Transformation-based Learning for Semantic parsing

F. Jurčíček, F. Mairesse, M. Gašić, S. Keizer, B. Thomson, K. Yu, and S. Young

Engineering Department
Cambridge University
Trumpington Street, Cambridge, CB2 1PZ, UK

{fj228, farm2, mg436, sk561, brmt2, ky219, sjy}@eng.cam.ac.uk

Abstract

In this paper, we present a semantic parser which transforms initial naive semantic hypothesis into correct semantics by using an ordered set of rules. These rules are learned automatically from the training corpus with no prior linguistic knowledge.

We show that TBL parser is competitive to the state-of-the-art semantic parser on the ATIS task without using any handcrafted linguistic knowledge.

Check how you call STEP/TBL parser!

Should I define semantic parsing? Parsing at all??

1 Introduction

Semantic parsing is important part of a spoken dialogue system (Williams and Young, 2007). The goal of a SLU module is to map natural language to formal meaning representation (semantics). Such semantics can be either defined by a grammar, e.g. LR grammar for GeoQuery domain (Kate, 2005) or frames with slots. The semantics, which is designed by a domain expert, is directly executable by a question answering system (Wong and Mooney, 2006) or by a dialogue manager in a spoken dialogue system. As dialogue managers, e.g. (Thomson et al, 2008), commonly use semantics in the form frames and slots (see Table 1), our approach learns how to directly map from natural language into frame and slot semantics.

A dialogue system needs a semantic parser which is acurate and robust, easy to build, and fast. First, a parser presented in this paper performs comparable to the state-of-the-art semantics parser and it can deal with ill formed utterances. Second, it does not need any handrafted linguistic knowledge and it can learn from data which is not semantically annotated at the word-level. Finaly, it learns a compact set of rules which results in limited number of operation per semantic concept.

"what are the lowest airfare from Washington DC to Salt Lake City"

GOAL	=	airfare
from.city	=	Washington
from.state	=	DC
to.city	=	Salt Lake City

Table 1: Example of frame and slot semantics from ATIS (Dahl et al, 1994) dataset.

In our approach, we adapt transformation-based learning (TBL) (Brill, 1995) to the problem of semantic parsing. We attempt to find an ordered list of transformation rules to improve a naive semantic annotation.

In the next section, we describe previous work on maping natural language into fromal meaning representation. Section 3.1 presents an example of TBL based semantic parsing. Section 3.3 describes the learning process. Section 4 compares TBL parser to the previously developed semantic parsers on ATIS (Dahl et al, 1994) and TownInfo (Williams and Young, 2007; Thomson et al, 2008) datasets.

2 Related work

In Section 4, we compare performance of our method with the four existing systems that were evaluated on the same dataset we consider.

First, hidden Vector State model (He and Young, 2006) have been used to model an approximation of a pushdown automaton which has semantic concepts as non-terminal symbols. Consequently, a deterministic algorithm was used to recover slot values.

Second, automatic induction of combinatory categorical grammar (Zettlemoyer and Collins, 2007) have been used to map sentences to lambda-calculus. The combinatory categorical grammar is generalized into probabilistic model by learning log-linear model. An online learning algorithm update weights of features representing a parse tree of an input sentence. They show that their technique produces the state-of-the-art performance on the Air Travel Information (ATIS) dataset (Dahl et al, 1994). **However,** apart from using the the lexical categories (city names, airport names, etc) readily available from the ATIS corpus, they also need considerable number of handcrafted entries in their initial lexicon.

Third, Markov logic networks (Meza et al, 2008a; Meza et al, 2008b) have been used to extract slot values by applying ideas of a Markov network to first-order logic. In this approach, weights are attached to first-order clauses which represent relationship between slot names and its values. Such weighted clauses are used as templates for features of Markov networks.

Finally, support vector machines (Mairesse et al, 2009) have been used to build a semantic trees by recursive calling classifiers to estimate probabilities of production rules using a linear kernel and word based features.

There has been lso done a large amount of research into maping natural language to semantics that is not directly comparable because it uses eaither different corpora or different meaning representation.

Wong (Wong and Mooney, 2006) used machine translation techniques with a syntax-based translation model based on the synchronous context-free grammars. Inductive logic programing (Tang et al, 2001) have been used to incrementally develop a theory including a set of predicates. In each iteration, the predicates were generalized from predicates in the theory and predicates automatically constructed from examples.

