Semantic Parsing with Joshua

Zhanwang chen

Abstract

This report presents using an open-source statistical machine translation decoder Joshua on semantic parsing for the FUNQL of Geoquery. It illustrates how mapping natural language to formal language using Joshua.

1 Introduction

The semantic parsing in this paper refers the task that mapping natural language to formal language as Figure 1 shows.

What is the biggest city in Kansas ?
answer(largest(city(loc_2(stateid('kansas')))))
Figure 1 : natural language and formal language in FUNQL

FUNQL is a functional, variable-free query language for Geoquery domain (Kate et al., 2005).
Sematic parsing here is cast as a machine translation task which translates one natural language to another natural language.
Hiero refers to Hierarchical phrase-based (David Chiang.2007.), it is a SCFG with a single nonterminal category, shows in Figure 2.

[X] ||| Arkansas ||| 'arkansas' ||| 0 0 1 0.01832 0 0 ||| 0-0
[X] ||| How big is [X,1] ? ||| size(stateid([X,1] ||| 0.68535 7.21837 1 0.01832 0 0.33647 ||| 0-1 1-0 2-1 4-1
Figure 2 : hiero grammar extracted using thrax in joshua

There six features in the 3th column: e_given_f_phrase , f_given_e_phrase , e_given_f_lex, f_given_e_lex, rarity ,phrase-penalty. The features are phrase translation probability.
The last column is alignment,

There are maximum of 7 columns in the Phrase-based grammar in Moses, shows in Figure 3:
1. Source phrase, 2. Target phrase, 3. Scores, 4. Alignment, 5. Counts, 6. Sparse feature scores, 7. Key-value properties

How long ||| len(||| 0.0247059 0.0455128 0.617647 1 ||| 1-0 ||| 50 2 2 ||| |||
How many Colorado rivers are ||| count(river(riverid('colorado' ||| 0.373244 0.000322623 0.373244 0.0229458 ||| 1-0 3-1 0-2 4-2 2-3 ||| 1 1 1 ||| |||
Figure 3 : phrase based grammar extracted using moses

2 Experiments

a) Shuffle corpus order

Similar sentence structures are ordered together in the corpus file, so we need shuffle the order before separate the training data and test data. Otherwise, some structures cannot be learned.

b) Data split

Here I used 550 sentences as training data, 50 sentences as tune data, and the rest of 280 sentences as test data.

c) Preparing parallel corpus

```
==> nl.txt <==
Give me the cities in Virginia
What are the high points of states surrounding Mississippi
Name the rivers in Arkansas

==> MRL.txt <==
city( loc_2( stateid( 'virginia'
high_point_1( state( next_to_2( stateid( 'mississippi'
river( loc_2( stateid( 'arkansas'
```

Since all target side Meaning represent sentences have the same form "answer()", it was being removed on training to get a higher translate precision.

In WASP, the SCFG learning algorithm assumes that an unambiguous, context-free grammar (CFG) of the target MRL is available, so it can learn grammars that can generate well-formed translations even they have deeply nested structure, as Figure 4 shows.

```
*n:River -> ({ the longest *n:River#1 })({ longest ( *n:River#1 ) }) weight 0.74
*n:State -> ({ Which state *n:State#1 })({ state ( *n:State#1 ) }) weight 0.68
*n:Query -> ({ *t:Bound What is *n:Num#1 *t:Bound })({ answer ( *n:Num#1 ) }) weight 0.81
```

Figure 4 : SCFG with weight in WASP

Unlike WASP, context-free grammar of the target MRL was not used in the whole process. It get a totally mess structure in translations without mechanism.
In this experiment, I removed all the right bracket and made the left bracket as part of the word, and complete the bracket form by adding right brackets after translation.

```
What is the adjacent state of California ?
answer(   state(   next_to_2(   stateid(   'california'
answer(state(next_to_2(stateid('california'))))
```

Figure 5 : to add right brackets after translation.

d) Tanning, extracting grammar, decoding.

Preprocess your data to adapt the Joshua or Moses file format.

Once it gets the grammar, then add your own rules to the grammar such as lake names, mountain names, and other place names.

