Evaluating Textual Features and Oversampling for Automatic Stance Detection

Robert Pugh
Indiana University
pughrob@iu.edu

Jongwon Lee
Indiana University
leejojo@iu.edu

Abstract

We describe a series of experiments focused on a number of basic textual features and their effectiveness at the task of automatic stance detection. Specifically, we evaluate the impact of bag-of-words (BoW) features, sentiment lexicon features, and syntactic features on the performance of an Support Vector Machine (SVM). Based on our analysis, we find that the words in a tweet offer the most insight into the stance, and that adding features from sentiment lexicons can improve the performance. Additionally, we find that one target showed a performance increase when adding syntactic dependency features. In addition, we identify challenges related to class imbalance, generally small data volume, and data quality.

1 Overview

Stance detection is the task of automatically determining from a given text whether the stance of the author is in favor of, in opposition to, or neutral toward a specific target, which can be a person, a topic, a product, etc. Both Stance detection and sentiment analysis classify the polarity of a given text. However, the two are different in that sentiment analysis will not expect a target. In this project, we present the results of stance detection on Twitter data using supervised approaches.

2 Data

Our dataset comes from the 2016 SemEval Shared task 6 (Mohammad et al., 2016), and includes tweets of five distinct targets with stance labeled. The training data has 2,913 tweets and the test data has 1,956 tweets, both distributed across 5 targets. Each data point is a tweet associated with a specific target and a "stance" label indicating whether the tweet expresses sentiment in favor of (*FAVOR*), in opposition to (*AGAINST*), or is neutral toward (*NONE*) the target. The five targets are: 'Atheism,' 'Climate Change is a Real Concern,' 'Feminist Movement,' 'Hillary Clinton,' and 'Legalization of Abortion'. Consider this

row:
Tweet: "if the fetus isn't inside you, than shut up love semst"
Target: Legalization of Abortion
Stance: FAVOR
Sentiment: Negative

A standard sentiment analyzer will be able to label this tweet to Negative. Also, humans can interpret this tweet to be AGAINST the target, 'Legalization of Abortion'. However, for a machine, it may be challenging to label the tweet with the correct stance.

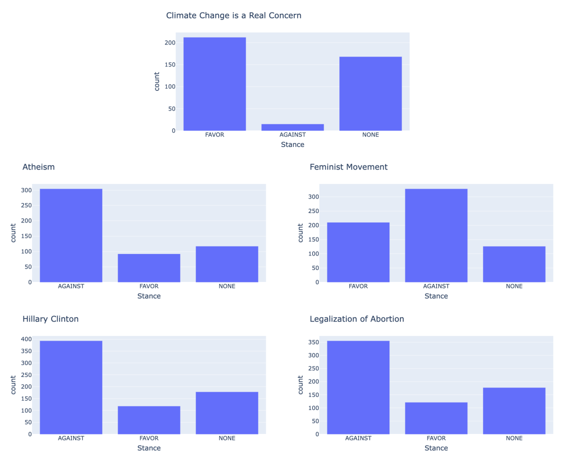

Figure 1: Stance Distribution of 5 Targets

We processed the dataset by lower-casing each tweet, removing URLs, and removing the #SemST hashtag (present in all tweets). We use the to-

		HillaryClntn	Abortion	Atheism	ClimateChng	FeministMvmt
Baseline	w1	0.307	0.523	0.313	0.408	0.553
Individual Features	w1	0.438	**0.579**	0.378	0.418	0.502
	sj	0.246	0.269	0.281	0.281	0.261
	ac	0.246	0.269	0.280	0.321	0.334
	ab	0.245	0.269	0.281	0.281	0.261
	d	0.323	0.372	0.275	0.371	0.338
	sn	0.267	0.34	0.332	0.17	0.338
	w1w2	0.472	0.535	**0.452**	0.414	0.441
	w1w2w3	**0.483**	0.511	0.383	0.414	0.427
Feature Combinations	w1+sj	0.454	0.57	0.387	0.419	0.514
	w1+sj+ac	0.459	0.572	0.398	**0.422**	0.515
	w1+sj+ab	0.439	0.56	0.385	0.396	0.507
	w1+sj+ac+d	0.385	0.559	0.375	0.396	0.548
	w1+sj+ab+d	0.429	0.559	0.361	0.418	**0.57**
	w1+sj+ac+sg	0.361	0.488	0.369	0.265	0.453
	w1+sj+ab+sg	0.355	0.538	0.372	0.269	0.508
	w1+sj+ac+sg+d	0.32	0.444	0.311	0.248	0.541
	w1+sj+ab+sg+d	0.331	0.444	0.333	0.257	0.548

