Benjamin Tibbetts and Benjamin Zifkin

Paraphrase Detection using Supervised Learning

Abstract:

We examined the task of detecting paraphrases from lexically divergent tweets using supervised learning. Tweets present many hurdles to many modern NLP techniques that rely on a large document or corpus to learn their models. We approach the issue with a variety of models and features. Our results, while satisfactory, illustrate the difficulty in engineering features for a task such as this.

Introduction:

There are several uses for paraphrase detection in the domain of natural language processing. Paraphrase identification can play a role in information retrieval, translation, question answering, and summarization. Most work on paraphrase detection has been done on more formal corpora. This is because larger, more grammatically rigorous documents are easier to engineer features around and train classifiers on, amongst other things. However, with the proliferation of social media and microblogging sites, e.g. Twitter, it has become increasingly important there exists some means by which the data contained in these sites can be accessed and organized. Paraphrase extraction is not only useful to researchers, but also to Twitter itself; where ideally it would be able to display informative and diverse tweets to its users, rather than repetitive or redundant ones. Additionally, it is exciting, from a computational linguistics standpoint, to develop tools and techniques on an emerging medium loaded with slang, colloquialisms, and abbreviations. In this paper we aim to explore techniques to identify paraphrases amongst tweets. In doing so we will also highlight some of the specific features that can be particularly useful in this task, as well as those that are superfluous.

Related Work:

Dr. Wei Xu and colleagues developed a very effective system called MultiP in a paper titled *Extracting Lexically Divergent Paraphrases from Twitter(2014)*. Dr. Xu begins by asserting that two tweets under the same topic are paraphrases if they contain at least one word pair indicative of sentimental paraphrase. These similar word pairs are called anchors by the author. Sentence pairs can have multiple anchors, and she calls the assumption of an anchor pair between paraphrases an *at-least-one-anchor* assumption. This assumption, along with the overall simplicity of the model, allows for greater manipulation of linguistic resources part-of-speech tags, and other arbitrary features. Though the author notes this approach might be ineffective for long sentences, it is very effective for tweets that are topically/temporally related. However, it is important to note that even in a pair of topically aligned tweets a word pair of similar meaning does not necessarily denote a paraphrase. For example in two sentences aligned on the topic of "Iron Man" the shared word "3" does not indicate a paraphrase :

- Iron Man 3 was brilliant fun
- Iron Man 3 tonight see what this is like

In order to ascertain which word pairs are genuine anchors the authors employ a discriminative model at the word-level to incorporate features to determine the probability of a word pair as an anchor.

This model creates a situation where the learner observes labels on bags of instances, (in this case a pair of tweets) as opposed to on individual instances(i.e. a word pair) themselves. This is called multiple instance learning, hence the name MultiP. The authors define an undirected graphical model that works at both the sentence and word level. At the sentence level there is an aggregate binary variable, observed in the labelled training data, which indicates whether two sentences are paraphrases. Additionally there is a latent variable for each word pair of the pair of sentences, denoting whether the word pair is a paraphrase anchor. To learn the parameters of the

word-level anchor classifier they maximize likelihood over sentence-level annotations. This is done through an iterative perceptron-based update function.

At the word-level the discriminative model extracts these features for every word pair:

String Features that indicate whether the two words are the same, similar, or dissimilar in their base, stemmed, and normalized forms.

POS Features which specifies what the POS tag is for the each of the two words in the pair and if they words share a POS tag. There was special consideration given to the words : “a”, “be”, “do”, “have”, “get”, “go”, “follow” and “please”.

Topical Features that relate to the strength of the word’s association to the topic. This is able to provide an estimate of the word’s relative importance within the sentence.

The authors also constructed a Twitter Paraphrase Corpus using a novel and efficient crowdsourcing method. For more information on the corpus see the Dataset Section. When learning and testing their model on the corpus the authors noted excellent performance of their model. With a F1 score of .724, precision of .722, and a recall of .726 their model was far ahead of other state-of-the-art models. The authors also combined their feature based classifier method with a latent space model using a technique called product of experts to achieve even higher results than with the MultiP algorithm alone.

