Transformation-based Learning for Semantic parsing

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

Engineering Department, Cambridge University, CB2 1PZ, UK

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

Abstract

This paper presents a semantic parser that transforms an initial semantic hypothesis into the correct semantics by applying an ordered list of transformation rules. These rules are learnt automatically from a training corpus with no prior linguistic knowledge and no alignment between words and semantic concepts. The learning algorithm produces a compact set of rules which enables the parser to be very efficient while retaining high accuracy. We show that this parser is competitive with respect to the state-of-the-art semantic parsers on the ATIS and TownInfo tasks.

Index Terms: spoken language understanding, semantics, natural language processing, transformation-based learning

1. Introduction

The goal of semantic parsing is to map natural language to a formal meaning representation - semantics. Such semantics can be either defined by a grammar, e.g. LR grammar for the GeoQuery domain [1], or by frames and slots, e.g. the TownInfo domain [2]. Table 1 shows an example of the frame and slot semantics from the ATIS dataset [3]. Each frame has a goal and a set of slots. Each slot is composed of a slot name, e.g. “from.city”, and a slot value, e.g. “Washington”. As dialogue managers commonly use semantics in the form of frames and slots [4, 5], our approach learns to map directly from natural language into the frame and slot semantics.

A dialogue system needs a semantic parser which is accurate and robust, easy to build, and fast. This paper presents a parsing technique which provides state-of-the-art performance and robustness to ill formed utterances. The parser does not need any handcrafted linguistic knowledge and it learns from data which has no alignment between words and semantic concepts. Finally, it learns a compact set of rules that allow it to perform real-time semantic parsing. Note that modern statistical dialogue systems typically exploit multiple ASR hypotheses. Hence, the semantic parser has to process an N-best list of user utterances every turn where N$\sim$10 to 100.

In our approach, we adapt Transformation-Based Learning (TBL) [6] to the problem of semantic parsing. We attempt to find an ordered list of transformation rules which iteratively improve the initial semantic annotation. In each iteration, a transformation rule corrects some of the remaining errors in the semantics. To handle long-range dependencies between words, we experiment with features extracted from dependency parse trees provided by the RASP syntactic parser [7].

In the next section, we describe previous work on mapping natural language into a formal meaning representation. Section 3 presents an example of TBL semantic parsing and describes the learning process. Section 4 compares the TBL parser to the previously developed semantic parsers on the ATIS [3] and TownInfo [2] domains. Finally, Section 5 concludes this work.

what are the lowest airfare from Washington DC to Boston		
GOAL	=	airfare
airfare.type	=	lowest
from.city	=	Washington
from.state	=	DC
to.city	=	Boston

Table 1: Example of frame and slot semantics from the ATIS dataset [3].

2. Related work

In Section 4, we compare the performance of our method with four existing systems that were evaluated on the same dataset. First, the Hidden Vector State (HVS) technique has been used to model an approximation of a pushdown automaton with semantic concepts as non-terminal symbols [8, 9]. Second, a Probabilistic parser using Combinatory Categorical Grammar (PCCG) has been used to map utterances to lambda-calculus [10]. This technique produces state-of-the-art performance on the ATIS dataset. However, apart from using the lexical categories (city names, airport names, etc) readily available from the ATIS corpus, this method also needs a considerable number of handcrafted entries in its initial lexicon. Third, Markov Logic Networks (MLN) have been used to extract slot values by combining probabilistic graphical models and first-order logic [11]. In this approach, weights are attached to first-order clauses which represent the relationship between slot names and their values. Such weighted clauses are used as templates for features of Markov networks. Finally, Semantic Tuple Classifiers (STC) based on support vector machines have been used to build semantic trees by recursively calling classifiers that predict fragments of the semantic representation from n-gram features [2].

In addition to the above, there is a large amount of research that is related but not directly comparable because of difference in corpora or meaning representation. For example, transformation techniques have been previously used to sequentially rewrite an utterance into semantics [1]. However, our approach differs in the way the semantics is constructed. Instead of rewriting an utterance, we transform an initial semantic hypothesis. As a result, the words in the utterance can be used several times to trigger transformations of the semantics.

3. Transformation-based parsing

The TBL parser transforms an initial semantic hypothesis into the correct semantics by applying transformations from a list of rules. Each rule is composed of a trigger and a transformation. The trigger is matched against both the utterance and the semantic hypothesis, and when successfully matched, the transformation is applied to the current hypothesis.

