Identifying Fake News Using Stance Detection

Morgan Cassels
Department of Computer Science
University of Victoria
casselsm@uvic.ca

Jared Foulds
Department of Computer Science
University of Victoria
jaredfoulds@uvic.ca

Abstract

The relationship between a news article and its headline is an important factor in fake news detection (Fake News Challenge 2017). Stance detection seeks to classify the nature of this relationship. The four major classes defined by the Fake News Challenge are "Agree", "Disagree", "Discuss", and "Unrelated" (2017). This task was broken into a series of binary classification tasks. Various methods for achieving these binary classifications were investigated, including n-gram overlap, word vector distance, polarity scoring, and subjectivity scoring. Very accurate results were achieved in determining whether the news article headline and its body were "Unrelated" or not. Further classifying the other classes proved to be a much more difficult task and suboptimal results were achieved.

1 Introduction

Fake news as defined by the New York Times is any "…made-up story with an intention to deceive, often geared toward getting clicks" for financial gain (Tavernise et al., 2016). According to the Telegraph writer Carson (2017), there are five different types of fake news:

- Intentionally deceptive
- Jokes taken at face value
- Large-scale hoaxes
- Slanted reporting of real facts
- Stories where the 'truth' is contentious

Although fake news may seem like a trivial annoyance, it is having a distressing effect on readers. A study done by Barthel et al. (2016) found that 64% of American adults found fake news "…caused a great deal of confusion about the basic facts of current events." Readers should not have to discern whether a news article is fact or fiction. Nor should they ever be confused about the basic facts on events unfolding around the world.

The Fake News Challenge is a programming competition with the goal of leveraging machine learning and natural language processing to assist human fact checkers with fake news detection (Fake News Challenge, 2017). The first task in the Fake News Challenge, "FNC-1", is to classify the stance of a news article headline and its associated body. This task is intended to serve as a "...building block in an AI-assisted fact-checking pipeline" (Fake News Challenge, 2017). Stance detection is essentially trying to uncover what the author of the news article is saying about a given topic in the headline. The Fake News Challenge (2017) defines four different classifications for this relationship:

1) Agrees: The body text agrees with the headline.
2) Disagrees: The body text disagrees with the headline.
3) Discusses: The body text discuss the same topic as the headline, but does not take a position
4) Unrelated: The body text discusses a different topic than the headline

Examples of each category classification can be found on the following page:

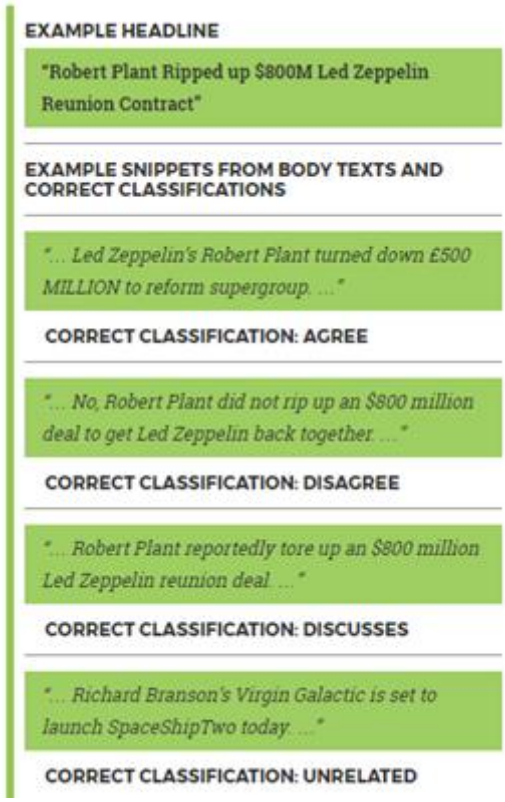

Figure 1: Example Stance Classifications (Fake News Challenge, 2017)

2 Data

For this project, we only used the training data provided by the Fake News Challenge (2017), which consisted of two files called "train_stances.csv" and "train_bodies.csv". The file "train_stances.csv" contained a grid listing out each news article headline, the body ID, and the stance classification, which could be "agrees", "disagrees", "discusses" or "unrelated". The body ID in "train_stances.csv" is used to reference the actual body text held in "train_bodies.csv".

