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.

Index Terms: spoken language understanding, semantics, nature language processing

Check how you call STEP/TBL parser! Should I define semantic parsing? Parsing at all?? Check how you use input, output, hypothesis semantics!

1. Introduction

Semantic parsing is important part of a spoken dialogue system [1]. The goal of a semantic parsing 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 [2] or by frames and slots e.g. TownInfo domain [3]. In Table 1, it is shown an example of the frame and slot semantics from ATIS dataset. Each frame has a goal and set of slots. Each slot is composed of a slot name, e.g. "from.city", a slot equal sign, e.g. "=" or "!=", and slot value, e.g. "Washington". As dialogue managers commonly use semantics in the form frames and slots, our approach learns how to directly map from natural language into frame and slot semantics.

A dialogue system needs a semantic parser which is accurate 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 handcrafted linguistic knowledge and it can learn from data which is not semantically annotated at the word-level. Finally, it learns a compact set of rules which results in limited number of operation per semantic concept.

In our approach, we adapt transformation-based learning (TBL) [4] to the problem of semantic parsing. We attempt to find an ordered list of transformation rules to improve a naive semantic annotation. A transformation parsing assumes:

- Initial annotation is obtained by a naive annotator.
- Transformations are applied in sequence.
- Transformations progressively change from general to specific.
- Results of previous transformations are visible to following transformations.
- Transformations correct errors of the previous transformations.

"what are the lowest airfare from Washington DC to Salt Lake City"		
GOAL	=	airfare
airfare.type	=	lowest
from.city	=	Washington
from.state	=	DC
to.city	=	Salt Lake City

Table 1: Example of frame and slot semantics from ATIS [5] dataset.

In the next section, we describe previous work on mapping natural language into formal meaning representation. Section 3 presents an example of TBL based semantic parsing, discuss templates for transformation rules, and describes the learning process. Section 4 compares TBL parser to the previously developed semantic parsers on ATIS [5] and TownInfo [6] 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 [7] 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 [8] 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 [5]. **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 [9, 10] 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 [6] 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 also done a large amount of research into mapping natural language to semantics that is not directly comparable because it uses either different corpora or different meaning representation.

Wong [11] used machine translation techniques with a syntax-based translation model based on the synchronous context-free grammars. Inductive logic programing [12] 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 [2] have been used to sequentially rewrite an utterance into semantics. However, 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 input words several times to trigger transformations of the semantics. This extends our ability to handle non-compositionality phenomena in spoken language.

Kate [13] used support vector machines and tree kernels to integrate knowledge contained in the gold standard dependency trees to capture long-range relationship between words.

3. Transformation-based parsing

This section describes the transformation-based parser. First of all, we give an example of the parsing algorithm. Secondly, we describe templates used to generate rules for the 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 transformations from an ordered set of rules. Each rule is composed of a trigger and a transformation and a trigger initiates a transformation of a 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. As a result, the initial semantics is as follows:

GOAL	=	flight

Secondly, the rules whose triggers match the sentence and the hypothesis are sequentially applied. Generally rules add slots, delete slots, and substitute slots. However, in this example the matching rules are only those which add slots.

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

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"

As a result of the transformations, 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 associate the date and time values with the "departure*" slots because the date and time values are mostly associated with slots describing properties of departure in the ATIS dataset. The incorrect 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 later stage of parsing the error correcting rules. For example, the following rule corrects the slot name of the slot value "Saturday".

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

In this case, we substitute 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

3.2. Slot alignment

So far we considered an 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 should 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=*"

One way to deal with this problem is to 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. Every time we apply a transformation of a slot, we store links between the words which triggered the transformation and the target slot. Such

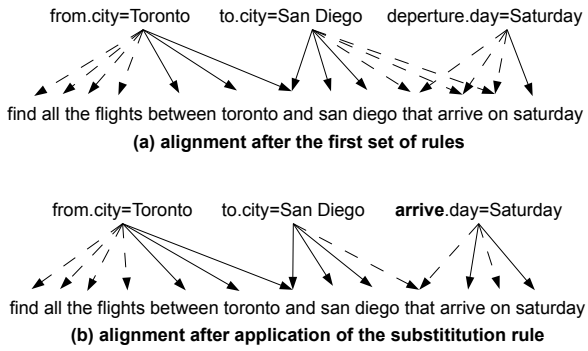

Figure 1: Alignment between words and the slots in the example utterance.

links are called as direct alignment.

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" must be 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".

3.3. Rule templates

In the previous section, we presented several rules which add, substitute, or delete slots. These rules were selected by a learning algorithm from a large set of all potential rules. Such rules are generated from a set of templates for triggers and transformations. The templates define what types of rules can be used for parsing.

