Jennifer Storozum
CS135 Final Project Report
Shallow Discourse Parsing with Neural Networks
12/15/2017

1 How to Run/Test the System

1. Run python start.py
2. Run python scorer.py <path-to-gold-file>/relations.json <path-to-output-file>/output.json

(NB: Please use the original scorer, as I did not test with the new scorer)

2 Code Structure

There are three functions in my program:

- feed_forward(x_train, y_train, x_test, y_test)
- process_data(dev_relations, training_relations, test_relations)
- main()

The feed_forward() function is the neural network, a multilayer perceptron for multi-class classification. The architecture and parameters design will be discussed below.

The process_data() function preprocesses the raw data from the input .json files and turns it into a usable format for the neural network.

First, all the data from test, train and dev sets is concatenated to create a sentence bank of all sentences in the data, and then processed to create a vocabulary of all words in the data. Similar to the name classification example Yuchen shared with us, each word is mapped to an index and a word/index pair for unknown words is also added. Then for each of the dev, train and test sets, parallel lists of raw text sentences and their senses were created.

For the sense labels, each label was mapped to a one hot vector representation, where the index of the one in the vector corresponds to the index of a list of senses.

Then the embeddings lists for the test, train and dev sets for both the data (x) and the labels (y) were created. For data, each word in each sentence of each set was mapped to its index to create word embeddings. Short sequences were padded to the length of the longest sentence. For the labels, lists of one hot vectors corresponding to each sense in the list of senses for each set of data were created.

An issue that came up during preprocessing was that some of the sense labels for data did not correspond exactly to the gold list of senses provided in the validator, which made it impossible to map to and from labels. I chose to throw out these data points. I could have padded and then truncated the input sequences, but I didn't see the need to as no input sequence was overly long.

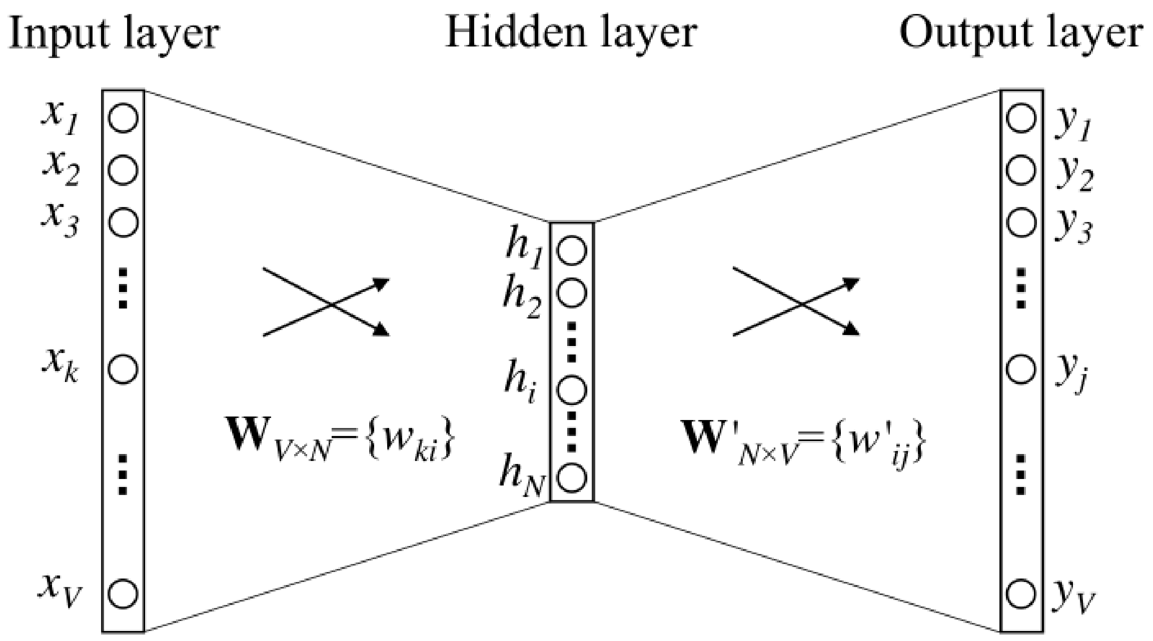
3 Feature Representation

Jennifer Storozum
CS135 Final Project Report
Shallow Discourse Parsing with Neural Networks
12/15/2017

