International Journal of Software Engineering and Knowledge Engineering

Context Model Acquisition from Spoken Utterances (083)

Sebastian Weigelt*, Tobias Hey† and Walter F. Tichy‡
Institute for Program Structures and Data Organization
Karlsruhe Institute of Technology
Karlsruhe, Germany
**weigelt@kit.edu*
†*hey@kit.edu*
‡*tichy@kit.edu*

Current systems with spoken language interfaces do not leverage contextual information. Therefore, they struggle with understanding speakers' intentions. We propose a system that creates a context model from user utterances to overcome this lack of information. It comprises eight types of contextual information organized in three layers: individual, conceptual, and hierarchical. We have implemented our approach as a part of the project PARSE. It aims at enabling laypersons to construct simple programs by dialog. Our implementation incrementally generates context including occurring entities and actions as well as their conceptualizations, state transitions, and other types of contextual information. Its analyses are knowledge- or rule-based (depending on the context type), but we make use of many well-known probabilistic NLP techniques. In a user study we have shown the feasibility of our approach, achieving F1 scores from 72% up to 98% depending on the type of contextual information. The context model enables us to resolve complex identity relations. However, quantifying this effect is subject to future work. Likewise, we plan to investigate whether our context model is useful for other language understanding tasks, e.g., anaphora resolution, topic analysis, or correction of automatic speech recognition errors.

Keywords: Spoken Language Interfaces; Spoken Language Understanding; Language Model; Programming In Natural Language; End-User Programming; Ontologies; Natural Language Processing; Knowledge Representation; Context Model; Context; Natural Language Understanding.

1. Introduction

The last decades have seen a growing interest in Spoken Language Interfaces (SLIs). The rise of intelligent assistants such as Siri, GoogleNow, and Cortana established SLIs in daily life [1]. One can easily book a restaurant, schedule a meeting, or make out whether one needs an umbrella by talking to a smart phone. However, such systems struggle with long utterances and complex relations. They are not always able to understand the speaker's intention. Humans leverage contextual information to interpret utterances. Among other things they take into account the choice of words as well as knowledge about the mentioned concepts and the setting of the conversa-

tion. Humans exploit their general knowledge about grammar, concepts that exist in the world, and their perception to create an inner model that comprises several forms of contextual information. Without these, a set of instructions, such as "Close the fridge [...] Open the dishwasher [...] Then close all open appliances," could not be interpreted correctly. Current approaches cannot infer that the dishwasher should be closed because they are not able to answer the following questions. Apparently, 'all appliances' is a group of objects, but which objects are appliances? What does 'open' mean and which objects can be open? What can be inferred from the actions 'open' and 'close' in regard to the state of objects?

We propose a system that infers the required contextual information from the utterance automatically. First, we define eight types of context, organized in three layers of increasing abstraction. Then, we implement multiple techniques to generate a context model from spoken utterances. Our approach uses NLP techniques to cope with contextual information that can be extracted from the actual wording. We integrate knowledge databases, such as WordNet [2], to find relations between entities in an utterance. In addition, we use a domain model to extract contextual information concerning the environment of the conversation. Finally, our system infers further information by combining the different sources. This context model enables us to resolve complex relations as in the above example. We discuss use cases in Section 7.

2. Related Work

Most spoken language interfaces do not incorporate explicit contextual information. However, many approaches model and use contextual information implicitly. Systems based on recurrent neural networks make use of the architecture of such networks [3]. Since 'old' information is constantly fed back into the network, current decisions are based on previous to some extent. Similarly, partially-observable markov decision process approaches model a kind of implicit contextual information in their belief systems [4]. Implicit context is also utilized by some knowledge-based systems. Active ontologies retain the information of previous activations in the network and are therefore able to use this information to interpret the current utterance [5]. Implicit knowledge is only accessible in the specific task and can neither be shared among different tasks nor addressed directly. Furthermore its impact is hard to quantify and evaluate.