Dataset and Annotations:

We reached out to Dr. Xu and she was kind enough to supply us with her Twitter Paraphrase Corpus. The corpus consists of 18,762 sentences pairs and 19,946 unique sentences in total. Sentence pairs are organized based off of Twitter’s own trending topics algorithm, and overall over 500 different trending topics are covered by the set. The training set consists of 13,063

sentences pairs, the test set has 972 sentence pairs. She also has an additional set called "dev" that has 4727 pairs, however we will not be using it. All of the sentence pairs are paired together based on them sharing a topic. These topics are not necessarily synonymous with the trending topic, and range from Africa, to Netflix, to Aaron Rodgers. In addition each sentence was tokenized and run through a POS tagger.

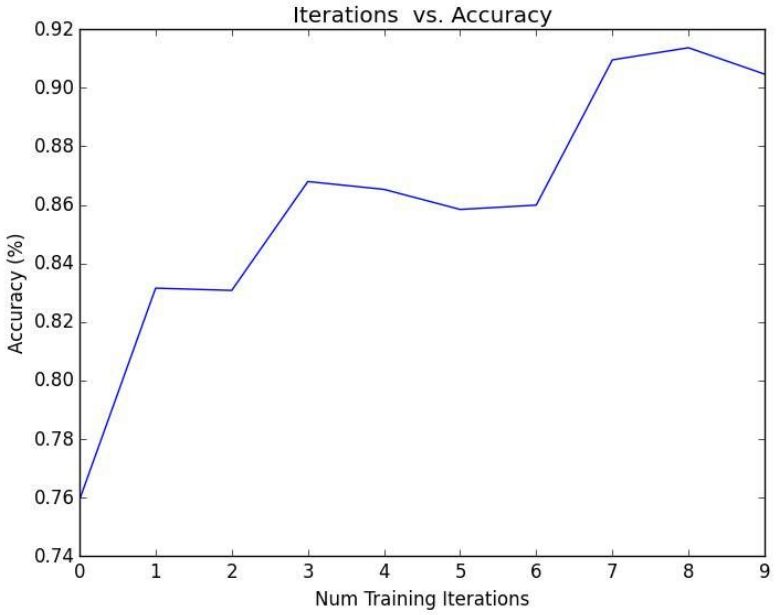
Each sentence pair was annotated by five different crowd-sourced workers. Amongst the crowd sourced workers, those annotators who consistently misidentified paraphrases as compared to their peers were dropped. The annotators were shown 1 original sentence and then 10 other sentences on the same topic. They were then asked to select any sentences whose meaning was the same to the original sentence. The sentence pairs were also evaluated by expert annotators. Based on these two metrics sentences were assigned a score ranging from zero to five. Any sentence pair with an annotation score of above three was considered a paraphrase. Likewise pairs were not considered a paraphrase if they received a score of zero or one. However, pairs with a score of two were considered borderline. For the purposes of simplicity in our data set we discarded these borderline pairs. Subtracting the borderline from our training set leaves us with 11530 sentence pairs, 34% of which were considered paraphrases, and 66% were not. Likewise our test set goes to 842 pairs with 36% considered paraphrases.

Method

Perceptron approach with "anchor"

Initially, our approach to the problem was very similar to Dr. Xu's "anchor" assumption. To us, this idea of finding the "anchor" between possible paraphrases seemed like a reasonable way to find a consistent pattern that determines whether the two tweets are indeed paraphrases. We explored similar features at the word-level, mostly utilizing string and part-of-speech features such as whether the words were the same after normalization (both stemming and lemmatization) as well as if they had the same part of speech. We utilized a simple binary perceptron to learn

the features that helped to determine anchor phrases. The results from this initial exploration were surprisingly good. Using a weighted perceptron, after ten iterations our model arrived at about 90% accuracy. The graph for these iteration is shown below. Although these results seem splendid, they are actually quite misleading, since due to the simplicity of our model, amongst other things, it would be highly implausible for us to outperform the state-of-the-art. There are several possible explanations for our model's unreasonable success. Firstly, all of the tweet pairs are aligned by topic, which means that they share at least *n* words, where *n* is the length of the topic, e.g. "germans" or "8 mile". Obviously, since they are identical they will skew both of our features towards paraphrase. In Dr. Xu's model the topic words were disregarded. Additionally, by removing the borderline tweet pairs we heighten the percentage of paraphrases in the corpus, further skewing the model. Dr. Xu noted that she and her colleagues also discarded the borderline tweet pairs, because these pairs would not have proven helpful for their model.