A trigger controls when a transformation of a semantic hypothesis can be performed and it 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 goal equals to G.
- The semantics contains slot S.

[2]In a skipping bigram, one or more words are skipped between two words. For example, ('arrive',*,'Boston') is a skipping 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 LARGEST NUMBER OF CORRECTED ERRORS
   (D) APPLY THE SELECTED RULE TO THE CURRENT STATE OF THE TRAINING SET.
   (E) STOP IF THE NUMBER OF CORRECTED ERRORS IS SMALLER THEN THRESHOLD T.
3. PRUNE RULES

Figure 2: Rule learning algorithm.

If a trigger contains more than one condition, then all conditions must be satisfied. A transformation performs one of these operations:

- Substitutes a goal to G.
- Adds a slot S.
- Deletes a slot S
- Substitutes a slot S.

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

3.4. Learning

The main idea of transformation-based learning is to learn an ordered list of rules which incrementally improve initially assigned naive semantic hypothesis (see the algorithm in Figure 2) and the initial assignment is made based on simple statistics. The semantics with the most common goal slots are added is used as initial semantics. The learning is conducted in a greedy fashion and at each step TBL chooses the transformation rule that reduces the largest number of errors in our hypothesis. Number of errors includes number of goal substitutions, number of slot insertions, number of slot deletions, number of slot substitutions. The learning ends when no rule that improves the hypothesis beyond the pre-set threshold can be found.

To limit overfitting the training data, we prune some rules which are learned at the end of the learning. We sequentially apply each rule on the development set and we measure the number of errors. At the end, we chose the N first rules for which the parser gets the lowest number of errors.

As in the previous work [8, 6, 10], we make use of database with lexical realizations of some slots, for example city and airport names. The use of such database results in more robust parser because the number of possible slot values for each slot is usually very high. In the input utterance, we replace lexical realizations of slot values with category labels, e.g. "i want to fly from CITY". Similarly, we replace slot values in the semantics.

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. We found that selection of multiple rules during learning does not affect the performance of the parser and at the same time it decrease the learning time.

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

In order to compare our results with previous work [7, 8, 10, 6], we apply our method to the Air Travel Information System dataset (ATIS) [5]. This dataset consists of user requests for flight information, for example "find flight from San Diego to Phoenix on 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 [14].

Our second 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 [1, 3].

For example, "what is the address of Char Sue" is represented as

GOAL	=	request
address	=	Char Sue

and "I would like a Chinese restaurant?" as

GOAL	=	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%). We compare our model with STC praser [6] and the handcrafted Phoenix grammar [15] used in the trials [1, 3]. The Phoenix parser implements a partial matching algorithm that was designed for robust spoken language understanding.

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 [13] used gold standard word dependencies to capture long-range relationship between words. At its simplest, dependency trees are one of the most concise way 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, the words "arriving" and "Monday" are neighbors in the dependency tree but they are four words apart in the sentence (see Figure 3).

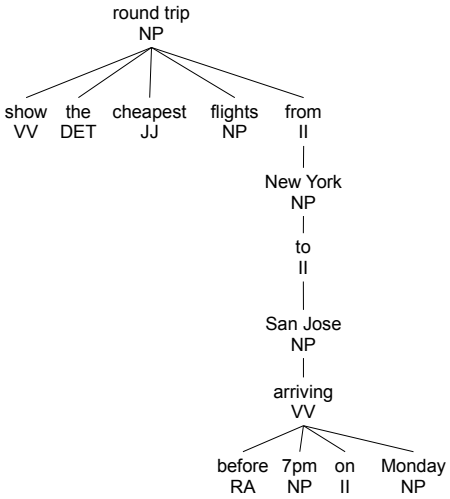

Figure 3: Dependency tree of the sentence "Show the cheapest flights from New York to San Jose arriving before 7pm on Monday" generated by the RASP parser [16].

Instead of using gold standard word dependencies, we used dependencies provided by RASP dependency parser [16]. 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 [9, 10] who handcrafted 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. We generalized this approach and we automatically find features which disambiguate semantics of 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 chooses 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.** For example, for the nearest verb for the word "Morning" in the sentence from Figure 3 is "arrive" and such bigram would look be written as ('arrive','VV','Monday') where VV stands for verb. This features assumes that the left-to-right tendency is dominant and the words in vicinity of lexical realization of a slot value affect the meaning the most.