I used a feature representation similar to the names example Yuchen provided, but with words instead of letters. Each word is mapped to a number in a dictionary of all words in the vocabulary. Sentences are tokenized and each word is stemmed with the Porter stemmer. Each word in each sentence is then replaced with its corresponding number to create word embeddings.

4 Neural Network Architecture

I implemented a feed-forward architecture for my neural network. The input size is the length of a sequence. All the input sequences are the same length due to the padding implemented in the preprocessing stage. The output size is the same as the number of classes, in this case senses, that the model is trained to predict. The architecture is roughly equivalent to the following structure:



Where there is one input layer, one hidden layer, and one output layer. There are 32 nodes in the input layer, 32 nodes in the hidden layer and 15 nodes in the output layer. The number of nodes in the input and hidden layers was determined empirically, and the number of nodes in the output layer is equal to the number of classes.

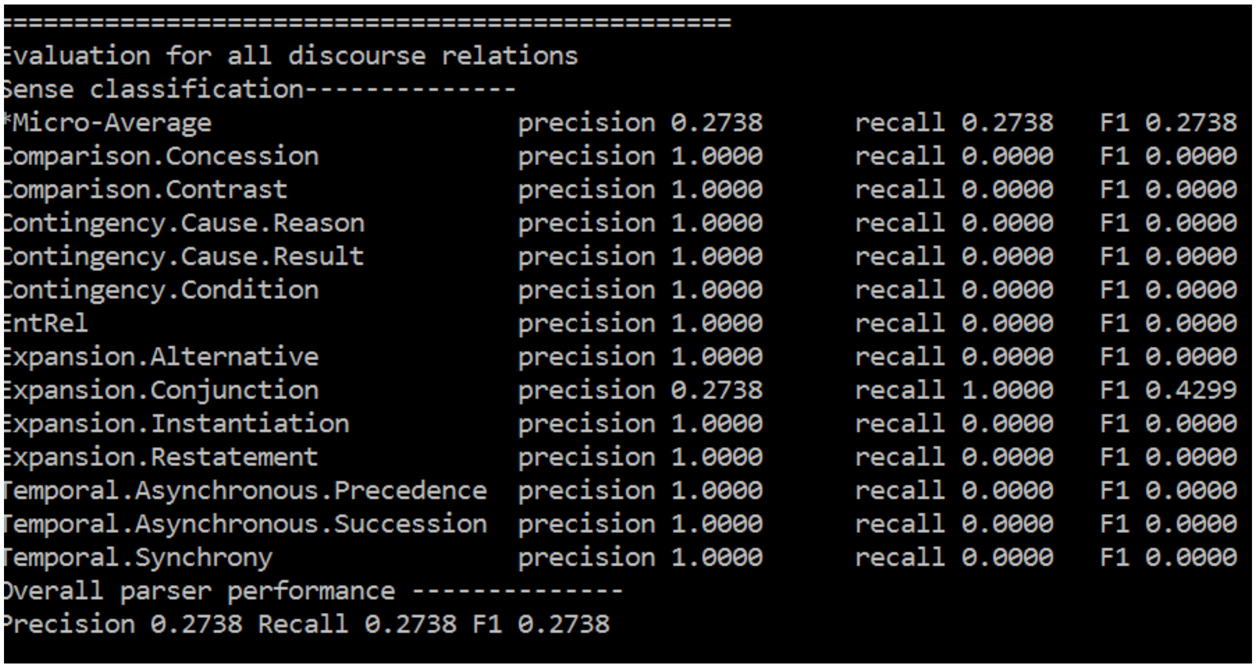
5 Hyperparameters Design

The hyperbolic tangent non-linear activation function is used for the input and hidden layers, and softmax is applied to the output layer. Dropout is .5 between each layer. The loss function

Adam (Adaptive Moment Estimation) was used to optimize stochastic gradient descent by computing an adaptive learning rate.

