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, 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 that transforms initial semantic hypothesis into correct semantics by using an ordered list of rules. These rules are learnt automatically from a training corpus with no prior linguistic knowledge and no alignment between words and semantic concepts is needed. We show that TBL 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

1. Introduction

Semantic parsing is important part of a spoken dialogue system [1]. The goal of 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 the ATIS dataset [4]. Each frame has a goal and set of slots. Each slot is composed of a slot name, e.g. "from.city", and slot value, e.g. "Washington". As dialogue managers commonly use semantics in the form of frames and slots [1, 3], our approach learns to map directly 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 operations per semantic concept.

In our approach, we adapt transformation-based learning (TBL) [5] to the problem of semantic parsing. We attempt to find an ordered list of transformation rules to iteratively improve initial semantic annotation. In each iteration, a transformation rule corrects some of remaining errors in semantics. To deal with long-range dependencies between words, we experiment with features extracted from dependency parse trees provided by RASP [6].

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, discusses templates for transformation rules, and describes the learning process. Section 4 compares TBL parser to the previously developed semantic parsers on the ATIS [4] and TownInfo [7] datasets. 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 ATIS [4] dataset.

2. Related work

In Section 4, we compare performance of our method with four existing systems that were evaluated on the same dataset we consider. First, the Hidden Vector State technique [8, 9] has been used to model an approximation of a pushdown automaton with semantic concepts as non-terminal symbols. Second, automatic induction of combinatory categorical grammar [10] has been used to map sentences to lambda-calculus. 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, the method also needs considerable number of handcrafted entries in their initial lexicon. Third, Markov Logic Networks [11] have been used to extract slot values by combining probabilistic graphical models and first-order logic. In this approach, weights are attached to first-order clauses which represent relationship between slot names and their values. Such weighted clauses are used as templates for features of Markov networks. Finally, Support Vector Machines [7] have been used to build semantic trees by recursive calling of classifiers to estimate probabilities of production rules using a linear kernel and word based features.

A large amount of research has been also done into mapping natural language to semantics that is not directly comparable because of either different corpora or different meaning representation. Transformation techniques [2] have been used to sequentially rewrite an utterance into semantics. 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, we can use words in an utterance several times to trigger transformations of the semantics. Support vector machines and tree kernels [12] have been used 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, we give an example of the parsing algorithm. Second, we detail locality constrains for transformation rules. Third, we de-

scribe templates used to generate rules for the inference process. Fourth, we describe features capturing long-range dependencies. Finally, the learning process is detailed.

3.1. Example of Parsing

The semantic parser transforms an initial semantic hypothesis into correct semantics by applying transformations from a list of rules. Each rule is composed of a trigger and a transformation and a trigger initiates transformation of a hypothesis. We demonstrate the parsing on an example: *"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

Second, the rules which triggers match the utterance and the hypothesis are sequentially applied. Generally, the rules add, delete , or substitute slots.

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"

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 generalisation. However, the TBL method learns to correct its errors. Therefore, the parser also applies the error correcting rules at later stage of parsing. For example, the following rule corrects the slot name of the slot value "Saturday".

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

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

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

3.2. Locality constrains

So far no relationship between slots and their lexical realisation was considered. For example, before we perform the substitution of the slot "departure.day" to "arrival.day", we should test whether word "arrive" is near the slot's lexical realisation. The reason is that we do not want to trigger the substitution of the slot "from.city=Toronto" to "to.city=Toronto". This could happen because because the parser has also learnt the following rule:

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

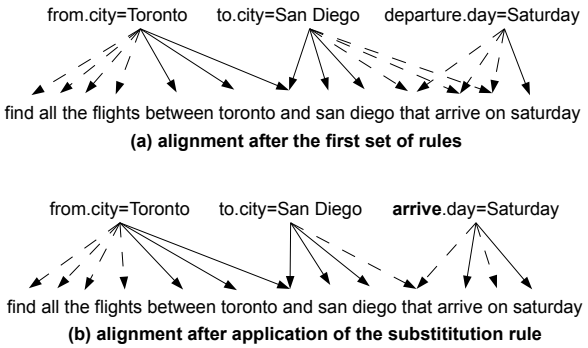

Figure 1: Alignment between the words and the slots in the example utterance.

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's lexical realisation. 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 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. 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; therefore, we have to derive such alignment (see Figure 1 (a) dashed lines). A word is aligned to a slot if the alignment does not cross direct alignment. In Figure 1 (a) dashed lines, 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 from the example of parsing. We can see change in the alignment of the phrase "that arrive on". First, the word "arrive" is aligned to the slot "arrival.day=Saturday". Second, the word "on" must be aligned to the same slot as the word "arrive". There is no change in the alignment of the word "that".