There were only 1684 body texts contained in "train_bodies.csv" while there were 49,973 headlines listed in "train_stances.csv". This was because for a single body text, they created multiple headlines with different stance classifications. The purpose of doing so was to increase the amount of useable training data. We also discovered that the the training data was not evenly distributed among the four stance classifications. Out of the 49,973 headlines, 3678 were labelled as "Agree", 840 were labelled as "Disagree" 8909 were labelled as "Discuss", and 36,545 were labelled as "Unrelated". Due to this staggering overrepresentation of headlines labelled as "Unrelated", we made sure to stratify our data when sampling for both our training and test data. Lastly, we noticed that a few entries in "train_bodies.csv" had unrecognizable characters or contained foreign languages. Since the number of problematic entries was very small, we decided ignore them as there were 49,973 headlines so their overall effect on our project would be minute.

3 Literature Review

Four scholarly sources related to our problem were reviewed in order to gain insight on how others approached solving issues that were similar to our project.

3.1 Fact Checking: Task Definition and Dataset Construction

Fact Checking: Task Definition and Dataset Construction (Vlachos and Riedel 2014) introduces the problem of fact checking as a natural language processing task and its baseline approaches. In fact checking, the assignment of truth values to claims is done as an ordinal classification since many claims are disputed and fall somewhere on a spectrum of truth (Vlachos and Riedel 2014). In this problem definition, five levels of truth were used: True, MostlyTrue, HalfTrue, MostlyFalse, and False. Not only must the content of the claim be used to make this classification, but the speaker and the time at which the claim was made must be considered (Vlachos and Riedel 2014). Further complicating matters is the high level of inter- annotator disagreement; not only may two different reliable sources disagree, but two different interpretations of the same source can lead to different conclusions (Vlachos and Riedel 2014).

In the construction of this dataset, only claims which had a reasonably objective truth value and could be checked against an online source were considered for fact checking. A dataset of statements which had previously been fact checked by PolitiFact and Channel 4's fact checking blog were screened and 106 suitable statements were collected.

This study offered three different approaches to fact checking. The first approach was to use supervised machine learning and learn from the statements themselves, while considering their truth

value. This was dismissed because the dataset of statements did not provide a large enough knowledge base to check those statements against. The second approach used a database of previously fact checked statements, which all had their own truth values. When given a new statement, it would find the semantically closest statement in the database and assign the new statement the same truth value. However, this method would clearly not be extendible to novel claims, because no statement in the database would be semantically close enough to produce a meaningful result (Vlachos and Riedel 2014).

The last proposed approach involves making an assumption that all true claims or paraphrases of all true claims already exist together as a text in some location such as Wikipedia. Then, if a match or a paraphrase of a given claim is not found within that text, that claim could be labelled false. However, this technique would not allow for interpretations based on the combinations of statements given in the text (Vlachos and Riedel 2014). For example, if Wikipedia contained the statements "Paul McCartney was born in 1942" and "John Lennon was born in 1940", the claim "John Lennon was older than Paul McCartney" may be labelled as false.

The main concern that Vlachos and Riedel have with the three approaches is that the user of a fact checking system likely wants to have some understanding of the logic behind the truth value assigned to claims. They suggest a system similar to that described in An Extensible Framework for Verification of Numerical Claims (Thorne and Vlachos 2017) wherein the system determines the appropriate question or entity-predicate pair. This system looks up the answer or value and compares the value from the source to the value in the claim in order to reach a verdict.

3.2 Stance Detection with Bidirectional Conditional Encoding

Stance Detection with Bidirectional Conditional Encoding (Augenstein et al. 2016) discusses a technique of determining the stance of a piece of text in relation to a given topic. In this study, the researchers extracted text from user's "tweets" from the social media platform, Twitter. The goal was to predict the stance of a tweet given some target, such as an issue or public figure, which may not necessarily be named in the tweet itself. For example, given a tweet reading "Trump will make America Great Again", a target of "Donald Trump" would produce a positive stance. However if the target were "Hillary Clinton", a negative stance would result. Even though Clinton is not mentioned in the the tweet, this stance detection system would learn that a positive stance towards Trump is equivalent to a negative stance towards Clinton.