6 Experiments

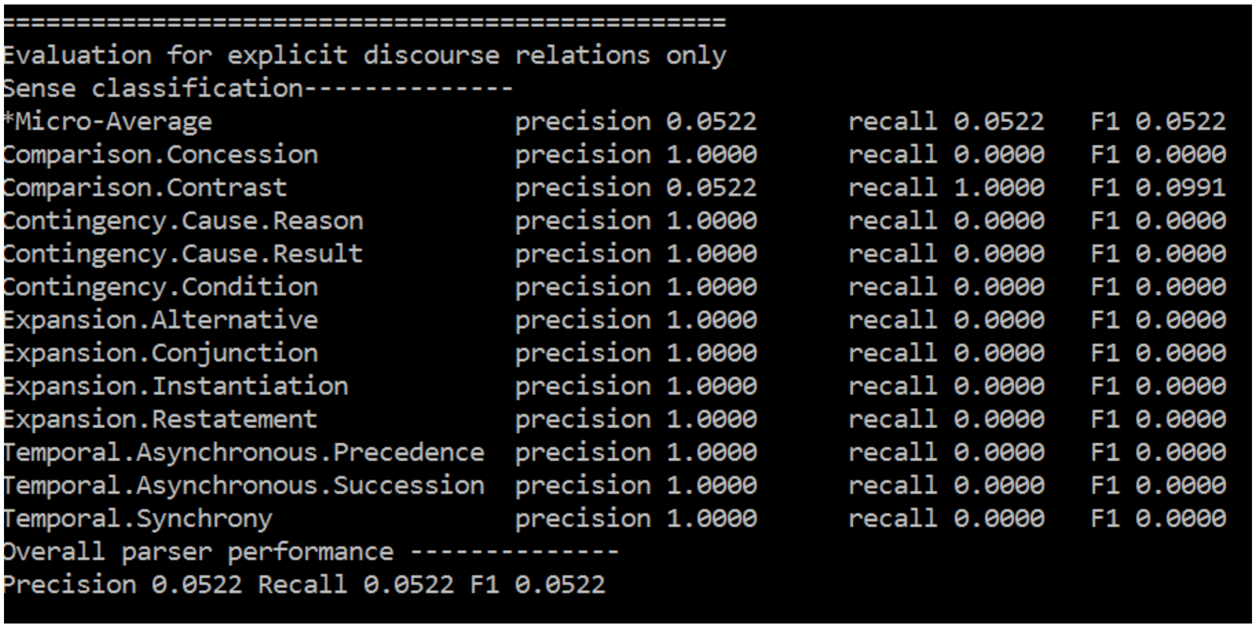
When fine-tuning my model, I experimented with both the architecture of the neural network and the hyperparameters. A major issue was that my model was only predicting one class.



```
===============================================
valuation for all discourse relations
ense classification--------------
*Micro-Average                   precision 0.2738      recall 0.2738   F1 0.2738
omparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
omparison.Contrast               precision 1.0000      recall 0.0000   F1 0.0000
ontingency.Cause.Reason          precision 1.0000      recall 0.0000   F1 0.0000
ontingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
ontingency.Condition             precision 1.0000      recall 0.0000   F1 0.0000
ntRel                            precision 1.0000      recall 0.0000   F1 0.0000
xpansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
xpansion.Conjunction             precision 0.2738      recall 1.0000   F1 0.4299
xpansion.Instantiation           precision 1.0000      recall 0.0000   F1 0.0000
xpansion.Restatement             precision 1.0000      recall 0.0000   F1 0.0000
emporal.Asynchronous.Precedence  precision 1.0000      recall 0.0000   F1 0.0000
emporal.Asynchronous.Succession  precision 1.0000      recall 0.0000   F1 0.0000
emporal.Synchrony                precision 1.0000      recall 0.0000   F1 0.0000
verall parser performance --------------
recision 0.2738 Recall 0.2738 F1 0.2738
```

When the class the model predicted was found frequently in the data, the model performed adequately.