In the TBL parser, a trigger contains one or more conditions as follows: the utterance contains N-gram N, the goal equals G, and the semantics contains slot S. If a trigger contains more than one condition, then all conditions must be satisfied. N-gram triggers can be unigrams, bigrams, trigrams or skipping bigrams which can skip up to 3 words. A transformation performs one of the following operations: replace the goal, add a slot, delete a slot, and replace a slot. A replacement transformation can replace a whole slot, a slot name, or a slot value.

Some example rules with triggers composed of unigrams, skipping bigrams, and goal matching are:

trigger	transformation
"tickets"	replace the goal by "airfare"
"flights * from" & GOAL=airfare	replace the goal by "flight"
"Seattle"	add the slot "to.city=Seattle"
"connecting"	replace the slot "to.city=*" by "stop.city=*"

The first rule replaces the goal by "airfare" if the word "tickets" is in the utterance. The second rule changes the goal from "airfare" to "flight" if the utterance contains the words "flights" and "from", which can be up to 3 words apart. The fourth rule adds the slot "to.city=Seattle" whenever the utterance contains the word "Seattle". Finally, every slot name "to.city" is replaced by "stop.city" if the utterance includes the word "connecting".

In the next section, we give an example of how the parsing algorithm works. Then, we detail locality constraints on the transformation rules. Next, we describe features capturing long-range dependencies. Finally, the automatic learning process is described.

3.1. Example of Parsing

This section demonstrates the parsing process on the example: *"find all the flights between Toronto and San Diego that arrive on Saturday"*

First, the goal "flight" with no slots is used as the initial semantics because it is the most common goal in the ATIS dataset. As a result, the initial semantics is as follows:

GOAL	=	flight

Second, the rules, whose triggers match the utterance and the hypothesised semantics, are sequentially applied.

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

After applying the transformations, we obtain the following semantic hypothesis:

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

As the date and time values are associated with the "departure.*" slots most of the time in the ATIS dataset, the parser learns to associate them with the "departure.*" slots. The incorrect classification of the word "Saturday" is a result of such a generalisation. However, the TBL method learns to correct its errors. Therefore, the parser also applies the error correcting rules at a later stage. For example, the following rule corrects the slot name of the slot value "Saturday".

#	trigger	transformation
4	"arrive"	replace the slot "departure.day=*" by "arrival.day=*"

In this case, we substitute the slot name with the correct name, to produce the following semantic hypothesis:

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

3.2. Locality constraints

So far the relationship between slots and their lexical realisation has not been considered. For example, before we replace the slot "departure.day" by "arrival.day", we should test whether the word "arrive" is near the slot's lexical realisation. Otherwise we may accidentally trigger the substitution of the slot "from.city=Toronto" by "to.city=Toronto". This could happen if the parser had also learnt the following rule:

#	trigger	transformation
5	"arrive"	replace the slot "from.city=*" by "to.city=*"

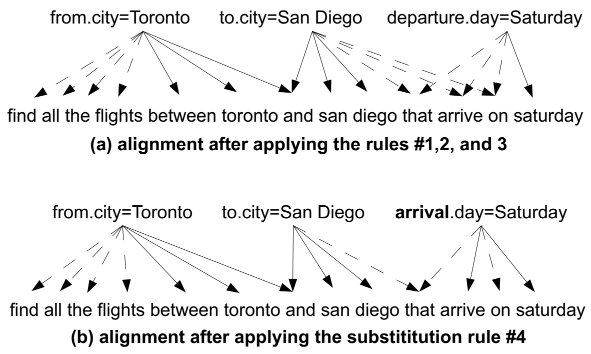
from.city=Toronto
to.city=San Diego
departure.day=Saturday
find all the flights between toronto and san diego that arrive on saturday
(a) alignment after applying the rules #1,2, and 3
from.city=Toronto
to.city=San Diego
arrival.day=Saturday
find all the flights between toronto and san diego that arrive on saturday
(b) alignment after applying the substititution rule #4

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

One way to handle this problem is to constrain triggers of rules performing substitutions to be activated only by the words aligned to the replaced slot. To do this; we track the words from the utterance that 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 links are referred to as "direct alignment".