Explicit contextual information is used by some approaches in natural language processing in robotics. The approach by Misra et al. uses situational contextual information to validate candidates for action grounding [6]. Fleischmann and Roy use the lexical context of actions to improve their approach to learn a mapping between spoken utterances and actions [7]. Another application is depicted by the approach of Bordes et al., which uses the identified concepts of preceding expressions to improve the language grounding of the current expression [8]. They all have in common that the contextual information primarily depicts the environment of the

Table 1. Overview of Context Layers and Types

Layer	Type	Description
Individual	Entity	occurring things that can exist
	Spatial Deixis	spatial relations between entities
	Action	events occurring in context
	State Transition	state changes induced by actions
Conceptual	Concept	abstraction of entities and actions
	State	states individuals can be in
Hierarchical	Super Concept	hypernym relations between concepts
	Part-Of Relation	meronym relations between concepts

utterance. In contrast, we define a comprehensive context model that not only considers the environment but also the utterances itself, the expressed concepts and relations between them.

3. Context Model

The term *context* has several definitions [9, 10, 11]. Most definitions share the notion that context describes information that is used to understand the meaning of an artifact. The artifacts are parts of a communication situation, i.e., spoken utterances. The context of these artifacts includes information about the setting of the utterance, such as the place, time, the relation between the communicating partners and their mutual knowledge assumptions. In addition, afore-stated expressions in an utterance form a setting for the statement.

Since our approach is based on artifacts of spoken language, it is necessary to analyze which contextual information is retrievable from this limited input. The input consists of spoken utterances provided in a discourse. Sentences contain information about actions performed by subjects and treated objects. Additional expressions form relations between them, including spatial, temporal, or conditional. The subjects, objects, and actions are instances of concepts. For example, the utterance "The fridge is running," refers to the concepts `refrigerator` and `run_operate` (in contrast to `run_move`). In an utterance, many entities belong to the same concept. Similarly, many of the concepts are specializations of the same (or closely related) super concepts. Hence, concepts of instances from the utterance form a hierarchy of concepts. For instance the concepts `refrigerator` and `dishwasher` are manifestations of the super concept `white_goods`.

Based on these observations, we define three layers of contextual information that are extractable from spoken utterances:

(1) *Individual Layer*: that deals with instances and the relations among them
(2) *Conceptual Layer*: that describes the concepts the instances form
(3) *Hierarchical Layer*: which considers generalizations of concepts

The layers constitute incremental layers of abstraction from the input. Table 1 summarizes the context layers as well as the types. We will discuss the layers and types in detail in the upcoming sections.

3.1. *Individual Layer*

The first layer of context describes the actual course of events. It concerns the actions, entities, etc. that occur in the spoken utterance. We define subjects and objects as acting and treated things and call this type *entity*. These entities might have additional properties (e.g. states) or are related to other entities (e.g. spatial relations) and actions. Locative relations, such as `cup:is_located_on:table` in the utterance "the cup on the table," are often used in spoken language. We denote this type of context as *spatial deixis*. Predicates determine the relation of entities. If a predicate expresses an action we call this information context of type *action*. For example, in "John, go to the table," the action is `go(who:John, where:table)`. The formalization of actions enables us to track which actions have been stated and which entities were affected by these. In some cases actions implicate a change of state of the treated entities; we call this kind of context *state transition*. For example, in the utterance "Open the fridge," the state of the entity `fridge` changes to `open`. State transitions provide valuable information to examine whether the utterance contains a valid sequence of actions. As we only consider spoken language, we have no information about the initial state of the entities. Thus the state transitions are the only way to obtain an image of the states the entities are in at distinct points in time.

3.2. *Conceptual Layer*

The *Conceptual Layer* contains the abstract concepts of entities, actions, and state transitions. To understand an utterance, humans shape concepts by leveraging the knowledge they have learned about the world and things that exist. This knowledge encompasses information about the entities and actions humans normally learn during their life, such as different meanings and synonyms of a word or relations between objects, e.g., that 'refrigerator' and 'fridge' refer to the same concept. Thus, we consider *concepts* as another context type. Note that whenever the same entity or action occur multiple times in utterances they shall refer to the same concept. Additionally, we define context of the type *state* as abstraction of state transitions. States are related to concepts, while state transitions are caused by actions. Thus, we see states as part of the *Conceptual Layer* rather than the *Individual Layer*. Note that different concepts can share the same state on the *Conceptual Layer*. E.g., the concepts `refrigerator` and `microwave_oven` might share the state `open`.

