NLP Project Report

Yang Jing

5130309572
Brotherjing@sjtu.edu.cn

This document is a report for the NLP project: implicit discourse relation recognition. This work basically implement most of the idea presented in [1], with some additional work to optimize the result.

Introduction

Implicit discourse relation recognition is to predict the relation between 2 sentences, without the help of explicit connective words (e.g., *but*). To do such recognition, we can extract some features from the 2 sentences, which might be related to the sense(discourse relation) between the 2 sentences. Then we can train a classifier to find a most possible sense for the sentences.

In this project, I implement most of the idea presented in [1]. First I extracted some syntax-level features like production rules and depencencies, and some word-level features like word pairs. I also used mutual information to reduce feature dimension. Next I trained a MaxEnt classifier, and finally test the model. With some additional feature and parameter configuration, The model gained a slight improvement compared to the original result.

The code are mainly written in Python. There is also a small Java program which wrap the Stanford Parser to accelerate the feature extraction process.

In the following sections, I will explain how I implement each step of feature extraction,

reduction and training, and finally show the test result.

Feature Extraction

Syntax-level Features. The syntax level features we are going to extract are production rules and dependencies. In the training data, the raw text of the 2 arguments of the relation is given. With the Stanford Parser [2], we are able to obtain the parse tree and dependencies of the sentences, from which we extract the features.

Since I only found Java API for Stanford Parser, I write a small Java program which read a file of sentences, and export the parse tree and dependencies for all sentences. The output parse tree is in one line:

```
( (S (ADVP (NP (DT This) (NN time)) (RB around)) (, ,)
(NP (PRP they)) (VP (VBP 're) (VP (VBG moving) (ADJP
(RB even) (JJR faster))))))
```

And the output dependencies is an array of:

```
det(time-2, This-1),
nmod:npmod(around-3, time-2),
advmod(moving-7, around-3),
nsubj(moving-7, they-5),
aux(moving-7, 're-6),
root(ROOT-0, moving-7),
advmod(faster-9, even-8),
xcomp(moving-7, faster-9)
```

According to Lin et al, the final production rule feature should be a binary feature containing each node and its children(e.g.,*S-NP_VP*), and the dependency feature should contain a depen-

dent and its dependency type(e.g., *moving-advmod_nsubj_aux_xcomp*). To convert a parse tree to an array of production rules, I simply build a tree structure, then traverse the tree to obtain all rules. It's also easy to extract all dependency features from the array of dependency. The detailed algorithm is in *features.py*.

Word-level Features. Word-level features include word pairs, and another called *First-Last, First3*, which is the first and last word of Arg1 and Arg2, and the first 3 words.

Word pair features are easy to extract. One thing to note is that we should use the stemmed word rather than the original one, which can reduce total amount of feature. The stemmed word is given in the training data as well as the original word.

First-Last, First3 feature [4] seems to be another good feature, although in implicit relation dataset the first and last words are not connectives. The MI feature selection result show some of these features that may indicate some discourse relations:

```
first2_now Contrast
first2_one Instantiation
first2_that Cause
first32_that_be_because Cause
first32_that_compare_with Contrast
```

Feature Selection

We cannot directly use the feature for training, since there are more than 10,000 production rules, and even more than 100,000 word pair features. To select feature that are actually related to some discourse relations, I use mutual information(MI) to evaluate each feature, by computing MI of a feature with each of the 11 discourse relation classes, and choosing the highest

value as its score. MI between two entities is computed by:

$$I(U;C) = \sum_{e_t \in \{0,1\}} \sum_{e_c \in \{0,1\}} P(U = e_t, C = e_c) \log_2 \frac{P(U = e_t, C = e_c)}{P(U = e_t)P(C = e_c)} \quad (1)$$

where $e_t = 1$ means t is shown in the document, and $e_t = 0$ means it's not.

