TTIC 31190 HW4: Paraphrase Identification

Lynn Zheng
University of Chicago
ruolinzheng@uchicago.edu

Abstract

This project is about pairwise paraphrase identification. We are given an original version of the task as well as a hard version. To approach this problem, I experimented with different ways of organizing and cleaning data for training. I compared the approach of hand-crafting simple features for classification to the approach of building end-to-end neural models. Much to my surprise, hand-crafted features substantially outperformed the neural model. This may imply that simple heuristics and statistics work well for the task of paraphrase identification.

1 Introduction

1.1 Dataset

We are given a large training corpus with no negative samples, as well as labeled DEV and DEVTEST sets for both the original and hard version of the task. We will refer to these datasets as TRAIN, DEV, DEVTEST, TEST and DEV-HARD, DEVTEST-HARD, TEST-HARD in this report.

1.2 Idea and Motivation

I approached this paraphrase identification problem as a binary classification problem. I experimented with both classifiers built on hand-crafted features and end-to-end neural models. I used two hand-crafted features: the number of unique overlapping unigrams in the two sentences, and their Levenshtein edit distance. My idea was that a paraphrase sentence pair will likely have many overlapping words and a relatively small edit distance. Using these two features, I trained Multi-Layer Perceptrons (MLP). I also compared different data configurations for the MLP models, such as the size, source of the training data and different methods of text cleaning including tokenization, removing punctuation and stopwords, and stemming. For the end-to-end neural model, I implemented a Recurrent Neural Network (RNN) that is initialized with the GloVe word embeddings and tunes it during training.

1.3 Results and Analysis

The MLP models with hand-crafted features substantially outperformed the RNN model both in terms of absolute accuracy and computation cost. Data preprocessing and cleaning, especially the removal of punctuation and stopwords, also contributed significantly to improving the performance of the classifier.

2 Methodology

2.1 Training Data Organization

TRAIN didn't contain negative examples and it is non-trivial to generate negative examples that mimic the ones in the HARD datasets using simple techniques like random insertion, deletion, substitution, or permutation. Hence, I decided to use the labeled datasets both for training and validation. However, the labeled datasets are small: DEV and DEVTEST contain 800+ pairs each, and DEV-HARD and DEVTEST-HARD contain 1000 pairs each. To obtain sufficient data for training, I concatenated DEV with DEVTEST, and DEV-HARD with DEVTEST-HARD, and then performed 25% train-test-split. I will refer to the resultant test sets as NORMAL and HARD, and the original DEV and DEV-HARD as NORMAL-SMALL and HARD-SMALL from here on. I also tried concatenating NORMAL with HARD to produce a larger training set NORMAL+HARD, in hope of fitting one model to solve both the original hard version of this paraphrase identification task.

2.2 Data Cleaning

In addition to word tokenization, I experimented with techniques like removing punctuation, stopwords, and stemming with the Porter stemmer. I used the stopwords from `nltk`. In particular, Removing stopwords significantly alters the value

of my hand-crafted features, the number of overlapping unigrams and the number of stopwords. Please refer to the Tables in the appendix for examples.

2.3 Feature Engineering

2.3.1 Number of Unique Overlapping Unigrams

For each sentence pair, I count the number of unique overlapping unigrams. My idea is that paraphrase pairs should have more overlapping unigrams than non-paraphrase pairs. As each sentence is short and is unlikely to have many duplicate unigrams, there shouldn't be much different between counting all occurrences of overlapping unigrams and counting only unique occurrences.

2.3.2 Levenshtein Edit Distance

For each sentence pair, I computed the word-wise edit distance between the two sentences. Intuitively, sentences that are paraphrase of each other should have similar morphological structures and hence a smaller edit distance than a non-paraphrase pair. This turns out especially useful for the HARD datasets, where we have sentences that might have the exact same words but in different orders, resulting in a different subject-object relationship. For instance, below is a negative example from DEVTEST-HARD:

```
Holmes described Moriarty as
   follows :
Moriarty described Holmes as
   follows :
```

2.3.3 Feature Visualization

We show the correlation matrices of tokenized DEV and DEV-HARD. For both datasets, the correlation between the two features is positive. Interestingly, this differs from my heuristics that a paraphrase pair should have a large number of overlapping unigrams but a small edit distance. One possible explanation is that both variables are correlated with the length of sentence and hence their correlation is confounded. For DEV, the number of unigrams is quite strongly positively correlated with the label, which aligns with my hypothesis that more overlapping unigrams contribute positively to the probability of a sentence pair being paraphrases.

