Paraphrase Detection Progress Report

Ari Conati
Carleton College
conatia@carleton.edu

Elliot Pickens
Carleton College
pickense@carleton.edu

Chiraag Gohel
Carleton College
gohelc@carleton.edu

1 Introduction

There is no one way to convey an idea through text. Any given sentence could be swapped out for an infinite number of replacements without changing its core meaning. Conversely, sentences with the same or similar word compositions could have very different meanings. This presents us with the interesting question of paraphrase detection. How can we effectively evaluate sentiment, or translate a piece of writing without some understanding that multiple sentences can have equivalent meaning, and beyond that how can we define similarity between sentences? We attempt to answer this question in paraphrase detection, in which our task is to determine whether two given sentences are roughly semantically equivalent.

2 Related Work

2.1 Automatically Constructing a Corpus of Sentential Paraphrases

A challenge in paraphrase detection is a lack of large corpora of paraphrases to train and evaluate models (Dolan and Brockett, 2005). Paraphrases do not often occur in the wild, so it is difficult to come up with a suitable corpus of naturally occurring sentence pairs. In particular, an ideal corpus should contain a mix of paraphrases and near-miss sentence pairs, so that detecting paraphrases is non-trivial. These sentence pairs, however, must be selected in a way that minimizes potential bias, so the data is representative enough such that algorithms trained on it is generalizable. Dolan and Brockett describe the method by which the Microsoft Research Natural Language Processing Group devised the Microsoft Research Paraphrase Corpus (MSRP) in response to this problem.

The starting point for the MSRP was a database of sentence pairs collected from topically clustered news articles, which were identified as a source for potential paraphrases. From here, candidate sentence pairs were chosen based on a number of heuristics, including word overlap (sentence pairs shared at least 3 words), length ratio (the sentence of the pair with fewer words was at least the length of the longer), and edit distance ($e > 8$). The goal was to arrive at sentence pairs which are similar as described above, but not overly so (in particular, sentence pairs which only have spelling or typographical differences).

Finally, an SVM classifier was trained on a separate set of 10,000 sentence pairs which labeled as paraphrases and non-paraphrases by two human annotators. The SVM classifier was restricted to a small set of features and designed to over-identify paraphrases (the goal of the classifier is not to identify paraphrases, but to extract a set of suitable set of pairs of similar sentences). The final dataset was randomly selected from the set of sentence pairs which the SVM classifier identified as paraphrases. The final dataset was labeled by a pair of human judges (with a third to tiebreak), who were asked to label sentences as paraphrases if they could be considered semantically equivalent.

2.2 Syntax-Aware Multi-Sense Word Embeddings for Deep Compositional Models of Meaning

Over the past 20 years a number of different methods have been developed to create distributional models of meaning that have applications in various language based tasks (Cheng and Kartsaklis, 2015). These models assume the distributional hypothesis that two words that occur in similar contexts have similar meanings (1532). In recent years various neural distributional models have sought to create word representations by treating

them as inputs to a word prediction problem that can be trained. One of the most well known examples of this is the skip-gram model that maximizes an objective function using a neural network. Distributional models list the skip-gram model have been shown to be very effective in modeling the semantic relationships between words, but they fail to reflect the syntactic positioning of words that can easily alter their meaning.

In this paper the authors seek to address that problem by expanding on the idea of creating word embeddings in order to create sense embeddings that better reflect the syntaxical composition of sentences. The model the authors propose in this paper is based around either a recursive or recurrent neural network (RecNN and RNN for short). Both of these forms of neural networks allow for word embeddings and layer parameters to be learned by optimizing a generic objective function that uses hinge loss (1533). In both RecNN and RNN models a compositional layer is applied repeatedly to sets of inputs using the function $p = g(Wx_{[1:2]} + b)$ in either a recursive manner (following a parse tree), or from left to right for RecNN and RNN models respectively. The authors then introduce a compositional layer that attempts to evaluate the linguistic probability of a phrase or sentence vector based on both semantics and syntaxes. They do this by generating a number of random negative samples by distorting input sentences in order to convert the task of plausibility to a supervised one that can be used to improve the consistency of the learned word embeddings. Lexical ambiguity is then addressed by applying a gated architecture (1534). They assume that each word has some fixed number n of senses, and that each word is associated with some main vector (and some number n vectors that denote centroid centers). Then when it comes time to choose a sense they are able to compare a context vector to the centroids associated with a word w_t by cosine similarity in order to choose best possible sense.

For the task of paraphrase detection the authors use the sense embedding architecture as an initial step to create sense embeddings that can then feed into a siamese architecture for the purpose of classification. The siamese architecture works by pushing the two sentences through two networks with equal siamese weights that compare the two vectors using cost function based on either the L_2 norm or cosine similarity.