As explained by Augenstein et al., the process of conditional encoding begins with building a fixed-length word vector representation of the target. Then a word vector representation of the tweet is built, initialized at the target vector. The stance of the tweet with respect to the target is found through a projection which uses these two vectors. Conditional encoding was found to produce better results than techniques in which tweets were not encoded relative to a given target.

This study found the best results by using bidirectional conditional encoding, a model which accounts for the surrounding context of each word being encoded. This means that it uses two vectors for both the target and the tweet; one which is found by processing the words from right to left and one which is found by processing the words from left to right. Then, one of the tweet vectors is initialized from the right-to-left target vector and the other tweet vector from the left-to-right target vector. The study found that bidirectional conditional encoding outperformed single-directional conditional encoding.

3.3 Emergent: A Novel Data-Set for Stance Classification

Emergent: A Novel Data-Set for Stance Classification (Ferreira and Vlachos 2016) presents a method of checking the veracity of a claim through analysis of a database of news articles.

Headlines summarizing the stance of each article had already been created by journalists for an earlier study; the goal of this project was to categorize headlines as for, against, or observing with respect to a given claim. A regression model using 10-fold cross-validation on all training data was built with two main sets of features. The first set of features were determined only by the headlines. These were primarily useful for distinguishing the observing case, since they often contained words from a closed class of 'hedging' words, such as "Reportedly" or "Allegedly". The second set of features was determined by a combination of the headline and the

claim. Semantic comparisons were done using a paraphrase database and direct comparison of word stems.

Four features came directly from the headline. The first feature was the bag of words representation and the second was a binary feature representing whether or not the headline ends with a question. A combination of the third and fourth features represented the minimum distance between the sentence root and 'refuting' or 'observing' words, such as "Never" or "Alleges". These minimum distances were found to be very useful in distinguishing between the observing and for stances.

The Kuhn-Munkres algorithm was used to extract a feature representing the relationship between the claim and the headline. As explained by Ferreira and Vlachos, Kuhn-Munkres creates a score for each headline-claim pair by pairing each of their constituent words together and scoring those pairs based on the semantic distance between the words. Then, it finds the word-pairing scheme which maximizes the total score of each headline-claim pair. The maximum score, which is normalized to the length of the shorter of the headline or claim, is used as the value of a feature. Another feature came from extracting the subject-verb-object combinations from both the headline and the claim. The two subjects, two verbs, and two objects were compared semantically and given one of five labels based on their semantic relationship. The combination of these three labels formed a feature value. Additionally, vector representations for both the claim and the headline were produced through multiplication of the vectors of each of their respective words. The cosine similarity between these two sentence vectors were used as another feature.

It was found that instances of the for stance had, on average, a higher lexical overlap between the claim and the headline than instances of the against case. This allowed the use of overlap thresholds, beyond which a headline-claim pair would be automatically labeled for, against, or observing; improving the accuracy of the results.

3.4 An Extensible Framework for Verification of Numerical Claims

An Extensible Framework for Verification of Numerical Claims (Thorne and Vlachos 2017) presents a method for fact checking simple numerical claims. This process involves extracting the piece of information in question and the proposed value for this piece of information from a given claim. Then, it retrieves the correct value for this piece of information from a knowledge base and compares the proposed value to the correct value. The key step in this system is determining which entry in the knowledge base should be used to support or refute a given claim (Thorne and Vlachos 2017).

The knowledge base used in this method had been translated into entity-predicate-value sets. For example, the fact "Confederation occurred in 1867" would be converted into the following entity-predicate-value set: confederation-year-1867. Thorne and Vlacho's method is as follows: When evaluating a claim, the first step is to find the entities mentioned in the claim and retrieve all entries in the knowledge base which contain that entity. The predicates of the entries are then compared to the text of the claim in order to determine whether an entry is relevant. If the wordforms in the text of a claim can describe the predicate in a knowledge base entry, as determined by a threshold in the difference between the feature vectors of the claim and the predicate, then that entry is classified as relevant.

