure 4.5. The probability is added to the probability

already associated with an edge, when this edge ends in a node representing a zero alignment. Otherwise it is subtracted from the probability already associated with the edge (figure 5.10). When adding or subtracting probabilities we make sure that the overall probability still remains between zero and one.

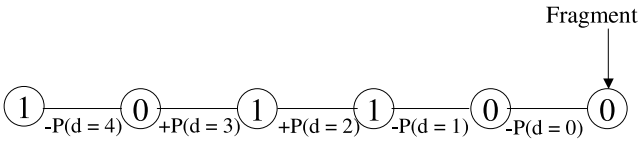

Figure 5.10: Assignment of the probability for a deletion or no deletion in the presence of a fragment

Word context in which a potential deletion occurs

The context model predicts the deletion of the word n_j given the context n_{j-1} and n_{j+1}, thus it predicts the alignment a_j. We assume again that the alignment a_j for a word n_j is either zero or one. The correspondence to the alignments in our translation model is straight forward as explained above. In order to learn the parameters for the model, we extract right and left bigrams and unigrams (section 4.2.6) from the data, together with the information whether the current word is disfluent or not. (The information whether the word n_j we extract from the data is disfluent or not is coded as alignment: $a_j = 0$ means that n_j is disfluent and therefore has to be deleted for correction; $a_j = 1$ means that n_j is fluent) Thus we store the following events together with their counts:

- triples for left bigrams: (n_{j-1}, n_j, a_j)
- triples for right bigrams: (n_j, n_{j+1}, a_j)
- tuples for unigrams: (n_j, a_j)

In order to derive probability estimates, left and right bigrams are combined if they haven been seen often enough during training. Otherwise we back off to unigrams or to even more simple estimates. In order to formulate the probability estimates for the probability $P(a_j|n_{j-1}, n_j, n_{j+1})$ from equation 3.33 we make the following definitions:

$$q_r(a_j, n_{j-1}, n_j, n_{j+1}) = \frac{C(n_j, n_{j+1}, a_j)}{C(n_j, n_{j+1}, a_j) + C(n_{j-1}, n_j, a_j)}$$
$$q_l(a_j, n_{j-1}, n_j, n_{j+1}) = \frac{C(n_{j-1}, n_j, a_j)}{C(n_j, n_{j+1}, a_j) + C(n_{j-1}, n_j, a_j)}$$
$$p_r(a_j, n_j, n_{j+1}) = \frac{C(n_j, n_{j+1}, a_j)}{C(n_j, n_{j+1})}$$
$$p_l(a_j, n_{j-1}, n_j) = \frac{C(n_{j-1}, n_j, a_j)}{C(n_j, n_{j+1})}$$

In the above definition $C(\cdot)$ denotes the count for the different events in the training text. The final probability is then defined as follows:

$$P(a_j|n_{j-1}, n_j, n_{j+1}) =$$
$$\begin{cases} q_r(a_j, n_{j-1}, n_j, n_{j+1}) \cdot p_r(a_j, n_j, n_{j+1}) + & \\ q_l(a_j, n_{j-1}, n_j, n_{j+1}) \cdot p_l(a_j, n_{j-1}, n_j) & \text{if } C(n_j, n_{j+1}) \geq 3 \text{ or } C(n_{j-1}, n_j) \geq 3 \\ p_u(n_j, a_j) & \text{if } C(n_j) \geq 3 \\ a_j & \text{otherwise} \end{cases} \quad (5.4)$$

The assignment of the context model probabilities to the words of the different hypotheses is straight forward. For each edge representing the alignment a_j we know the words n_{j-1}, n_j and n_{j+1} from the source sentence. Thus we simply calculate the estimate for $P(a_j|n_{j-1}, n_j, n_{j+1})$ for each edge according to the definition in equation 5.4.

Deletion history of a potential deletion

Finally we explain the implementation of the binary context model. Remember that this model predicts the alignment a_j for the word n_j given the two previous alignment a_{j-2} and a_{j-1}. The parameters for this model are learned from the data, in the same way as for the context model. Thus we store the following events together with their counts (the information whether a word is disfluent or not is again coded as alignment):

- deletion histories of two words: $(a_{j-2}, a_{j-1}, n_j, a_j)$
- deletion histories of one word: (a_{j-1}, n_j, a_j)
- deletion histories of zero words: (n_j, a_j)

For the calculation of the probability estimates we always use the longest history which has been seen often enough during training. Otherwise we back off to a shorter history. In order to express the estimates for $P(a_j|n_j, a_{j-1}, a_{j-2})$ from equation 3.34, we make the following definitions:

$$p_2(a_j, n_j, a_{j-1}, a_{j-2}) = \frac{C(a_j, n_j, a_{j-1}, a_{j-2})}{C(n_j, a_{j-1}, a_{j-2})}$$
$$p_1(a_j, n_j, a_{j-1}) = \frac{C(a_j, n_j, a_{j-1})}{C(n_j, a_{j-1})}$$
$$p_0(a_j, n_j) = \frac{C(a_j, n_j)}{C(n_j)}$$

Then we can define the probability estimate for the binary context model as follows:

$$P(a_j|n_j, a_{j-1}, a_{j-2}) = \begin{cases} p_2(a_j, n_j, a_{j-1}, a_{j-2}) & \text{if } C(n_j, a_{j-1}, a_{j-2}) \geq 3 \\ p_1(a_j, n_j, a_{j-1}) & \text{if } C(n_j, a_{j-1}) \geq 3 \\ p_0(a_j, n_j) & \text{if } C(n_j) \geq 3 \\ a_j & \text{otherwise} \end{cases} \quad (5.5)$$

The probability estimates for the words of the different hypotheses are calculated using simply equation 5.5. The variables a_{j-2} and a_{j-1} are assigned hypothesized values for the alignments of the two previous words.

Chapter 6

Results

In this chapter the results from experiments with our system are presented. In section 6.1 we explain some measures which we use to evaluate the performance of our system in addition to recall and precision. The results of the experiments on the English Verbmobil corpus are discussed in section 6.2. In section 6.3 some results of experiments on the Mandarin Chinese CallHome corpus are presented. Finally we compare the results we obtained from our system with results reported in some other works about disfluency detection and correction (section 6.4).

6.1 Measures for Evaluating the Disfluency Correction Performance

Recall and precision which are explained in section 2.3.2 are very important to evaluate the performance of a disfluency correction system. Furthermore we present sometimes the absolute numbers of hits and false positives which occur in a test set. In our terminology hits are those disfluencies where the deletion region is deleted completely by our system. False positives occur when only words which do not belong to the deletion region of a disfluency are deleted.
In addition to these figures we consider the cases in which our system deletes only parts of the deletion region of a disfluency or where a deletion region is overlapped by a deletion of our correction system (figure 6.1). In the latter case, there is a number of possibilities how a deletion region of a disfluency can be overlapped by a correction system deletion. The actual deletion region can be covered partly by the correction system deletion, so that only at one of its boundaries non-disfluent words are deleted or it can be covered completely. In this case non-disfluent words can be deleted either only at one boundary or at both boundaries of the deletion region. Nevertheless from the point of view of the correction system only one deletion of a contiguous word sequence is done. Therefore we classify this event as deletion overlapped by the correction system, and not as two or three deletions which may consist of hits, false positives and partly deletions of deletion regions.
However, partly deleted and overlapped deletion regions usually don't play a very important role in our results and therefore are not presented for all system aspects we discuss. Nevertheless it is important to be aware of the fact, that these two phenomena exist.