Bag-of-words approach:

Our second approach made a very different assumption from our first. Instead of assuming that there was one "anchor" pair between the two tweets that would help classify them as paraphrases, we made the assumption that we could use bags-of-words to extract features from

each pair of tweet to classify them. Right from the beginning, this approach was easier to implement and more conceptually intuitive. The main reason for this was that it granted us the ability to extract sentence-level features instead of only word-level features. Additionally, we could still combine features at the word level for classification, which was very helpful.

Feature Overview

We used a number of features with various classifiers to determine which would be most effective. We predicted that the features independently would not be very informative to determine similarity between pairs of tweets, but that the combination of the features would be effective. To use with our classifiers, we created a feature vector for each tweet pair. The features we explored are shown below.

Surface similarity: We used cosine similarity of the surface text to determine the similarity between two tweets. However, we were sure to remove the topic words from the tweet beforehand. We made the assumption that the similarity between surface texts of two tweets would be informative as a feature.

POS similarity: A similar feature to sentence similarity, we used the cosine similarity of the parts of speech of the two tweets. Because of the lexical divergence between tweets, we made the assumption that a higher cosine similarity for the parts of speech between two texts would be informative as a feature. If two tweets have the same or similar constructions but different words, they will still share many of the same parts of speech.

Stem similarity between verbs and nouns: Since the sentence similarity only considered words in their unnormalized form, two words with the same stem but different inflections would not increase the cosine similarity between two tweets. In this feature, we assumed that we would only need to get the stems of the verbs and nouns because other types of words typically aren't

inflected. Again, we used the cosine similarity of these words as a score. For the stemming of verbs and nouns we used the Lancaster Stemmer from NLTK.

Lemmatized noun overlap: For each pair of tweets, we extracted nouns from each tweet separately and lemmatized them. Then we calculated the overlap of these lemmas to use as a feature. The motivation for this was similar to the stem similarity, but we made the assumption that separating the lemmatized nouns from verbs would be more informative as a feature. For lemmatization of nouns and verbs we used WordNetLemmatizer from NLTK.

Lemmatized verb overlap: As explained above, we also extracted verbs from each tweet and lemmatized them, calculating the overlap. We would only expect this feature and the lemmatized noun overlap feature to be informative if the tweets exhibit more formal language, because that is the intention of the WordNetLemmatizer.

Difference in length: A very simple feature, we used the difference of length between tweets. We didn't expect this to be very informative, but we thought there might be some correlation between number of words and sentence similarity.

Syntactic similarity: This feature was the only one that did not adhere to the bag-of-words approach. Since the dataset has the parts-of-speech tags for each tweet, we assumed that we could use this syntactic information to help detect paraphrases. We started by creating a Context Free Grammar and attempting to parse the parts-of-speech tags into trees using a Chart Parser from NLTK. If successful in parsing the trees for both tweets in a pair, we compared their structure. This was a troublesome feature because the structure of tweets often strays from the typical structure of English, not to mention all of the interjections and various symbols. Since the parser with our grammar was unable to parse many of these troublesome structures, we had to abandon this feature. Likewise, we had originally planned on incorporating WordNet to identify synonyms in our sentences pairs. However as we became more acquainted with our dataset we

decided against using WordNet due to the prevalence of slang, abbreviations, and misspellings in Twitter's use of language.

Classifiers

After choosing the features for our model, we then wanted to see the effectiveness of a number of classifiers with different features. We used classifier implementations from sci-kit learn. We decided to use a supervised learning approach, experimenting with four classifier models with our features: Gaussian Naive Bayes, a Support Vector Machine, logistic regression, and a random forest. Each classifier is defined in terms of our project below.

Gaussian Naive Bayes: We used the sci-kit learn implementation of a Gaussian Naive Bayes classifier. This required the assumption that the feature values would be distributed according to a Gaussian distribution.

Support Vector Machine: The sci-kit learn implementation of a Support Vector Machine that we used was the Linear Support Vector Classifier. It supports both sparse and dense input, which was useful to us. It maps the feature vectors into a high dimensional vector space and calculates the dot product of the two vectors. There are also non-linear systems, but we chose to only use the linear model.

Logistic Regression: We also used the Logistic Regression (also called MaxEnt) classifier from sci-kit learn. This classifier simply measures the relationship between the categorical dependent variable (paraphrase or not) and the values in the feature vectors by estimating the probability using a logistic function.