For both DEV and DEV-HARD, edit distance is fairly negatively correlated with the label, which agrees with my hypothesis that paraphrase pairs tend to have smaller edit distances.

	unigrams	edit_dist	labels
unigrams	1.0000	0.2877	0.4417
edit_dist	0.2877	1.0000	-0.2765
labels	0.4417	-0.2765	1.0000

	unigrams	edit_dist	labels
unigrams	1.0000	0.2335	-0.0429
edit_dist	0.2335	1.0000	-0.2896
labels	-0.0429	-0.2896	1.0000

Table 1: Correlation matrices of tokenized (top) DEV and (bottom) DEV-HARD

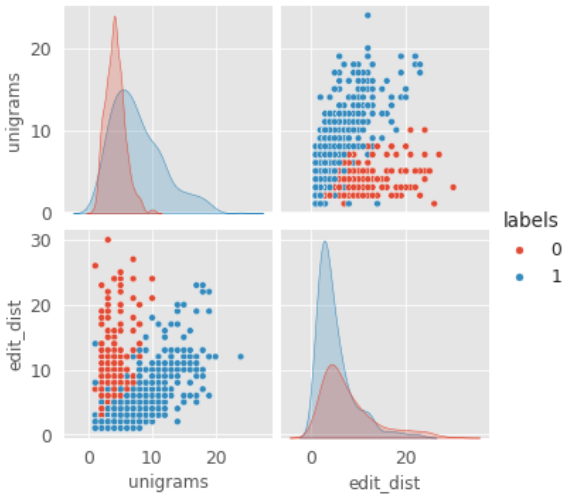

Figure 1: Features of tokenized DEV

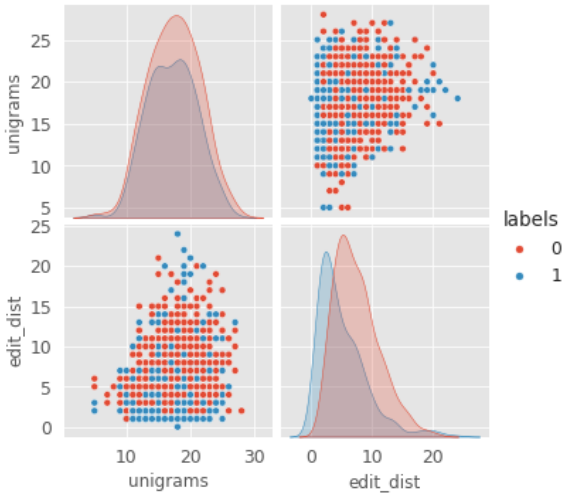

Figure 2: Features of tokenized DEV-HARD

When we apply further data cleaning like the removal of punctuation and stopwords, and stemming, the magnitude of the correlation between the edit distance and labels increased, indicating that edit distance is now a stronger predictor.

	unigrams	edit_dist	labels
unigrams	1.0000	0.2471	-0.0343
edit_dist	0.2471	1.0000	-0.4299
labels	-0.0343	-0.4299	1.0000

Table 2: Correlation matrix of DEV-HARD with tokenization, removal of punctuation and stopwords, and stemming

2.4 Models

2.4.1 MLP

I used the `MLPClassifier` from `sklearn` which has one hidden layer with 100 hidden units, uses the ReLU activation function, and the Adam optimizer.

2.4.2 RNN

I used the GloVe pre-trained embedding `glove.6B.50d.txt` which is trained on six billion tokens and of embedding dimension 50. I randomly initialized embeddings for special tokens like the start and end token and the pad token. I fine-tuned the embedding during training. In its forward pass, the model takes a sentence pair and pass each through its RNN unit. It then concatenates the two RNN outputs and pass them through two fully-connected layers to obtaina single scalar output. The scalar output then passes through the sigmoid activation function. I used the binary cross entropy loss , the Adam optimizer with a learning rate or 0.0001, and minibatches of size 1.

```
RNNClassifier(
  (emb): Embedding(400004, 50)
  (rnn): GRU(50, 128)
  (fc1): Linear(in_features=256,
     out_features=128, bias=True)
  (fc2): Linear(in_features=128,
     out_features=1, bias=True)
)
```