3.3. Rule templates

In the previous sections, 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.

A trigger questions 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[1] bigram B, the goal equals to G, the semantics contains slot S. If a trigger contains more than one condition, then all conditions must be satisfied. In our method, we use uni-grams, bi-grams, and tri-grams and skipping bigrams can skip up to 3 words.

[1] In a skipping bigram, one, two or three words are skipped between two words.

A transformation performs one of the following operations: substitutes a goal to G, adds a slot S, deletes a slot S, substitutes a slot S. A substitution transformation can substitute a whole slot, a slot name, or a slot value.

3.4. Improving disambiguation of long-range dependencies

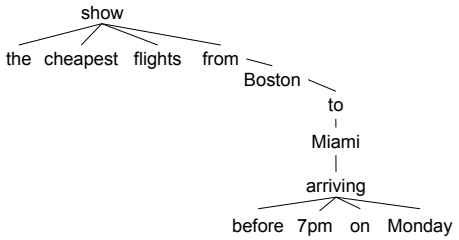

Figure 2: Dependency tree of the sentence "show the cheapest flights from Boston to Miami arriving before 7pm on Monday" generated by RASP.

Besides simple n-grams and skipping bigrams more complex lexical features can be used. Kate [12] 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 - the sentence root. 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 sentence (see Figure 2).

Instead of using gold standard word dependencies [12], we used dependencies provided by the RASP dependency parser [6]. New n-gram features were generated in which a word history of a word is given by links between words. For example, the algorithm would generate bi-gram ('arriving','Monday') for word "Monday". We however note that RASP was used off-the-shelf and more accurate dependencies should be obtained by adapting the RASP parser to the ATIS and TownInfo domains.

3.5. Learning

The main idea of transformation-based learning is to learn an ordered[2] list of rules which incrementally improve initially assigned initial semantic hypothesis (see the algorithm in Figure 3). 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 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.

As in the previous work [7, 8, 10, 11], we make use of 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 such database results in more robust parser. In our method, we replace lexical realisations of slot values with category labels in the utterance, e.g. "i want to fly from CITY". Note that after parsing we use a deterministic algorithm to recover the original values for category labels. More details about this algorithm can be found in [7].

[2]The list of rules must be ordered because each learnt rule corrects errors remaining after application of 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 the state-of-the-art techniques and a handcrafted Phoenix parser [13].

4.1. Datasets

In order to compare our results with previous work [7, 8, 10, 11], we apply our method to the ATIS dataset [4]. 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 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. These dialogues were previously used for training dialogue management strategies [1, 3].

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 parser [7] and the handcrafted Phoenix parser [13] used during 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. 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 the value must be correct to count as correct classification.

Results on the ATIS dataset show that our method with F-measure 95.74% is competitive with respect to Zettlemoyer &

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 constrains	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.

Collins' PCCG model [10] (95.9%). Note that PCCG model makes use of a considerably large number of handcrafted entries in their initial lexicon. In addition, our method outperforms STC, MLN and HVS parsers. Concerning the TownInfo dataset, Table 2 shows that TBL produces 87.87% F-measure, which represents a 3.28% improvement over the handcrafted Phoenix parser and it is competitive with respect to STC model on the TownInfo dataset - TBL's performance is only 0.76% lower compared with the STC model.

Table 3 shows contrast between the full system and the system wit no features extracted from dependency trees and the system with no locality constrains. 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 of our method degrades.

As is typical for TBL methods, the number of learnt rules is considerably low. 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.

Even though the TBL approach has to generate a significantly large number of transformation rules (up to 1M) in each iteration and the method was implemented in Python[3], for both datasets the learning time is about 24 hours on an Intel Pentium 2.8GHz. The TBL method is able to decode 6000 utterances in about 40s.

5. Conclusion

This paper presents novel application of TBL for semantic parsing. Our method learns sequence of rules which transforms initial semantics into correct semantics. It significantly differs 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. On the ATIS dataset, 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 by 3.28% on ASR output on the TownInfo dataset.

Many can argue that TBL approach does not directly offer N-best list of hypothesis and confidence scores; however, several methods has been developed to alleviate this problem. For example, transformation rules has been converted into decision trees [14] and informative probability distributions on the class labels has been obtained. In our future work, we will investigate how to obtain multiple hypotheses and confidence scores.

[3]Scripting language: http://www.python.org/

6. 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] 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.

[5] 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.

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

[7] 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.

[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. Jurcicek, 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*, 2005.

[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.