On average, too many entries were determined to be relevant for each claim. However, this did not lead to false positives in the test cases. Thorne and Vlachos explained that this is because the values associated with the extraneous entries were so dissimilar to the value associated with the claim that they had a percentage error too great to allow a false positive. Since those extraneous entries were discarded due to their dissimilarity to the claim, the

Extensible Framework for Verification of proper corresponding entry existed was left, which could accurately verify or falsify the claim.

4 Methods

Since the "Unrelated" class was overrepresented in the data set, we undersampled from each class to build training and testing sets in which each class was equally represented. The "Disagree" class was the least represented in the data, with only 840 samples, so we built a data set of 3,360 samples, with 840 samples from each of the four classes. We used 2,688 samples for training (672 from each

class) and 672 for testing (168 from each class), for an approximate 80/20 split.

The baseline technique we used was based on simple unigram overlap between the news article headline text and the body text. First, we performed stop word removal on both the headline and body text and then calculated the percentage of unigrams in the headline text which appeared at least once in the body text. We used scikit-learn's Support Vector Machine (SVM) (Pedregosa et al., 2011) to classify our data based on unigram overlap, which produced the following results:

	Actually Agree	Actually Disagree	Actually Discuss	Actually Unrelated
Classified as Agree	0	0	0	0
Classified as Disagree	53	62	47	10
Classified as Discuss	108	94	109	0
Classified as Unrelated	7	12	12	158
Total Accuracy:	49%			

Table 1: Unigram Overlap Results into Four Classes

This technique was very successful in identifying "Unrelated" samples, but was largely unsuccessful in identifying the other three classes. As a result, we decided to split the task into a series of binary classification tasks, beginning with separating the "Unrelated" class from the other three classes. We referred to these remaining three classes collectively as the "Related" class. The first binary classification task involved utilizing unigram overlap with stop word removal to classify news articles as either "Related" or "Unrelated", but before doing so, we had to stratify our data.

Since there were more "Unrelated" than "Related" samples in the data set, we undersampled and created a balanced data set with 26,854 samples. We used 21,448 news articles for training (10,742 from each class) and 5,370 for testing (2,685 from each class), which gave us an approximate 80/20 split. The results of this binary classification are shown below:

	Actually Related	Actually Unrelated
Classified as Related	2517	134
Classified as Unrelated	168	2551
Total Accuracy:	94.4%	

Table 2: Unigram Overlap Results Between Related and Unrelated Classes

This 94.4% accuracy indicated to us that unigram overlap with stop word removal was a promising method for separating out the "Unrelated" samples. We tested to see if these results could be improved upon by using bigrams instead of unigrams. We used the ngram function from the TextBlob python library (Loria 2013) and achieved the following results:

	Actually Related	Actually Unrelated
Classified as Related	1236	49
Classified as Unrelated	1449	2636
Total Accuracy:	72.1%	

Table 3: Bigram Overlap Results Between Related and Unrelated Classes

Since bigram overlap produced a lower accuracy than unigram overlap, we decided to keep unigram overlap with stop word removal as the method for separating "Unrelated" and "Related" samples.

We explored two alternative designs for breaking down the remaining three classifications into a series of binary classification tasks. Our first idea was as follows:

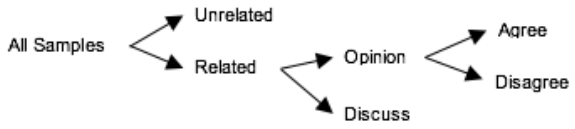

Figure 1: Binary Classification Design for Pipeline 1

After our class presentation, an alternative design for our binary classification pipeline was suggested to us by our classmate Rahnuma Nishat (2017):

Figure 2: Binary Classification Design for Pipeline 2

We explored many different techniques in an attempt to create meaningful splits between "Opinion" and "Discuss", "Agree" and "Disagree", "Not_Disagree" and "Disagree", as well as "Agree" and "Discuss". This included using word vector representations, subjectivity scoring, polarity scoring, part-of-speech (POS) tagging, and named entity recognition (NER).

