Understanding Route Directions in Human-Robot Dialogue

SemDial 2011: Proceedings of the 15th Workshop on the Semantics and Pragmatics of Dialogue, pages 19–27.
Los Angeles, California, 21–23 September 2011.

Understanding Route Directions in Human-Robot Dialogue

Martin Johansson **Gabriel Skantze** **Joakim Gustafson**
KTH Speech Music and Hearing
Stockholm, Sweden

vhmj@kth.se,gabriel@speech.kth.se,jocke@speech.kth.se

Abstract

This paper discusses some of the challenges in building a robot that is supposed to autonomously navigate in an urban environment by asking pedestrians for route directions. We present a novel Wizard-of-Oz setup for collecting route direction dialogues, and a novel approach for data-driven semantic interpretation of route descriptions into route graphs. The results indicate that it is indeed possible to get people to freely describe routes which can be automatically interpreted into a route graph that may be useful for robot navigation.

1 Introduction

Robots are gradually moving from industrial settings into our daily lives. This change from constrained, well-controlled environments into situations where objectives and situations may radically change over time means that it will be next to impossible to provide robots with all necessary knowledge a-priori. Even if robots are able to learn from experience, sufficient information will not always be available in the environment to fill the knowledge gaps. Humans, however, are a rich source of information. If robots are equipped with the knowledge of how to extract this information, it will give them a powerful means to improve their adaptability and cope with new situations as they arise.

The purpose of the IURO project[1] – a successor of the ACE project (Bauer et al., 2009) – is to explore how robots can be endowed with capabilities for extracting missing information from humans through spoken interaction. The test scenario for the project is to build a robot that can autonomously navigate in a real urban environment and enquire human passers-by for route directions (see Figure 1).

Figure 1: ACE, the precursor to IURO (left) and a design draft of the IURO robot (right).

One of the central challenges in this project is that of interpreting the spoken route directions into a semantic formalism that is useful for the robot. In this paper we present a feasibility study where we explore a novel approach to data-driven semantic interpretation in this setting. For this study, we do not address the problems of automatic speech recognition, but will base the experiments of transcriptions. We present a novel Wizard-of-Oz setup that is used to collect initial data on human-robot route descriptions. This serves to validate whether it is at all possible to make people describe a route

[1] Interactive Urban Robot (www.iuro-project.eu)

in a way that is understandable to a robot. A domain model of route graphs based on previous research in the area has been developed and the collected data has been annotated according to it. A data-driven semantic chunking parser which utilizes the domain model is then applied to explore whether the route graphs can be automatically extracted from the route descriptions.

2 Dialogue for route directions

The task of the IURO robot is quite different from a robot which is supposed to do as told by a human instructor. For the IURO robot, the route directions retrieved from human interlocutors are only possible means for accomplishing the task; there is no end in itself in following them. If the input from a human interlocutor doesn't contribute to the robot's solving of the task, the robot may (in a polite way) turn to another human, which makes the task much more feasible. This is very different from assistive robots, which are supposed to respond and react to all requests from the user.

Studies on human-human dialogue with an error prone speech recognition channel have shown that humans that have a very clear goal of the interaction may accurately pick out pieces of information from the speech recognition results that are relevant to the task (even when the speech recognition accuracy is very poor), and ask relevant task-related questions to recover from the problem (Skantze, 2005). Experiments on automated call-routing have also shown that the system may give positive backchannels (such as "mhm") to get more input from the user, even if the system only understands parts of what has been said (Gustafson et al., 2008). In order to increase the acceptability of the direction-giving dialogues among the human interlocutors, we will use a non-committal dialogue strategy where the system produces feedback that has the purpose of progressing without revealing lack of understanding.

This approach calls for techniques for robust integration and selection of input modalities, as well as accurate confidence scoring, so that the system may pick out the pieces of information that it can actually understand. The system may also combine descriptions from several humans, to derive a more solid hypothesis.