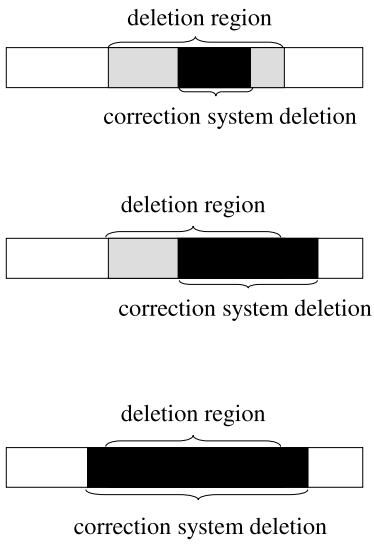

Figure 6.1: Deletion regions deleted partly (first picture) or overlapped by deletions of the correction system (two possibilities in the second and third picture)

6.2 Results for the English Verbmobil Corpus

This section deals with the experiments we conducted on the English VERBMOBIL corpus. First we describe briefly the setup for our experiments. Then the results of the experiments are presented. To conclude this section we illustrate the performance of our system with some examples.
In section 3.3 we showed that a number of different models are used to assign the overall score to a hypothesis. We try to improve the performance of our system by increasing or decreasing the weight with which these models contribute to the overall score. However, given the time constraints of this work, we didn't design an algorithm for the automatic adjustment of the optimal weights, rather we adjust them manually. Since there are at least seven different parameters to adjust, the parameter combination we obtain by manual adjustments is possibly suboptimal.
The way our results are presented in the following reflects the attempts to find an optimal parameter combination. First we present results for a baseline system, where only the weight of the language model score is varied. Then we show the effect of varying other parameters and finally present this parameter combination which led to the best results.

6.2.1 Setup

As already briefly explained in section 4.1.3, we divided the 127 dialogs which are available from the English VERBMOBIL corpus in 10 disjoint test sets. Each test set consists of 10% of the whole number of sentences in the dialogs. The remaining 90% are used for training. Thus we have 10 different partitions of our corpus which were created by choosing randomly 10 different permutations of the 16583 sentences and splitting them into a test and a training set. Our approach for disfluency correction considers sentence by sentence. No information about the turn or the dialog in which the sentence occurs is necessary. Hence we even broke

up turns, so that different sentences of one turn may occur in different test sets.
Each experiment was run on all 10 test sets and the results which are reported below represent averages from the results of all 10 test sets. The reason to do this is to avoid biases in the results which may arise because of certain characteristics of a test set, even if it is chosen randomly. When appropriate the range of these results is reported.
The average number of disfluencies which occur in the 10 test sets is 1102.7 which is equal to the total number of disfluencies in the corpus divided by 10. This number is the same for all experiments with the VERBMOBIL corpus and it serves as orientation when we report the average number of hits, false positives etc. When we report total numbers of disfluencies of a certain type or with deletion regions of a certain length, the average number is again equal to the total number of these events in the corpus divided by 10.

6.2.2 Baseline System

In the following we present the baseline system where only the weight of the language model score is varied. In order to introduce a weight factor for the language model score we modify our search criterion from equation 5.2 as follows:

$$\hat{C} = \arg\min_{C} \left(-\log P(N|C) - \lambda_{lm} \cdot \log P(C) \right) \tag{6.1}$$

This baseline system uses all models introduced in section 3.3 except the binary context model[1]. We varied λ_{lm} in steps of 0.1 from 0.0 to 1.0. The results are illustrated in figure 6.2. The light gray bars show the number of false positives and the dark grey bars the number of hits for different language model weights.
One can see that the number of hits increases up to $\lambda_{lm} = 0.4$ and decreases slowly for higher language model weights. The number of false positives increases rapidly with increasing language model weight. Good results are characterized by a high number of hits and a low number of false positives. Here this is the case for $\lambda_{lm} = 0.2$ where we have 837 hits (compared to a maximum of 853.0 for $\lambda_{lm} = 0.4$) and 140.7 false positives (compared to a minimum of 67.2 for $\lambda_{lm} = 0.0$). That corresponds to a recall of 75.7% and a precision of 85.6%. These figures will serve as baseline results which are compared to the results of the experiments we present in the following.
A reason for the number of false positives increasing rapidly with increasing language model weight may be that the language model favors the deletion of events which were not or only rarely seen during training. This effect becomes visible for higher language model weights, as the language model then dominates the other scores.
The figures for deletion regions which are deleted partly and which are overlapped by deletions of the correction system are presented in figure 6.3. One can observe that the number of partly deleted deletion regions decreases slowly with increasing language model weight. The number of deletion regions which are overlapped by deletions of the system increases with increasing language model weight. The latter observation may be due to the same reason as the increasing number of false positives for high language model weights.
Since the language model is trained on fluent speech, disfluencies are events which have not been seen during training of the language model. Therefore the number of successfully deleted deletion regions (hits) increases up to $\lambda_{lm} = 0.4$. However one would expect a further increase of the number of hits for higher language model weights. One possible explanation

[1] The binary context model was not originally part of our system. We implemented it later as an attempt to improve the system performance.

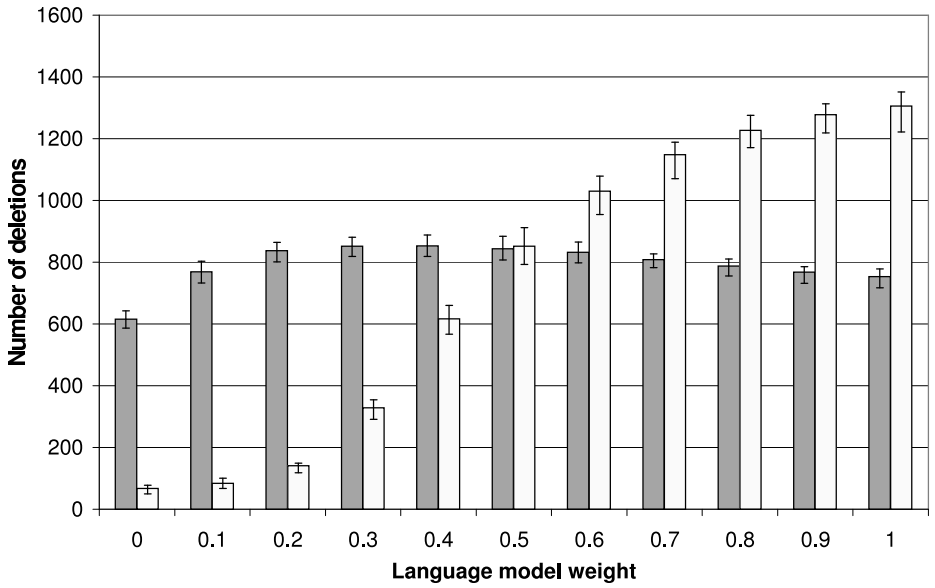

Figure 6.2: Results for different language model weights; The error bars indicate the deviation among the different test sets

that this expectation is not true can be the increase of deletion regions which are overlapped by correction system deletions. Deletion regions which are deleted completely for lower language model weights are now overlapped at one or both boundaries by correction system deletions.
When we report recall and precision so far we only considered complete deletions of deletion regions as hits and deletions of word sequences which to not belong to a deletion region false positives. However one could argue that deletion regions which are not deleted correctly (i.e. they are deleted partly or overlapped by cleaner deletions) degrade the quality of the text. Hence one would count them as false positives. On the other hand one could say that they help at least to *detect* disfluencies and count them as hits. Table 6.1 shows how recall and precision change in these two cases for $\lambda_{lm} = 0.2$.