Transformation techniques (Kate, 2005) have been used to sequentially rewrite an utterance into semantics. Our approach differ in the way how the semantics is constructed. Instead of rewriting an utterance, we transform initial naive semantic hypothesis. As a result, we can use in input words several times to trigger transformations of the output. This extends our ability to handle non-compositionality phenomena in spoken language.

Kate (Kate, 2008) used support vector machines and tree kernels to integrate knowledge contained in the gold standatrd dependency trees to capture long-range relationship between words.

3 Transformation-based parsing

This section describes the transformation-based parser.

A transformation parsing assumes:

- Existence of initial annotation.
- Transformations are applied in sequence.
- Transforamtions progresively change from general to specific.
- Results of previous transformations are visible to following transformations.
- Transformations correct errors of the previous transformations.

First of all, we give an example of the parsing algorithm. Secondly, we describe rule templates used to generate rules for the rule inference process. Finally, we detail the learning process.

3.1 Example of Parsing

The semantic parser transforms initial naive semantic hypothesis into correct semantics by applying rules from an ordered set of rules. Each rules is composed of a trigger and a transforamtion. A trigger initiate a transformation of the hypothesis.

The parsing consists of **three** steps:

1. initial semantics is assigned as hypothesis

2. sequentially apply all rules[1]

3. output hypothesis semantics

We demonstrate the parsing on an example. Think of the utterance: *"find all the flights between toronto and san diego that arrive on saturday"*

First, the goal "flight" is used as the initial goal because it is the most common goal in the ATIS dataset and no slots are added in the semantics.

GOAL	=	flight

Secondly, the rules whose triggers match the sentence and the hypothesis are sequentially applied.

trigger	transformation
"between toronto and"	add slot "from.city=Toronto"
"and san diego"	add slot "to.city=San Diego"
"saturday"	add slot "departure.day=Saturday"

Conesquently, we obtain the following semantic hypothesis.

GOAL	=	flight
from.city	=	Toronto
to.city	=	San Diego
departure.day	=	Saturday

The trigger **"and Sand Diego" is example of non-compositionality**, in which the words in an utterance do not have a one-to-one correspondence with the slots in the semantics. The word "and" indicates that the city "San Diego" is slot value of the slot "to.city".

As the TBL method tends to learn and apply general rules first, the parser learns to assicate the date and time values with the "departure*" slots becasue the date and time values are mostly associated with slots describing properties of departure in the ATIS dataset. The icorrent classification of the word "saturday" is a result of such generalization.

However, TBL method learns to correct its errors. Therefore, the parser also applies at laiter stage of parsing the error correcting rules.

[1]Input utterance is not modified by rules. As a result, words from the utterance can be trigger several different transformations.

trigger	transformation
"arrive"	substitute slot from "departure.day=*" to "arrive.day=*"

In this case, we subtitute the slot name with the correct values. The final semantic hypothesis is as follows.

GOAL	=	flight
from.city	=	Toronto
to.city	=	San Diego
saturday.day	=	Saturday

Up to now, we considered the utterance as a bag of words and no notion of locality was considered. For example, before we perform the substitution of the slot "departure.day" to "arrival.day", we shoud test whether word "arrive" is in the vicinity of the slot. The reason is that we do not want to trigger the substitution of the slot "from.city=Toronto" to "to.city=Toronto" because the parser can also learn the rule as follows.

trigger	transformation
"arrive"	substitute slot from "from.city=*" to "to.city=*"

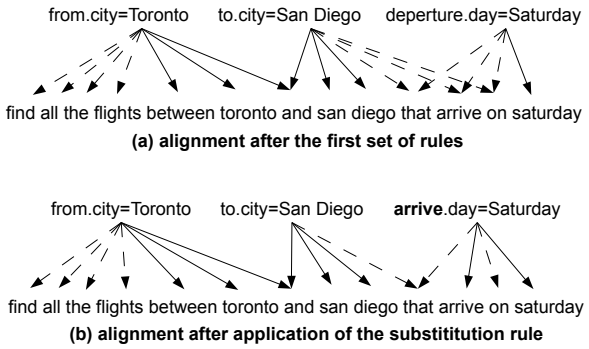

Figure 1: Alignment between words and the slots in the example utterance.

XXX: I mentioned delete transformation but never defined.