The robot may use a controlled dialogue strategy where the human is asked for one piece of information at a time, but it may also allow the human to describe the route more freely, while the robot responds with encouraging backchannels. This is somewhat similar to the call-routing domain, where the user is often asked to describe the problem in a free way, and relevant concepts are extracted using data-driven methods (Gorin et al., 1997). However, whereas the semantics of utterances in the call routing domain is typically represented as a "bag of concepts", the semantics of route descriptions is highly structured and cannot be treated in this way. A central requirement of our model is also that it should be able to generalise to some extent when encountered with unseen data, and to be able to back off to more general concepts, without breaking the conceptual structure. We therefore need a domain model (ontology) which defines concepts on different levels of specificity and specifies how they may be structured, and we need a data-driven semantic interpreter which takes this domain model into account.

3 Representing navigational knowledge

A common way of representing navigational knowledge is the *route graph*, which is a directed graph that represents one or several possible ways of reaching a goal from a starting point. However, the details of this representation have varied, partly depending on what level of knowledge it is supposed to represent.

3.1 Topological and metric route graphs

According to Bauer et al. (2009), a *topological route graph* is a directed graph where nodes represent intersections on the route and edges straight paths which connect these intersections. If metric coordinates are assigned to the nodes (for example by the use of sensory data), a *metric route graph* is constructed, which the robot may use to derive distances and angels in order to follow the route graph.

3.2 Conceptual route graphs

While the topological and metric route graphs are useful for representing a route from a bird's eye perspective and to guide the robot's locomotion, they are not representative for how humans describe routes. Thus, a *conceptual route graph* is needed that can be used to represent human route descriptions semantically. In the scheme proposed by Müller et al. (2000), conceptual route graphs are similar to topological graphs in that nodes represent places where a change in direction takes place and edges connect these places. The route graph may be divided into *route segments*, where each segment consists of an edge and an ending node where an action to change direction takes place. Conceptually, each segment consists of the following components:

- **Controllers**: A set of descriptors that ensures that the traversal along the edge is maintained. These may be indicators of the travel distance, and important landmarks to keep track of (e.g., "*continue for 100 meters*", "*go through the tunnel*", "*you should have a large building on your left*").
- **Routers**: A set of place descriptors that helps to identify the ending node.
- **Action**: The action to take at the ending node in order to change direction.

Note that Controllers, Routers and Actions are not in themselves mandatory components for each segment, but that at least one of them is required. This representation has been further developed by Mandel et al. (2006), which applied it to a corpus of route descriptions involving 27 subjects, and showed that it had good coverage for the various types of descriptions in the corpus.

3.3 A domain model for route descriptions

The dialogue framework that we use in the IURO project is Jindigo (Skantze & Hjalmarsson, 2010). Jindigo is a Java-based framework for implementing incremental dialogue systems. The framework provides methods for defining domain models (a simple form of ontology) for limited domains in an XML format. The domain models are then compiled into Java classes. A type hierarchy is specified, where each concept type may have a set of typed attributes. For example, there is a generic concept LANDMARK, which is subtyped by TRAVERSABLE, which is in turn subtyped by the concepts STREET and LAWN. LANDMARK is also subtyped by BUILDING, which is in turn subtyped by CHURCH. As another example, the type hierarchy of CONTROLLER is shown in Figure 2. We currently have about 60 concept types in the domain model.

A route graph instance can be illustrated with a conceptual tree, where nodes represent concepts and edges represent arguments. An example representation is shown in Figure 3.

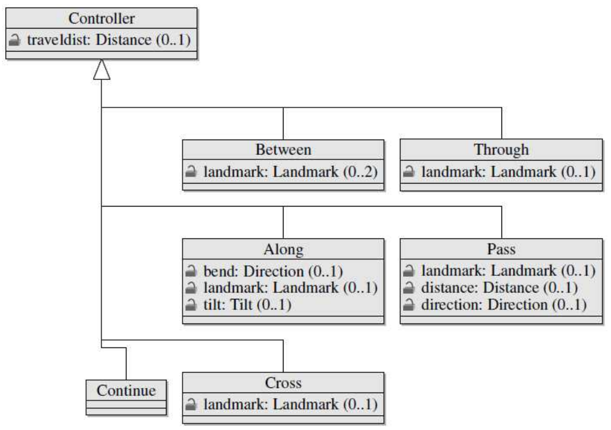
Controller
traveldist: Distance (0..1)
Between
landmark: Landmark (0..2)
Through
landmark: Landmark (0..1)
Along
bend: Direction (0..1)
landmark: Landmark (0..1)
tilt: Tilt (0..1)
Pass
landmark: Landmark (0..1)
distance: Distance (0..1)
direction: Direction (0..1)
Continue
Cross
landmark: Landmark (0..1)

Figure 2: The concept CONTROLLER and its subtypes.