3.3. *Hierarchical Layer*

The *Hierarchical Layer* depicts super-conceptual and part-of relations of concepts. Humans do not stop the process of 'context shaping' with the concepts directly

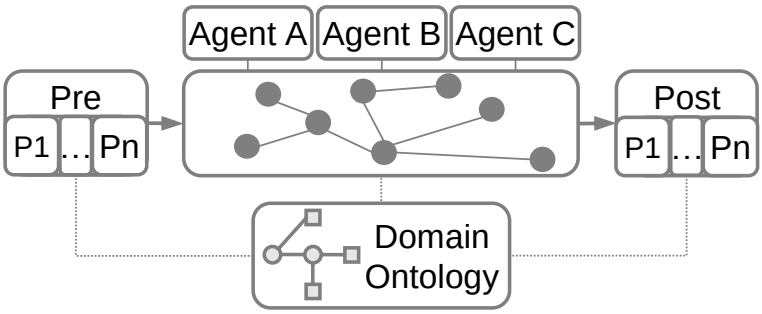

Fig. 1. Architecture of PARSE

instanced in the utterance. In fact they use their knowledge to relate the concepts to each other. This happens in two ways. First, humans form superordinated concepts to perceive connections between concepts. We consider these relations as another context type named *super concept*. E.g. concepts such as `refrigerator` and `dishwasher` share the super concept `white_goods` and therefore are concept-wise closely related to each other. Second, humans determine whether a concept depicts a part of another concept. We address this kind of relation with the context type *part-of relation*. E.g., in the utterance "Go to the fridge and open its door," the concept `door` is a part of the concept `refrigerator`.

4. PARSE

Our work on context model acquisition from spoken utterances is part of the project *PARSE* [12]. The goal of the project is to enable laypersons to program in plain spoken English. Typical application areas of *PARSE* are robotics, home automation systems, and alike. To facilitate programming with spoken language the system must understand the user's intention. Thus, *PARSE* is actually a system for Spoken Language Understanding (SLU) [13]. To achieve deep SLU *PARSE* takes the approach of independent agents. Every agent is responsible for a certain SLU task. As SLU tasks are generally interdependent all agents work in parallel and therefore might benefit from results of each other. The strict separation of concerns additionally enables us to build agents either knowledge-based or probabilistically according to the SLU task at hand and evaluate it intrinsically. The architecture of *PARSE*, which is illustrated in Figure 1, is separated in three independent parts: a pipeline for preprocessing, an agent-based main execution, and a pipeline for postprocessing. A graph serves as shared data structure for the agents. The preprocessing pipeline is meant for common natural language processing tasks, e.g., automatic speech recognition, shallow parsing, and named entity recognition. The user utterance is processed sequentially here; finally a basic graph for the main execution is built. The main execution is responsible for SLU. Agents for deep SLU transform the graph to publish their results. Hereby, a semantic representation of the input is created incre-

Table 2. Domain Ontology Structure

Class	Description
System	Systems and sub-systems, i.e. API classes
Method	System functions, i.e. API methods
Parameter	Parameter names used by the system
Data Type	Data Types used by the system
Value	Value enumerations and ranges of data type values
Object	External objects, e.g., *a fridge* or *a cup*
State	States of the external objects, e.g., *opened* and *closed*

mentally. SLU tasks encompass detection of actions, loops and conditions, context, topic and coreference analysis, etc. If the graph cannot be transformed to a proper intention model, the utterance is likely to be incomplete or ambiguous. In such situations the user is queried for clarification. The postprocessing pipeline maps the user's intention – modeled in the graph – to functions of the target system. Target systems are modeled in ontologies as proposed in our previous project *NLCI* [14]. We define a class hierarchy suitable for all target systems as shown in Table 2. In [14] we have shown, that domain ontologies can be extracted semi-automatically from most APIs for end-user programming with small effort.

5. Context Acquisition

To generate the representation of contextual information described in Section 3 we use an incremental approach to construct the layers. We have implemented the context acquisition as an agent for *PARSE*. Thus, we can draw from information created by the preprocessing pipeline. This information includes parts of speech, lemmata, chunks, named entities, and semantic roles.