Random Forest: The sci-kit learn random forest implementation we used was called Random Forest Classifier. This classifier creates decision tree classifiers and fits them on various values of the feature vectors, averaging them to improve predictive accuracy and reduce over-fitting.

Experiments

Baseline

For our baseline, we decided to use a supervised logistic regression classifier (mentioned in the previous section) with only the cosine similarity of the tweet pairs as a feature. Shown in Table 1 are the results of the baseline, with the f-score, precision, and recall.

Table 1

	F-score	Precision	Recall
BaselineLogReg	.543	.429	.479

Results

First, we trained the four classifiers using all of the features (except syntactic similarity) using the train portion of the dataset, and then evaluated the predictions of the classifiers. The results are shown below in Table 2, broken down into f-score, precision, and recall. (Note that this logistic regression model uses all of the features).

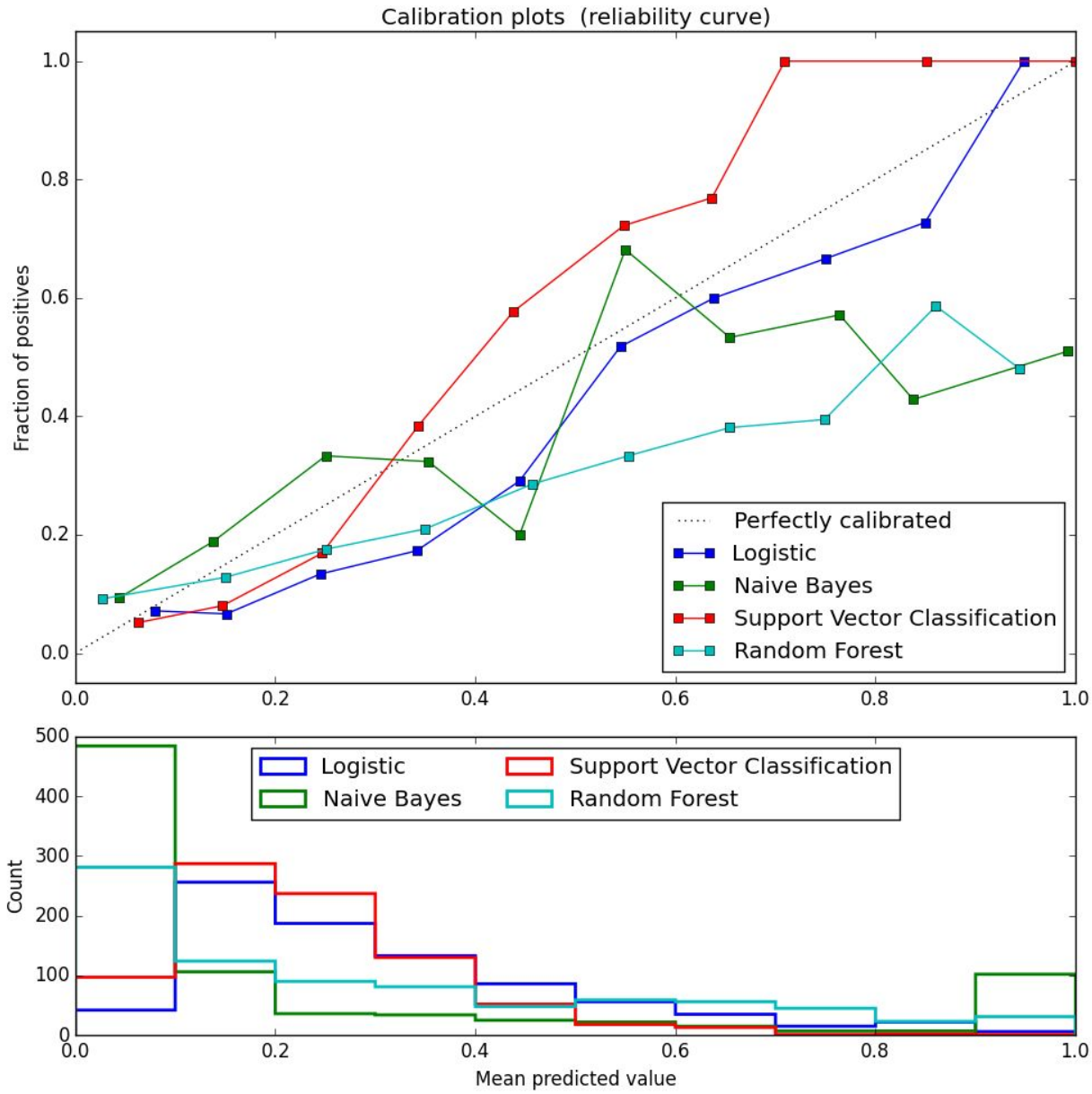
Table 2

Classifier	F-score	Precision	Recall
SVM	.505	.603	.434
GNB	.500	.536	.469
Rand. Forest	.421	.486	.451
Log.Regression	.612	.469	.531