4.1 Word Vectors

We used the 50d GloVe model vectors (Pennington, Socher, and Manning 2014) to build vector representations of the headline text and the body text. Since the body text of the news articles were generally long and the time complexity of our word vector generation and comparison was expensive, we tried a few different methods of producing a shortened representation of the body text.

We tried using the first three sentences, first sentence of each paragraph, and first two and last two sentences of the article. All three methods produced very similar results but those produced by using only the first three sentences were slightly better on average. We then tried extracting the keywords from the headline text and the shortened body text by removing stop words and the corresponding GloVE model vector was found for each remaining keyword. The headline and shortened body text were then each represented as a single vector by averaging all of their keyword vectors. The cosine distance between the headline and body vectors was found using the scipy spatial library (Jones, Oliphant, and Peterson et al. 2001). We tried using this cosine distance between the headline and body texts for several different tasks to see where it could be useful. We first attempted using this approach as a feature in the SVM to classify between all four classes to see what kind of baseline it would produce. This approach unfortunately only had 33.2% accuracy and caused most test samples to be classified as "Discuss". We then tried using it for the "Agree" and "Disagree" binary classification, which would be the last stage in Pipeline Design 1. This produced meaningless results partly because antonyms actually have very similar word vectors to each other because they often appear in similar contexts.

After our project presentation, we received a suggestion from classmate Jeremy Krenbrink (2017). He suggested we try doing word vector comparisons using only verbs since they usually are more significant in determining the stance of a given sentence. For example, in determining whether the sentences "Clinton lost the election" and "Clinton won the election" agree, only the verbs are relevant. Including the other keywords, "Clinton" and "election", just adds noise since they both appear in both sentences.

Verbs were identified using Natural Language Toolkit's (NLTK) POS tagger (Bird, Klein, and Loper 2009). We constructed vector representations of the headline and body texts in the same way as before. We used only the keywords which were tagged as verbs. Any form of the verb "to be" was excluded. Using the cosine distance between the headline and the body texts, we trained the SVM to classify between "Disagree" and "Not_Disagree", which would be the second stage in Pipeline Design 2. The results were not meaningfully different than random chance.

4.2 Subjectivity Scoring

From our perspective, the difference between a "Discuss" and an "Agree" classification is primarily based on the article's objectivity. The example given by The Fake News Challenge (2017) is as follows: Given a headline of "Robert Plant Ripped up $800M Led Zeppelin Reunion Contract", paired with a body text reading "...Robert Plant reportedly tore up an $800 million Led Zeppelin reunion

deal...” would be classified as “Discuss”. If you paired that same headline with the following body text “...Led Zeppelin’s Robert Plant turned down £500 MILLION to reform supergroup…” then you would get an “Agree” classification. This example suggests that the use of the word “reportedly” makes the sentence objective and therefore determines its classification as “Discuss”.

We explored two different subjectivity scorers, one from NLTK (Bird, Klein, and Loper 2009) and one from Textblob (Loria 2013). NLTK classifies sentences as either “subj” or “obj”, while Textblob produces a score between 0 and 1, where 0 is very objective and 1 is very subjective. We used undersampling to create the largest possible stratified data set where “Agree” and “Discuss were equally represented, and an approximate 80/20 split between training and testing samples. The Textblob subjectivity score of the first three sentences of the article body was used as a feature in the SVM to perform a binary classification between “Discuss” and “Agree” (the last stage of Pipeline Design 2). Our accuracy achieved with this technique was 56%. We then tried the same task using NLTK’s subjectivity classifier. These results weren’t any better, with just 52% accuracy. NLTK’s subjectivity classifier classified the overwhelming majority of samples as objective and the SVM classified most of the test samples as discuss.

We also tried using the subjectivity score in the second stage of Pipeline 1 to separate the “Discuss” class from the “Opinion” class. The reasoning behind this idea was that the first three sentences in both “Agree” and “Disagree” articles would be more subjective than the first three sentences in “Discuss” articles. We undersampled from the “Agree” class to create the largest possible “Opinion” data set in which “Agree” and “Disagree” were equally represented. Then, we undersampled from “Discuss” to create the largest possible data set where “Opinion” and “Discuss” were equally represented. An approximate 80/20 split between training and testing samples was used and with both TextBlob’s and NLTK’s subjectivity scorers our results were equivalent to random chance.