In Figure 1 (a), we see the alignment between the words and the slots in the example utterance after applying the rules #1,2, and 3. The full arrows denote direct alignment. Because no rules were triggered by the words "find all the flights" and "that arrive on", those words could not be aligned directly to any of the slots. Therefore, we have to infer an appropriate alignment (see Figure 1 (a) dashed arrows). A word is aligned to a slot if the alignment does not cross any direct alignment. In Figure 1 (a), the phrase "find all the flights" can be aligned to the slot "from.city=Toronto" only (dashed arrows). The phrase "that arrive on" can be aligned to two slots "to.city=San Diego" and "departure.day=Saturday".

In Figure 1 (a), we see that the rule #4 meets the locality constraint because the word "arrive" is aligned to the slot "departure.day". As a result of applying the rule, the slot and the alignment of the phrase "that arrive on" have changed (see Figure 1 (b)). The rule #5 is not triggered because the word "arrive" is not aligned to the slot "from.city".

3.3. Improving the disambiguation of long-range dependencies

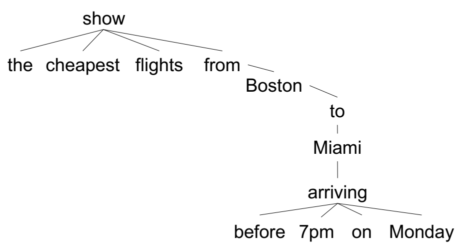

Figure 2: Dependency tree of the utterance "show the cheapest flights from Boston to Miami arriving before 7pm on Monday".

Besides simple n-grams and skipping bigrams, more complex lexical features can be used. Kate [12] used manually annotated dependency trees to capture long-range relationships between words. In a dependency tree, each word is viewed as the dependant of one other word, with the exception of the root. Dependency links represent grammatical relationships between words. 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 neighbours in the dependency tree but they are four words apart in the utterance (see Figure 2).

Instead of using manually annotated word dependencies [12], we used dependencies provided by the RASP dependency parser [7]. New n-gram features were generated in which a word history is given by links between words. For example, the algorithm would generate bi-gram ('arriving','Monday') for the word "Monday". Note however that RASP was used "off-the-shelf" and more accurate dependencies could be obtained by adapting it to the target domain.

3.4. Learning

The main idea behind transformation-based learning [6] is to learn an ordered list of rules which incrementally improve an initial semantic hypotheses (see the algorithm in Figure 3)[1]. The initial assignment is made based on simple statistics - the most common goal is used as initial semantics. The learning is conducted in a greedy fashion, and at each step the algorithm chooses the transformation rule that reduces the largest number of errors in hypotheses. Errors include goal substitutions, slot insertions, slot deletions, and slot substitutions. The learning process stops when the algorithm cannot find a rule that improves the hypotheses beyond some pre-set threshold. Note that no prior alignment between words and semantic concepts is needed.

As in the previous work [2, 8, 10, 11], we make use of a database with lexical realisations of some slots, e.g. city and airport names. Since the number of possible slot values for each slot is usually very high, the use of a database results in a more robust parser. In our method, we replace lexical realisations of slot values with category labels before parsing, e.g. "i want to fly from CITY". After parsing we use a deterministic algorithm to recover the original values for category labels, which is detailed in [2].

[1]The list of rules must be ordered because each learnt rule corrects some of the remaining errors after applying the preceding rules.

```
1. ASSIGN INITIAL SEMANTICS TO EACH UTTERANCE
2. REPEAT AS LONG AS THE NUMBER OF ERRORS ON THE
   TRAINING SET DECREASES
     (A) GENERATE ALL RULES WHICH CORRECT AT LEAST
         ONE ERROR IN THE TRAINING SET
     (B) MEASURE THE 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 THAN THRESHOLD T.
```

Figure 3: Rule learning algorithm.

4. Evaluation

In this section, we evaluate our parser on two distinct corpora, and compare our results with state-of-the-art techniques and a handcrafted Phoenix parser [13].

4.1. Datasets

In order to compare our results with previous work [2, 8, 10, 11], we apply our method to the ATIS dataset [3]. 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 criterion is the F-measure of the number of reference slot/value pairs that appear in the output semantics (e.g., from.city = New York). He & Young detail the test data extraction process in [8].

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. 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 a 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 the STC parser [2] and the handcrafted Phoenix parser [13]. The Phoenix parser implements a partial matching algorithm that was designed for robust spoken language understanding.