Table 1: Results (macro-averaged f1-scores) for baseline (no hpo), individual features with hpo, and feature combinations with hpo. The best performance per target is listed in bold. *BoW*= *w1*=Bag-of-words; *w1w2*=Unigrams and bigrams; *w1w2w3*=Unigrams, bigrams, and trigrams; *sj*=Subjectivity lexicon features; *ac*=Arguing lexicon features with multiple categories; *ab*=Arguing lexicon combined; *d*=Bag-of-Dependencies; *sn*=Bag-of-Syntactic-Ngrams

kenizer, part-of-speech tagger, and dependency parser in the `en_core_web_md` model provided by the SpaCy package for Python (Honnibal and Montani, 2017). We mostly used the default settings in this model's pipeline, with the exception of the tokenizer: we customized the tokenization behavior to not split hashtags into two tokens.

We found the Stance label distribution for 'Climate Change is a Real Concern' is overly skewed. Also, other targets' stance label distribution are considered skewed (Figure 1). We attempt to address (or at the very least explore) a solution to this challenge via a popular oversampling technique, described in more detail in Section 5.

3 Support-Vector Machines

The Support-vector machine (SVM) is a discriminative machine-learning algorithm that attempts to identify a decision boundary between data from different classes, using a "kernel" to efficiently achieve this goal for classes with feature values that may not be linearly separable. SVMs have been proven very effective for text classification, and have been a staple in the field of natural language processing for decades (Rennie and Rifkin, 2001). The experiments described in the following sections all used the Support Vector Machine (`SVC`) implementation, based on `libsvm`, from the Scikit-learn Python package (Pedregosa et al., 2011).

4 Feature Sets

In this section we provide a brief description of the different feature sets we investigated.

Filtered Bag-of-Words The BoW feature set is simply the set of words that occurs in a given post. These are represented as a large sparse vector, where each dimension corresponds to a unique word. For our experiments, we count each word in a post and use this frequency as the value of the corresponding dimension. To reduce the noisiness of this feature, we only include Nouns, Verbs, Adjectives, and Hashtags.

Subjectivity Lexicon Features Since our dataset is relatively small for a given target, it is unlikely that our model can learn all of the nuances relating word-usage and sentiment by just

	HillaryClinton	LegalizationAbortion	Atheism	ClimateChange	FeministMovement
w1	0.355(-0.083)	0.321(-0.258)	0.366(-0.012)	0.298(-0.12)	0.314(-0.188)
sj	0.331(+0.085)	0.27(+0.001)	0.318(+0.037)	0.268(-0.013)	0.31(+0.049)
ac	0.313(+0.067)	0.311(+0.042)	0.33(+0.05)	0.167(-0.154)	0.327(-0.007)
ab	0.315(+0.07)	0.325(+0.056)	0.353(+0.072)	0.17(-0.111)	0.32(+0.059)
d	0.324(+0.001)	0.348(-0.024)	0.307(+0.032)	0.363(-0.008)	0.338
sn	0.299(+0.032)	0.342(+0.002)	0.335(+0.003)	0.17	0.338
w1w2	**0.506**(+0.034)	0.535	0.452	0.414	0.441
w1w2w3	0.483	0.511	0.383	0.414	0.427
w1+sj	0.399(-0.055)	0.57	0.382(-0.005)	0.409(-0.01)	0.5(-0.014)
w1+sj+ab	0.431(-0.028)	0.572	0.412(+0.014)	0.405(-0.017)	0.517(+0.002)
w1+sj+ac	0.428(-0.011)	0.56	0.387(+0.002)	0.396	0.507
w1+sj+ab+d	0.434(+0.049)	0.556(-0.003)	0.377(+0.002)	**0.45**(+0.054)	0.546(-0.002)
w1+sj+ac+d	0.444(+0.015)	0.547(-0.012)	0.361	0.439(+0.021)	0.57
w1+sj+ab+sg	0.361	0.511(+0.023)	0.369	0.265	0.575(+0.122)
w1+sj+ac+sg	0.355	0.463(-0.075)	0.378(+0.006)	0.269	**0.599**(+0.091)
w1+sj+ab+sg+d	0.315(-0.005)	0.444	0.395(+0.084)	0.248	0.545(+0.004)
w1+sj+ac+sg+d	0.354(+0.023)	0.444	0.42(+0.087)	0.257	0.571(+0.023)