	Recall	Precision
ddp and dod not considered at all	75.7%	85.6%
ddp and dod as false positives	75.7%	81.0%
ddp and dod as hits	80.9%	86.4%

Table 6.1: Recall and precision when deletion regions which are deleted partly (ddp) or which are overlapped by correction system deletions (dod) are counted to hits or false positives respectively ($\lambda_{lm} = 0.2$)

In absolute numbers we have 38.3 deletion regions which are deleted partly and 16.8 deletion regions which are overlapped by deletions of the correction system. Although these figures seem low compared to the number of hits, they change recall and precision, when they are seen as hits or false positives respectively.
We decided however to count those disfluencies which are not corrected completely by the

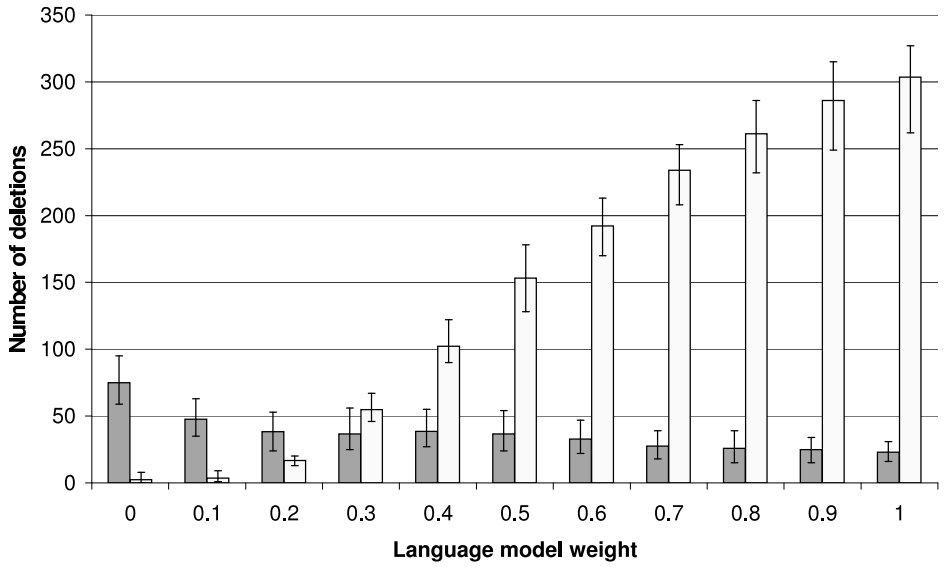

Figure 6.3: Deletion regions deleted partly (gray bars) and deletions overlapped by deletions of the correction system (white bars); The error bars show the deviation among the different test sets

correction system neither to the hits nor to the false positives. We rather report these figures explicitely in the following if they differ much from those which are shown in figure 6.3.
Table 6.2 shows the correction results for the different types of disfluencies for $\lambda_{lm} = 0.2$. In the last line the percentage of hits of the total number of disfluencies of a particular type (i.e. the recall for this type) is shown.

	REP	FS	ET	DM	UH	IN
Total disfluencies	230.3	120.9	15.5	124.8	577.6	33.6
Hits	90.4	28.3	13.5	111.5	559.7	33.6
Percentage of total disfluencies	39.3%	23.4%	87.1%	89.3%	96.9%	100.0%

Table 6.2: Correction results for different types of disfluencies ($\lambda_{lm} = 0.2$)

The figures in table 6.2 show that the complex disfluency types repetition/correction and false start are much more difficult to correct for our system than the other types. Taking editing terms, filler words (discourse markers/filled pauses) and interjections together, we note that 95.6% of the total disfluencies of these types are among the hits. The poor performance for the complex disfluency types could possibly be increased by taking word correspondences between reparandum and repair into account.
Table 6.3 shows the length of deletion regions for the hits compared to the length of deletion regions of disfluencies which really occurred.
One can see from the figures in table 6.3 that the baseline system only produces hits for disfluencies with deletion regions having a maximal length of four words. Taking only these disfluencies into account, the system deletes successfully 77.6% of them. Since disfluencies with deletion regions longer than four words make only 2.2% of all occurrences, we decided to focus rather on other issues than improving the system for longer deletion regions.

Length of deletion region	1	2	3	4	5	6	7	8	9	10	11
Total disfluencies	840.2	174.7	44.3	19.3	11.1	7.3	2.6	1.9	0.9	0.3	0.1
Hits	748.6	84.1	3.9	0.4							

Table 6.3: Lengths of deletion regions for hits compared to all deletion regions ($\lambda_{lm} = 0.2$)

For the baseline system and for all other experiments we conducted on the VERBMOBIL corpus, we conclude that a language model weight of $\lambda_{lm} = 0.2$ seems to be the best choice in order to keep false positives low and hits high at the same time. Hence all results in the following are reported for this language model weight.

6.2.3 Effect of the language model

The language model we use for a particular test set is build using the training corpus which belongs to this test set. Construction of the language model requires the division of the training corpus into a language model training corpus and a language model cross validation corpus which is used to determine the discounting factors for the models of different order. Each language model training corpus comprises approximately 82000 words. We obtained perplexities of approximately 32 on the cross validation corpora.
One attempt to improve the performance of our system was to increase the size of the language model corpus. This was done in two different ways.
First we used some data from the VERBMOBIL 1 project which also consists of spontaneously spoken dialogs. Disfluencies were removed from these dialogs using the annotations which were obtained in the same way as we obtained the annotations the other VERBMOBIL dialogs. Then we enlarged the existing cross validation and training corpora by adding disfluency free text from the VERBMOBIL 1 dialogs. The new language model training corpora comprise approximately 160000 words. The perplexities on the new cross validation corpora remain at about 32.
Experiments with the new language models show, that $\lambda_{lm} = 0.2$ is still the best weight factor for the language model score. The numbers of hits and false positives almost don't change. That indicates that even increasing the size of the language model corpus does not improve the ability of the language model to predict rather fluent than disfluent sentences.
Another attempt to improve the performance of the language model component was to take a very large language model corpus based on a formal English text. We took 38 million words of the Wall Street Journal corpus and built a language model with a perplexity of 153.11 on the cross validation corpus. Using the language model we found the recall and precision decrease significantly. However a few disfluencies with deletion regions longer than 4 words could be corrected completely now.

6.2.4 Effect of models for position of a disfluency in a sentence, length of deletion regions and number of deletions per sentence

In the following we analyze the effect of the following for models for our system:

1. Position of a disfluency in a sentence
2. Length of the deletion region of a disfluency

3. Number of deletions per sentence

Models 1 and 2 are are modeled by the probabilities P_1 and P_2 which are part of the overall translation probability (section 3.3.2). The number of deletions per sentence (model 3) is modeled by the probability $P(m|J,C)$ (section 3.3.1).
Remember that the baseline system uses the three models we analyze here. Thus we present in table 6.4 how results change, when one or all models are *not* used.

