Detecting stance using various classification techniques

Harika Putti[1],Manoj Joshi[2]

Abstract
Social media websites are used by millions of users to express their views on everything like brands, products, social issues, political issues etc. With so much information available, companies can utilize it and develop tools and computational models to detect stance towards a particular target. In our paper we discuss the various classification techniques used to detect stance. We compare the techniques using a diverse feature set generated using tweets on targets such as Feminism, Legalization of abortion, Hillary Clinton, Climate change and Atheism. Using the words and characters from the tweets as features, we have classified the data using Timbl and random forest under default and custom settings. The train data contains the actual tweets, the target of those tweets, and stance of that tweet. Using the train and test data we explored different ways of increasing the accuracy of the classifier. The data set used was created as part of the Proceedings of the 9th International workshop on Semantic Evaluation[1]

Keywords
Stance detection — Dependency parsing — Random Forest — Timbl — Sentiment analysis

Computer Science, School of Informatics and Computing, Indiana University, Bloomington, IN, USA

Contents

Introduction

Stance detection aims to identify whether the author or speaker of a piece of text, for a particular issue, is in favor or against of that issue. There is third class in which the stance is neither against nor in favor i.e., neutral. Stance detection is almost like sentiment analysis but there are some major differences - the significant difference is that, in sentiment analysis we determine if the text is positive, negative or neutral while in stance detection we simply determine the favorability towards the target, when the target is not specified.

In recent years, Twitter has emerged as one of the largest platforms for user content facilitating research and development for analyzing that data for computational and analysis purposes. The data from twitter is highly useful for analyzing user opinions of targets. We can build highly efficient natural language systems which take the text and the desired target as input and identifies the authors opinions towards the target. The task at hand is to use the tweets that have already been labeled with a stance as the target as training data and testing various classifiers such as Timbl and Random forest with different sets of features. This is called supervised training where we tested stance detection towards five target - *Atheism, Climate Change is a Real Concern, Feminist Movement, Hillary Clinton and Legalization of Abortion*.

Our task was to built multiple models with different feature sets and test the accuracy of each model. The first model was generated by POS tagging all the words in a tweet from the training set, tokenizing it, then extracting nouns, verbs and adjectives from it and then creating the feature file to be used on the training and test data and running the results on

Timbl. We modified this model by taking all the words instead of just the nouns, verbs and adjectives. The next model, we used the tagged words and the MPQA subjectivity lexicon and created a weighted feature set based on the scores assigned to each of the clue in the lexicon. The next model we created was passing the tagged data through a dependency parser and extracting dependency triplets containing word, head and the label and using those as features. We modified this model by taking the dependency head chains instead of the triplets. Another model was extracting character 7-grams from the tweets and using those as features. Finally we used Random forest instead of Timbl for all these model for classification.

1. Data

The given train data contained tweets pertaining to a particular target (i.e., Atheism, Climate Change is a Real Concern, Feminist Movement, Hillary Clinton and Legalization of Abortion) and the stance label (i.e., for, against or neutral). The train data is used to train our models and test files are then tested and the accuracy of the analysis is found. The output at the end of processing each model is used as our feature file which was finally used in the classifier for testing the accuracy.

1.1 Data Pre-processing

Since tweets can be noisy(language wise) and ambiguous, it could lead to extraction of useless patterns or trends from the data. We have tried to eliminate some of the factors that might effect the stance detection process. Data pre-processing is an important step because working with less attributes reduces the complexity of the problem and the running time. With less attributes, the generalization capability increases. The first step we performed was to separate tweets from each of the targets from the single train file. The features were created for the data-sets using the different methods. The methods that we used to alleviate the chances of error are-

1. *Eliminating user handles:* Since the user handles in a tweet are relatively less relevant, we have eliminated the user handle by using a regular expression in our code while cleaning the data.
2. *Converting words to lowercase:* After separating the tweets, while extracting each word from the tweet, we have converted the words to lowercase in order to avoid repetitions, reduce redundancy and to have a feature set that is more clear. This will help reduce the data set by eliminating repetitions, for example - religion and Religion being different words is not useful. Hence the process helps improve efficiency of the classifier
3. *Eliminating repeated words and creating lexicon:* After all words were extracted, we added them to a dictionary in order to create a lexicon of words for a particular target

The same process was repeated for all the test files. This has largely helped with giving us unbiased and better results from the classifiers.