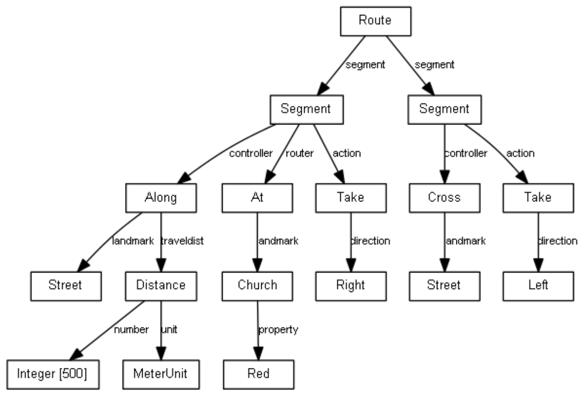

Figure 3: A conceptual route graph representing *"follow the street for five hundred meters up to eh the red church then go right cross the street and turn left"*

4 A Wizard-of-Oz data collection

At this stage in the project the robot platform is not yet ready for human-robot interaction. Thus, we have to start to collect preliminary data on human-robot route direction dialogue by other means. There have been several approaches to such data collections. Kyriacou et al. (2005) present an experiment where subjects are given the task of giving navigational instructions to a small robot in a miniature city. This is somewhat different from the IURO setting, since the subject can see the whole city from a bird's eye perspective when giving the instructions. Another approach is to let subjects watch a video of a route and simultaneously describe what they see (Rieser & Poesio, 2009). The drawback with that approach is that it might not be very representative for how people describe a route that is retrieved from memory.

In this data collection, we used a Wizard-of-Oz setup where the subject first watches a recorded video of a route from a first-person perspective. The Wizard then initiates a route description dialogue with the subject, where the subject has to recall the route from memory. (In the present study, the subject only got to see the video once, but this could be varied in order to simulate varying experience of the route). During the dialogue, both the Wizard and the subject are shown the initial perspective (as shown in Figure 4), giving them an opportunity to talk about visual cues at the start of the route. This is intended to resemble a real situation to some extent, where the human has some more or less vague notion about the route, which is likely to trigger disfluencies and erroneous descriptions. For each step in the dialogue, the Wizard's interface is updated with controls that correspond to the contents of the user's utterance. The robot's next utterance is automatically selected according to a state chart and played to the subject using text-to-speech.

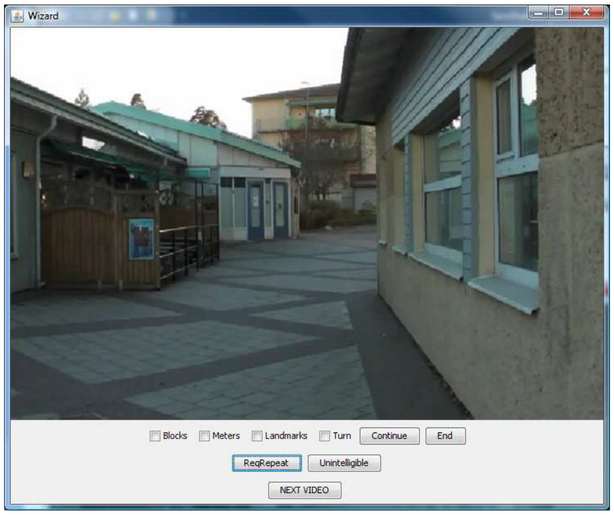

Figure 4: The Wizard-of-Oz interface used in the data collection.