	Hits	False Positives	Recall	Precision
Baseline (includes models 1,2,3)	837	140.7	75.7%	85.6%
Only models 1 and 3	844.9	146.6	76.4%	85.2%
Only models 1 and 2	856.3	165.5	77.4%	83.8%
Only models 2 and 3	826.9	119.5	74.8%	87.3%
None of the models 1, 2 and 3	833.1	160.1	75.3%	83.8%

Table 6.4: Results for different combinations of the models 1, 2 and 3 compared to the baseline system (which uses models 1, 2 and 3); λ_{lm} is set to 0.2

Comparison of the first and the last line of table 6.4 shows that the baseline system performs better than a system where none of the models 1, 2 and 3 is used.
However one can see that by omitting the models for the length of deletion regions of disfluencies and for the number of disfluencies in a sentence (second and third row of table 6.4), it is possible to improve the recall compared to the baseline system. However an improvement of recall means in both cases a more or less strong decrease of precision. A converse trend can be observed when omitting the model for the position of disfluencies (fourth row of table 6.4): In this case recall decreases but precision increases.
The number of disfluencies where deletion regions are only partly deleted is lower for the system where none of the models 1, 2 and 3 are used. It decreases from 38.3 to 26.6. The number of disfluencies where deletion regions are overlapped by the correction system increases however a lot from 16.8 for the baseline system to 66.8 for the system which does not use the models 1, 2 and 3. Especially an increase of the latter events is undesirable. When deletions of the correction system overlap deletion regions of a disfluency it is possible that important content words of the sentence are deleted.
One can conclude from the results above that all three models together seem to be helpful for disfluency correction. However either recall or precision can be improved when certain models are not used.
When the model for the number of deletions per sentence is not considered recall increases from 75.7% for the baseline system to 77.4% (1.7% difference). However the percentage of complex disfluencies (repetitions/corrections and false starts) which are corrected completely (i.e. the recall for these disfluencies) increases from 33.8% to 36.8% which is as difference of 2.8%. In table 4.5 we showed that complex disfluencies tend to occur more often per sentence than other disfluencies. This might explain the stronger increase of recall for complex disfluencies when no model is used which predicts the number of deletions per sentence. As in most disfluent sentences only one deletion occurs (table 4.5) this model keeps the number of deletions per sentence low which is not appropriate for complex disfluencies.
Another interesting finding can be made when the model for the position of a disfluency in a sentence is not considered. In this case the system makes very long deletions and it can

correct some disfluencies involving deletion regions which are longer than 4 words. This is surprising as one would expect that the system would make longer deletions if no model restricting the length of a deletion region was used. This is however not the case. A reason for this behavior might be that the influence of other models (e.g. the language model) which favor long deletions becomes stronger when initial sentence positions are not favored anymore as potential deletion sites.
Although improvements for complex disfluencies or disfluencies involving longer deletion regions can be achieved, when the models for the number or the position of disfluencies in a sentence are not used, we decided to use both models in the following analysis. The reason for this is the overall improvement in recall and precision which could be achieved by using these models (table 6.4).

6.2.5 Effect of fragments

In order to examine the effect of the model for deletion regions containing a fragment at the end, we modify the probability which combines the models contributing to the overall translation model (equation 3.35) by introducing a factor λ_f:

$$P_{alignment} = \frac{\left(\sum_{u=1,u\neq 3}^{5} P_u\right) \pm \lambda_f \cdot P_3}{\sum_{u=1,u\neq 3}^{5} \alpha(P_u)} \tag{6.2}$$

We use here the same notation as in section 3.3. $P_{alignment}$ denotes the overall alignment probability for an arbitrary word. For simplicity we skip the argument (0 or i) here. The probability P_3 is added or subtracted according to the alignment (deletion or no deletion) which is made. The factor λ_f scales the fragment probability. We conducted experiments with $\lambda_f = 0$, $\lambda_f = 2$ and $\lambda_f = 5$. For the baseline system we have $\lambda_f = 1$. Table 6.5 shows the results for the different values of λ_f.

	Hits	False Positives	Recall	Precision
$\lambda_f = 1$ (Baseline system)	837	140.7	75.7%	85.6%
$\lambda_f = 0$	834.4	131.9	75.5%	86.3%
$\lambda_f = 2$	827.2	162.4	74.8%	83.6%
$\lambda_f = 5$	808.5	168.8	73.2%	82.7%

Table 6.5: Results for different values of λ_f ($\lambda_{lm} = 0.2$)

Comparison of the baseline system ($\lambda_f = 1$) with the system where no fragments are used ($\lambda_f = 0$) reveals that the fragment model does not contribute to much to the correction of disfluencies. There is a slight increase of recall when the fragment model is used, however precision decreases. The decrease of precision when the fragment model is used can be explained when considering those fragments which do not occur at the end of a deletion region. The fragment model favors deletions in front of those fragments what increases the number of false positives our system produces.
For $\lambda_f > 1$ precision *and* recall become worse. For $\lambda_f = 5$ we note however that some disfluencies involving deletion regions longer than four words could be corrected successfully.

6.2.6 Effect of binary context model

In order to examine the effect of the binary context model on the results of our system, we conducted experiments with and without binary context model for language model weights between 0.0 and 1.0. The system without the binary context model corresponds to the baseline system.
Figure 6.4 shows recall and precision for the two systems. The recall of the system using the binary context model is above the recall of the baseline system up to $\lambda_{lm} = 0.3$ and falls below the baseline system for higher language model weights. The precision of the system using the binary context model is always below the precision of the baseline system. For reasonably good values of recall (above 70%), one can find no language model weight where the gain of recall compared to the baseline system can compensate the strong strong loss in precision.
A reason that the binary context model does not improve the results may be that the information about the two previous deletions and the current word is to coarse in order to predict the deletion of the current word.
Table 6.6 compares the correction results for repetitions/corrections and false starts between the baseline system and the system which uses the binary context model. Despite the improvement in performance for complex disfluencies which can be seen from the table, we don't consider the binary context model in the following analysis, as our goal is to improve the overall performance of our system in terms of recall and precision.

	REP	FS
Total DFs	230.3	120.9
Baseline System	90.4 (39.3%)	28.3 (23.4%)
System with binary context model	103.2 (44.8%)	32.2 (26.6%)

Table 6.6: Correction results for complex disfluencies of the baseline system and the system using the binary context model; λ_{lm} is set to 0.2; In parentheses the recall for the particular types is shown