Classifier Calibration:

In addition to measuring the f-score, precision, and recall, we also measured the calibration of the classifiers. This measure informed us of the most well-calibrated classifier (Logistic

Regression), because it resembles the “perfectly calibrated” classifier the most. This means that it has the highest confidence throughout most predictions. A graph of the calibration of the classifiers is shown below.



Feature Effectiveness:

For measuring the effectiveness of each feature, we used the Logistic Regression classifier and measured the features. We measured the features separately, except in all cases we included the sentence similarity as a feature. We did this because some of the features didn't seem to be effective at all independently of sentence similarity. Shown below in Table 3 are the results of this experiment.

Table 3

Feature	F-Score	Precision	Recall
Sent Similarity	.479	.543	.429
POS Similarity	.519	.591	.463
Stem Similarity	.479	.543	.429
Length Difference	.479	.522	.423
Lemma N Overlap	.518	.605	.451
Lemma V Overlap	.472	.517	.434

Discussion:

Overall, our system performed decently when compared to similar models, especially considering that we didn't end up using an external database such as Wordnet, like we originally intended. Interestingly, the effectiveness of features was seemingly negligible when we trained them independently, but the combination of the features had a larger impact on the results. The most important features besides the cosine similarity of the sentence, were the overlap of lemmatized nouns and the cosine similarity of the parts of speech. We are unsure of why the overlap of the lemmatized verbs did not have as much of an impact as nouns, especially because there is more variation between types of inflections on verbs (-ing, -ed, etc.) than nouns

(pluralization). The impact of the similarity of parts of speech as a feature is logical because it is a very similar measure to the similarity of the sentence.

Future Work:

The work we have presented here represents a good initial attempt into detecting paraphrases amongst tweets. However there are still many areas of our models that could be strengthened or sophisticated, as well as other directions we might attack the issue from. For example we would be very interested adapting our syntactic similarity feature. For this to be a reality, we would have to develop a grammar that is specific to structures that we observe in Twitter or similar platforms, which could prove difficult because language on the internet is often without rigid structure or patterns. At this point in the project we have only evaluated our features against each other using the logistic regression classifier; we might find that the importance of features changes based on using different classifiers. Additionally, we would like to explore other approaches besides a bag-of-words approach, such as a neural network that can generate a paraphrase from a tweet. Not only could we compare the result of this to the original sentence and a candidate sentence, such a project would be informative to understanding what the crucial properties of paraphrases in tweets are.

References:

Bjerva, J., Bos, J., Van der Goot, R., & Nissim, M. (2014). The Meaning Factory: Formal Semantics for Recognizing Textual Entailment and Determining Semantic Similarity. *Proceedings of the 8th Internitonal Workshop on Semantic Evaluation,* 642-646.

Li, Yuhua et al. "Sentence similarity based on semantic nets and corpus statistics." *Knowledge and Data Engineering, IEEE Transactions on* 18.8 (2006): 1138-1150.

Li, Yuhua, Zuhair Bandar, and David McLean. "An approach for measuring semantic similarity between words using multiple information sources." *Knowledge and Data Engineering, IEEE Transactions on* 15.4 (2003): 871-882.

Mitchell, Jeff, and Mirella Lapata. "Vector-based Models of Semantic Composition." *ACL* 15 Jun. 2008: 236-244.

Satyapanich, Taneeya, Hang Gao, and Tim Finin. "Ebiquity: Paraphrase and Semantic Similarity in Twitter using Skipgram}." *Proc. 9th Int. Workshop on Semantic Evaluation* 2014: 109-122.

Xu, W., Ritter, A., Callison-Burch, C., Dolan, W., & Ji, Y. (2014). Extracting Lexically Divergent Paraphrases from Twitter. *Association for Computational Linguistics, 2*, 435-448.