3 Experiments

3.1 MLP

I experimented with the following configurations of organizing and cleaning data:

1. Dataset
   - NORMAL-SMALL and HARD-SMALL
   - NORMAL and HARD
   - NORMAL+HARD to train a single model for both tasks
2. Data cleaning
   - Tokenization
   - Removing punctuation
   - Removing stopwords
   - Stemming with the Porter stemmer

3.2 RNN

As training RNNs is computationally heavy, I used NORMAL+HARD to train a single model for the original and hard version of the task. I trained the model for 50 epochs, which took approximately 3 GPU hours.

4 Results

4.1 Accuracy

Model	Train Set	Kaggle Test	Score
MLP	NORMAL	TEST	0.9320
MLP	HARD	TEST-HARD	0.7347
RNN	NORMAL+HARD	TEST	0.7000
RNN	NORMAL+HARD	TEST-HARD	0.6140

Table 3: Kaggle TEST and TEST-HARD scores

For the original version of the task, the MLP model trained on tokenized NORMAL achieved the highest score. For the hard version of the task, the MLP model trained on HARD with tokenization, removal of punctuation and stopwords, and stemming performed the best. Please refer to the Tables in the appendix for the full accuracy table.

4.2 MLP

4.2.1 Error Analysis

For the hard version of the task, when the data-cleaning only involves tokenization, the model erred a lot on false negatives. When we remove punctutaion and stopwords and apply stemming, false negatives decreased substantially.

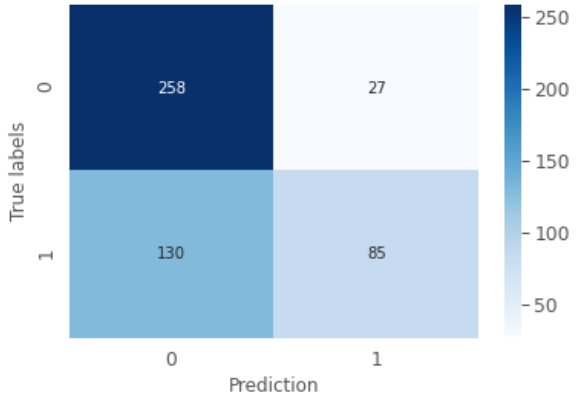

Figure 3: Confusion matrix of the validation set for a model trained on tokenized HARD

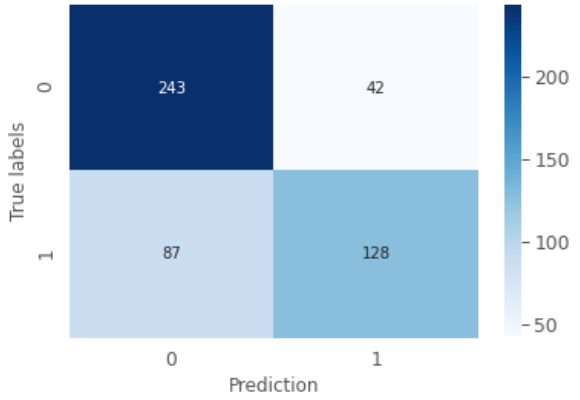

Figure 4: Confusion matrix of the validation set for the model that achieved the highest score on Kaggle TEST-HARD, trained on HARD with tokenization, removal of punctuation and stopwords, and stemming

Below is a false negative example from DEV-HARD corrected by text cleaning.

```
After his service , Lockhart
   lived in Florida , but moved
   to Texas recently .
After his service Lockhart lived
   in Florida but recently moved
   to Texas .

['servic', 'lockhart', 'live',
   'florida', 'move', 'texa',
   'recent']
['servic', 'lockhart', 'live',
   'florida', 'recent', 'move',
   'texa']
```

The edit distance between the tokenized version of the top two sentences is 4 and that between the bottom two sentences with tokenization, removal of punctuation and stopwords, and stemming is 2. Given the strongly negative correlation between the edit distance and the label, this indicates a higher probability that the model will label this pair a paraphrase pair.

4.3 RNN

As the plot below shows, the change in accuracy is noisy from epoch to epoch. This might mean that the training could have benefited from a smaller learning rate.

