Error Corrective 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

a 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 linguistic knowledge.

1 Introduction

Check how you cal STEP. ECL parser?

Semantic parsing is important part of a spoken dialogue system (Williams and Young, 2007; Thomson et al, 2008). The goal of a semantic parsing is to construct formal meaning representation (semantics) which is directly executable by a dialogue manager. Such semantics is usually defined by a grammar, e.g. LR grammar for GeoQuery domain (Wong and Mooney, 2006) **better paper???**, which is designed by a domain expert and easy to interpret by a dialogue manager or question answering system.

Semantic parsing can be understood as machine translation from a natural language to a formal language. First, we do not not have formal grammar for natural language is ungrammatical, include hesitations, and very often only fragments of complete sentences, e.g. "Boston to Miami tomorrow".

In our approach, we adapt transformation-based learning (TBL) (Brill, 1995) to the problem of semantic parsing **slot classification, attribute/value pair extraction**. TBL attempts to find an ordered list of transformation rules to improve a baseline annotation.

The rules decompose the search space into a set of consecutive words (windows) within which alignment links are added, to or deleted from, the initial alignment.

TBL is an appropriate choice for this problem for the following reasons: 1. It can be optimized directly with respect to an evaluation metric. 2. It learns rules that improve the initial prediction iteratively, so that it is capable of correcting previous errors in subsequent iterations. 3. It provides a readable description (or classification) of errors made by the initial system, thereby enabling alignment refinements.

The rest of the paper is organized as follows: In the next section we describe previous work on semantic parsing. Section 3.2 presents am example of TBL based semantic parsing. Section 3.3 describes the learning process. Section 4 compares ECL to the previously developed semantic parser on ATIS (Dahl et al, 1994) and TownInfo (Williams and Young, 2007; Thomson et al, 2008) tasks.

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

2 Related work

There has been a large amount of work done on learning to map sentences into their semantics. Many different techniques have been considered including machine translation (Wong and Mooney, 2006)

techniques using inductive logic programing (**?**)

support vector machines (Mairesse et al, 2009) and tree kernels (Kate, 2008) markov logic networks (Meza et al, 2008a; Meza et al, 2008b)

automatic induction of combinatory categorical grammar (Zettlemoyer and Collins, 2007). Z & C mention their problems with spoken speech =¿ non-compositionality After Zettlemoyer & Collins implemented second-pass decoding which relaxed constraints on the utterances, they achieved state-of-the-art results.

(He and Young, 2006) developed a parser for the ATIS domain also takes utterances paired with semantics as input. Their parser approximate a push-down automaton with semantic concepts as non-terminal symbols.

transformation techniques used in the system SILT (Kate, 2005). SILT learns transformation rules which sequentially rewrites an utterance into its formal representation. Our ECL parser differ in the way the semantics is constructed instead of transforming utterance. 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 the ability to handle non-compositionality phenomena in spoken language.

Transformation-based Error-driven Learning - Some Advances in Transformation-Based Part of Speech Tagging - Brill (1994)

(Kate, 2008)

R. J. Kate, Y. W. Wong, and R. J. Mooney. 2005. Learning to transform natural to formal languages. In Proc. of AAAI-05, pages 1062?1068, Pittsburgh, PA, July.

L. R. Tang and R. J. Mooney. 2001. Using multiple clause constructors in inductive logic programming for semantic parsing. In Proc. of ECML-01, pages 466-477, Freiburg, Germany.

3 Transformation-based parser

This section describes the transformation-based parser. First of all, we describe rule templates used to generate rules for the rule inference process. Secondly, we describe the learning process. Finally, we describe parsing algorithm.

TBL chooses the transformation that reduces the errors the most.

Existence of initial annotator. Transformations are applied in sequence. Results of previous transformations are visible to following transformations. Existence of current label. It can handle dynamic problems well.

Most of the effective methods can be roughly divided into rule-based and probabilistic algorithms. In general, the rule-based methods have the advantage of capturing the necessary information in a small and concise set of rules. For example, in part-of-speech tagging rule-based and probabilistic methods achieve comparable accuracies, but rule-based methods capture the knowledge in a hundreds or so simple rules, while the probabilistic methods have very high-dimensional parameter space (millions of parameters)

3.1 Rule templates

The learning algorithm uses rule templates to instantiate rules which are subsequently tested by the learning algorithm. Each rule template is composed of a trigger and transformation. **A trigger controls whether a transformation of hypothesis can be performed**. Each trigger question either input utterance or output semantics. As a results each trigger contains one or several condition:

- The utterance contains n-gram N?
- The utterance contains skipping[1] 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. **Get rid of the questions.**

A transformation performs one of these operation:

- substitute a dialogue act type
- add a slot
- delete a slot
- substitute a slot

As substitution can either substitute a whole slot, an equal sign in the slot, or a slot name.

[1]A skipping bigram is bigram which skips one or more words between words in the bigram

3.2 Example of Parsing

The semantic parser transforms initial naive semantic hypothesis into correct semantics by applying rules from an ordered set of rules.

The parsing consists of **three** steps:

1. initial semantics is assigned as hypothesis
2. sequentially apply all rules[2]
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 semantic 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"

The trigger **"and Sand Diego" is example of non-compositionality**, in which the the words in an utterance do not have a one-to-one correspondence with the slots in the semantics. The word "and" is used two times and in the second time it indicate that the city "San Diego" is slot value of the slot "to.city". After application of the transformations, we obtain the following semantic hypothesis.