Two types of dialogue strategies were used by the Wizard. First, a controlled dialogue strategy, where the controllers, routers and actions were asked for one at a time by the Wizard. Second, a free dialogue strategy where the subject is asked to describe the route freely using whatever strategy or format he or she deem suitable, with encouraging backchannels from the robot. If the subject does not continue, the Wizard initiates a specific question to the subject about how to continue. A very short example is shown in Table 1. In the experiment presented in this paper, we will use data from the free dialogues, but the controlled dialogue data will be useful when looking at fallback dialogue strategies for the robot.

Robot	Excuse me, could you help me to find my way to the library?
Human	Yes
Robot	Great, how should I go?
Human	Take the first street left
Robot	Yes
Human	Then the second left
Robot	Yes
	And then?
Human	You are at the library
Robot	Thank you, bye!

Table 1: An example dialogue in the free dialogue setting, translated from Swedish.

5 Data-driven semantic interpretation

In this section, we will explore a novel approach for interpreting the route descriptions presented in the previous section into the semantic representations described in section 3.

5.1 Previous work

The problem of semantic interpretation of spoken utterances – or Spoken Language Understanding (SLU), as it is also referred to – is a vital step in dialogue system processing. The problem can be formulated as taking a speech recognition result and produce a semantic representation that can be used by the dialogue manager to decide what to do next. A simple and robust approach is that of keyword spotting, where specific words in the input are associated with certain concepts or slot-value pairs. The drawback with that approach is of course that it cannot utilize any syntactic or contextual features and that the resulting semantic representation is not structured in any way. Grammar-based approaches which may be applicable to written text are often not appropriate, since they are poor at coping with disfluencies in spoken language and are not robust to speech recognition errors. More robust, grammar-based approaches have been presented (e.g., Skantze & Edlund, 2004). However, such approaches still need hand-written grammars which are tailored to the domain.

A promising alternative is to use data-driven methods, which do not need any hand-crafted grammars, may better cope with the irregularities of spoken language, and be more robust against speech recognition errors. In Meza-Ruiz et al. (2008), a method for SLU using Markov Logic Networks is presented, but the resulting semantic representations are limited to a set of slot-value pairs (i.e., they are not structured). Another approach is presented by Wong & Mooney (2007), where a context-free grammar (CFG) augmented with semantic instructions (based on first-order logic) is learned. However, the approach assumes that the input may be described with a CFG, which, as we discussed above, may not be the case for speech recognition results. He & Young (2006) presents a Hidden Vector State model (an extended HMM) which may produce deeply structured semantic interpretations. It is shown to have good performance in a travel-booking domain. However, it is not clear whether it may utilize an ontology and back off to more general concepts in order to learn generalizations, which we will aim for in the approach presented here.

5.2 The chunking parser

The *chunking parser* was introduced by Abney (1991), as a variant of a typical natural language parser where the syntactical analyser is comprised of two stages: the *Chunker* and the *Attacher*. The task of the Chunker is to convert a stream of words into a stream of chunks, which is taken as input by the Attacher. The Attacher then adds connections between individual chunks, thus producing a parse tree. The approach has gained a lot of interest, since it can be easily formulated as a classification problem, which makes it possible to apply machine learning methods (Sang & Buchholz, 2000). However, the approach has mainly been used for syntactic analysis, and not for semantic interpretation.

In this paper, we introduce a novel application of the chunking parser to data-driven semantic interpretation in limited domains. In this approach, the Chunker is given the task of finding base concepts in the sequence of words. The Attacher is then given the task of assigning more specific concepts (given the type hierarchy of the domain) and to attach concepts as arguments. Consider the example given in Figure 3, which could be chunked in the following way:

[$_{\text{CONTROLLER}}$ follow] [$_{\text{LANDMARK}}$ the street] [$_{\text{DISTANCE}}$ for five hundred meters] [$_{\text{ROUTER}}$ up to] [$_{\text{FP}}$ eh] [$_{\text{LANDMARK}}$ the red church] [$_{\text{DM}}$ then] [$_{\text{ACTION}}$ go] [$_{\text{DIRECTION}}$ right] [$_{\text{CONTROLLER}}$ cross] [$_{\text{LANDMARK}}$ the street] [$_{\text{DM}}$ and] [$_{\text{ACTION}}$ turn] [$_{\text{DIRECTION}}$ left]

As the example shows, this is similar to the chunks used in shallow parsing (e.g., a LANDMARK roughly corresponds to an NP), but here the chunks are semantically motivated. To turn the chunking into a classification problem, a common practice is to define two labels for each type of chunk: one with the prefix B- for the first word in the chunk and one with prefix I- for the following words. This is illustrated in Table 2.