We start our analysis on the *Individual Layer*, the layer with the lowest degree of abstraction. We identify the described entities by analyzing the noun phrases. Additionally, we use a rule-based approach on parts of speech and chunks to add information such as the grammatical number, adjectives, quantifiers, and determiners. After extracting the entities we are able to search for spatial deixes describing relations between the identified entities. We accomplish this task by keyword matching on the expressions between the identified entities. This approach is reasonable, since in English grammar locative relations are usually expressed by prepositional phrases between nominal phrases they relate [15]. Finally, we extract the actions expressed in the utterance. To identify actions we combine a rule-based analysis of verb phrases with the information we obtain from the semantic role labeler SENNA [16] included in the preprocessing pipeline. Furthermore, we translate arguments of predicates to thematic roles from VerbNet [17] to relate actions and treated entities.

With the instances present, we are now able to infer the concepts and states on the *Conceptual Layer*. The process is depicted in Algorithm 1. To generate

Algorithm 1 Concept Building

```
1: procedure BUILD_CONCEPTS
2:     matched_concept ← match instance and concept_set
3:     if matched_concept ≠ null then
4:         assign instance to matched_concept
5:     else
6:         domain_individuals ← query ontology
7:         concept_name ← match instance and domain_individuals
8:         if concept_name = null then
9:             concept_name ← match (sub-)phrases of instance with WordNet
10:        end if
11:        if concept_name ≠ null then
12:            build new_concept from concept_name
13:            add information from ontology and WordNet to new_concept
14:            assign instance to new_concept
15:            add new_concept to concept_set
16:        end if
17:    end if
18: end procedure
```

concepts we first try to match entities with individuals from our domain ontology (lines 6 and 7). We use the Jaro-Winkler distance [18] with a threshold of 0.92 and synonyms from WordNet to allow fuzzy matching. The threshold was determined empirically. If no proper match is found for an entity we query WordNet (lines 8 to 10). First we use the full phrase that represents the entity. If unsuccessful we query WordNet again with sub-phrases of decreasing length (the head of the phrase is always included). The first match forms the concept. If there is no WordNet entry that matches the (sub-)phrase, no concept is created. For the following entities we first try to match the head of the phrase with an existing concept (line 2). Again we use the Jaro-Winkler distance and synonyms. If a matching concept is found we link the entity to the concept and proceed with the next entity (line 3 to 5). Otherwise the previously described generation process is invoked. For all concepts we additionally retrieve synonyms from WordNet (line 13). This approach results in associating different entities, such as `white fridge` and `small fridge`, to the same concept `refrigerator`. For all created concepts we retrieve their possible states from the domain ontology and relate them to the concept (line 13).

Then we examine whether the actions in the utterance cause any state transitions. We obtain this information from the domain ontology. Exemplary, the ontology contains the fact that the action `open` causes the state `closed` to change to `open`. For all state-changing actions we analyze the treated entities. We verify whether the entity's concept can be in this state. If so, we link the state to the entity. We allocate the same state to all following entities of the same concept until

Algorithm 2 Concept Hierarchy Construction

```
1: procedure HIERARCHY_CONSTRUCTION
2:     repeat
3:         new_super_concept ← false
4:         for all tuple_of_concepts do
5:             if tuple_of_concepts has no super_concept then
6:                 wu_palmer ← calculate wup similarity of tuple_of_concepts
7:                 if wu_palmer ≥ 0.7 then
8:                     lcs ← calculate LCS of tuple_of_concepts
9:                     if depth of lcs > threshold then
10:                        super_concept ← match lcs and concept_set
11:                        if super_concept = null then
12:                            build super_concept from lcs
13:                            add information from WordNet to super_concept
14:                            add super_concept to concept_set
15:                        end if
16:                        assign super_concept to tuple_of_concepts
17:                        new_super_concept ← true
18:                    end if
19:                end if
20:            end if
21:        end for
22:    until new_super_concept = false
23: end procedure
```

another state-changing action occurs. Then the analysis starts anew.