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

The word "saturday" is incorrectly classified because the date and time values are associated with slot "departure*" most of the time. As TBL **classifier** learns to correct its errors, the parser also applies the error correcting rules. In our case it is:

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

The final semantic hypothesis is the following.

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. To implement this, we constrain what features are used by triggers based on the distance from the words which actually triggered the slots first of all.

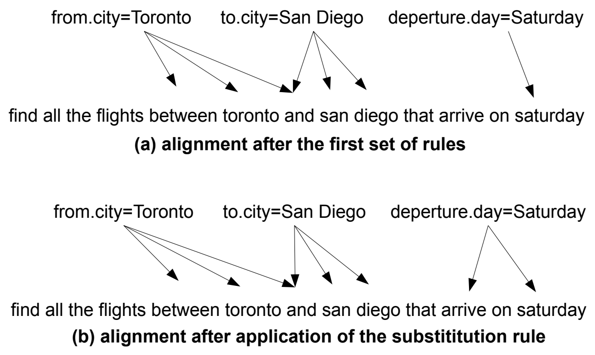
from.city=Toronto
to.city=San Diego
deperture.day=Saturday
find all the flights between toronto and san diego that arrive on saturday
(a) alignment after the first set of rules
from.city=Toronto
to.city=San Diego
deperture.day=Saturday
find all the flights between toronto and san diego that arrive on saturday
(b) alignment after application of the substititution rule

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

For every slot we track the words from the utterance, which were used in the triggers. In Figure 1, you can see the alignment. If we perform the substitution of the slot "departure.day" to "arrival.day" we test whether word "arrive" is in the vicinity of the slot. The main reason is that we do not want to trigger the substitution of the slot "from.city=Toronto" to "to.city=Toronto" because the parser also learns rule.

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

And to protect the rule from triggering we use the

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

vicinity constrain.

The vicinity of the slot is computed as it follows:

1. Let A_S a set of words aligned to the slot S.

3.3 Learning

The central idea of transformation-based learning is to learn and ordered list of rules which progressively improve upon the current state of the training set.

The in initial assignment is made based on simple statistics, and then the rules are greedily learned to correct the mistakes, until no improvement can be made.

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.
- 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.

Th system learns a list of rules in a greedy fashion, according the objective function. When no rule that improves the current state of the training set beyond the pre-set can be found, the training phase ends.

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.

Rules are learned sequentially:

1. initial semantics is assigned as hypothesis to each utterance
2. repeat as long as the number of errors[3] on training set decreases
   (a) generate all possible rules which correct at least one error in the training set
   (b) measure number of corrected errors for each rule
   (c) select the best rule
   (d) apply the selected rule
3. prune rules

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

First, the number of plausible 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

[3]Number of errors include number of dialogue act substitutions, number of slot insertions, number of slot deletions, number of slot substitutions.

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

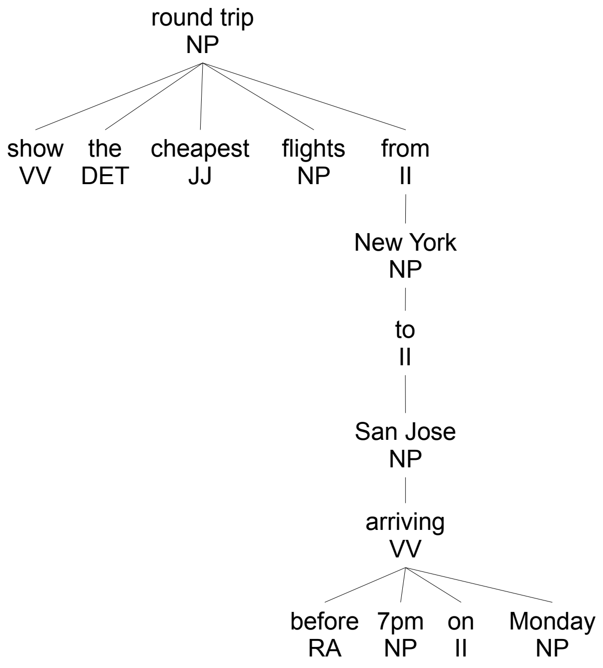

Figure 2: Dependency tree of the sentence "Show the cheapest flights from New York to San Jose arriving before 7pm on Monday" generated by the RASP parser (Briscoe et al, 2006).

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, the words ”arriving“ and ”Monday“ are neighbors in the dependency tree but they are four words apart in the sentence (see Figure 2).

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[4] Our motivation was work of (Meza et al, 2008a; Meza et al, 2008b) 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. 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.

[4]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:			
ECL	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:			
ECL	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:			
ECL	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 1: Slot/value precision (Prec), recall (Rec) and F-measure (F) for the ATIS and TownInfo datasets. ECL 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

The number of learned rules is very small. As is shown in the figure 3, 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,

Figure 3: The learning curve shows the relation between number of learned rules and the F-measure for both TI and ATIS corpora.

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. Makes decision in one step using all the features rather than making several decisions by several rules as STEP does it.

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. 2008. *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.