Word	**Chunker**	**Attacher**
follow	B-CONTROLLER	*class*: ALONG *landmark*: → *traveldist*: →
the	B-LANDMARK	*class*: STREET
street	I-LANDMARK	
for	B-DISTANCE	*value:* 500 *unit*: METERUNIT
five	I-DISTANCE	
hundred	I-DISTANCE	
meters	I-DISTANCE	
up	B-ROUTER	*class*: AT *landmark*: →
to	I-ROUTER	
eh	B-FP	
the	B-LANDMARK	*class*: CHURCH *property*: RED
red	I-LANDMARK	
church	I-LANDMARK	
then	B-DM	
go	B-ACTION	*class*: TAKE *direction*: →
right	B-DIRECTION	*class*: RIGHT
cross	B-CONTROLLER	*class*: CROSS *landmark*: →
the	B-LANDMARK	*class*: STREET
street	I-LANDMARK	
and	B-DM	
turn	B-ACTION	*class*: TAKE *direction*: →
left	B-DIRECTION	*class*: LEFT

Table 2: The correct output of the Chunker and Attacher for the example in Figure 3.

The Attacher does two things. First, it may assign a more specific concept class (like *class*: CHURCH).

To allow it to generalise, it also assigns all ancestor classes, based on the domain model (i.e., BUILDING for CHURCH; this, however, is not shown in the example above). The second task of the Attacher is to assign attributes. Some attributes are filled in with new concepts (like *property*: RED), while others are attached to the nearest concept that fits the argument type according to the domain model (like *distance:* →, which means that the interpreter should look for a matching argument in the right context). Thus, while the chunking can be described as a *single-label* classification, the attachment is a *multi-label* classification where none, one or several labels may be assigned to the chunk.

5.3 Machine-learning algorithms

To implement the classifier, we used the Learning Based Java (LBJ) framework (Rizzolo & Roth, 2010), which has shown competitive performance on the CoNLL 2000 shared task (Sang & Buchholz, 2000). Two basic types of machine learning algorithms were tested: Naive Bayes and Linear Threshold Units (LTUs). Only the latter can be used for the multi-label learning in the Attacher. LTUs represent their input with a weight vector whose dimensions correspond to features of the input, and outputs a real number score for each possible label as output. For single-label learning, the label with the highest score is selected. For multi-label learning, all labels with a score above a certain threshold (which is learned) are selected. Three types of LTUs were tested: Sparse Perceptron, Sparse Avaraged Perceptron (Collins, 2002) and Sparse Winnow (Littlestone, 1988).

As a final step, a set of simple heuristic rules are used to group the CONTROLLERS, ROUTERS and ACTIONS into SEGMENTS (and a ROUTE). Errors in the Chunker and Attacher will also result in loose concepts, as well as surplus connections and concepts. These are also handled by the heuristic rules, so that a full route graph may be constructed. To handle numbers ("five hundred"), a simple rule-driven parser is applied in the attachment stage, which would otherwise require a large amount of data.

5.4 Features

One of the requirements of the interpreter is that it should be able to generalise to some extent when encountered with unseen data, and to be able to back off to more general concepts, without breaking the conceptual structure. To do this, we use not only the specific words as features, but also their Part-of-speech and affixes. Thus, the classifier may for example learn that the Swedish word "Drottninggatan" (Eng: "the Queen Street") is a STREET, given the suffix and context.