With the concepts at hand, we are able to form relations on the *Hierarchical Layer*. The process is depicted in Algorithm 2. First, we use WordNet to build up a hierarchy of concepts. Apparently, the lowest common subsumer (LCS) of a pair of concepts in WordNet's hyperym hierarchy is a good candidate for a super concept (line 8). However, this naive approach would lead to many imprecise super concepts, such as `artifact` or `entity`. Therefore, we use the similarity metric defined by Wu & Palmer [19] with a threshold of 0.7 and a depth filter to avoid terms that are too generic (line 6 to 9). We create a new super concept from the LCS (lines 10 to 14) and link it to both sub concepts (line 16). If one of the considered concepts is a hypernym of the other, the second becomes the super concept of the first. The hierarchy can be extended further. The created super concept can be used to find new super concepts (repeat-until in lines 2 and 22). For example, if in the first iteration the super concepts `white_goods` and `kitchen_appliance` are created, the next iteration produces the super concept `home_appliance`.

Second, we generate part-of relations. Some of these are included in our domain model and can be retrieved directly. If the ontology yields no information, we use

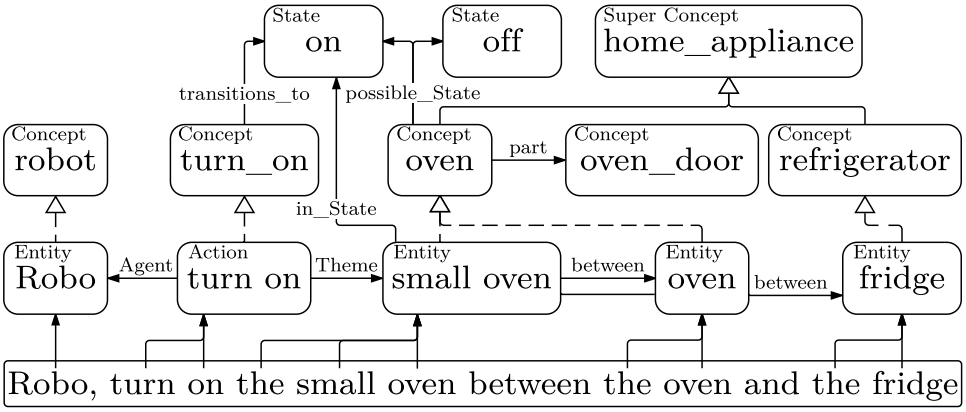

Fig. 2. Extract of the context model for an exemplary utterance.

meronyms from WordNet. These are solely used to create part-of relations among existing concepts to avoid false positives. However, parts retrieved from the domain ontology are added to the context model even if they had no concept representation before. The final context model consists of three layers with five context types denoted as nodes (entity, action, concept, super concept, and state) and three denoted as edges (spatial deixis, state transition, and part-of relation). Figure 2 shows an extract of the context model for the exemplary utterance "Robo, turn on the small oven between the oven and the fridge."

6. Evaluation

To evaluate our approach to context acquisition we conducted a user study with 10 participants. We let all subjects describe tasks for a robot in two scenarios. We took recordings, transcribed them and provided a gold standard per context type. For each context type we calculate precision, recall and the F_1 score. In the upcoming subsections we first show the layout and implementation of our study, afterwards we discuss the results.

6.1. *Experimental Design*

Our user study comprises two scenarios containing a household robot in a kitchen setting. In both scenarios the robot should fulfill a certain task; in the first scenario it should fill a cup with water from the fridge, bring it to the user and afterwards put cups from the dishwasher into the cupboard. In the second scenario the robot is supposed to prepare an instant meal that is located in the fridge by putting it on a plate from the dishwasher and heat it in the microwave. Subjects are encouraged to describe the steps to accomplish the task to the robot. The instructions the participants received contained no concrete wording, just a high level description of

the tasks and figures that showed the setting. We took continuous recordings, one per subject and scenario.