1.2 Feature Extraction

The main set of features that we have used are word unigrams which we created using the 'Bag Of Words'(BOW) approach which comprises of words that include Nouns, Adjectives and Verbs. Another set of features are the set of all words extracted from all the tweets of the corresponding target. Other sets of features were created using the MPQA subjectivity lexicon that were implemented as part of this task. Features were also drawn using character n-grams of length 7. One set of features were also drawn from taking the dependency triplets that include the head, the word and the label. The sections below explain how each of these set of features were created.

1.2.1 Tokenizing

Tokenization is the process of replacing sensitive data with unique identification symbols that retain all the essential information about the data without compromising its security, it is done by cutting a string into identifiable linguistic units that constitute a piece of language data. From the tweets, tokens were created using a customized tokenizer that tokenizes the tweets accurately. Since, tweets, in most cases, do not follow English grammar, there was a need to build to a tokenizer that analyses the tweets and split them into tokens that made sense. For instance, tokenizers that were part of Python's NLTK package and other available packages do not take emoticons, hash-tags, mentions etc. into account while tokenizing them. NLTK provides an off-the-shelf tokenizer nltk.word_tokenize() which we have used. Files with a single token on each line and an empty line following each tweet was created.

1.2.2 POS tagging and Bag of Words

The process of classifying words into their parts of speech and labeling them accordingly is known as part-of-speech tagging, POS-tagging, or simply tagging. We have used the NLTK default tagger nltk.pos_tag() which uses 'taggers/maxent-treebank-pos-tagger/english.pickle' which is a Maximum Entropy tagger trained on the Penn Treebank corpus. The tagger is based on Maximum Entropy (ME) principles and the model allows arbitrary binary-valued features on the context, so it can use word-specific features to tag correctly. The Maximum Entropy model is an extremely flexible technique for linguistic modelling since it can use a virtually unrestricted and rich feature set in the framework of a probability model. Its a statistical tagger that assigns a word its most likely tag, based on the unigram or bigram frequencies in the training corpus. After successfully tagging the tweets, we have extracted the words of all classes that were tagged as verbs, nouns or adjectives. We then created a distinct list of words for the different targets.

1.2.3 Character n-grams

An n-gram model is generally used for predicting the next item in a sequence in the form of (n-1) order Markov Model. They are useful because of their simplicity and scalability – with larger n, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently. For our task we have used 7-character n-grams extracted from each tweet of the training set and then used those as our features along with the verbs, nouns and adjectives obtained previously for training and testing our data.

1.2.4 Vector Creation

For each of the targets, word vector files were created which included the extracted nouns, adjectives and verbs as individual features and values were given based on their presence in each tweet. The value for that respective column would be 1 if the word/token is present in the tweet, 0 otherwise. This process was carried out for each of the target files for both the train and test data-sets.

1.3 Feature Engineering

Feature engineering is the process of using domain knowledge of the data to create features that make machine learning algorithms work. It helps data scientists reduce data exploration time allowing them to try and error many ideas in short time. On the other hand, it enables non-experts, who are not familiar with data science, to quickly extract value from their data with a little effort, time and cost. We have used MPQA Subjectivity lexicon and dependency parsing using MALTParser for creating features to help us increase the accuracy of the classifiers.

1.3.1 MPQA Subjectivity Lexicon

MPQA or Multi-Perspective Question Answering contains opinion corpora, subjectivity lexicons and various other lexicons that contain perspective information such as level of subjectivity or objectivity or patterns that represent arguing. These are generally very useful in sentiment analysis which is why we have used the Subjectivity lexicon to incorporate the subjectivity of a word to each word in our list of tweets. The lexicon contains clues which are in the following format -

- *type*: A clue that is subjective in most context is considered strongly subjective (strongsubj), and those that may only have certain subjective usages are considered weakly subjective(weaksubj).
- *len*: Denotes the number of words or tokens
- *word*: "Word1" denotes the token or stem of the clue
- *pos*: Denotes the part-of-speech of the corresponding word.
- *stemmed*: if the word is stemmed or not - y or n
- *priorpolarity*: the polarity of the word whether its positive, negative, neutral or both is given by "priorpolarity".