But it the class the model predicted rarely occurred in the data, the accuracy and F1 scores were (predictably) terrible. To fix this glaring issue in the design of my model, I ran a lot of experiments trying to figure out the source of the problem.

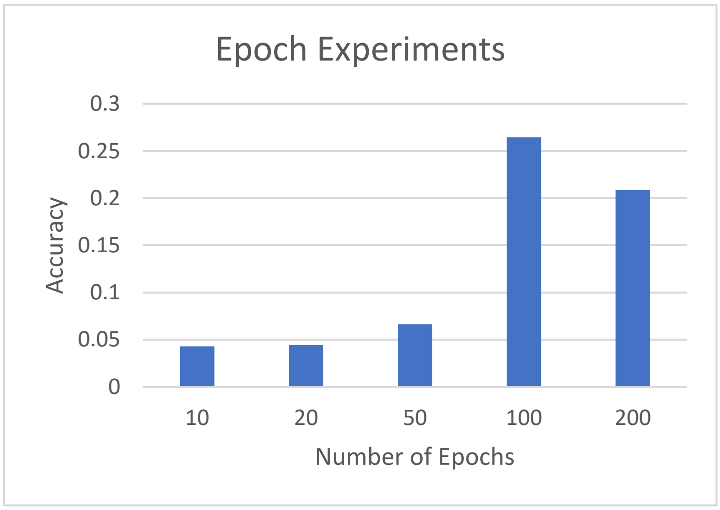


```
================================================
Evaluation for explicit discourse relations only
Sense classification--------------
*Micro-Average                     precision 0.0522      recall 0.0522   F1 0.0522
Comparison.Concession              precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast                precision 0.0522      recall 1.0000   F1 0.0991
Contingency.Cause.Reason           precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Cause.Result           precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Condition              precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Alternative              precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction              precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Instantiation            precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Restatement              precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Precedence   precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Succession   precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Synchrony                 precision 1.0000      recall 0.0000   F1 0.0000
Overall parser performance --------------
Precision 0.0522 Recall 0.0522 F1 0.0522
```

Through experimentation, I found that the problem with my model was the activation function. I initially used ReLU as my activation function for the input and hidden layer. I realized that "dying ReLU" might be what was happening with my model, so I changed the activation function to tanh instead. This greatly improved the accuracy of my model.

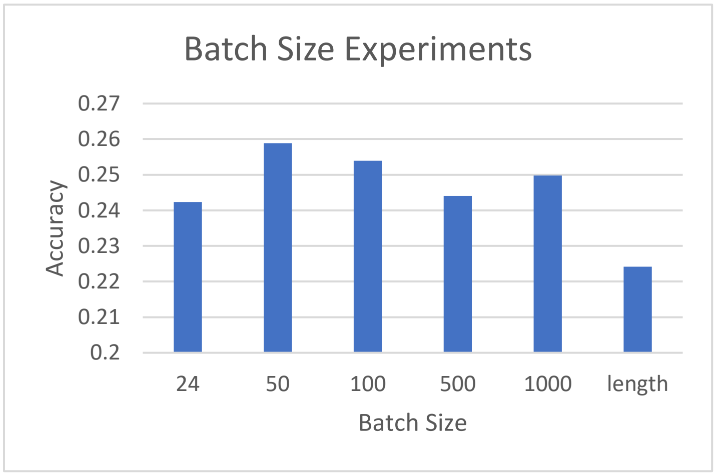
Another huge problem was that I was using way too many nodes in each layer to get good results. I started out with 64 nodes in the input and hidden layers, and experimented on different numbers of nodes to see if it would improve my model. Through experimentation, 32 nodes seemed to be an appropriate size.

