Error Corrective Learning for Semantic parsing

Filip Jurcicek, XXX
Department of Engineering
University of Cambridge
fj228@cam.ac.uk, XXX

Abstract

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

1 Introduction

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. CFG 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".

2 Related work

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.

Subsection **??**).

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.

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?

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

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.

3.2 Learning

Rules are learned sequentially:

1. initial semantics is assigned as hypothesis to each input utterance
2. repeat as long as the number of errors[2] 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 input utterance. For example, in utterance "find all the flights from cincinnati to the new york city" the lexical realization are replaced as follows: "please find all the flights from city-0 to the city-1". Similarly, we replace slot values in the semantics.

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

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

To speed up training, we select not only one best rule but also rules which correct at minimum 80% errors of the best rule.

3.3 Parsing

The semantic parser transforms initial naive semantic hypothesis into correct semantics by using a ordered set of rules. The parsing is composed of **three** steps:

1. initial semantics is assigned as hypothesis
2. sequentially apply all rules[3]
3. output hypothesis semantics

Although we use the

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

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

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

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 Experiments - dep trees

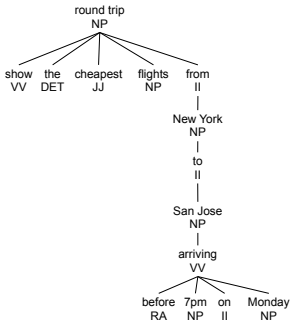

Figure 1: Dependency tree of the sentence "Show the cheapest flights from New York to San Jose arriving before 7pm on Monday" generated by the RASP parser (Briscoe et al, 2006).

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 relation 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 neighbors in the dependency tree but they are four words apart in the sentence (see Figure 1).

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. As result, the dependencies generated by the parsers 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.

Second, we generated long-range features by using POS tags[4] Our motivation was work of (Meza et al, 2008) who handcrafted features using words "arrive", "arriving", "leave", and "leaving". These handcrafted features effectively disambiguate largest number of semantic parsing errors in ATIS data because the most confusions in the output of the semantic parser are whether the time is time of arrival or departure, etc. To generalize this approach, we generate features (bigrams) for every word in utterance and every POS tag in data. Simply, for every word we search for the nearest left/right word with specific POS tag. Once we find such POS tag, we generate a bigram between those two words and search for nearest for another POS tag until all POS tag are processed.

Which effectively covers large portion of diam

generate bigrams for each word in the utterance and for the nearest word

We generate new n-grams reflecting the adjecy in

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

the semantic tree.

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

We extracted

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.

5 Discussion

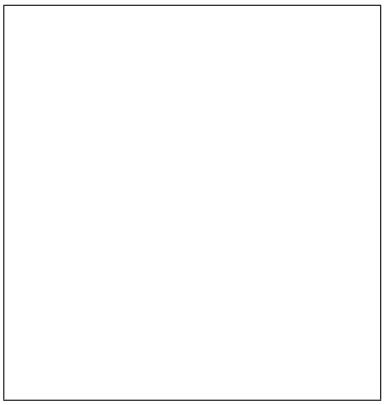

Figure 2: The learning curve shows the relation between number of learned rules and the F-measure for both TI and ATIS corpora.

The number of learned rules is very small. As is shown in the figure 2, 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 sometimes very complex entries for general English and yet additional lexical entries must be learned.

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

Lexical realizations of a slot can overlap with lexical realization of neighbouring slots. It is shows to be important pattern, for example in the trigram (city-0,and,city-1) is very common for sentence including "between city-0 and city-1". The lexical realizations city-0, city-1 respectively would be classified as from.city, and city-1 just because we know the

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

Acknowledgments

Do not number the acknowledgment section.

References

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.

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.

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

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

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.

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

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

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

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

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