CS134 Discourse Relation Sense Classification

with a Neural Network approach

Jiajie Yan

Motivation

In this project, we built a CNN model to classify discourse sense relation. We chose CNN instead of DNN and RNN based on following reasons:

1. DNN is not able to capture locality information, while CNN can learn the correlative words grammatically.
2. Sequence information is not necessary in sentence relation classification. Though RNN may have better performance, CNN still wins in terms of speed.

Feature Representation

We applied word embedding to represent the word features. In detail, we project each word in the sentence pair to a low dimension space, with various number of dimensions of features. Due to time reason, we only chose the first N words of each sentence as input to the model.

Architecture

■ Embedding layer

We concatenate the sentences in pair as one data instance (pad if shorter than defined total length), and feed the model with a batch of data points. Thus the input embedding layer is a 3D matrix, as [batch_size, sentence_length, num_feature].
The word embedding is fine-tuned, which means we don't train the word vectors from training data or use any pre-trained embedding, we trained the word embedding together with the model.

■ Convolutional layer

We applied different sizes of convolutional filters, and each size filter is multiple designed. These layers are to slide over the total sentence data point and extract locality features from the layers.

■ Pooling layer

There is a max pooling layer after the convolutional layers, which aims to down sample the features from convolutional layers.

■ Softmax output layer

The last layer is the output layer, where we use softmax to assign normalized probabilities for all relation senses. This output is a one hot vector, the predicted sense is 1, with all others 0.

■ Hyper-parameter design

There are many combinations of hyper parameter settings in this model, which may influence the performance of the model. Followings are default settings of this project:

Sentence length	50	Batch size	100
Embedding dimension	300	Learning rate	0.0001
Filter sizes	3, 4, 5	Iteration time	15
Number of filters	3		

Code structure

There are several modules responsible for different goals in the construction of the model.

■ CNN.py

This is the main body of the model, a CNN architecture that defines an embedding layer, multiple convolutional layers, a pooling layer and a softmax layer. This module is built by TensorFlow. Hyper-parameters are passed in by the constructor.

■ prep_data.py

This module prepares and provides some utility functions to offer convenient data, like loading train/de/test data as a list of relation dictionaries from JSON files, converting words into indices, generating train/dev/test batch in matrices, and converting a sentence of a word into a sequence of indices.

■ train.py

Use this module to directly begin setting hyper-parameters, training and evaluating the model. The result of the classification will be output as a JSON file by the given output path.

>> How to run the model

Go to the 'train.py' file under 'model' folder, set the 'root' and 'path' to train/dev/test relation files, then set the path of the output file, 'Run' the script. Configure the parameters rolling down if necessary.

Performance

Below is the performance of the model according to the default hyper-parameter settings:

Evaluation for all discourse relations:

```
Sense classification--------------
*Micro-Average                    precision 0.2423      recall 0.2423   F1 0.2423
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0278      recall 0.0556   F1 0.0370
Contingency.Cause.Reason          precision 0.1600      recall 0.1000   F1 0.1231
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Condition             precision 0.0513      recall 0.0741   F1 0.0606
EntRel                            precision 0.2917      recall 0.2450   F1 0.2663
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.2770      recall 0.6862   F1 0.3947
Expansion.Instantiation           precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 0.3182      recall 0.0464   F1 0.0809
Temporal.Asynchronous.Precedence  precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Succession  precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Synchrony                precision 0.0769      recall 0.0200   F1 0.0317
Overall parser performance --------------
Precision 0.2423 Recall 0.2423 F1 0.2423
```

```
------------------------------------------------
Evaluation for explicit discourse relations only
Sense classification--------------
*Micro-Average                    precision 0.3018      recall 0.2752   F1 0.2879
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0189      recall 0.0357   F1 0.0247
Contingency.Cause.Reason          precision 0.1290      recall 0.1053   F1 0.1159
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Condition             precision 0.0556      recall 0.0741   F1 0.0635
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.3930      recall 0.6840   F1 0.4991
Expansion.Instantiation           precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Precedence  precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Succession  precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Synchrony                precision 0.0909      recall 0.0213   F1 0.0345
Overall parser performance --------------
Precision 0.3018 Recall 0.2752 F1 0.2879

================================================
Evaluation for non-explicit discourse relations only (Implicit, EntRel, AltLex)
Sense classification--------------
*Micro-Average                    precision 0.2154      recall 0.2144   F1 0.2149
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0364      recall 0.0769   F1 0.0494
Contingency.Cause.Reason          precision 0.2105      recall 0.0952   F1 0.1311
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
EntRel                            precision 0.4118      recall 0.2450   F1 0.3072
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.1789      recall 0.6903   F1 0.2842
Expansion.Instantiation           precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 0.3889      recall 0.0479   F1 0.0854
Temporal.Asynchronous.Precedence  precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Synchrony                precision 0.0000      recall 0.0000   F1 0.0000
Overall parser performance --------------
Precision 0.2154 Recall 0.2144 F1 0.2149
```

1. The overall performance is not as expected. The magic of deep learning doesn't happen here. The major reason may be lack of data. If bigger data is fed in, the performance might be better.
2. Classification of implicit relations are worse than explicit ones. This happens probably because when explicit connectives appear in the sentence pair, there are also other words indicating the relation sense co-occurring.
3. Not every relation sense is detected. This is mostly due to the lack of data.

Experiments

We mainly tuned different hyper-parameters to see what is the most important. The table below shows what we have tried. Note when one difference indicated, other parameters are the same.

Sentence length of 50	24.3% *(all are F1 score)*
Sentence length of 100	21.2%
Feature dimensions of 300	24.3%

Feature dimensions of 100	20%
Iteration time of 15	24.3%
Iteration time of 30	20.1%
Number of filters of each size of 3	24.3%
Number of filters of each size of 6	21.3%

1. Long sentence length lowers the F1 score. This is theoretically not normal, because the longer the sentence we give, the more evidence the model has to enhance its prediction ability. More research needs to be done in the future to look into this issue.
2. Too many feature dimensions lower the score. This is because too many features make the model complex, and thus easily leads to overfitting.
3. Too many iterations on the data lowers the score. It seems this model converges fast, 10 to 15 iterations on the entire training data are already enough, which reaches its best performance; and if more than 15, like to 30, the performance on dev data becomes worse and worse. This means the model already overfits the training data, so behaves bad on new data.
4. Too many filters lower the score. The same reason as too many features, this setting complicates the model and makes it easy to overfit the training data. Less filter makes the model simple and have a good adaption on new data.