Before training their classifier they pre-train the RecNN (or LSTM RNN) using the British National Corpus, so they can get better quality senses from the 10,000 sentences in the $MRP_{araphrase}$ dataset. For the purpose of paraphrase detection they also choose to add two extra steps to their model. First they add a pooling layer to sense embedding architecture to better reflect local sentence features that can be lost in during the compositional phase of RecNN and RNN models. Second they choose to ensemble the siamese classifier with a logistic regression classifier that has been trained using a set of hand crafted features like n-gram overlap and the difference in sentence length. The final most effective version of their model uses a RecNN with a pooling layer to create the sense embeddings, and an ensemble of the siamese classifier trained on the embeddings, and a logistic regression classifier trained using the hand crafted features.

2.3 Discriminative Improvements to Distributional Sentence Similarity

Bag of word representations of short units of text for the purposes of measuring semantic similarity suffer from high sparsity (Ji and Eisenstein, 2013). Distributional methods address this issue by transforming term-context count matrices into lower-dimensional latent spaces. Unfortunately, dimension reduction involves loss of information that may impede natural language processing tasks. Ji and Eisenstein describe how labeled data may improve distributional methods for measuring semantic similarity in three ways.

Ji and Eisenstein begin by creating a new discriminative term-weighting metric named TF-KLD. Similar to Linear Discriminant Analysis, the scheme attempts to increase the weights of discriminative distributional features, and decrease those that are not, based on supervised data. They use the Kullback-Leibler divergence measure to determine the discriminability of words in a given training set as to reweight the features in the sparse matrix before factorization. Such increases the weights of words whose likelihood of appearing in a pair of sentences is strongly influenced by whether the sentences are paraphrases.

TF-KLD outperformed previous TF-IDF benchmarks, regardless of factorization technique, when predicting whether a pair of sentences is a paraphrase by measuring their cosine similarity in latent space (dimensionality K = 100). Furthermore TF-KLD supervised classification methods, with added fine-grained lexical features, outperformed the prior state-of-the-art.

3 Data Set

The data set we are using is the Microsoft Research Paraphrase Corpus (MSRP), which is a database of 5801 sentence pairs (Dolan and Brockett, 2005). These sentences were collected from news clusters across the World Wide Web, with sentences being added to the database based on a number of heuristics (including for instance, edit distance and word overlap) intended to ensure that the sentences had some lexical similarities. Each of these sentence pairs is accompanied by a judgment of whether the two sentences are paraphrases of each other (that is, they are roughly semantically equivalent). These judgments were decided by two human raters (with a third to settle disagreements), who were instructed to determine whether the sentences 'mean the same thing' on a high level. The criterion for semantic equivalence is intentionally loose and somewhat vague, to prevent the set of paraphrases from containing only sentences which are practically equivalent at the string level.

The sentences were randomly divided into a training set and a test set, with the training set containing roughly 70% of the sentence pairs, and the other 30% comprising the test set.

Training Set Length	4076
# of Paraphrases in Training Set	2753
Testing Set Length	1725
# of Paraphrases in Testing Set	1147
Average Training Sentence Length	19.7728
Average Testing Sentence Length	19.648

4 Evaluation Metrics

For this problem we are simply trying to predict whether or not a pair of sentences are paraphrases of one another. Thus, we are able to classify every pair of sentences as being a 1 or 0 based on its paraphrase status. Given the existence of these convenient labels we are able to use fairly straight forward binary classification metrics to evaluate our model. Those metrics are accuracy, precision, recall, and the f1 score. We use these 4 metrics to analyze our model, because due to the class imbalances we have in our data set we need to be able to investigate the ways in which our model is handling both classes.

5 Baseline

For our baseline we compared the n-gram overlap of our sentence pairs. The n-gram overlap is done by first finding the number of n-grams that exits in both sentences, and then finding the n-gram ratio by taking that value over the average length of the two sentences. We then set a classification threshold by finding the average n-gram overlap of the of the sentence pairs of class 1 (the paraphrases). Finally we use our threshold to classify the sentence pairs in our data set by classifying pairs with a overlap value above the threshold as being part of class 1, and all others as being part of class 0.

We ran this classification system for n-grams of lengths 1 through 5, and achieved values for all 4 metrics that look to be at least slightly better than random guessing. In general we saw accuracy values for all lengths above 60%, and surprisingly high precision values that hovered in the high 70% range.

Model	Accuracy	Precision	F1
1-gram	**65.95**	**80.28**	**71.63**
2-gram	63.69	78.95	69.37
3-gram	60.15	77.22	65.46
4-gram	58.41	76.78	63.17
5-gram	56.96	76.51	61.11

Table 1: Results from baseline model using various n-gram lengths.

6 Improved Baseline

Our better than baseline model implements gensims doc2vec function to create sentence embeddings by training the doc2vec model on the training subset of the $MRP_{araphrase}$ data. We then create a feature set for each pair of sentences in the training set by taking the outer product of each pair of embeddings. The outer products are then scaled, and truncated using an SVD method to reduce the number of dimensions down to 100. These features are then feed into a cross validated grid search methods for both logistic regression, and a linear SVM. We also tried using random

forests fit using Extra Trees, and the standard decision tree method based on parameters found using a randomized grid search. It should be noted that the random forest methods were fit using an unscaled, un-truncated version of the features.

We then predicted class labels for our test set by first inferring sentence embeddings using the trained doc2vec model, and then creating features using the same set of methods described above.