Table 2: Results (macro-averaged f1-scores) for individual features and feature combinations when using Synthetic Minority Oversampling (SMOTE oversampling) of the minority class. The best performance per target in this experiment is listed in bold, and any scores that are the best of all experiments on a given target (including comparisons to Table 1) is underlined. Scores that improved compared when using SMOTE are listed in green, and those that decreased are in red. *w1*=Bag-of-words; *sj*=Subjectivity lexicon features; *ac*=Arguing lexicon features with multiple categories; *ab*=Arguing lexicon combined; *d*=Bag-of-Dependencies; *sn*=Bag-of-Syntactic-Ngrams

observing the patterns in the training set. Thus, we leverage lexical resources containing sentiment information about several words. The Subjectivity Lexicon (Wilson et al., 2005) contains polarity (e.g. Positive or Negative) and strength (Strong or Weak) for over 8,000 words. To combine the BoW feature set with the Subjectivity Lexicon, we concatenate the BoW vector to a 2-dimensional vector (Positive and Negative) with the counts of words for each polarity.

Arguing Lexicon Features We use the Arguing Lexicon (Somasundaran et al., 2007), which contains patterns indicative of arguing in 17 different domains (e.g. *assessments*, *authority*, etc.), as the basis of an additional feature. We used these patterns in two ways:

1. We created a 17-dimension vector (1 per category of arguing), and added the frequency with which any of the patterns corresponding to that category as the value of the respective dimension. Thus if we found two matching patterns for *assessments* and one for *difficulty*, then the vector would have 15 zeros, one dimension with a value of 2, and one dimension with a value of 1. This 17-dimension vector was concatenated with the other relevant feature vectors for our experiments.

2. Since there are not many patterns per category, and it wasn't clear exactly how the arguing categories might be particularly useful for stance detection, we also tried simply calculating the sum of all matching patterns, regardless of category. In this case, we simply concatenated a 1-dimension vector of match frequency to the other relevant feature vectors.

Bag-of-Dependencies Additionally, we explored the value of incorporating syntactic information to our system. The first method for doing this involved generating head-dependent-relation triples from a dependency parse of the tweet. This resulted in a "bag-of-dependencies", the vector for which could be concatenated with other feature vectors.

Bag-of-Syntactic-ngrams We also explore syntactic n-gram features, which are similar to word n-grams except that their neighborhood is defined based on their syntactic relationships instead of the surface-level ordering (Sidorov et al., 2014). These features have been shown to be effective in a number of text classification tasks (Sidorov, 2019). We use the simplest form of syntactic ngrams, namely bigrams and trigrams of words in the dependency tree (as opposed to including the syntactic relation in the ngrams).

Unigrams, Bigrams, and Trigrams In addition to the features listed above, we also tested tf*idf-weighted ngrams. We investigated the combination of unigrams and bigrams, and unigrams, bigrams, and trigrams.

5 Experiments

For all of the experiments described below, we trained 5 separate models (one per target). As an initial baseline, we trained the SVM on bag-of-word count features using the default SVM settings[1]. To start, we trained an SVM on each of the individual features listed above. Next, in order to get a sense of the improvement achieved by combining features, we added features one-by-one in the order they were listed in the assignment[2] With the exception of the initial baseline experiment (BoW features only), we performed 5-fold cross-validation grid search on the training set for each experiment to select the hyper-parameters (we refer to this process, hyper-parameter optimization, as hpo throughout the paper). Specifically, we explored the regularization parameter (`C`), linear vs. rbf kernel, and the `gamma` parameter for the rbf kernel. For evaluation, we follow the original shared task in using the macro-averaged f1-score. This metric is especially useful given skewed label distributions, since a poor model that always predicts the majority class may still have decent accuracy or micro-averaged f1-scores. One difference between our metric and the one reported by the original shared task paper is that, unlike the shared task paper, we include the `NONE` class in our f1-score calculation. This decision was motivatd by the real-world use-case of having to classify stance on social media data. Unless there is an independent model deciding whether each tweet is about a topic, the stance-detection model itself should be able to distinguish between all three classes well.

5.1 SMOTE oversampling

As we noted above, many of the targets in the stance detection data set are highly skewed. We explore whether we can mitigate the impact of this label imbalance by re-running all of our experiments using the SMOTE oversampling technique (Chawla et al., 2002), which generates synthetic examples from the minority class that are very close in the feature space to the original data points. Specifically, we use the SMOTE implementation provided by the Imbalanced Learn Python package (Lemaître et al., 2017), and oversample the less-frequent of either `FAVOR` or `AGAINST`. We chose not to oversample the `NONE` class since these tweets are much noisier and it is harder to model their out-of-dataset distribution (i.e. there are nearly infinite ways in which a tweet could NOT have a stance, thus generating synthetic samples of this class would likely further increase noise).