4.3 Polarity Scoring

Polarity scoring is another method we tried and both NLTK (Bird, Klein, and Loper 2009) and Textblob (Loria 2013) had libraries for it. Polarity is a measure of the overall sentiment of a given piece of text and is measured on a scale from -1 to 1. A piece of text with a positive sentiment would be given a score close to 1 while a piece of text with a very negative mood would be given a score close to -1. We hypothesized that a headline and body pair that agrees would have similar polarity scores, while a pair that disagrees would have different polarity scores.

NLTK’s VADER polarity scoring tool produces four different scores which indicate the given text’s levels of negativity, positivity, neutrality, and a “compound” score. We used only the compound score where -1 is very negative and 1 is very positive. Separating “Agree” and “Disagree” using the NLTK polarity score was accurate 56.3% of the time. The TextBlob polarity tool produces a single score between -1 and 1 where -1 is very negative and 1 is very positive. Using Textblob, our results were equivalent to random chance.

After we explained our poor results from polarity comparison in our project presentation, classmate Kyle Hildebrandt (2017) suggested a modification. Instead of just comparing the polarity of the headline with the first few sentences of the article, he suggested that we identify the entities mentioned in the headline. Then, compare the polarity of the headline with the polarities of only the sentences in the article body which mention those entities.

To first identify the entities mentioned in the headline, we tried using NLTK’s named entity annotator, called “ne_chunk.” Unfortunately, we discovered that NLTK’s recognition of named entities relies heavily on capitalization. This posed problems because usually every keyword in a headline is capitalized. The capitalizations confused the NER system and lead to superfluous and meaningless annotations. When the entire headline was instead set to lowercase, the NER system recognized almost no entities at all.

We decided to try using noun phrases instead of entities, which we extracted from the headlines using TextBlob’s noun phrase extraction tool. Unfortunately, TextBlob’s noun phrase extraction tool often produced inaccurate annotations. For example, it often labelled verb phrases as noun phrases. Another problem was that the noun phrases identified in the headline would often not appear at all in the article body since the phrases could be several words long. For example, for the headline “America’s 45th President Donald J. Trump Tweets

Too Much", the noun phrase identified would be "America's 45th President Donald J. Trump". In the body of the article, "America's 45th President Donald J. Trump" might be referred to as just "Trump", meaning that there would be no matches and so nowhere to do polarity comparison.

We decided to instead use the NLTK POS tagger on the headline to just identify nouns. Then, we identified instances of these same nouns in the article body. We collected the sentences in which they appeared and used NLTK's compound polarity score to find the average polarity of these sentences and compared this average to the polarity of the headline. We used this difference as a feature in an SVM to classify between "Disagree" and "Not_Disagree" in the second stage of Pipeline 2. Unfortunately, these results were still roughly equivalent to random chance.

4.4 Pipeline Implementation

Based on our experimentation, we determined that the only binary classification for which we had a very useful method was the separation into the "Unrelated" and "Related" classes. In order to produce a final result, we decided to implement Pipeline 2. The pipeline first separated out the "Unrelated" class using unigram overlap with stop word removal. Then, it separated out the "Disagree" class from the "Not_Disagree" class using the difference in average polarity around mentions of key nouns. Finally, it separated the "Agree" class from the "Discuss" class using TextBlob's subjectivity score. Undersampling was used to stratify the data set so that the "Unrelated", "Disagree", "Agree", and "Discuss" classes were all equally represented in the training and testing data sets.