For the Chunker, the following features were used:

- **Word**: The word instance, as well as a window of the two previous and next words.
- **Previous tags**. The two previous chunking tags.
- **Word affixes**: The initial 2-4 letters of the word (prefix), and/or last 3-4 letters of the word (suffix).
- **Part-of-speech, Lemma**: The Part-of-speech tags and Lemmas of the five-word-window, automatically extracted using the software Granska (Domeij et al., 1999).

For the Attacher, the following features were used:

- **Chunk label:** The label produced by the Chunker, as well as a window of the two previous and next labels.
- **Lemmas**: The lemmas of the words in the chunk, both ordered and unordered ("bag of lemmas").
- **Suffixes**: The suffixes of the words in the chunk.

5.5 Concept Error Rate

For the evaluation of automatic speech recognition, Word Error Rate (WER) is a common measure of performance, where the string of words in the reference is compared to the string of recognized words using minimum edit distance, and the number of insertions, deletions and substitutions are counted (Jurafsky & Martin, 2000). Similarly, Concept Error Rate (CER) can been used to evaluate a semantic interpreter. However, this is not entirely straightforward. First, the concepts are tree structured, which means that they have to be flattened in a consistent manner. To accomplish this, a depth-first traversal of the tree is applied, where the order of the concepts are preserved if the order is important (e.g., the SEGMENT children of a ROUTE), and otherwise alphabetically ordered. Second, not all concept substitutions should be treated equal. For example, substituting CHURCH with BUILDING should not be penalized as much as substituting STREET with BUILDING. To handle this, the domain model is used in the evaluation, where an insertion or deletion gives the error score of 1, a wrong base concept gives 1.5, a too specific concept gives 1, a too generic concept gives 0.5, and a wrong branch in the concept hierarchy gives 1. The total error score is then divided with the number of concepts in the reference and multiplied by 100 to derive the CER. Although these error scores are quite arbitrary, they might give a better indication of the performance than one crude overall error score.

5.6 Data

The main part of the data used in the training and evaluation comes from the Swedish corpus described in section 4. In this experiment, we simply concatenated the user utterances of each dialogue and ignored the Wizard's utterances, which resulted in a total of 35 route descriptions with an average length of 54 words and 24.8 concepts. These routes were partitioned into a training set *t* using eighty percent of the route descriptions through random selection and a validation set *v* of the remaining twenty percent. In addition to these data sets, a separate set *w* consisting of eight route descriptions *written down* by a single test subject, one for each recorded route, was used to indicate performance on similar but slightly different data. All data was manually annotated with the correct chunking and attachment.

5.7 Results: Chunker

In general, the LTUs performed better than Naive Bayes for the Chunking task. The best performance was achieved with the Sparse Perceptron learner, as shown in Table 3, where the CER for the validation set (CER_V) and the written data (CER_W) is shown with additive feature sets. It also shows the accuracy of the labels assigned by the classifier. To compare with, a simple majority class baseline accuracy of 52.77 may be devised by looking at the performance of the Naive Bayes classifier with only the word instance as feature.

Features	**$Accuracy_V$**	**CER_V**	**CER_W**
Word Instance	52.44	83.34	67.50
+ Word window	71.25	43.76	32.81
+ Previous tags	87.62	26.50	34.38
+ Word suffix	88.11	25.15	31.25
+ Word prefix	88.44	23.93	31.25
+ POS window	**90.55**	**21.47**	32.19
+ Lemma window	90.39	27.72	**30.31**

Table 3: Performance of the Chunker based on the Sparse Perceptron learner for additive feature sets.

As discussed in section 4, the method chosen to collect the data forces the subject to retrieve the route from memory while describing it, which gives rise to a lot of disfluencies. To test whether it would be beneficial to remove these disfluencies from the data, a simple filter was devised which removed filled pauses and repetitions from both the training and testing data. The results on the Sparse Averaged Perceptron learner with full feature set are shown in Table 4. Interestingly, the behaviour of CER_V indicates that the filled pauses in the spoken data are indeed *useful* in the chunking process, while repeated words are not. This is in line with previous findings that filled pauses are more common at clause boundaries than within clauses (Swerts, 1998). As expected, removing both filled pauses and repeats appear to make the spoken data somewhat more similar to written route descriptions according to CER_W.