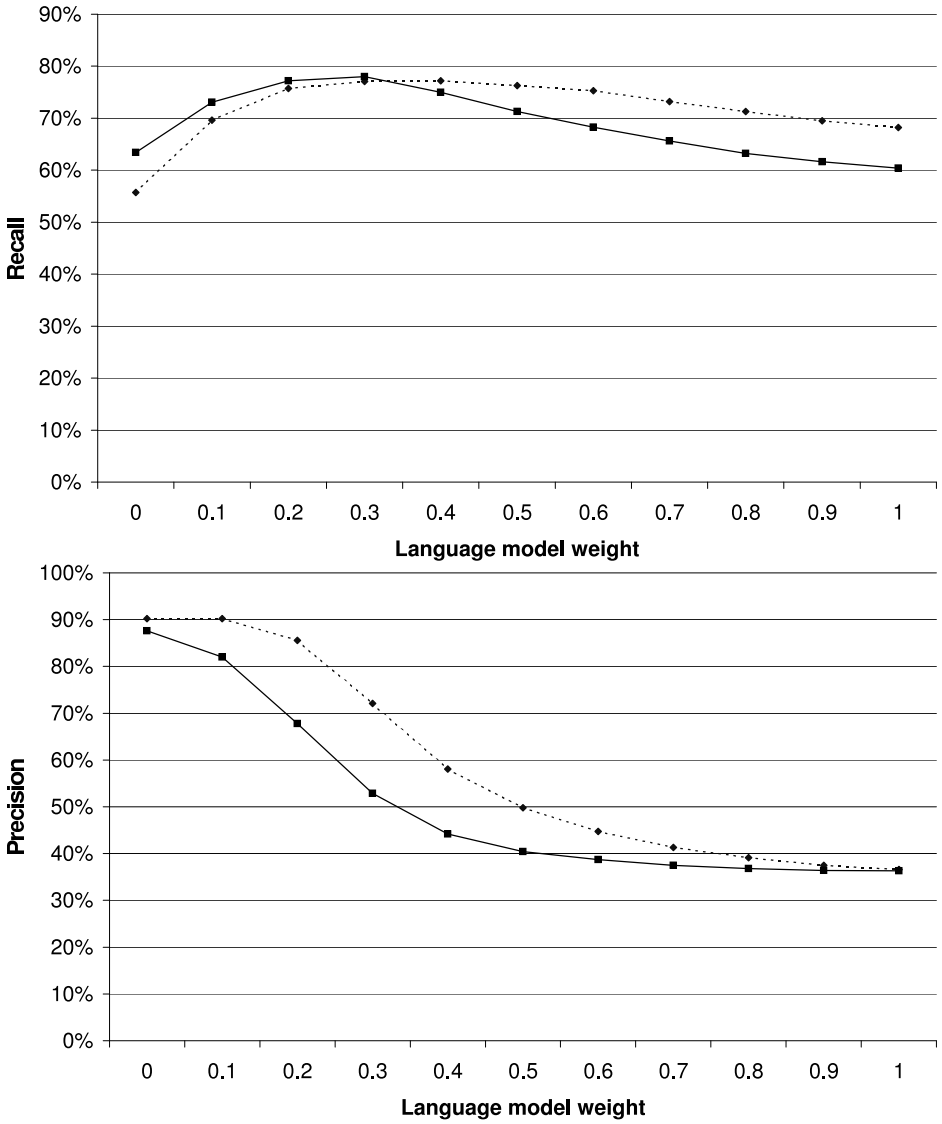

Figure 6.4: Recall and precision of the system with binary context model (solid line) compared to the baseline system (dashed line) for different language model weights

6.2.7 Effect of context model

The effect of the context model on the results of our system is examined by modifying again the probability from equation (3.35). A weight factor for the context model probability P_5 is introduced:

$$P_{alignment} = \frac{P_1 + P_2 + P_4 + \lambda_{context} \cdot P_5 \pm \lambda_f P_3}{\alpha(P_1) + \alpha(P2) + \alpha(P_4) + \lambda_{context} \cdot \alpha(P_5)} \tag{6.3}$$

The factor λ_f is set to one during the experiments with the context model.
In our experiments we varied $\lambda_{context}$ between zero and 15 in steps of one. A first finding is that the context model improves the results of our system significantly. Table 6.7 compares the results for the baseline system ($\lambda_{context} = 1$) with a system using no context model ($\lambda_{context} = 0$).

	Hits	False Positives	Recall	Precision
Baseline system ($\lambda_{context} = 1$)	837	140.7	75.5%	85.6%
System using no context model	699.4	437	63.3%	61.5%

Table 6.7: Results for a system using no context model compared to the baseline system ($\lambda_{context} = 1$); λ_{lm} is set to 0.2

The strong effect of the context model can be explained when considering filler words and short repetitions/corrections. Filler words can be usually identified by the word itself or by the local context. Consider for example the sentences "This is well done." and "Alright, well, this is a good idea." Because of the right context "done" in the first sentence the context model predicts that a deletion of "well" is very unlikely. In the second sentence the prediction of a deletion of "well" is more likely because of the left context "alright". For repetitions like "the the" or corrections like "I we" predictions of the context model are helpful as well.
We can observe a strong increase for the number of hits among repetitions/corrections and filler words for the system which uses the context model compared to the system using no context model. An interesting finding is however that the system using no context model corrects 40.5 false starts while the baseline system can only correct 28.3 false starts. That may indicate that false starts cannot be predicted by taking only the local context into account.
Finally we note that the number of disfluencies where deletion regions are partly deleted or overlapped by deletions of the correction system is higher for the system using no context model than for the baseline system. Thus one can conclude that the context model is very helpful for disfluency correction.
In section 4.2 we mentioned already our attempt to make the predictions of the context model more reliable by using wordclasses. The wordclasses for the context model are used as follows: Let e_k be the word equivalence class for the word n_k. Then the word class based context model predicts the deletion of the word n_j which belongs to the class e_j given the classes e_{j-1} and e_{j+1} of the words surrounding the word n_j. We made experiments with 500 and 50 wordclasses which were constructed using only the training corpora. The results however became worse compared to the system using no word classes. A reason for this might be that the corpora on which the classes were constructed are too small to obtain equivalence classes which are useful for predictions.
For context model weights greater than one we note that the number of hits only improves slightly. However for the number of false positives we observe a strong decrease for increasing

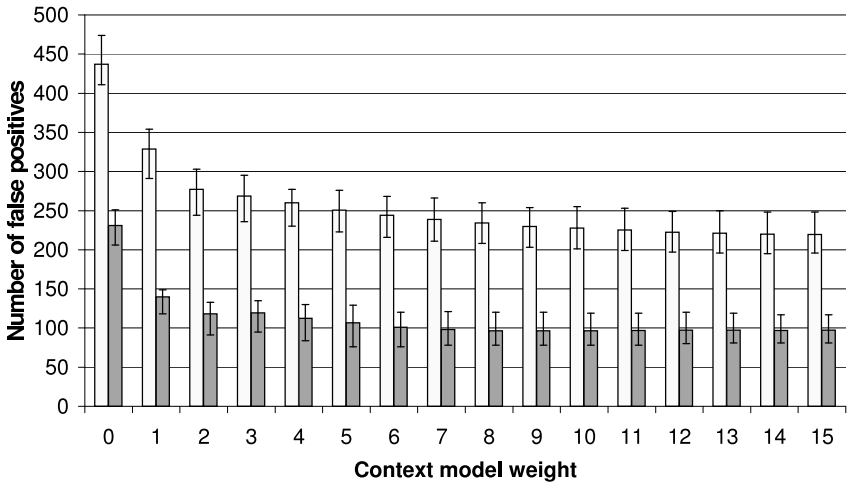

Figure 6.5: False positives for increasing context model weight for $\lambda_{lm} = 0.2$ (white bars) and $\lambda_{lm} = 0.3$ (grey bars); The error bars show the deviation among the different test sets.

context model weight. Figure 6.5 shows the number of false positives for $\lambda_{lm} = 0.2$ and $\lambda_{lm} = 0.3$. For the higher language model weight, the decrease of false positives is even stronger. The best results for the language model weight $\lambda_{lm} = 0.2$ are obtained for a context model weight $\lambda_{context} = 10$.

6.2.8 Best system

From the different experiments presented above, we conclude that especially the change of two parameters improves recall and precision of our system compared to the baseline system: A large value of the context model weight results in a higher precision. A low value for the fragment model weight makes precision and recall go up even a bit more. Hence the parameter settings which lead to the the best system are $\lambda_{lm} = 0.2$, $\lambda_f = 0$, $\lambda_{context} = 10$. Furthermore the binary context model is not used, the models for position of a disfluencies, number of disfluencies per sentences and length of the deletion region of a disfluency are used. Note that these settings might still be suboptimal, as this parameter combination is determined manually and we did not conduct experiments with each possible parameter combination. Table 6.8 illustrates for $\lambda_{lm} = 0.2$ the different improvements ($\lambda_f = 0$ and $\lambda_{context} = 10$) which lead to the best system.

