Using Lexical Features and Wordnet for Headline-Body Stance Detection

Nathaniel Blake, Nick Miller, and Matthew O'Brien
University of Pittsburgh
{nsb23, nam99, mro25}@pitt.edu

Abstract

We present the details of our system for stance detection of news headline–body pairs as a tool to assist manual fact checking or as a basis for an automated truth-labeling system. Training from a human-labeled sample of 40,000 headline–body pairs, our system calculates feature vectors for each pair and then trains an SVM classifier on these vector–stance pairs. Using this trained classifier we achieve 88.8% accuracy on the development set and 87.8% accuracy on the test set in the task of assigning a stance *unrelated*, *discusses*, *agrees*, or *disagrees* to each headline–body pair. We close by proposing directions for future work, including ways to improve the system's ability to distinguish among bodies which *agree*, *disagree*, and *discuss*.

1 Introduction

The task of detecting low-quality or misleading news content in a fully or partly automated way has gained significant attention recently, as the limits of manual fact-checking become increasingly apparent with the ever-larger deluge of information of dubious quality released daily across the Web. Our work follows the partial automation approach to this quality assessment task, specifically addressing the task of labeling the relationship between a news item's body text to its headline as *unrelated*, *discusses*, *agrees*, or *disagrees*. Using a tool such as ours, a manual editor could quickly dismiss articles with misleading headlines and spend care on fact-checking in those articles which make coherent, falsifiable claims.

Our efforts follow the precedent set by (**?**), who showed the viability of classifying known-related headline–body pairs, labeling each body as *for*, *against*, or *observing* the claim made by the corresponding headline. In a similar task, (**?**) automatically labeled tweets as holding a *positive*, *negative*, or *neutral* stance toward an unspecified target, such that their system had first to identify the target and then the stance.

Last, we drew inspiration from the related task of question answering, treating the headline as a sort of query and the body text as a potentially related document. To this end, we leveraged features developed in (**?**), (**?**), and (**?**). Each of these works uses semantic features to ascertain the relationship of the document body to the query—an insight we attempted to capture in our work.

2 Approach

Our system makes use of two Python libraries: NLTK and scikit-learn. NLTK is a general natural language library which offers a variety of general functions such as tokenization (for both sentences and words) and part-of-speech tagging, as well as access to corpora and other lexical resources such as WordNet. (**?**)

The scikit-learn library (`sklearn`) contains tools for data analysis. In particular, our system leverages scikit-learn's implementation of a support vector machine (SVM) to build a classifier using the features extracted from each headline-body pair. (**?**) Sklearn was chosen based off the results of (**?**) who's system used sklearn's LogisticRegression classifier. The LogisticRegression from sklearn is a linear classifier, despite its name, so the SVM classifier was initially chosen for its ability to handle non-linear combinations of features. Despite this initial choice, it was found that a linear kernel performed almost as well as a non-linear kernel and achieved much better runtime performance.

2.1 System Design

The system performs training in two steps: feature extraction and classifier fitting. This is because the SVM classifier implemented by sklearn trains off of a full set of feature vectors and corresponding labels, so it requires the feature vectors and label for each training example before it fit the classifier. The classifier used was kept on default settings, with the exception of switching the kernel mode to linear from radial basis function, which was the default. The switch to linear was observed to have negligible affect on performance, while resulting in a more stable run time.

2.2 Feature Selection

Features were chosen based on fulfilling one of two criteria: related/unrelated disambiguation and agree/disagree/discusses disambiguation. Some features were chosen from features used by related systems, others were chosen by observed patterns in the test set. Each feature tested was implemented alongside the current established base feature set and the impact on accuracy and performance examined to determine whether the feature would be used. If a feature performed well, it was incorporated into the base feature set.

3 Features

The following are the features present in the final model:

- **Bag of Words Comparison**
  For this feature, the headline and article body are both converted into bags of words, and each set has all stop words removed (stop words as defined by the NLTK stop word set (**?**)). The feature is the number of words in the headline word set which occur in the body. This was tested both normalized to the number of words in the headline set (putting the feature value between 0 and 1) and as a raw count, and there appeared to be no major difference between the two. This feature was chosen for it's success rate in other systems (including (**?**)), and as expected it significantly increased the system's accuracy in distinguishing between unrelated and related headline/body pairs, though it did not affect intra-related disambiguation.

- **Cosine Similarity**
  For this feature, the headline and article body were both stripped of stop words and punctuation. The system then loops through each sentence in the body and compares to the headline sentence provided. The comparison is done by vectorizing each sentence and then performing cosine similarity between the sentences. The first feature returned is the average cosine similarity of each sentence in the body compared to the headline. The second feature returned is the max cosine similarity out of each sentence in the body compared to the headline. The third feature returned is once again getting the max similarity from each body compared to the headline, but the similarity measurement is a Sequence-Matcher provided by difflib that checks for the total number of sequence matches between the headline and the sentence from the body provided.

4 Results

4.1 Specifications

For training, a corpora with 40,000 lines with the pattern of entry, body id, and stance was used. Also for training, all the different stances (agrees, disagrees, discusses, unrelated) are used. The test.csv testing data, testing gold-set of answers, and the python accuracy tool were provided by Yuhuan Jiang. The system used to evaluate the run-time was a 2.9 GHz Intel Core i5 processor. Table 1 references the results by Accuracy, Test Score, and Time taken to complete.

4.2 Results

To evaluate the the accuracy and score, the use of several different combinations of features was needed to evaluate if the improved accuracy was worth the added time. The first feature added was the Bag-of-Words feature, which set the standard accuracy to .815 and a test score of 1761.0 with a run-time to model the training data at about 4 minutes. The uniqueness of headlines meant that features that may not seem like they would increase accuracy actually may, so this lead to adding the second feature which was the length of the headline. This added no run-time but increased the accuracy to .835 with a test score of 1907.25.

The second set of features (Table 1: +SC) that were added were the max cosine similarity of headline to body-sentence, the average cosine similarity of headline to body, and the max head-

Table 1: Results

	BOW	+HL	+SC	+NSB
Accuracy	.815	.835	.870	.878
Test Score	1761.0	1907.25	2100.25	2149.25
Time	∼4 Min.	∼4 Min.	∼15 Min.	∼21 Min.

line to body-sentence ratio using a python library sequence-matcher. The run-time for this set of features took 14 to 15 minutes, and with the run-time added to the previous set of features it took about 18 minutes. The accuracy with these features applied to the training data was .870 with a test score of 2100.25. The additional run-time was worth the .035 increase in accuracy and the 193 point test score increase.

The third set of features that were added are the NSB features. There were many different NSB features, but the features selected [NATHANIEL ENTER***] were the features that improved the model. The run-time for this set of features took 5 to 6 minutes and with the previous run-times added on took about 20 to 21 minutes. The accuracy with these features applied to the training data increased the accuracy to .878 and increased the test score to 2149.25.

5 Conclusion

We present a model for detecting headline and body validity by using an SVM classifier. The classifier has specifically chosen features that only improve the accuracy and test score. Our system model is able to achieve a high accuracy in disambiguating between all the stances while being exceptionally well trained in disambiguating the relevant/irrelevant stances. We believe that this approach could be applied with more sophisticated features that could improve the ability for our classifier to disambiguate between agree, disagree, and discuss which would improve our overall system. For future work, we could experiment with a system to increase speed when adding the NSB and cosine similarity feature, as well as looking into a better disambiguation system which could improve our test score.

Acknowledgments

We probably don't have any.

A Supplemental Material

If we need an appendix…