Filter	**Acc.**	**CER_V**	**CER_W**
Unfiltered	91.04	25.88	31.56
No repeats	**91.20**	**25.05**	31.56
No filled pauses	90.71	29.75	35.00
No filled pauses or repeats	90.20	30.37	**28.44**

Table 4: Performance of the Chunker based on the Sparse Averaged Perceptron learner with filtered data, full feature set.

5.8 Results: Chunker + Attacher

The best performance in the Attacher is achieved with the Sparse Averaged Perceptron, as shown in Table 5. It is important to note that the Attacher has to base its classification on the erroneous output of the Chunker. As can be seen, the best performance (25.60) is relatively close to that of the Chunker (21.47), which indicates that most errors are introduced in the chunking stage.

Features	**CER$_V$**	**CER$_W$**
Chunk label	50.60	68.87
+ Chunk label window	36.75	67.61
+ Lemmas	29.52	41.19
+ Bag of lemmas	26.20	**39.94**
+ Suffixes	**25.60**	40.57

Table 5: The performance of the Attacher based on the Sparse Averaged Perceptron learner, for additive feature sets.

For this feasibility study, we have used a relatively small amount of data and the question is how much room there is for improvement, given that more data was provided. Figure 5 shows how the CER of the Attacher decreases as more data is added. The slope of the curve indicates that the performance is likely to improve if even more data is added.

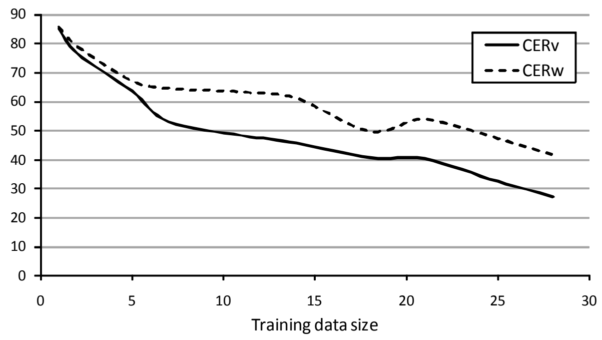

Figure 5: The performance of the Chunker and Attacher depending on the amount of data used for training.

6 Conclusions and Future work

Given the limited amount of data, the results show promising performance (25.6% CER on unseen data). Most problems are introduced in the chunking stage, but the results indicate that more data is likely to improve the performance. The next step is to apply the method to a larger corpus and to another language. The IBL corpus is a good candidate here (Kyriacou et al., 2005). In this experiment, we simply concatenated the user's utterances and did not use the Wizard's. The performance of the chunker is likely to improve if the Wizard's utterances are also taken into account.

One important step that we have not addressed yet is that of automatic speech recognition. Although a lot more errors are likely to be introduced, a data-driven approach for semantic interpretation is much more likely to degrade gracefully than a grammar-based approach. This, however, will need to be investigated in future studies. Another important issue we have yet to investigate is that of confidence scores in the interpretation step. As discussed in section 2, accurate confidence scoring is a vital issue for the approach that we will take in the IURO project. There are also other machine learning approaches that have shown good performance in chunking tasks which we will investigate, such as Hidden Markov Models and Conditional Random Fields (Collins, 2002). The method should also lend itself to incremental processing (Schlangen & Skantze, 2011). While some features used in the experiment are part of the right context, we have not directly addressed the question of how much these contribute to the performance. It is also possible to revise previous output as more input is processed (Ibid.).

As a feasibility study, we think that the results indicate that it is indeed possible to get people to freely describe routes which can be automatically interpreted into a route graph that may be useful for robot navigation. We also hope that the novel approach to semantic chunking parsing presented here will inspire others to similar approaches in other domains.

Acknowledgments

This work has been carried out at the Centre for Speech Technology at KTH, and is supported by the European Commission project IURO (Interactive Urban Robot), grant agreement no. 248314. The authors would also like to thank the reviewers for their valuable comments.

References

Abney, S. (1991). Parsing by chunks. In Berwick, R. C., Abney, S. P., & Tenny, C. (Eds.), *Principle-Based Parsing: Computation and Psycholinguistics* (pp. 257 278). Dordrecht: Kluwer.