5 Results

The final results of the Pipeline 2 were as follows:

	Actually Agree	Actually Disagree	Actually Discuss	Actually Unrelated
Classified as Agree	29	34	24	6
Classified as Disagree	104	111	103	9
Classified as Discuss	31	20	39	12
Classified as Unrelated	4	3	2	141
Total Accuracy:	47.6%			

Table 1: Pipeline Results

The accuracies at each stage were as follows:

Stage	Accuracy
1. "Unrelated" / "Related"	94.6%
2. "Disagree" / "Not_Disagree"	44.8%
3. "Agree" / "Discuss"	34.9%

Table 2: Pipeline Accuracy by Stage

Due to the design of the pipeline, errors were compounded between each stage, reducing the overall accuracy. Any "Unrelated" samples which mistakenly made it to the second stage of the pipeline further reduced the accuracy of the "Disagree" and "Not_Disagree" classifier. This was because no matter how the "Unrelated" samples were classified at the second stage, their classification was always wrong since it should have been "Unrelated". Low accuracy in the second stage exacerbated this same issue in the third stage, leading to an even lower accuracy.

In trying to determine which method to use for each stage of our pipeline, we attempted many methods and variations on those methods. As can be seen from our results, while determining the relatedness of a headline and article body is a

straightforward task, classifying related texts as either “Disagree”, “Agree”, or “Discuss” is very difficult. The major difficulty is that article/headline pairs classified as either “Disagree”, “Agree”, or “Discuss” already have a lot in common with each other.

Separating “Agree”, “Disagree”, and “Discuss” using word vectors poses problems since their vector representations are all similar due to their similar content. For example, the vector representations of the phrases “ceasefire deal” and “ongoing conflict” would likely be close, because they have a similar distribution in the English language, appearing in roughly the same contexts. However, for the purposes of determining the nature of the relationship between the headline and the body text, the differences between those two phrases might be vital.

Not only are the contents of “Agree”, “Disagree”, and “Discuss” headlines similar, but their tone could be as well. If all headlines are an unbiased reporting of an event, it could be very difficult to separate them based on their subjectivity or other tone or discourse-marker related features. For example, given an article about Michael Phelps winning a gold medal, the “Agree” headline “Michael Phelps Wins Gold”, the “Discuss” headline “Michael Phelps Wins Gold, Reports Say”, and the “Disagree” headline “Michael Phelps Wins Silver” all have a very similar tone. So all three headlines would have high polarity and low subjectivity scores.

Our results from using polarity scores to separate out “Disagree” instances were disappointing, likely because training on the difference between the polarity scores of the headline and body of an article relies on an assumption which is not always right. Namely, that if an article’s body text and its headline disagree, one of them will have a negative polarity and one will have a positive polarity. This would be true in a headline/body pair such as “Triumphant Michael Phelps Wins Gold” and “A devastated Michael Phelps lost all his events yesterday...”, which have a high and low polarity respectively. However, if the headline was “Michael Phelps Wins Gold” and the body was “Michael Phelps won a silver medal yesterday…”, the polarity scores of the headline and the body would likely be almost identical even though they should be classified as “Disagree”.

6 Conclusion

Over the course of this project we explored many different Python libraries and natural language processing techniques in order to test possible approaches that we could use in our pipeline. As a result, we found that our Python coding abilities improved tremendously. We also gained experience with several libraries that could be useful in future projects. We learned how to implement concepts that we learned in class by incorporating them into our project, such as n-gram comparison, word vectors, part-of-speech tagging, and sentiment analysis.

If we had more time to complete this project, we would definitely conduct more research on methods that could be used to more effectively classify news articles as “Agree”, “Disagree”, or “Discuss”. We tried several approaches to differentiate these classifications to no avail. In addition, we would also try implementing another word vector representation model such as “Word2Vec” and retry the unsuccessful classification approaches we tried initially. Another issue we faced in our project was simply its runtime. Therefore, another step we could take in project would involve finding a cloud computing service to reduce the amount of time wasted waiting for our program to execute or simply reviewing our code to find ways we could reduce its time complexity.

At the conclusion of the project, we reflected on the societal implications of the project and considered how it could be used to disadvantage various social groups. We envision that our project could possibly be used in online discussion forums such as “Reddit” where users can express their opinions towards a given topic. Using stance detection, user’s comments could easily be filtered based on their stance towards a given topic and if a particular stance is undesirable by the forum owner, it would not be displayed. This could also be extended to corporation websites that contain comments or reviews by their users. The company could use our program to filter out any negative remarks made by users so that they would not be displayed to anyone else using the website even if they are valid. In order to prevent the abuse of our software, we could simply patent it so we could control who uses our software. However, we are not confident in this approach as the concept of stance

detection is not an original one and can easily be replicated.