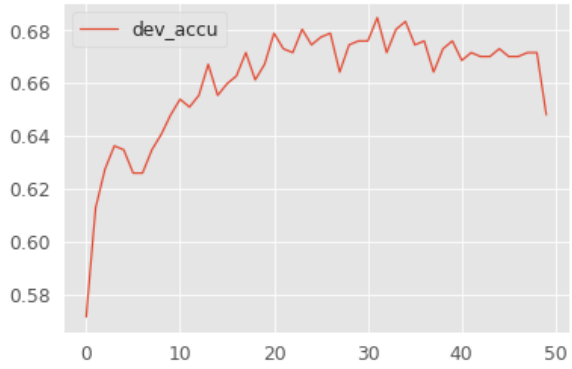

Figure 5: Validation Accuracy of the RNN Trained on NORMAL+HARD

Furthermore, the RNN is trained on NORMAL+HARD to save computation cost and to guarantee that there is enough data for the neural model to learn from. A better approach will be to train separately on NORMAL and HARD, as the two datasets are very different in nature. Our first model, MLP, also performs less well when trained on NORMAL+HARD as shown in the Tables in the appendix.

5 Conclusion

This paraphrase identification task demonstrates the power of hand-crafted features. In comparison to neural models, they are interpretable, perform well, and cost a lot less computation resource to train. This last attribute also greatly facilitates my experimentation with organizing and cleaning data for both the original and hard version of the task. I found that text cleaning has great significance for feature engineering. Operations as simple as removing punctutaion, stopwords, and stemming could notably increase the correlation between the predictor, edit distance, and the target label of whether a sentence pair is paraphrase of each other.

A Tables

Table 4: Data cleaning example using a negative sentence pair from DEVTEST-HARD

	DEVTEST-HARD Example	DEV-HARD Negative Example	Edit Distance
Original	Steam can also be used , and does not need to be pumped .	Steam can also be pumped and need not be used .	-
Tokenize	['steam', 'can', 'also', 'be', 'used', ',', 'and', 'does', 'not', 'need', 'to', 'be', 'pumped', '.']	['steam', 'can', 'also', 'be', 'pumped', 'and', 'need', 'not', 'be', 'used', '.']	6
Remove Punctuation	['steam', 'can', 'also', 'be', 'used', 'and', 'does', 'not', 'need', 'to', 'be', 'pumped']	['steam', 'can', 'also', 'be', 'pumped', 'and', 'need', 'not', 'be', 'used']	5
Remove Stopwords	['steam', 'also', 'used', ',', 'need', 'pumped', '.']	['steam', 'also', 'pumped', 'need', 'used', '.']	3
Porter Stemmer	['steam', 'can', 'also', 'be', 'use', ',', 'and', 'doe', 'not', 'need', 'to', 'be', 'pump', '.']	['steam', 'can', 'also', 'be', 'pump', 'and', 'need', 'not', 'be', 'use', '.']	6

Table 5: Accuracy of different MLP configurations ordered by increasing training accuracy

Training Dataset	Tokenized	Remove Punctuation	Remove Stopwords	Stemmed	Train Accuracy	Validation Accuracy
NORMAL+HARD	+	-	-	-	0.6901	0.6678
NORMAL+HARD	+	-	-	-	0.6901	0.8128
NORMAL+HARD	+	-	-	-	0.6901	0.5500
NORMAL	+	+	+	-	0.7757	0.8128
NORMAL	+	+	+	+	0.7946	0.8177
NORMAL	+	+	-	-	0.8233	0.8522
NORMAL	+	+	-	+	0.8332	**0.8596**
NORMAL-SMALL	-	-	-	-	0.8411	0.8137
NORMAL	+	-	-	-	0.8422	0.8374
NORMAL-SMALL	+	-	-	-	0.8447	0.8298
HARD	+	-	-	-	0.6813	0.6860
HARD-SMALL	+	-	-	-	0.6890	0.6680
HARD	+	+	-	-	0.6933	0.7060
HARD	+	+	-	+	0.6980	0.7100
HARD	+	+	+	+	0.7120	**0.7420**
HARD	+	+	+	-	0.7127	**0.7420**
HARD-SMALL	-	-	-	-	0.7250	0.7000