To select features using MI, we can count the number of sentence where feature and class both appear(N_{11}), and also N_{00}, N_{01}, N_{10}, then applying the above method:

$$I(U;C) = \frac{N_{11}}{N} \log_2 \frac{NN_{11}}{(N_{10}+N_{11})(N_{01}+N_{11})} + \frac{N_{01}}{N} \log_2 \frac{NN_{01}}{(N_{00}+N_{01})(N_{01}+N_{11})} + \quad (2)$$
$$\frac{N_{10}}{N} \log_2 \frac{NN_{10}}{(N_{10}+N_{11})(N_{00}+N_{10})} + \frac{N_{00}}{N} \log_2 \frac{NN_{00}}{(N_{00}+N_{01})(N_{00}+N_{10})}$$

I also apply a frequency cut-off of 5 to remove infrequent features.

Then I rank the 4 kinds of features by their scores, and store the 4 lists in separate file.

Training

Next I generate a training set, each line of which is a list of feature and a label. The features are selected from the top n features of each kind. Different n yields different result. Here I use top 250 production rules, top 100 dependencies, top 700 word pairs and top 70 first-last.

The training process of MaxEnt classifier is simple. Just feed in the training data and it will output a model. The GIS iteration is also an important parameter. If it's set too high, it gives good result on training set, but it may be overfitting. An iteration of 50 is appropriate.

Result

I test the trained model on the dev set. It give an F1-accuracy of 42.33%, which is slightly higher than the original work.

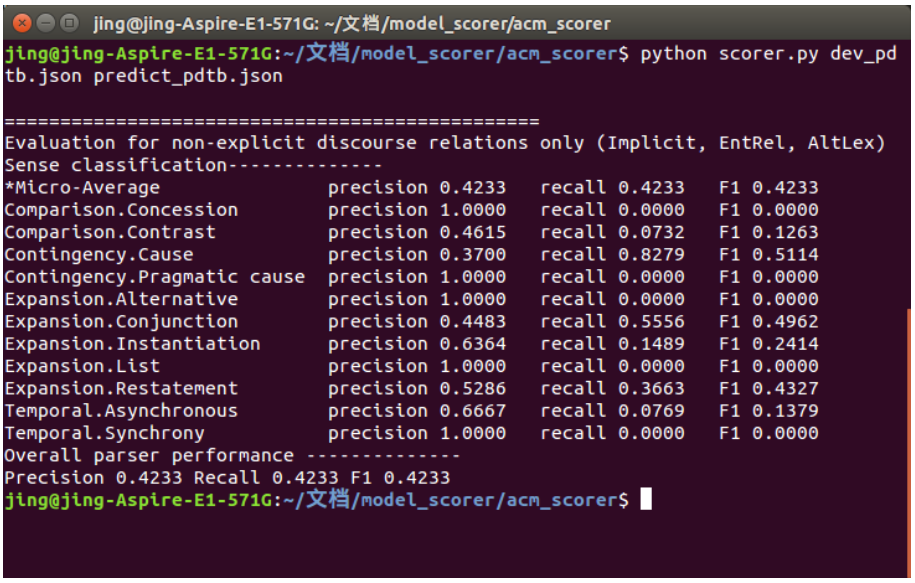

Figure 1: The result on develop set

Conclusion

In this work, I implement the idea presented in [1], and add some more feature to optimize the result. However, this still need more improvement. More features presented in [4](e.g., Polarity, modality, verbs) can be used to improve accuracy for some discourse relation. For example, if two arguments of a sentence have opposite polarity, it might indicate a Contrast relation. Also, semantic-level features(e.g., entity mention in [3]) are very important. A good tool for representing semantic of word is word2vec. More complicated machine learning model rather than a simple MaxEnt classifier would also improve performance.

References and Notes

[1] Lin, Z., Kan, M. Y., & Ng, H. T. (2009, August). Recognizing implicit discourse relations in the Penn Discourse Treebank. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1-Volume 1* (pp. 343-351). Association for Computational Linguistics.

[2] Chen, D., & Manning, C. D. (2014, October). A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP* (pp. 740-750).

[3] Ji, Y., & Eisenstein, J. (2014). One vector is not enough: Entity-augmented distributional semantics for discourse relations. *arXiv preprint arXiv*:1411.6699.

[4] Pitler, E., Louis, A., & Nenkova, A. (2009, August). Automatic sense prediction for implicit discourse relations in text. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2* (pp. 683-691). Association for Computational Linguistics.