Throughout the project, we had a difficult time defining what makes an article be classified as "Discuss". The Fake News Challenge defined this class as when the "...body text discuss the same topic as the headline, but does not take a position" (2017). However, the issue arises when news articles just simply report current events and do not have or require an opinion or position towards the topic. For example, say you have a news article titled, "Fire in Vancouver" and a body text that just simply reports the the fire that just happened. Would that be classified as "Agree" because the headline follows the body of the article or would it be classified as "Discuss" because the article body just reports or discusses what happened? This was very unclear and we did not see a meaningful difference between "Agree" and "Discuss" or even the purpose of trying to determine if an article discusses a headline. If we were to do this project again, we would probably just remove the "Discuss" class entirely unless its purpose was made clear by the Fake News Challenge.

Overall, this project was very successful in augmenting our understanding with regards to natural language processing. Although we did not achieve good results in separating "Agree", "Disagree", and "Discuss", we conducted a thorough investigation of a variety of methods and eliminated many methods which initially seemed promising.

Work Distribution

For the project proposal, Jared wrote the following sections: "Introduction", "Baseline Technique", "Baseline Technique Improvements", and "Timeline". Morgan wrote the "Data" section as well as the "Literature Review" section. For the programming portion of the project, Morgan wrote a large majority of the program with exception of the methods involving unigram overlap and word vector cosine similarity comparison. For the project presentation, we divided the work evenly with each person writing the slides that they were going to present. Lastly for this final report, Morgan wrote the "Methods" and "Results" section while Jared wrote the remaining sections.

References

Augenstein, I., Rocktäschel, T., Vlachos, A., & Bontcheva, K. (2016, September 26). Stance Detection with Bidirectional Conditional Encoding. Retrieved from https://arxiv.org /abs/1606.05464

Barthel, M., Mitchell, A., & Holcomb, J. (2016, December 15). Many Americans Believe Fake News Is Sowing Confusion. Retrieved from http://www.journalism.org/2016/12/15/many-americans-believe-fake-news-is-sowing-confusion/

Bird, S., Loper, R. and Klein, E. (2009), Natural Language Processing with Python. O'Reilly Media Inc. Retrieved from http://victoria.lviv.ua/html/fl5/NaturalLanguage ProcessingWithPython.pdf.

Carson, J. (2017, February 08). What is fake news? Its origins and how it grew in 2016. Retrieved from http://www.telegraph.co.uk/ technology/0/fake-news-origins-grew-2016/

Ferreira, W., & Vlachos, A. (n.d.). Emergent: a novel data-set for stance classification. Retrieved from http://eprints.whiterose.ac.uk/97 416/1/emergent2016.pdf

Hildebrandt, K. (2017, June 26) Connex Feedback.

Jones, E., Oliphant, E., Peterson, P., et al. SciPy: Open Source Scientific Tools for Python, 2001-, http://www.scipy.org/ [Online; accessed 2017-05-30]

Krenbrink, J. (2017, June 26) Connex Feedback.

Loria, S. TextBlob: Simplified Text Processing, 2013-, http://www.textblob.readthedocs.io/ [Online; accessed 2017-05-30]

Nishat, R. (2017, June 26) Connex Feedback.

Pedregos, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830. Retrieved from http://jmlr.csail.mit.edu/papers/v12/pedregosa11 a.html.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global Vectors for Word Representation. Empirical Methods in Natural Language Processing, 1532-1543. Retrieved from https://nlp.stanford.edu/pubs/glove.pdf.

Tavernise, S. (2016, December 06). As Fake News Spreads Lies, More Readers Shrug at the Truth. from https://www.nytimes.com/2016/12/06/us/fake-news-partisan-republican-democrat.html

Thorne, J., & Vlachos, A. (n.d.). An Extensible Framework for Verification of Numerical Claims. Retrieved from http://aclweb.org/anthology/E17-3010

Vlachos, A., & Riedel, S. (n.d.). Fact Checking: Task definition and dataset construction. Retrieved from http://www.aclweb.org/anthology/W14-2508