6 Results

The performance of individual features with hyper-parameter optimization (HPO) and BoW with default settings are listed in the top part Table 1. There are three points of note in these results:

1. HPO has a positive impact on the performance of virtually all targets. The one exception to this is the target ”Feminist Movement”, where it appears that grid-searching derived by hyper-parameters were sub-optimal on the test set (the performance of the BoW features with the default hyper-parameters performs quite well for this target). We suspect that this is due to differences in the data between the training and test set, since optimizing on the training data resulted in worse test-set performance. Nevertheless, there seems to be a clear signal that grid search on the training data is effective.

2. Simple word-level features are by far the most effective. In every target, the best-performing individual feature is either filtered unigrams or a tf*idf-weighted n-gram combination.

[1] C=1.0, kernel=`rbf`, gamma=$(1/(n\_features * training\_variance))$

[2] Ideally we would perform a complete ablation test, trying all possible feature set combinations. However, given time constraints we decided to skip this.

3. Dependency-based features are less-effective than the simple word features, but more effective than the two lexicon features by themselves. In some sense, both of our dependency features carry similar information as the word n-gram features (i.e. they tell our model which words occur in the text). The major difference is that the dependency features offer additional structural information which can by beneficial (as we will see below), but also be more sparse than simple n-grams, making them less effective individually.

The concatenation of these features occasionally resulted in better performance than the individual feature experiments, though in three out of the five targets the best performance was achieved by individual word-level features. For ”Climate Change”, the combination of unigrams, subjectivity lexicon features, and categorical arguing lexicon features (`w1+sj+ac`) achieved the best performance, and for ”Feminist Movement” the best-performance was achieved by unigrams, subjectivity lexicon, arguing lexicon, and bag-of-dependencies features. It appears that the addition of curated linguistic features (the subjectivity and arguing lexicons) can offer important insight that perhaps is not learned by simple word features, particularly, we suspect, given the relatively small number of samples.

Finally, the results of using SMOTE for oversampling the minority class can be seen in Table 3. Generally, scores for each featureset increased or remained the same, with a minority of experiments (23/85) showing decreased performance. More importantly, we saw that using SMOTE achieved the highest overall score for three of the five targets, including Climate Change, the most skewed targets. These results suggest that, in cases with imbalanced labels, data generation and augmentation techniques such as synthetic oversampling should probably be explored in addition to linguistic feature engineering.

7 Future Work

For future work, we are interested in exploring other feature types, namely character-based features and word/sentence embeddings. One proposed advantage of using character features (such as character ngrams) is that they can carry significant information even when there are misspellings or orthographic variation. For example, the word ”feminits” would likely be out-of-vocabulary (OOV) using word n-grams, or at the very least would have few examples. Using character n-grams, on the other hand, we could still extract a feature for ”femin”, which would correctly be associated with words like ”feminine”, ”feminist”, etc. Additionally, character-ngrams allow us to extract potential words from within hashtags, an important word-type on Twitter. A hashtag like ”sheaintmypresident” probably only occurs once or twice (if at all) in any reasonably-sized training set. Using character n-grams, however, we can extract the sequence ”presid” or ”she”, both of which may have important signals learned from other non-hashtag words. Word and sentence embeddings give us the advantage of learning implicit knowledge about words and sequences of words which we would be unable to learn from the training data alone. By training word vectors on millions of Tweets, we may learn an important relationship between words like ”clinton” and, e.g. ”benghazi”, or the similarity of ”bikini” and ”swimsuit” (words that appear in the Feminist Movement target with non-trivial frequency). Thus, learning good representations for words and/or sentences on high volumes of data may grant us access into relationships between the words that we likely would not otherwise obtain via simple BoW features and weights trained only on a training set of less than 1,000 words. Finally, based on our modest success at improving scores using SMOTE, we are interested in exploring additional methods for automatically creating new labeled data, giving our model more examples from which to learn. For instance, we are curious to explore the impact of data augmentation via natural-language transformation such as the methods explored in Dhole et al. (2021). This would involve automatically making small changes to the language used in some examples, small enough so that the label doesn’t change, but large enough to give our model more examples. For instance, replacing adjectives with close synonyms, adding/removing punctuation, and back-translating (automatically translating from English to another language, and then translating it back). Finally, we would like to pursue methods of data-programming to produce more labeled data without the time and effort required to manually label each data point. Frameworks such as Snorkel

(Ratner et al., 2017) make it relatively straightforward to leverage human intuitions to create a large set of weak, noisy labeling functions, and leverage information about their intersections to create probabilistically-labeled datasets.