Jennifer Storozum
CS135 Final Project Report
Shallow Discourse Parsing with Neural Networks
12/15/2017



I tried adjusting the number of epochs, which was initially set to 10 in order to speed up computation despite being aware that 10 epochs were definitely not enough. I ran experiments with 10, 20, 50, 100 and 200 epochs.

Next, I tried adjusting the batch size. I found that a batch size of 50 was ideal for my model, and that having batches that were too large would decrease performance.

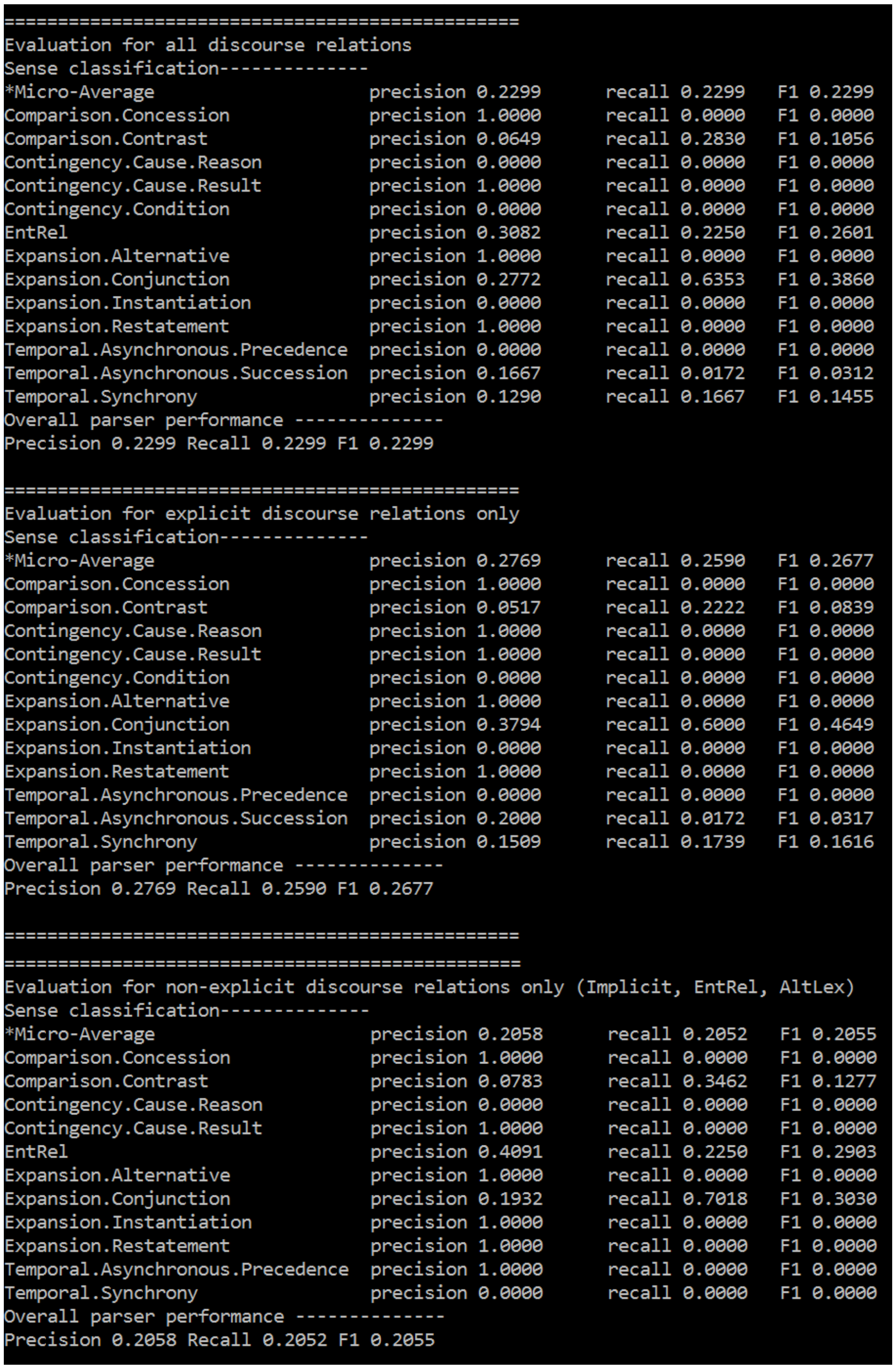


I also tried adjusting the dropout rate to see what impact it would have. My baseline test was dropout = .5, and I tested dropout levels of .2, .7 and .9. None of this seemed to have an impact on performance, so I left dropout at .5.

Initially, I used stochastic gradient descent for optimization. But after a number experiments where changing the learning rate (using .1, .01, and .001) failed to impact performance, I abandoned SGD in favor of Adam, which is a loss function that optimizes stochastic gradient descent by computing an adaptive learning rate for each parameter.

Sample results with my best model are as follows:

Jennifer Storozum
CS135 Final Project Report
Shallow Discourse Parsing with Neural Networks
12/15/2017



```
================================================
Evaluation for all discourse relations
Sense classification--------------
*Micro-Average                    precision 0.2299      recall 0.2299   F1 0.2299
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0649      recall 0.2830   F1 0.1056
Contingency.Cause.Reason          precision 0.0000      recall 0.0000   F1 0.0000
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Condition             precision 0.0000      recall 0.0000   F1 0.0000
EntRel                            precision 0.3082      recall 0.2250   F1 0.2601
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.2772      recall 0.6353   F1 0.3860
Expansion.Instantiation           precision 0.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Precedence  precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Succession  precision 0.1667      recall 0.0172   F1 0.0312
Temporal.Synchrony                precision 0.1290      recall 0.1667   F1 0.1455
Overall parser performance --------------