Ten subjects participated in the study, four female and six male. Nine were grad students from different departments and one was a computer science PhD student. All were native German speaker. However, all but one assessed their own English skills to be at least 'experienced' (*CEF* level C1). All participants gave descriptions for both scenarios. One participant repeated the recording for the second scenario because of an omission. Therefore we ended up with 21 recordings. As the subjects were able to describe the task as they liked, we received quite different recordings. They vary in length from 15 seconds up to 80 seconds and in instructions for the robot from a minimal set of 5 up to 22.

We transcribed all recordings according to the guideline by Kiesling et al. [20] and prepared a gold standard for each context type. We annotate the solutions as per-phrase labels. Thus, a phrase can have none, one, or more labels (depending on how many types of context it depicts). Annotating entities, actions and spatial deixes is straightforward. The gold standard for actions includes its argument relations. Thus, we can evaluate not only if the expected action is present but also whether all entities are identified correctly. We annotate states induced by the state transitions at the treated entities. This approach enables us to evaluate not only the state transitions but also the state inference mechanism. A recurring state annotated at different instances of an entity indicates that no state transition was identified. The expected states are those defined in the domain ontology. To assess the state generation on the *Conceptual Layer* we distinguish states and state references: 'States' refers to the created states in the resp. scenario and 'state references' to the created relations to concepts. For example, "Open the door and the window" produces one state *open* but two state references: door $\rightarrow$ open and window $\rightarrow$ open. We annotate concepts and super concepts as well as the possible states of the concepts at all phrases that can cause them. Hence, our evaluation is insensitive to consequential errors. For part-of relations we annotate the phrase that mentions the part. For example, "Go to the fridge and open [the door]." As ground truth for the part-of relations we used the relations present in our domain ontology and meronyms from WordNet. For our scenarios, only a few relevant meronym relations can be found in WordNet. Thus, the majority of expected part-of relations are based on our domain ontology. Table 3 summarizes the recordings' meta data and shows the number of expected instances per context type. To retrieve the results of our agent we first run the preprocessing pipeline of *PARSE*. Afterwards we run our agent multiple times – until no further changes can be observed – without interference of other agents.

6.2. *Results*

To assess the soundness of our approach we calculate precision, recall and the F_1 score. Table 4 summarizes the results of our evaluation. The results are promising.

Table 3. Evaluation: Data Overview

	Scenario 1	**Scenario 2**	**Total**
Recordings	10	11	21
Words	734	811	1545
Phrases	467	543	1010
Instructions	121	143	264
Entities	199	233	432
Spatial Deixes	41	43	84
Actions	120	154	274
State Transitions	33	48	81
Concepts	274	320	594
States (References)	8 (306)	10 (338)	18 (644)
Super Concepts	88	56	144
Part-Of Relations	35	43	78

Table 4. Evaluation: Results by Context Type

Context Type	**Precision**	**Recall**	$\mathbf{F_1}$
Entity	0.972	0.975	0.973
Spatial Deixis	0.945	0.793	0.862
Action	0.852	0.762	0.804
State Transition	0.854	0.627	0.723
Concept	0.986	0.974	0.981
State	1.000	0.955	0.977
Super Concept	0.680	0.932	0.786
Part-Of Relation	0.897	0.959	0.927

However, when we use more complex – and therefore error-prone – heuristics to generate context types the results diminish. Additionally, we generate results incrementally. Thus, failures occurring in the lower abstraction levels (or during preprocessing), such as incorrect entities or actions, cause failures in the higher levels. For example, if a verb 'open' is falsely labeled as an adjective by the part-of-speech tagger this results in a missing action, a falsely identified entity (having open as describing adjective), and also causing a state transition expected not to be present because there is no action that triggers it. This kind of failure is propagated even further. Since we evaluate state transitions by examining the states of all occurrences of the same entity, a missing state transition results potentially in multiple incorrect 'current states' and therefore in false state transitions. Hence, the low recall of the context type state transition can partially be explained by aftereffects of the action analyses. Since the generation of actions is primarily based on the semantic role