To implement this, we constrain triggers performing substitution to be activated only if they are in the vicinity of the slot lexical realization. We track the words from the utterance, which were used in triggers. Anytime we apply a transformation of a slot,

we store links to the words which triggered the transformation. We call this links as direct alignment. For the delete transformation no tracking information is kept because a slot is removed from hypothesis and never used again.

In Figure 1 (a), we see the alignment between the words and the slots in the example utterance after applying the first set of rules. The full lines denote direct alignment created by the add-transformations. The dashed lines denote derived alignment - alignment computed from the direct alignment. Because no rules were triggered by words "find all the flights" and "that arrive on" those words could not be aligned directly to any of the slots, we have to derive such alignment. To compute derived alignment, first we order the slots so that the slot aligned with the left-most word is the first and the ordering results in minimum instances of direct alignment crossing. Than, every unaligned word is aligned with the nearest left and the right slot. In Figure1 (a), the phrase "find all the flights" can be aligned to the slot "from.city=Toronto" only. The phrase "that arrive on" can be aligned to two slots "to.city=San Diego" and "departure.day=Saturday".

In Figure 1 (b), we see the alignment after applying the substitution. We can see change in the alignment of the phrase "that arrive on". First, the word "arrive" is aligned to the slot "arrive.day=Saturday". Second, the word "on" is aligned to the same slot as the word "arrive", otherwise the derived alignment would cross the direct alignment. There is no change in the alignment of the word "that".

3.2 Rule templates

The learning algorithm uses templates to instantiate rules which are subsequently tested by the learning algorithm. Each rule template is composed of a trigger and a transformation.

A trigger controls when a transformation of a hypothesis can be performed. A trigger can question an input utterance, an output semantics, or both. In our method, a trigger contains one or more conditions as follows.

- The utterance contains n-gram N.
- The utterance contains skipping[2] bigram B.
- The semantics dialogue act equals to D.
- The semantics contains slot S.

If a trigger contains more than one condition, then all conditions must be satisfied.

A transformation performs one of these operation:

- Substitute a dialogue act type.
- Add a slot
- Delete a slot
- Substitute a slot

A substitution transformation can either substitute a whole slot, a slot name, or a slot value.

3.3 Learning

The main idea of transformation-based learning is to learn an ordered list of rules which incrementaly improve upon the current state of the training set (see Figure 2). The learning is conducted in a greedy fashion according the objective function. We define the objective function as a measure of the number of errors in our hypothesis. Number of errors includes number of dialogue act substitutions, number of slot insertions, number of slot deletions, number of slot substitutions. As a result, TBL chooses the transformation rule that reduces the errors the most. When no rule that improves the current state of the training set beyond the pre-set threshold can be found, the training ends.

The in initial assignment is made based on simple statistics. **The most common goal is assigned and no slots are added**.

Using TBL approach to solve a classification problem assumes the existence of:

- An initial call assignment. This can be done by assigning the most common simple semantics, e.g. inform().
- A set of templates for rules. These templates determine the predicates the rules will test, and they have the largest impact on the performance of the classifier.

[2]A skipping bigram is bigram which skips one or more words between words in the bigram. For example, ('arrive',*,'Boston') is a skiping bigram which skips one word.

1. INITIAL SEMANTICS IS ASSIGNED AS HYPOTHESIS TO EACH UTTERANCE
2. REPEAT AS LONG AS THE NUMBER OF ERRORS ON THE TRAINING SET DECREASES
   (A) GENERATE ALL POSSIBLE RULES WHICH CORRECT AT LEAST ONE ERROR IN THE TRAINING SET
   (B) MEASURE NUMBER OF CORRECTED ERRORS BY EACH RULE
   (C) SELECT THE RULE WITH THE MOST CORRECTED ERRORS
   (D) APPLY THE SELECTED RULE TO THE CURRENT STATE IF THE TRAINING SET.
   (E) STOP IF THE NUMBER OF CORRECTED ERRORS IS SMALLER THEN 3.
3. PRUNE RULES

Figure 2: Rule transformations learning algorithm.

- An objective function for learning. The typical objective function is the difference in performance resulting from applying the rule.

At the beginning of the learning phase, the training set is first given an initial class assignment. The system then iteratively executes the following steps:

1. Generate all productive rules.
2. For each rule:
   (a) Apply to a copy of the most recent state of the training set.
   (b) Score the result using the objective function.
3. Select the rule with the best score.
4. Apply the rule to the current state of the training set. updating it to reflect the change.
5. Stop if the score is smaller than some pre-set threshold T
6. Repeat from the step 1.