Precision 0.2299 Recall 0.2299 F1 0.2299

================================================
Evaluation for explicit discourse relations only
Sense classification--------------
*Micro-Average                    precision 0.2769      recall 0.2590   F1 0.2677
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0517      recall 0.2222   F1 0.0839
Contingency.Cause.Reason          precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
Contingency.Condition             precision 0.0000      recall 0.0000   F1 0.0000
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.3794      recall 0.6000   F1 0.4649
Expansion.Instantiation           precision 0.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Precedence  precision 0.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Succession  precision 0.2000      recall 0.0172   F1 0.0317
Temporal.Synchrony                precision 0.1509      recall 0.1739   F1 0.1616
Overall parser performance --------------
Precision 0.2769 Recall 0.2590 F1 0.2677

================================================
================================================
Evaluation for non-explicit discourse relations only (Implicit, EntRel, AltLex)
Sense classification--------------
*Micro-Average                    precision 0.2058      recall 0.2052   F1 0.2055
Comparison.Concession             precision 1.0000      recall 0.0000   F1 0.0000
Comparison.Contrast               precision 0.0783      recall 0.3462   F1 0.1277
Contingency.Cause.Reason          precision 0.0000      recall 0.0000   F1 0.0000
Contingency.Cause.Result          precision 1.0000      recall 0.0000   F1 0.0000
EntRel                            precision 0.4091      recall 0.2250   F1 0.2903
Expansion.Alternative             precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Conjunction             precision 0.1932      recall 0.7018   F1 0.3030
Expansion.Instantiation           precision 1.0000      recall 0.0000   F1 0.0000
Expansion.Restatement             precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Asynchronous.Precedence  precision 1.0000      recall 0.0000   F1 0.0000
Temporal.Synchrony                precision 0.0000      recall 0.0000   F1 0.0000
Overall parser performance --------------
Precision 0.2058 Recall 0.2052 F1 0.2055
```