8 Conclusion

We provided a detailed account of numerous textual features' impact on the automatic classification of stance towards five targets. We found that, individually, word unigram, bigram, and trigram counts were the strongest features across the board. However, combining them with additional lexical features and, on occasion, syntactic features, can improve the system's performance even further. Finally, we demonstrated the effectiveness of synthetic minority oversampling on performance for many feature sets, and in particular on an especially skewed dataset.

References

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. 2002. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357.

Kaustubh D Dhole, Varun Gangal, Sebastian Gehrmann, Aadesh Gupta, Zhenhao Li, Saad Mahamood, Abinaya Mahendiran, Simon Mille, Ashish Srivastava, Samson Tan, et al. 2021. Nl-augmenter: A framework for task-sensitive natural language augmentation. *arXiv preprint arXiv:2112.02721*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Guillaume Lemaître, Fernando Nogueira, and Christos K Aridas. 2017. Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. *The Journal of Machine Learning Research*, 18(1):559–563.

Saif Mohammad, Svetlana Kiritchenko, Parinaz Sobhani, Xiaodan Zhu, and Colin Cherry. 2016. Semeval-2016 task 6: Detecting stance in tweets. In *Proceedings of the 10th international workshop on semantic evaluation (SemEval-2016)*, pages 31–41.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830.

Alexander Ratner, Stephen H Bach, Henry Ehrenberg, Jason Fries, Sen Wu, and Christopher Ré. 2017. Snorkel: Rapid training data creation with weak supervision. In *Proceedings of the VLDB Endowment. International Conference on Very Large Data Bases*, volume 11, page 269. NIH Public Access.

Jason DM Rennie and Ryan Rifkin. 2001. Improving multiclass text classification with the support vector machine.

Grigori Sidorov. 2019. *Syntactic n-grams in computational linguistics*. Springer.

Grigori Sidorov, Francisco Velasquez, Efstathios Stamatatos, Alexander Gelbukh, and Liliana Chanona-Hernández. 2014. Syntactic n-grams as machine learning features for natural language processing. *Expert Systems with Applications*, 41(3):853–860.

Swapna Somasundaran, Josef Ruppenhofer, and Janyce Wiebe. 2007. Detecting arguing and sentiment in meetings. In *Proceedings of the 8th SIGdial Workshop on Discourse and Dialogue*, pages 26–34.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of human language technology conference and conference on empirical methods in natural language processing*, pages 347–354.

9 Appendix

Below, we present the results of running the same experiments described in this paper, using a Random Forest classifier instead of an SVM. For some experiments, the Random Forest performs better than the SVM, but the best-performing SVM experiment is better than the best-performing Random Forest experiment for all 5 targets. Similar to the SVM experiment results, we find that the word features appear to be the most valuable, with only one of the five targets having its best performance outside of the simple n-gram features.

	HillaryClinton	LegalizationAbortion	Atheism	ClimateChange	FeministMovement
w1 (no hpo)	0.394	**0.562**	0.312	0.363	0.408
w1	**0.436**	0.526	0.41	0.377	0.524
sj	0.255	0.268	0.317	0.288	0.254
a1	0.26	0.269	0.279	0.321	0.319
a	0.246	0.269	0.281	0.3	0.261
d	0.302	0.33	0.318	0.338	0.341
sg	0.284	0.322	0.324	**0.419**	0.367
w1w2	0.438	0.477	**0.468**	0.307	0.449
w1w2w3	0.398	0.441	0.451	0.304	0.446
w1+sj	0.432	0.516	0.426	0.371	0.509
w1+sj+a	0.345	0.507	0.429	0.367	0.504
w1+sj+a1	0.416	0.515	0.425	0.371	0.482
w1+sj+a+d	0.321	0.505	0.352	0.364	0.501
w1+sj+a1+d	0.296	0.499	0.417	0.401	0.49
w1+sj+a+sg	0.307	0.52	0.411	0.36	**0.532**
w1+sj+a1+sg	0.291	0.515	0.397	0.357	0.515
w1+sj+a+sg+d	0.303	0.513	0.417	0.352	0.526
w1+sj+a1+sg+d	0.3	0.526	0.395	0.35	0.517

Table 3: Results (macro-averaged f1-scores) ffor individual features and feature combinations when using Random Forest. For all but the first row (the baseline), we selected hyper-parameters via 5-fold cross-validation on the training data. s*w1*=Bag-of-words; *sj*=Subjectivity lexicon features; *ac*=Arguing lexicon features with multiple categories; *ab*=Arguing lexicon combined; *d*=Bag-of-Dependencies; *sn*=Bag-of-Syntactic-Ngrams