	Hits	False Positives	Recall	Precision	Δ Recall	Δ Precision
Baseline system	837	140.7	75.7%	85.6%	-	-
System without fragments ($\lambda_f = 0$)	834.4	131.9	75.5%	86.3%	-0.3%	0.8%
Best system ($\lambda_f = 0, \lambda_{context} = 10$)	853.3	92.4	77.2%	90.2%	2.3%	4.5%

Table 6.8: Different improvements of the baseline system which lead to the best system; Δ Recall and Δ Precision are relative improvements of precision and recall, each time compared to the system in the line above; λ_{lm} is set to 0.2

6.2.9 Examples

Finally we illustrate the performance of our system with some examples. First we present a number of sentences for which our system produces output sentences in which disfluencies are corrected as desired. Then we show some other cases where the output of our system contains false positives. Sometimes these deletions are tolerable because they do not change the meaning of the sentence. In other cases the meaning of the sentence is changed when fluent words are deleted. In these cases the output of the system becomes mostly ungrammatical and even less well formed than the input. Words with "=" at the end represent word fragments.

"Good" Sentences:

Original:	boy I sure wish our schedule w= we need to to plan ahead for these things so we can have our schedules a couple of weeks ahead of time clear
System Output:	we need to plan ahead for these things so we can have our schedules a couple of weeks ahead of time clear

Original:	I I the the the travel agent might be able to help you out with that too
System output:	the travel agent might be able to help you out with that too

Original:	so we could we want to try for a an arrival Friday morning I think
System Output:	we want to try for an arrival Friday morning I think

Original:	you know so we have to find a space that there is three days yeah
System Output:	so we have to find a space that there is three days

"Bad" sentences:

Original sentence:	yeah it doesn't really matter much to me
System output:	it doesn't really matter to me

Original sentence	nineteenth of this month
System output:	of this month

Original sentence:	nothing really got my attention
System output	got my attention

6.3 Results for the Mandarin Chinese CallHome Corpus

In this section we describe the results from the experiments we conducted on the Mandarin Chinese CallHome corpus. In order to find the best parameter settings we used the experience gained during the experiments with the English Verbmobil corpus and varied the language model weight and the context model weight and examined the effect of omitting the models the position of a disfluency in a sentence, the length of deletion regions and the number of deletions per sentence.

6.3.1 Baseline system

Figure 6.6 shows the results of the experiments with a baseline system on the Mandarin CallHome corpus. This system works with the same parameter settings as the baseline system we used for the English VERBMOBIL corpus. Only the model for the length of deletion regions with fragments at the end is not included as no information about fragments is provided in our transcriptions.

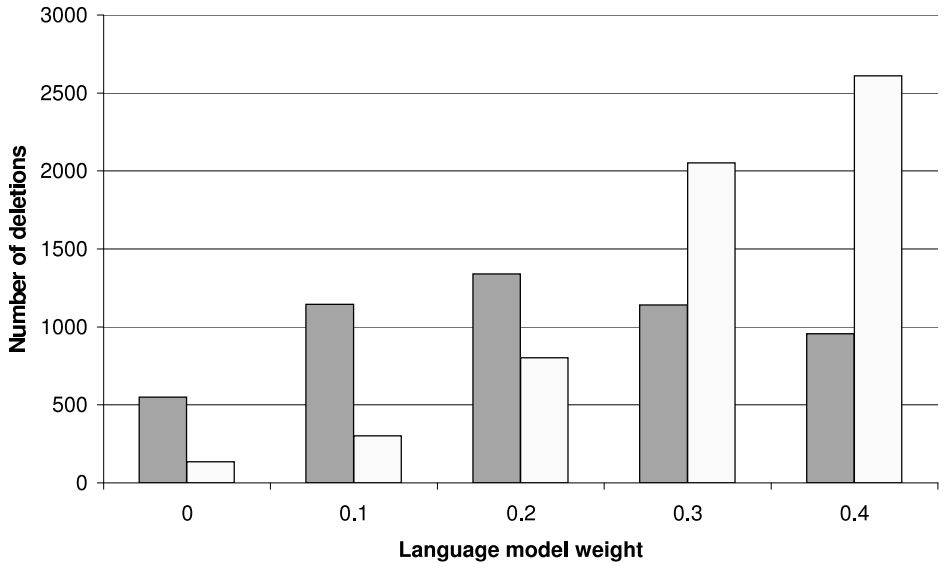

Figure 6.6: Hits (grey bars) and false positives (white bars) for disfluency correction with the baseline system on the Mandarin CallHome corpus

The total number of disfluencies in the Mandarin test set is 3008. The maximum number of disfluencies our system is able to correct is 1340 (for $\lambda_{lm} = 0.2$). This corresponds to a recall of 44.5% which is much worse than the recall achieved for the English VERBMOBIL corpus with equivalent parameter settings. The best language model weight for the Mandarin CallHome corpus is however $\lambda_{lm} = 0.1$, since the number of hits is not too much lower here than for $\lambda_{lm} = 0.2$ and the number of false positives is still low. Table 6.9 shows the results for the best language model weight for the baseline system compared to the language model weights where the most hits ($\lambda_{lm} = 0.2$) and the least false positives ($\lambda_{lm} = 0.0$) occurred. In the last two columns the results for English VERBMOBIL corpus with the same parameter settings are displayed for comparison. Remember the we determined $\lambda_{lm} = 0.2$ to be the best language model weight for the English VERBMOBIL corpus, thus for both corpora the optimal language model weight is in a similar region.
The figures for deletion regions which are deleted partly or which are overlapped by deletions of the correction system are compared to the corresponding figures for the English VERBMOBIL corpus in table 6.10. We note that the number of disfluencies with partly deleted deletion regions is high compared to the English VERBMOBIL corpus. The number of deletion regions which are overlapped by correction system deletions is in the same region for both corpora.
Table 6.11 compares which types of disfluencies are corrected completely for the two corpora. For the English VERBMOBIL corpus we display the figures for discourse markers/filled pauses in the column "FP". The figures for editing terms and interjections which are only considered in the English VERBMOBIL corpus are not shown in this table. We note from the table that

	Mandarin CallHome				English VERBMOBIL	
	Hits	False Positives	Recall	Precision	Recall	Precision
$\lambda_{lm} = 0.1$	1145	302	38.1%	79.1%	69.6%	66.7%
$\lambda_{lm} = 0.0$	549	135	18.3%	80.2%	55.7%	90.2%
$\lambda_{lm} = 0.2$	1340	803	44.5%	62.5%	75.7%	86.5%

Table 6.9: Results for different language model weights for the Mandarin CallHome corpus compared to the English VERBMOBIL corpus

	Mandarin CallHome	English VERBMOBIL
ddp	247	38.3
dod	19	16.8

Table 6.10: Deletion regions deleted partly (ddp) or overlapped by correction system deletions (dod) for the Mandarin CallHome corpus ($\lambda_{lm} = 0.1$) compared to the English VERBMOBIL corpus ($\lambda_{lm} = 0.2$)

the correction of complex disfluencies (repetitions/corrections and false starts) seems to be the hardest task for both corpora.