labeler SENNA, we expected to achieve F_1 scores around 75%[a]. Because our input examples are less complex than those from the CoNLL-2005 shared task, the generation of actions performs even better. However, our additional verb-phrase-based heuristic further improves recall by 5%. Nonetheless, missing or falsely created actions are unpleasant because they influence other context types. The relatively low recall of spatial deixes is caused by unexpected choice of words. The subjects used diverse phrases to express locative relations, some of which our approach is not capable to solve. For example, nested prepositional phrases, such as "put the meal from the fridge on the plate," may cause the locative relation to be ambiguous. This kind of ambiguity cannot easily be addressed by a keyword matching approach. We will investigate whether probabilistic methods are more suitable for such expressions. The generation of concepts from entities as well as the connection to the possible states is highly accurate; false positives and negatives result from failures in the preprocessing pipeline. Part-of relations are generated accurately; false positives are due to the use of WordNet. Since we do not resolve word ambiguities, we use all WordNet senses to find part-of relations. Sometimes this procedure causes failures, such as `plate` being a part of `table` instead of `dish`. The result of the evaluation of super concepts shows that our approach is focused on recall rather than precision. This objective is reasonable because we want to explore new context information and make it available to other language analyses. Wrong information can later be ignored, but information that is never generated remains unused forever. As with part-of relations, missing disambiguation of WordNet senses of the concepts is the reason of most of the false positives.

In summary our approach produces highly valuable results for the context types entity, spatial deixis, concept, state, and part-of relation. For the context types action and state transition an improved approach to semantic role labeling, i.e., adjusted to spoken utterances, would yield significantly better results. The evaluation shows that the current approach adds valuable context information. The result of the context type super concept indicates that the approach is feasible but depends on correct WordNet senses.

7. Use Cases

Our comprehensive context model is useful for resolving complex relations in spoken utterances. In this section we will show exemplary how we use the model in certain situations. For brevity we focus on identity relations.

First, we consider the sample utterance "Open the cupboard. Take the cup and close it." A naive approach would relate 'it' and 'the cup'. However, from the context model we know, that the `cupboard` is open and can be closed, but a `cup` cannot. Therefore we can infer that 'it' more likely refers to 'the cupboard'.

The second example is, "There is a tumbler on the table. Take the glass and

[a]SENNA achieves an F_1 score of 75.49% in the CoNLL shared task [16].

bring it to me." Here, the challenge is to understand that the expression 'the glass' is a generalization of the 'tumbler'. Since 'tumbler' and 'glass' are not synonymous, a simple WordNet query does not help. Neither a hypernym look-up in WordNet is helpful, as one cannot easily decide whether a hypernym is meaningful. Our model yields the information that `glass` is an appropriate super concept of `tumbler` and thus can be used as a substitute.

In the third example, "To prepare meringue take egg white, powdered sugar, and lemon extract. Put all the ingredients into the bowl," 'ingredients' refers to a group of previously mentioned entities. Our context model provides the information that `egg white`, `powdered sugar`, and `lemon extract` share the super concept `ingredient`. Together with the quantifier we are able to infer that 'all ingredients' refers to 'egg white', 'powdered sugar', and 'lemon extract'.

The last example, "Close the fridge [...] Open the dishwasher [...] Then close all open appliances," contains multiple challenges. First, one must resolve which are the 'appliances'. According to our context model `fridge` and the `dishwasher` share the super concept `appliance`. Second, the context model comprises the information that the fridge is closed and the dishwasher is open at the time 'all open appliances' is mentioned. Thus, we can infer that 'all open appliances' refers to the dishwasher.

8. Conclusion and Future Work

Acquiring a context model of spoken utterances is essential for understanding the speaker's intention. Without such information a spoken language interface could never fully interpret the relations in complex expressions. We have presented a new approach to generating a comprehensive context model from spoken utterances. Our approach generates eight context types: occurring entities and actions, the concepts of which they are instances, spatial deixes, states and state transitions, and relations between concepts as super and part concepts. The context types are organized in three layers representing different levels of abstraction.

The evaluation showed that the approach is promising: we achieve F_1 scores from 72% up to 98%, depending on the context type. However, the scores might be improved in several ways. Our generation of (super) concepts and part-of relations depends on the correct selection of the WordNet sense for a concept. Therefore, a comprehensive word sense disambiguation would improve our results. Moreover, our approach would benefit from a semantic role labeler adapted to spoken utterances. Since the semantic role labeler used by us is trained on textual input, it struggles with ungrammatical phrases common in spoken utterances. Consequentially, our generation of actions contains false positives. These are propagated to other context analyses, e.g., state transition.