During the decoding phase, the test set is initialized with the same initial class's assignment. Each rule is than applied, in the order it was learned, to the test set. The final classification is the one attained when all rules have been applied.

The make the parser more robust, we increase robustness of the parser by the following steps.

First, the number of possible slot values for each slot is usually very high. As a result, we replace all lexical realizations from the database, available to **a dialogue manager**, by its slot name in the utterance. For example, in utterance "find all the flights from cincinnati to the new york city" the lexical realization are replaced as follows: "please find all the flights from city-0 to the city-1". Similarly, we replace slot values in the semantics.

Secondly, to limit overfitting the training data, we prune the rules which are learned at the end of the learning. We sequentially apply each rule on the development set. And we chose the number of rules for which the parser gets the highest score on the development data.

First of all, very naive rules are learn are for example Classifier learns to correct its errors STEC can delete an incorrect slot STEC can substitute A slot name of an incorrect slot An equal sign of an incorrect slot

To speed up the training process, we select multiple best performing rules and the performance of worst selected rule has to be at least at least 80% of the best rule.

4 Evaluation

In this section, we evaluate our parser on two distinct corpora, and compare our results with the state-of-the-art techniques and handcrafted rule-based parser.

4.1 Datasets

Our first dataset consists of tourist information dialogues in a fictitious town (TownInfo). The dialogues were collected through user trials in which users searched for information about a specific venue by interacting with a dialogue system in a noisy background. These dialogues were previously used for training dialogue management strategies (Williams and Young, 2007; Thomson et al, 2008).

The semantic representation of the user utterance consists of a root dialogue act type and a set of slots which are either unbound or associated with a child value. For example, “What is the address of Char Sue” is represented as request(address=’Char Sue’), and “I would like a Chinese restaurant?” as inform(food=’Chinese’,type=’restaurant’). The TownInfo training, development, and test sets respectively contain 8396, 986 and 1023 transcribed utterances. The data includes the transcription of the top hypothesis of the ATK speech recogniser, which allows us to evaluate the robustness of our models to recognition errors (word error rate = 34.4%).

In order to compare our results with previous work (He and Young, 2006; Zettlemoyer and Collins, 2007), we apply our method to the Air Travel Information System dataset (ATIS) (Dahl et al, 1994). This dataset consists of user requests for flight information, for example “Find flight from San Diego to Phoenix on Monday is rerepresented as flight(from.city=”San Diego”,to.city=”Phoenix”,departure.day=”Monday”). We use 5012 utterances for training, and the DEC94 dataset as development data. As in previous work, we test our method on the 448 utterances of the NOV93 dataset, and the evaluation criteria is the F-measure of the number of reference slot/value pairs that appear in the output semantic (e.g., from.city = New York). He & Young detail the test data extraction process in (He and Young, 2005).

For both corpora are available databases with lexical entries for slot values e.g. city names, airport names, etc.

4.2 Improving disambiguation of long-range dependencies

Besides simple n-grams and skipping bigrams more complex lexical features can be used. (Kate, 2008) used gold standard word dependencies to capture long-range relationship between words. At its simplest, dependencies tree is one of the most concise ways to describe language syntax. Essentially, each word is viewed as the dependent of one other word, with the exception of a single word which that is the root of the sentence. Kate showed that word dependencies significantly improve semantic parsing because long-range dependencies from an utterance tend to be local in a dependency tree. For example,

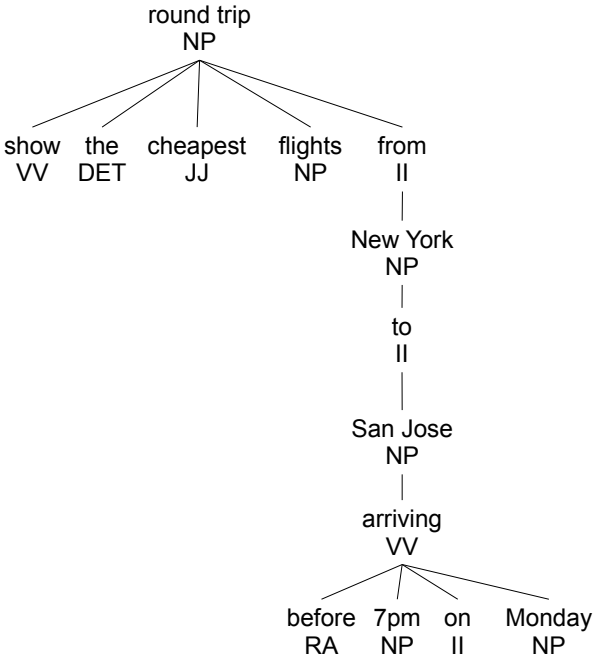

Figure 3: Dependency tree of the sentence ”Show the cheapest flights from New York to San Jose arriving before 7pm on Monday“ generated by the RASP parser (Briscoe et al, 2006).