4.2. Results

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

Results on the ATIS dataset show that the TBL parser (F-measure = 95.74%) is competitive with respect to the Zettlemoyer & Collins' PCCG model [10] (95.9%). Note that this PCCG model makes use of a considerably large number of handcrafted entries in their initial lexicon. In addition, TBL outperforms the STC [2], HVS [8] and MLN [11] parsers. Concerning the TownInfo dataset, Table 2 shows that TBL produces 87.82% of F-measure, which represents a 3.28% improvement over the handcrafted Phoenix parser, while being competitive with the STC model - TBL's performance is only 0.76% lower.

Table 3 shows a contrast between the full system and the

Parser	Prec	Rec	F
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
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

Table 2: Slot/value precision (Prec), recall (Rec) and F-measure (F) for the ATIS and TownInfo datasets.

Parser	Prec	Rec	F
ATIS development dataset:			
TBL	93.95	93.70	93.82
No locality constraints	93.38	92.64	93.01
No dependency tree features	92.78	92.04	92.41

Table 3: Comparison of different aspects of the TBL method on the ATIS development dataset.

system with no features extracted from dependency trees and the system with no locality constraints. Experiments were carried out on the ATIS development dataset. The results show that if the dependency tree features are removed or the locality constraints are not used, the performance degrades.

The learning time of the TBL parser[2] is acceptable and the parsing process is efficient. First, the learning time is about 24 hours on an Intel Pentium 2.8GHz for each dataset. The TBL parser generates up to 1M potential transformation rules in each iteration; however, only a fraction of these rules have to be tested because the search space can be efficiently organised [6]. Second, the TBL parser is able to parse an utterance in 6ms while the STC parser needs 200ms on average [2]. We cannot report on speed the other approaches because such information is not publicly available.

The TBL parser is very efficient on domains such as ATIS and TownInfo because the final list of learnt rules is small. There are 17 unique dialogue acts and 66 unique slots in the ATIS dataset and the total number of learnt rules is 372. This results in 4.5 rules per semantic concept on average. In the TownInfo dataset, we have 14 dialogue acts and 14 slots and the total number of learnt rules is 195. The average number of rules per semantic concept is 6.9. The number of semantic concepts per utterance is 5 on average.

5. Conclusion

This paper presents a novel application of TBL for semantic parsing. Our method learns a sequence of rules which iteratively transforms the initial semantics into the correct semantics. The TBL parser was applied to two very different domains and it was shown that its performance is competitive with respect to the state-of-the-art semantic parsers on both datasets. On the ATIS dataset, TBL outperforms STC, HVS and MLN parsers by 1.27%, 2.75%, and 5.44% respectively [2, 8, 11]. We also show that TBL outperforms the handcrafted Phoenix parser by 3.28% on ASR output of the TownInfo dataset [2].

Although the TBL approach cannot directly generate an N-best list of hypotheses with confidence scores, several methods have been developed to alleviate this problem. For example, transformation rules can be converted into decision trees from which informative probability distributions on the class labels can be obtained [14]. In future work, we plan to investigate how to adapt the TBL method to obtain multiple hypotheses and confidence scores, and extend the model to richer domains where the ability to model long-range dependencies might be more important.

[2]The source code is available under GNU GPL at `http://code.google.com/p/tbed-parser/`.

6. Acknowledgment

This research was partly funded by the UK EPSRC under grant agreement EP/F013930/1 and by the EU FP7 Programme under grant agreement 216594 (CLASSIC project: www.classic-project.org).

7. References

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

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

[3] 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.

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

[5] B. Thomson, M. Gašić, 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.

[6] 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.

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

[8] Y. He and S. Young, "Semantic processing using the Hidden Vector State model," *Computer Speech & Language*, vol. 19, no. 1, pp. 85–106, 2005.

[9] F. Jurčíček, J. Svec, and L. Muller, "Extension of HVS semantic parser by allowing left-right branching," in *Proceedings of ICASSP*, 2008.

[10] L. Zettlemoyer and M. Collins, "Online learning of relaxed CCG grammars for parsing to logical form," in *Proceedings of EMNLP-CoNLL*, 2007.

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

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

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

[14] R. Florian, J. C. Henderson, and G. Ngai, "Coaxing confidence from an old friend: Probabilistic classifications from transformation rule lists," in *Proceedings EMNLP*, 2000.