However, the evaluation shows that the current approach adds valuable context information. We have shown that our context model can be used to resolve complex identity relations. The next step is to quantify this effect and extend this approach to coreference resolution in general.

Future work will focus on exploring further areas of application for our context model. We expect it to be useful for tasks such as topic extraction, time line analysis, correction of automatic speech recognition errors, and even word sense disambiguation. A partial context model might support word sense disambiguation, which in turn improves the context model.

References

[1] J. R. Bellegarda, "Spoken Language Understanding for Natural Interaction: The Siri Experience," in *Natural Interaction with Robots, Knowbots and Smartphones*, J. Mariani, S. Rosset, M. Garnier-Rizet, and L. Devillers, Eds. Springer New York, 2014.

[2] C. Fellbaum, *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[3] P. Xu and R. Sarikaya, "Contextual domain classification in spoken language understanding systems using recurrent neural network," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014.

[4] S. Young, M. Gašić, S. Keizer, F. Mairesse, J. Schatzmann, B. Thomson, and K. Yu, "The Hidden Information State model: A practical framework for POMDP-based spoken dialogue management," *Computer Speech & Language*, vol. 24, no. 2, Apr. 2010.

[5] D. Guzzoni, C. Baur, and A. Cheyer, "Modeling Human-Agent Interaction with Active Ontologies." in *AAAI Spring Symposium: Interaction Challenges for Intelligent Assistants*, 2007.

[6] D. K. Misra, K. Tao, P. Liang, and A. Saxena, "Environment-Driven Lexicon Induction for High-Level Instructions," *ACL (1)*, 2015.

[7] M. Fleischman and D. Roy, "Intentional Context in Situated Natural Language Learning," in *Proceedings of the Ninth Conference on Computational Natural Language Learning*, ser. CONLL '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005.

[8] A. Bordes, N. Usunier, R. Collobert, and J. Weston, "Towards understanding situated natural language," in *International Conference on Artificial Intelligence and Statistics*, 2010.

[9] A. Duranti, *Rethinking Context: Language as an Interactive Phenomenon*. Cambridge University Press, May 1992.

[10] A. Fetzer, *Recontextualizing Context: Grammaticality Meets Appropriateness*. John Benjamins Publishing, 2004.

[11] R. Stalnaker, *Context and Content: Essays on Intentionality in Speech and Thought*. Oxford University Press, 1999.

[12] S. Weigelt and W. F. Tichy, "Poster: ProNat: An Agent-Based System Design for Programming in Spoken Natural Language," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering (ICSE)*, vol. 2, May 2015.

[13] G. Tur and R. De Mori, *Spoken Language Understanding: Systems for Extracting Semantic Information from Speech*. Wiley, Apr. 2011.

[14] M. Landhäußer, S. Weigelt, and W. F. Tichy, "NLCI: A Natural Language Command Interpreter," *Automated Software Engineering*, Aug. 2016.

[15] H. H. Clark, "Space, time, semantics, and the child," in *Cognitive Development and the Acquisition of Language*. Oxford, England: Academic Press, 1973.

[16] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, "Natural Language Processing (Almost) from Scratch," *Journal of Machine Learning Research*, vol. 12, no. Aug, 2011.

[17] K. K. Schuler, "VerbNet: A broad-coverage, comprehensive verb lexicon," *Dissertations available from ProQuest*, Jan. 2005.

[18] W. E. Winkler, "The State of Record Linkage and Current Research Problems," Statistical Research Division, U.S. Census Bureau, Tech. Rep., 1999.

[19] Z. Wu and M. Palmer, "Verbs Semantics and Lexical Selection," in *Proceedings of the 32Nd Annual Meeting on Association for Computational Linguistics*, ser. ACL '94. Stroudsburg, PA, USA: Association for Computational Linguistics, 1994.

[20] S. Kiesling, L. Dilley, and W. D. Raymond, "The variation in conversation (vic) project: Creation of the buckeye corpus of conversational speech," *Ohio State University, Columbus, OH*, 2006.