the words ”arriving“ and ”Monday“ are neighbors in the dependency tree but they are four words apart in the sentence (see Figure 3).

Instead of using gold standard word dependencies, we used dependencies provided by RASP dependency parser (Briscoe et al, 2006). First of all, we had to add capitalization and punctuation into the ATIS data to be able to use the RASP parser. The RASP parser without proper capitalization fails to tag ”new“ and ”york“ as NP and instead of this it tags ”new“ as ”JJ“ and ’york’ as NP and the dependencies generated by the parser are unsatisfactory. Secondly, we generated new n-gram features from dependency trees. Even though the dependencies generated the RASP parser are no absolutely accurate, the new features increase performance in F-measure on ATIS data.

Secondly, we generated long-range features by using POS tags[3] Our motivation was work of (Meza et al, 2008a; Meza et al, 2008b) who handcrafted

[3]We used POS tags provided by the RASP parser; however, any POS tagger can be used instead.

features using words ”arrive“, ”arriving“, ”leave“, and ”leaving“. These handcrafted features disambiguate large number of semantic parsing errors in ATIS data because large portion or errors is caused by confusions between concepts ”arrival.time“ and ”departure.time“, ”arrival.day“ and ”departure.day“, etc. To generalize this approach, we want to automatically find features which could disambiguate words like ”Monday“, ”7pm“, and ”Boston“. As a result, we generate a new type of bigrams for a word and the nearest verb, preposition, etc. We use all parts-of-speech provided by RASP and the learning algorithm chose the most discriminative features. Among those learned are not only the words used by Meza-Rui but also words like ”stop“, ”reach“, ”buy“ and prepositions like ”at“, ”from“, ”to“, etc.

Mention why I do not use tree similarity measure as Kate.

4.3 Results

We also compare our models with the handcrafted Phoenix grammar (Ward, 1991) used in the trials (Williams and Young, 2007; Thomson et al, 2008). The Phoenix parser implements a partial matching algorithm that was designed for robust spoken language understanding.

Parser	Prec	Rec	F
TownInfo dataset with transcribed utterances:			
TBL	96.05	94.66	95.35
STC	97.39	94.05	**95.69**
Phoenix	96.33	94.22	95.26
TownInfo dataset with ASR output:			
TBL	92.72	83.42	87.82
STC	94.03	83.73	**88.58**
Phoenix	90.28	79.49	84.54
ATIS dataset with transcribed utterances:			
TBL	96.37	95.12	95.74
STC	96.73	92.37	94.50
HVS	-	-	90.3
MLN	-	-	92.99
PCCG	95.11	96.71	**95.9**

Table 2: Slot/value precision (Prec), recall (Rec) and F-measure (F) for the ATIS and TownInfo datasets. TBL parser is compared with Phoenix parser and STC classifier (Mairesse et al, 2009) on the TownInfo dataset and compared with HVS parser (He and Young, 2006), MLN parser (Meza et al, 2008b), STC classifier, and PCCG parser (Zettlemoyer and Collins, 2007) on the ATIS dataset

5 Discussion

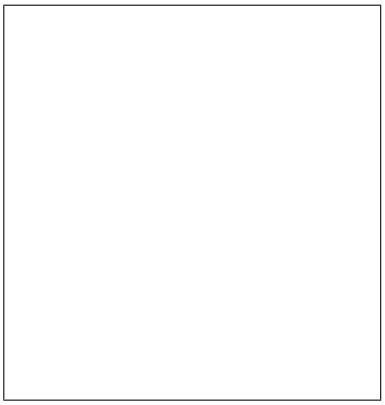

Figure 4: The learning curve shows the relation between number of learned rules and the F-measure for both TI and ATIS corpora.