	REP		FS		FP	
	MCC	EVM	MCC	EVM	MCC	EVM
Total	1326	230.3	13	120.9	1669	702.4
Hits	246	90.4	1	28.3	898	671.2
Recall	18.6%	39.3%	7.7%	24.3%	53.8%	95.6%

Table 6.11: Hits grouped by disfluency types for the Mandarin CallHome (MCC) corpus ($\lambda_{lm} = 0.1$) and the English VERBMOBIL (EVM) corpus ($\lambda_{lm} = 0.2$)

6.3.2 Effect of the context model

The influence of the context model weight on the results for the Mandarin CallHome corpus in shown in figure 6.7 for $\lambda_{lm} = 0.1$. We varied the parameter $\lambda_{context}$ from zero to ten in steps of one. Recall (dashed line) increases strongly up to $\lambda_{lm} = 2$. Then the increase of the recall becomes weaker and finally there is no growth anymore. Precision (solid line) increases only up to $\lambda_{lm} = 1$ and decreases slowly for higher context model weights. Therefore possible good settings for $\lambda_{context}$ are four, five or six, since recall is already high in this region and precision is not yet too low.
Table 6.12 compares the results between the Mandarin CallHome corpus and the English VERBMOBIL for $\lambda_{context} = 1$ (baseline system), $\lambda_{context} = 5$ (best value for the Mandarin CallHome corpus) and $\lambda_{context} = 10$ (best value for the English VERBMOBIL corpus).
We note from table 6.12 that the recall rate shows the same trends for both corpora. It increases first (more or less strongly) and stays then almost at an equal level. For precision we note that the maximum for the Mandarin CallHome corpus is already reached at $\lambda_{lm} = 1$, compared to the English VERBMOBIL corpus where the maximum is at $\lambda_{lm} = 10$. One

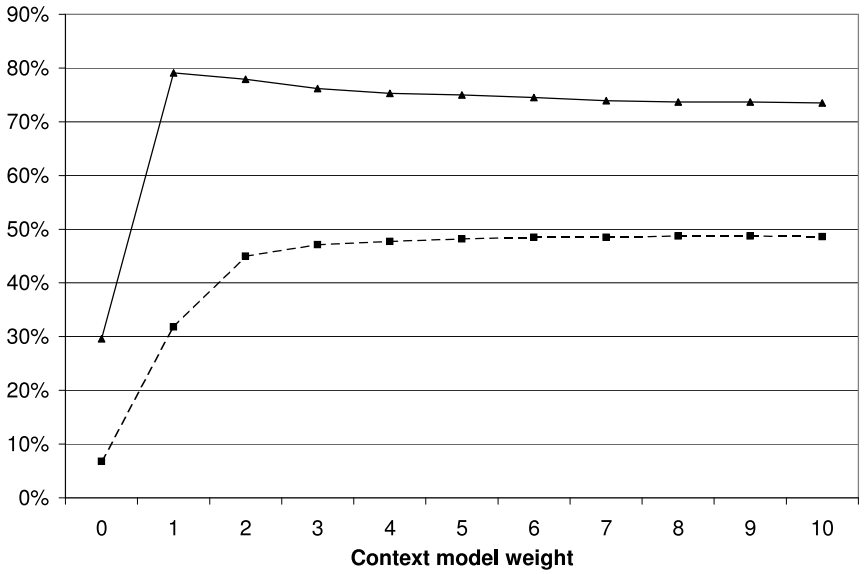

Figure 6.7: Recall (dashed line) and precision (solid line) for different context model weights for the Mandarin CallHome corpus. ($\lambda_{lm} = 0.1$)

	Mandarin CallHome				English VERBMOBIL	
	Hits	False Positives	Recall	Precision	Recall	Precision
$\lambda_{context} = 1$	1145	302	38.1%	79.1%	75.7%	86.5%
$\lambda_{context} = 5$	1451	483	48.2%	75.0%	77.2%	88.9%
$\lambda_{context} = 10$	1463	528	48.6%	73.4%	77.1%	89.8%

Table 6.12: Results for different values for $\lambda_{context}$ for the Mandarin CallHome corpus ($\lambda_{lm} = 0.1$) and the English VERBMOBIL corpus ($\lambda_{lm} = 0.2$)

reason that the maximum for the Mandarin CallHome corpus is reached earlier than for the English VERBMOBIL corpus might be that we use a lower language model weight for the Mandarin CallHome corpus. Thus the influence of the language model is reduced too much already for lower context model weights. Another reason might be that the predictions of the context model for the Mandarin CallHome corpus are more unreliable than for the English VERBMOBIL corpus (due to sparse data). Thus a negative influence of the context model for the Mandarin CallHome corpus can be seen already for lower context model weights.

6.3.3 Best system

We found that the models for the length of the deletion region of a disfluency and for the position of a disfluency in a sentence improve the results of our system. The model for the number of deletions per sentence however makes recall and precision go down. Possibly another choice of equivalence classes for sentence lengths and the number of deletions per sentence could change that. However we concluded from the inspection of the English VERBMOBIL data and the Mandarin CallHome data that the same choice of equivalence classes for both corpora is appropriate.
Hence the system which produces the best results for the Mandarin CallHome corpus uses no model for the number of deletions per sentence. The language model weight is set to $\lambda_{lm} = 0.1$ and the context model weight to $\lambda_{context} = 5$ for this system. Table 6.13 compares the results from this system with the results from the baseline system. The slight decrease of precision for the best system is compensated by the big increase of recall.

	Hits	False Positives	Recall	Precision
Baseline system	1145	302	38.1%	79.1%
Best system	1486	448	49.4%	78.8%

Table 6.13: Overall best system for the Mandarin Chinese CallHome corpus compared to the baseline system. For the best system we set $\lambda_{lm} = 0.1$, $\lambda_{context} = 5$ and no model for the number of deletions per sentence is used

Generally one can observe that our system performs worse for the Mandarin CallHome corpus than for the English VERBMOBIL corpus. As suggested in section 4.1.2, the main reason for this seems to be that the vocabulary of the Mandarin CallHome corpus is more than three times bigger than the vocabulary of the English VERBMOBIL corpus, while the amount of training data we have for the Mandarin CallHome data is not even twice as big as the amount of training data for the English VERBMOBIL corpus.
The portation of our system from English to Mandarin Chinese was very straight forward. We used exactly the same statistical models for both languages where only the parameters had to be adapted for the different languages and different corpora. A deeper knowledge of particular characteristics of Mandarin Chinese spontaneous speech would perhaps help to improve the performance of our system on the Mandarin CallHome corpus even more, since our models could be modified to meet special requirements of this language. Nevertheless it is remarkable that it was possible to use our system on Mandarin Chinese without having any knowledge about this language. The annotated training was sufficient to train and to use the system.

6.4 Comparison to Other Work

In this section we compare the results of our best systems with results of some other works about disfluency processing we described in section 2.3.2. Although recall and precision are important measures for the evaluation of the performance of a disfluency processing system, one can not compare the results of different works by only considering these two figures. As one can see from the experiments we conducted on different corpora it is always important to take the corpus which is used into account when results are compared.

Table 6.14 compares our results with five other works about automatic correction and detection of disfluencies. If available, we give results for disfluency correction and detection. Note that in our system disfluency correction and detection can not be separated. Hence we can report only a correction result for our experiments.

Authors	Corpus	C/D	Recall	Precision
Our system	English VERBMOBIL	C	77.2%	90.2%
Our system	Mandarin CallHome	C	49.4%	76.8%
[Bear et al., 1992]	ATIS	D	76%	62%
		C	44%	35%
[Nakatani and Hirschberg, 1994]	ATIS	D	86.1%	91.2%
[Heeman, 1997]	Trains	D	76.8%	86.7%
		C	65.9%	74.3%
[Zechner, 2001][2]	English CallHome/CallFriend	C	51.9%	59.4%
[Spilker et al., 2000]	German VERBMOBIL	D	71%	85%
		C	62%	83%

Table 6.14: Comparison of the results for different works about disfluency processing. In the column C/D it is marked whether the results apply for disfluency correction (C) or detection (D).