Bauer, A., Klasing, K., Lidoris, G., Mühlbauer, Q., Rohrmüller, F., Sosnowski, S., Xu, T., Kühnlenz, K., Wollherr, D., & Buss, M. (2009). The autonomous city explorer: Towards natural human-robot interaction in urban environments. *International Journal of Social Robotics, 1*(2), 127-140.

Collins, M. (2002). Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proceedings of ACL* (pp. 1-8). Philadelphia, PA.

Domeij, R., Knutsson, O., Carlberger, J., & Kann, V. (1999). Granska an efficient hybrid system for swedish grammar checking. In *Proceedings of Nordic Conference of Computational Linguistics* (pp. 49-56). Trondheim, Norway.

Gorin, A. L., Riccardi, G., & Wright, J. H. (1997). How may I help you?. *Speech Communication, 23*, 113-127.

Gustafson, J., Heldner, M., & Edlund, J. (2008). Potential benefits of human-like dialogue behaviour in the call routing domain. In *Proceedings of Perception and Interactive Technologies for Speech-Based Systems (PIT 2008)* (pp. 240-251). Berlin/Heidelberg: Springer.

He, Y., & Young, S. (2006). Spoken language understanding using the hidden vector state model. *Speech Communication, 48*(3-4), 262-275.

Jurafsky, D., & Martin, J. (2000). *Speech and language processing.* Englewood, NJ, US: Prentice Hall, Inc.

Kyriacou, T., Bugmann, G., & Lauria, S. (2005). Vision-based urban navigation procedures for verbally instructed robots. *Robotics and Autonomous Systems, 51*(1), 69-80.

Littlestone, N. (1988). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine learning, 2*(4), 285-318.

Mandel, C., Frese, U., & Rofer, T. (2006). Robot navigation based on the mapping of coarse qualitative route descriptions to route graphs. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 205-210). Beijing, China.

Meza-Ruiz, I. V., Riedel, S., & Lemon, O. (2008). Accurate statistical spoken language understanding from limited development resources. In *Proceedings of ICASSP 2008* (pp. 5021-5024). Las Vegas, Nevada.

Müller, R., Röfer, T., Lankenau, A., Musto, A., Stein, K., & Eisenkolb, A. (2000). Coarse qualitative descriptions in robot navigation. In Freksa, C., Brauer, W., Habel, C., & Wender, K-F. (Eds.), *Spatial Cognition II* (pp. 265-276). Springer.

Rieser, H., & Poesio, M. (2009). Interactive gesture in dialogue: a PTT model. In *Proceedings of SIGdial* (pp. 87-96). London, UK.

Rizzolo, N., & Roth, D. (2010). Learning Based Java for Rapid Development of NLP Systems. *Language Resources and Evaluation*.

Sang, E. F. T. K., & Buchholz, S. (2000). Introduction to the CoNLL-2000 shared task: Chunking. In *Proceedings of CoNLL* (pp. 127-132). Lisbon, Portugal.

Schlangen, D., & Skantze, G. (2011). A General, Abstract Model of Incremental Dialogue Processing. *Dialogue & Discourse, 2*(1), 83-111.

Skantze, G., & Edlund, J. (2004). Robust interpretation in the Higgins spoken dialogue system. In *Proceedings of ISCA Tutorial and Research Workshop (ITRW) on Robustness Issues in Conversational Interaction*. Norwich, UK.

Skantze, G., & Hjalmarsson, A. (2010). Towards Incremental Speech Generation in Dialogue Systems. In *Proceedings of SIGdial*. Tokyo, Japan.

Skantze, G. (2005). Exploring human error recovery strategies: implications for spoken dialogue systems. *Speech Communication, 45*(3), 325-341.

Swerts, M. (1998). Filled pauses as markers of discourse structure. *Journal of Pragmatics, 30*(4), 485-496.

Wong, Y. W., & Mooney, R. (2007). Learning synchronous grammars for semantic parsing with lambda calculus. In *Proceedings of ACL-07* (pp. 960-967). Prague, Czech Republic.