The number of learned rules is very small. As is shown in the figure 4, learning curves for both training data and development data are very steep. Although our current strategy for choosing the final number of rules for decoding is to keep only the rules for which we obtain highest F-measure on the development data, we could use much less rules without scarifying accuracy. For example, we accepted 0.1% lower F-measure on the development data than we would need only YYY rules in comparison with XXX rules if select the number of rules based in the highest F-measure. In contrast, the initial lexicon the CCG parser (Zettlemoyer and Collins, 2007) contains about 180 complex entries for general English words or phrases and yet additional lexical entries must be learned. **explain better**

Also, the number of rules per semantic concept (dialogue act or slot name) is very low. In TI data, we have XXX different dialogue acts and XXX slot and the average number of rules per semantic concept is XXX. In case of ATIS data, we have XXX dialogue acts and XXX slots and the average number of rules per semantic concept is XXX.

Lexical realizations of a slot can overlap with lexical realization of neighbouring slots. It is shows

to be important pattern, for example in the trigram (city-0,and,city-1) is very common for sentence including "between city-0 and city-1". The lexical realizations city-0, city-1 respectively would be classified as from.city, and city-1 just because we know the

We found that the dialogue act type recognition accuracy of the STEP parser is lower than STC's; as a result, we tried to use SVM as STC does to classify dialogue act types. We believe that STC is better in dialogue act type recognition better because SVN classifier use all features at one time. STC makes decision in one step using all the features rather than making several decisions by several rules as STEP.

We hoped for an increase of F-measure as result of increased dialogue act type accuracy. However, we did not get any increase in F-measure.

6 Conclusion

Our approach adapts TBL to the problem of word-level alignment by examining word features as well as neighboring links.

Acknowledgments

We would like to thank to Luke Zettlemoyer and I.V. Meza-Ruiz for their help with understanding their methods.

References

E. Brill. 1995. *Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging*. Computational Linguistics, 21(4):543?565.

E. Briscoe, J. Carroll and R. Watson. 2006. *The Second Release of the RASP System*. Proceedings of COLING/ACL.

D.A. Dahl, M. Bates, M. Brown, W. Fisher, K. Hunicke-Smith, D. Pallett, C. Pao, A. Rudnicky, and E. Shriberg. 1994. *Expanding the scope of the ATIS task: The ATIS-3 corpus*. Proceedings of the ARPA HLT Workshop.

Y. He and S. Young. 2005. *Semantic processing using the hidden vector state model*. Computer Speech & Language,vol. 19, no. 1, pp. 85-106.

Y. He and S. Young. 2006. *Spoken language understanding using the hidden vector state model*. Computer Speech & Language, vol. 19, no. 1, pp. 85-106.

Y.W. Wong and R.J. Mooney. 2006. *Learning for Semantic Parsing with Statistical Machine Translation*. Proceedings of HLT/NAACL.

R.J. Kate, Y.W. Wong and R.J. Mooney. 2005. *Learning to Transform Natural to Formal Languages*. Proceedings of AAAI.

R.J. Kate. 2008. *A Dependency-based Word Subsequence Kernel*. Proceedings of EMNLP.

F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young. 2009. *Spoken Language Understanding from Unaligned Data using Discriminative Classification Models*. Proceedings of ICASSP.

I.V. Meza-Ruiz, S. Riedel and O. Lemon. 2008. *Accurate statistical spoken language understanding from limited development resources*. Proceedings of ICASSP.

I.V. Meza-Ruiz, S. Riedel and O. Lemon. 2008. *Spoken Language Understanding in dialogue systems, using a 2-layer Markov Logic Network: improving semantic accuracy*. Proceedings of Londial.

L.R. Tang and R. J. Mooney 2001. *Using Multiple Clause Constructors in Inductive Logic Programming for Semantic Parsing*. Proceedings of ECML.

B. Thomson, M. Gasic, S. Keizer, F. Mairesse, J. Schatzmann, K. Yu, and S. Young. 2008. *User study of the Bayesian update of dialogue state approach to dialogue management*. Proceedings of Interspeech.

W.H. Ward. 1991. *The Phoenix system: Understanding spontaneous speech*. Proceedings of ICASSP.

J. Williams and S. Young. 2007. *Partially observable markov decision processes for spoken dialog systems*. Computer Speech and Language, vol. 21, no. 2, pp. 231-422.

L.S. Zettlemoyer and M. Collins. 2005. *Online learning of relaxed CCG grammars for parsing to logical form*. Proceedings of EMNLP-CoNLL.