The ATIS corpus (air travel planing) and the Trains corpus (railroad freight transportation) are comparable or even easier for the task of disfluency correction than the English VERBMOBIL corpus. The variety of topics and the size of the vocabularies are smaller than in the English VERBMOBIL corpus. The Mandarin CallHome corpus and the English CallHome/CallFriend corpora with a comparatively large vocabulary size and a large variety of topics which are treated in the dialogs are more difficult than the other corpora.
We note from table 6.14 that our system for the English VERBMOBIL corpus outperforms all other systems for disfluency correction. When we compare our system for the Mandarin CallHome corpus to the DIASUMM-System [Zechner, 2001] which is the only system working on a "difficult" corpus, we see that we have a higher precision, but Zechner's system achieves a higher recall.

[2]The results for the system form [Zechner, 2001] are not reported in that work. We evaluated Zechner's system in our lab, using the data, which Zechner used as well for evaluation in his work.

Chapter 7

Conclusions and Further Work

In our work we presented a system for disfluency correction in spontaneous speech dialogs using a noisy-channel approach. The system is trained using information extracted out of a text in which disfluencies are annotated. After training it is able to perform the correction of disfluencies for an arbitrary sentence. We conducted experiments on the English VERBMOBIL corpus and the Mandarin Chinese CallHome corpus. The results of our experiments indicate that our approach is working well for disfluency correction. The results on the Mandarin CallHome Corpus even suggest, that the approach is applicable to different languages.
Nevertheless, there remains still some work to be done in order to improve the system.
In order to fully integrate our disfluency correction system into a spontaneous speech processing framework, it needs to function on speech recognizer output rather than on manually transcribed speech.
Furthermore the influence of some more features for disfluency correction and detection should be examined for our system. A simple model which can correct repetitions and correction by considering matching words or POS-tags in a sentence might improve the performance for the correction of disfluencies of this type. The effect of acoustic features (e.g. duration of pauses and words, intonational boundaries) on the performance of our system also remains to be examined.
With a growing number of features the number of free parameters to be adjusted grows as well. Thus it is important to develop an algorithm which finds automatically an optimal parameter combination given an optimization criterion.
Hence our current system can be seen as a first step forward building a disfluency correction system using a noisy-channel approach. We believe that the extensions proposed above can transform system to an even more valuable component in a spontaneous speech processing framework.

Bibliography

[Alexandersson et al., 1997] Alexandersson, J., Reithinger, N., and Maier, E. (1997). Insights into the Dialogue Processing of Verbmobil. In *Proceedings of the fifth Conference on Applied Natural Language Processing*, pages 33–40, Washington D.C.

[Bear et al., 1992] Bear, J., Dowding, J., and Shriberg, E. (1992). Integrating Multiple Knowledge Sources for Detection and Correction of Repairs in Human-Computer Dialog. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 56–63. Association for Computational Linguistics.

[Brill, 1994] Brill, E. (1994). Some Advances in Transformation-Bases Part of Speech Tagging. In *Proceedings of the 12 th National Conference on Artificial Intelligence*.

[Brown et al., 1993] Brown, P. F., Pietra, S. A. D., Pietra, V. J. D., and Mercer, R. L. (1993). The Mathematics of Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

[Burger, 1997] Burger, S. (1997). Transliteration spontansprachlicher Daten. Verbmobil Technisches Dokument 56, Universtität München.

[Carbonell and Hayes, 1983] Carbonell, J. and Hayes, P. (1983). Recovery Strategies of Parsing Extragrammatical Language. *American Journal of Computational Linguistics*, 9(3-4):123–146.

[Heeman, 1997] Heeman, P. A. (1997). *Speech Repairs, Intonational Boundaries and Discourse Markers: Modeling Speakers' Utterances in Spoken Dialog*. PhD thesis, Department of Computer Science, University of Rochester.

[Hindle, 1983] Hindle, D. (1983). Deterministic Parsing of Syntactic Non-fluencies. In *Proceedings of the 21th Annual Meeting of the Association for Computational Linguistics*, pages 123–128.

[Huang, 2003] Huang, F. (2003). Annotation Guideline for Callhome Mandarin Corpus. Guideline given to human annotators performing the disfluency annotation of the corpus.

[Huang et al., 2001] Huang, X., Acero, A., and Hon, H.-W. (2001). *Spoken language processing*. Prentice Hall PTR.

[Jelinek, 1985] Jelinek, F. (1985). Self-organized Language Modeling for Speech-Recognition. Technical report, IBM T.J. Watson Research Center.

[Kneser and Ney, 1993] Kneser, R. and Ney, H. (1993). Improved Clustering Techniques for Class-Based Statistical Language Modeling. In *Proceedings of the European Conference on Speech Technology*, pages 973 – 976.

[Kurematsu et al., 2000] Kurematsu, A., Akegami, Y., Burger, S., Jekat, S., Lause, B., MacLaren, V., Oppermann, D., and Schultz, T. (2000). VERBMOBIL Dialogues: Multifaced Analysis. In *Proceedings of International Conference on Speech and Language Processing*, Beijing, China.

[Nakatani and Hirschberg, 1994] Nakatani, C. and Hirschberg, J. (1994). A Corpus-based Study of Repair Cues in Spontaneous Speech. *Journal of the Acoustical Society of America*, 95(3):1603 – 1616.

[Ries et al., 1996] Ries, K., Suhm, B., and Geutner, P. (1996). Language Modeling in JANUS. Technical report, School of Computer Science, Carnegie Mellon University.

[Shriberg, 1994] Shriberg, E. (1994). *Preliminaries to a Theory of Speech Disfluencies*. PhD thesis, University of California at Berkeley.

[Spilker et al., 2000] Spilker, J., Klarner, M., and Görz, G. (2000). Processing Self-Corrections in a Speech-to-Speech System. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer Verlag, Berlin.

[Stephan Vogel and Hermann Ney and Christoph Tillmann, 1996] Stephan Vogel and Hermann Ney and Christoph Tillmann (1996). HMM-Based Word Alignment in Statistical Translation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 836–841, Copenhagen, Denmark.

[Stolcke and Shriberg, 1996] Stolcke, A. and Shriberg, E. (1996). Statistical Language Modeling for Speech Disfluencies. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, volume 1, pages 405–408.

[Stolcke et al., 1998] Stolcke, A., Shriberg, E., Bates, R., Ostendorf, M., Hakkani, D., Plauche, M., Tür, G., and Lu, Y. (1998). Automatic Detection of Sentence Boundaries and Disfluencies Based on Recognized Words. In *Proceedings of the International Conference of Spoken Language Processing*, volume 5, pages 2247–2250.

[Vogel et al., 2000] Vogel, S., Och, F. J., Tillmann, C., Nieen, S., Sawaf, H., and Ney, H. (2000). Statistical Methods for Machine Translation. In Wahlster, W., editor, *Verbmobil: Foundations of Speech-to-Speech Translation*, pages 377–393. Springer Verlag, Berlin.

[Wahlster, 2000] Wahlster, W., editor (2000). *Verbmobil: Foundations of Speech-to-Speech Translation*. Springer Berlin, Heidelberg.

[Wang and Waibel, 1997] Wang, Y.-Y. and Waibel, A. (1997). Decoding Algorithm in Statistical Machine Translation. In *Proceedings of the 35^{th} Annual Meeting of the Association for Computational Linguistics*.

[Wheatley, 1997] Wheatley, B. (1997). CALLHOME Mandarin Chinese. Speech corpus published by the LDC.

[Zechner, 2001] Zechner, K. (2001). *Automatic Summarization of Spoken Dialogues in Unrestricted Domains*. PhD thesis, Language Technologies Institute, School of Computer Science, Carnegie Mellon University, Pittsburgh.