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

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
PCCG	95.11	96.71	**95.9**
STC	96.73	92.37	94.50
HVS	-	-	90.3
MLN	-	-	92.99

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 [6] on the TownInfo dataset and compared with HVS parser [7], MLN parser [10], STC classifier, and PCCG parser [8] on the ATIS dataset

4.3. Results

The results for both detasets are shown in Table 2. The model accuracy is measured in terms of F-measure of the slot/value pairs. Both slot and the value must be correct to count as correct classification. We report precision, recall, and F-mesure (harmonic mean of precision and recall).

Results on the AITS dataset show that our method with F-measure 95.74% is competitive with rerspect to Zettlemoyer & Collins' PCCG model [8] (F-mesure=95.9%). Note that PCCG model makes use of considerable large number of handracfted entries in their initial lexicon. In addition, our method ouperforms STC, MLN and HVS parsers.

Concerning the TownInfo dataset, Table 2 shows that TBL produces 87.87% F-mesure, wich represents a 3.28%improvemnt over achieved by the handcrafted Phoenix parser, but 0.76% lower compared with STC model.

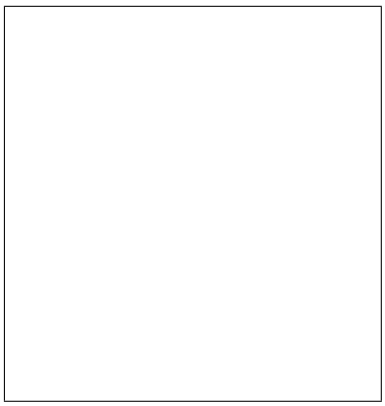

Figure 4: The learning curve shows the relation between number of learned rules and the F-measure for both TI and ATIS corpora.

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 [8] 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

Speed

5. Conclusion

This paper presents novel application of TBL for semantic parsing. Our method learns sequence of rules which transforms initial naive semantics into correct semantics. It significantly differ from the method presented by Kate et al [2] where they were rewriting utterance and replacing its words with semantic concepts.

Our method was applied to two very different domains and it was shown that it performs competitive on both datasets with respect to the state-of-the-art semantic parsers. Results show that our method is comeptitive with respect to Zettlemoyer & Collins' PCCG model [8] on ATIS dataset. In addition, our method outperforms STC, MLN and HVS by 1.27%, 2.75%, and 5.44% respectively. We also show that our method outperforms the handcrafted Phoenix parser on ASR output and it is competitive with respect to STC on TownInfo dataset - TBL's performance is only 0.76% lower compared with STC model.

6. Acknowledgments

We would like to thank to Luke Zettlemoyer and I.V. Meza-Ruiz for valuable discussions about their methods.

7. References

[1] J. Williams and S. Young., "Partially observable markov decision for spoken dialog systems," *Computer Speech and Language*, vol. 21, no. 2, pp. 231–422, 2007.

[2] R. Kate, Y. Wong, and R. Mooney, "Learning to transform natural to formal languages," in *Proceedings of AAAI*, 2005.

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

[4] E. Brill, "Transformation-based error-driven learning and natural language processing: A case study in part-of-speech tagging," *Computational Linguistics*, vol. 21, no. 4, pp. 543–565, 1995.

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

[6] F. Mairesse, M. Gasic, F. Jurcicek, S. Keizer, B. Thomson, K. Yu, and S. Young, "Spoken language understanding from unaligned data using discriminative classification models," in *Proceedings of ICASSP*, 2009.

[7] Y. He and S. Young, "Spoken language understanding using the hidden vector state model," *Computer Speech & Language*, vol. 19, no. 1, pp. 85–106, 2008.

[8] L. Zettlemoyer and M. Collins, "Online learning of relaxed ccg grammars for parsing to logical form," in *Proceedings of EMNLP-CoNLL*, 2005.

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

[10] ——, "Spoken language understanding in dialogue systems, using a 2-layer markov logic network: improving semantic accuracy," in *Proceedings of Londial*, 2008.

[11] Y. Wong and R. Mooney, "Learning for semantic parsing with statistical machine translation," in *Proceedings of HLT/NAACL*, 2006.

[12] L. Tang and R. Mooney, "Using multiple clause constructors in inductive logic programming for semantic parsing," in *Proceedings of ECML*, 2001.

[13] R. Kate, "A dependency-based word subsequence kernel," in *Proceedings of EMNLP*, 2008.

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

[15] W. Ward, "The phoenix system: Understanding spontaneous *Proceedings of ICASSP*, 1991.

[16] E. Briscoe, J. Carroll, and R. Watson, "The second release of the rasp system," in *Proceedings of COLING/ACL*, 2006.