3 Evaluation

A translation is correct if we can get the same answer from the GEOQUERY database as the reference query.
Here in the evaluation module in WASP, it uses a Prolog development system SICStus to retrieve the answer from GEOQUERY database.

Hireo:
Got 126 correct translations of 280 test sentences
Precision 45%

Phrase-based in Moses:
Got 33 correct translations of 280 test sentences
Precision 11.8%

thrax-phrase:
Got 41 correct translations of 280 test sentences
Precision 14.6%

Appendix

Here is the command for hiero translate model, to use phase-base in mose, change --type hiero to –tpye moses

```
$JOSHUA/bin/pipeline.pl --source nl --target mr    --type hiero --no-prepare --aligner
berkeley --lm-gen berkeleylm    --lm berkeleylm --corpus nl-mr/training.nl-mr --tune
nl-mr/dev.nl-mr --test nl-mr/devtest.nl-mr
```

Here is the list the main steps in hiero translate model, instead of using pipeline.pl directly, you excuse these command one by one according to your needs such as change the parameter in config file, add your initial grammar.

```
  [Alignments]
  java -d64 -Xmx10g -jar /home/zhanwang/joshua-6.0.5/lib/berkeleyaligner.jar
++alignments/0/word-align.conf


  [thrax-run] (run thrax to extract Hireo grammars)
hadoop/bin/hadoop jar /home/zhanwang/joshua-6.0.5/thrax/bin/thrax.jar -D
mapred.child.java.opts='-Xmx2g' -D hadoop.tmp.dir=/tmp thrax-hiero.conf
pipeline-nl-mr-hiero-_root_models_test9 > thrax.log 2>&1; rm -f grammar
grammar.gz; hadoop/bin/hadoop fs -getmerge
pipeline-nl-mr-hiero-_root_models_test9/final/ grammar.gz

[filter-tune] (you can add your owe grammar to grammar.filtered.gz)
/home/zhanwang/joshua-6.0.5/scripts/support/filter_grammar.sh -g grammar.gz -f -v
/root/models/test9/nl-mr/dev.nl-mr.nl |
/home/zhanwang/joshua-6.0.5/scripts/training/filter-rules.pl -bus3 | gzip -9n >
/root/models/test9/data/tune/grammar.filtered.gz


  [mert-1]
/home/zhanwang/joshua-6.0.5/scripts/training/run_tuner.py
/root/models/test9/nl-mr/dev.nl-mr.nl /root/models/test9/nl-mr/dev.nl-mr.mr
--tunedir /root/models/test9/tune --tuner mert --decoder
/root/models/test9/tune/decoder_command --decoder-config
/root/models/test9/tune/joshua.config --decoder-output-file
/root/models/test9/tune/output.nbest --decoder-log-file
/root/models/test9/tune/joshua.log --iterations 15 --metric 'BLEU 4 closest'
   took 137 seconds (2m17s)
```

[test-decode-1] (Generate the translate result of test set)
/root/models/test9/test/decoder_command

[test-bleu-1]
/home/zhanwang/joshua-6.0.5/bin/bleu /root/models/test9/test/output /root/models/test9/nl-mr/devtest.nl-mr.mr > /root/models/test9/test/bleu

[analyze-test-1](Generate the translate analysis of test set, with BLEU score of every sentence)
/home/zhanwang/joshua-6.0.5/scripts/analysis/sentence-by-sentence.pl -s /root/models/test9/nl-mr/devtest.nl-mr.nl -r /root/models/test9/nl-mr/devtest.nl-mr.mr /root/models/test9/test/output > /root/models/test9/test/analysis/sentence-by-sentence.html

[about WASP]
Rewrite two methods in WASP, one for extract corpus from FUNQL Geoquery, another in parser evaluation module in WASP.

References

David Chiang (2005). A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL-2005)*, pp. 263–270. Ann Arbor, MI.

Y. W. Wong and R. J. Mooney. 2006. Learning for semantic parsing with statistical machine translation. In Proc. of HLT/NAACL-06, pages 439–446, New York City, NY.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, Evan Herbst, Moses: Open Source Toolkit for Statistical Machine Translation, Annual Meeting of the Association for Computational Linguistics (ACL), demonstration session, Prague, Czech Republic, June 2007.