Model	Accuracy	Precision	F1
SVC	61.80	**70.68**	71.68
Log. Reg.	67.53	69.84	78.68
Extra Trees	**68.34**	68.38	80.38
Random Forest	68.23	67.78	**80.65**

Table 2: Better than baseline results from various models run on doc2vec sentence embeddings.

7 External API

BERT (Bidirectional Encoder Representations from Transformers) makes use of Transformer to create sentence embeddings. Bert is used in the current state of the art paraphrase detection models, so we decided to try and employ it in our API based model. Pre-trained versions of the BERT transformer along with a myriad of python scripts that can be used to apply it can be found on Google Research's Github page. These scripts even include a pre-written classifier that can be run directly on the $MRP_{araphrase}$ data. Their pre-canned classifier outputs accuracy values in the mid to high 80s, and easily out preforms almost all paraphrase detection models that are not an absolutely enormous ensemble method designed by Microsoft or Alibaba.

We wanted to do something using the BERT embeddings other than just run Google's classifier. To do this we started by getting token by token embeddings for all of the sentences in the $MRP_{araphrase}$ corpus using the 1024 layer uncased version of BERT. This produced nearly 20 Gb of token embeddings in the form of json lists. Given that we could not load all of these embeddings into memory at once we choose to only used the weights given by the final layer of the transformer, and to used to average weight of each token instead of including all 300 weights originally assigned to each token. These average weights are then combined to form the sentence features. The sentence feature lists are, however, of different lengths due to the different token lengths of the sentences. To fix jaggedness of the array that results from stacking all of these sentence features we pad out all of the features list using zeros. Finally to get features we once again take the outer product of out sentence vectors.

We then tried a few different methods of classification. First we ran the data through the same logistic regression, and random forest classifiers we used in our better than baseline model. These methods preformed fairly well but we wanted to do something better. We were able to improve our model by taking some inspiration from the classifier used in the "syntax aware sense embeddings," and creating an ensemble using a classifier run on the outer product of the BERT embeddings and a logistic regression classifier trained on the hand crafted features of unigram overlap, and the difference in sentence length. We tried a number of of different classifiers to use in our ensemble, and have had promising results using both regular random forests, and AdaBoost.

Model	*Accuracy*	*Precision*	*F1*
Log. Reg.	66.49	66.49	79.88
Random Forest	68.17	68.12	**80.37**
RF + LR	69.33	69.99	80.36
AdaBoost + LR	**69.97**	**71.98**	79.91
GradientBoost + LR	68.93	69.98	79.97

Table 3: External API results from various models run on BERT sentence embeddings.

8 Shortcomings and Future Work

Words can have drastically or subtly different meanings depending on the context they appear in (Cheng and Kartsaklis, 2015). The doc2vec model that we use makes the simplifying assumption that a word has just one possible embedding when generating its sentence embeddings. Looking into methods of capturing the context of words and adjusting our sentence embeddings accordingly could greatly improve the performance of our models.

There is also a great deal of room for improvement in our implementation of the BERT API. In particular, we are currently failing to create truly informative sentence embeddings due to the differing lengths of the sentences in our corpus. We could likely greatly improve our model by building truly normalized sentence level embeddings rather than simply padding a very jagged array. We tried using the average embedding values of the sentence as

padding instead of simply using zeros for all sentences, but this failed to improve our model leading us to believe we need a more drastic fix.

9 Conclusion

We implemented a number of different model variants for each of the three categories (baseline, baseline**+**, API). The three selected models (the overall best performing from each category) perform well in different areas. The unigram baseline model boasts the highest precision of 80.23%, indicating a conservative approach with a higher threshold for labeling a set of sentences as paraphrases. Such can be understood through the nature of the model; setting a threshold based on n-gram overlap reduces the probability of the model assigning two sentences as paraphrases without a certain proportion of n-gram equivalence. The model fails to account for the semantic equivalence of n-grams.

Model	*Accuracy*	*Precision*	*F1*
Baseline (unigram)	65.95	**80.28**	71.63
Baseline**+** (Extra Trees)	68.34	68.38	**80.38**
API (AdaBoost + LR)	**69.97**	71.98	79.91

Table 4: Selected models from each baseline level.

The extra trees model run on *doc2vec* sentence embeddings performs similarly on the corpus to the AdaBoost and logistic regression model using Bert sentence embeddings. Both produce F1 scores higher than that of the baseline model, with our better than baseline model touting the highest (80.38%). The external API model produces the highest accuracy (69.97%).

The inability of one sole model to perform sufficiently better than the others in regard to desired metrics demonstrates the intricacy of the paraphrase detection task. Models based on word embeddings show promise in their ability to represent sentence semantics.

References

Jianpeng Cheng and Dimitri Kartsaklis. 2015. Syntax-aware multi-sense word embeddings for deep compositional models of meaning. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1531–1542, Lisbon, Portugal. Association for Computational Linguistics.

William B. Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

Yangfeng Ji and Jacob Eisenstein. 2013. Discriminative improvements to distributional sentence similarity. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 891–896, Seattle, Washington, USA. Association for Computational Linguistics.