Table 1. Score based on the subjectivity lexicon

Priorpolarity	Subjectivity	
	Strong	Weak
Positive	2	1
Negative	-2	-1
Both	0	0
Neutral	0	0

A scoring process was formulated to use these clue words from the MPQA subjectivity lexicon as described in the Table 1. For each word in the clue file that was present in a tweet, aggregated subjectivity score was created for all the tweets present in each of the targets' train and test data-sets.

1.3.2 Dependency Parsing

A dependency parse connects words according to their relationships. Each vertex of the tree represents a word, child nodes are dependent on the parent, and edges are labeled by the relationship. We have used the MaltParser, which can be used to induce a parsing model from treebank data and to parse new data using an induced model. It is a data driven dependency parser which uses machine learning methods to parse data with the help of relationships in a graph database format. This improves the non-dependency parsing methods since the relationships from a graph database are used and also by use of machine learning methods such as SVM. We used the English malt parser which was trained using the Penn treebank corpus. The input to the parser should be in the CoNLL format [2]. Example for the input format -

1	Hillary	_	NNP	NNP	_
2	is	_	VBZ	VBZ	_
3	our	_	PRP$	PRP$	_
4	best	_	JJS	JJS	_
5	choice	_	NN	NN	_

The output of the parser is also in CoNLL format as well. After passing all of our train file in the input format to the parser, we obtained the parsed file which contained the dependency edge label and the dependency head. Using the combination of the word,label and the head or the dependency triplets as our features, we have created the feature vector for the train data and the test data.

2. Classification

2.1 TIMBL

TiMBl is a Memory-Based Learner(MBL)which is based on the idea that intelligent behavior can be obtained by analogical reasoning, rather than by the application of abstract mental rules as in rule induction and rule-based processing. Timbl has a wide variety of learning algorithms which makes it very useful for finding an algorithm that best improves the accuracy of classification based on the test data provided to it. It is also one of the fastest k-NN implementing tools around, it also is one of the few tools that allows a wide range of formats for the input and output files. Timbl offers a lot of options

for parameter tuning, the different algorithms being one, we can modify the number of nearest neighbors for extrapolation, set threshold values for tree-based IB, possibility of feature-weighting etc. We have provided all separate feature files for training and testing to TiMBL using the default settings firstly and then experimenting with the parameters to obtain the best results.

2.2 Random forest

Random Forest is an ensemble based classifier. In Random forests, multiple random decision trees are created using only a small random subset of the features. Samples of the training dataset are taken with replacement, but the trees are constructed in a way that reduces the correlation between individual classifiers. Specifically, rather than greedily choosing the best split point in the construction of the tree, only a random subset of features are considered for each split. Using Random Forests also removes the necessity of pruning trees which would otherwise be required to prevent over-fitting of the model to the training dataset. For our task we have given each of the different train and test files that were given to Timbl to compare the classifiers based on the same kind of input. The Random Forest classifier was built using scikit learn package in python 3.0 - 'sklearn.ensemble.RandomForestClassifier'. Since this builds n number of trees and averages all the results, it is expected to produce better results than the memory based learning classifier methods. The models were built with the maximum depth of 3 and the number of trees was set to 100.

3. Experiments and Results

3.1 Detecting stance using the TiMBL classifier with various feature sets

For our task we have compared various feature sets of the targets using Timbl. For each target we have modified the parameters individually to see which of the algorithms or what k-value gives the best results for that particular target. This was to check which algorithm best suited a particular genre of tweets and to check if the algorithm matters when classifying a certain type of language.

3.1.1 Using all words and Bag of Words as features

For each of the five targets, all the words in the tweets and the Bag of words extracted from the training data-sets were used to create feature vectors which were then fed to the TIMBL classifier. The default settings in TIMBL were used initially and later modified for each target separately to increase accuracy of each of them. The resulting accuracy is shown in Table 2 and 3. A huge difference in accuracies was observed by changing the algorithms and the k values of the classifier. The largest jump in accuracies was observed for Hillary Clinton for both sets of features. The best results for Hillary Clinton data was observed when a combination of IB1 and the IGTREE algorithms called the TRIBL algorithm was used. The best results for climate change and feminist movement was observed when IGTREE algorithm was used. The most surprising results were found for Atheism tweets which gave highest accuracies irrespective of the type of algorithm or the type of feature used. Upon scrutiny we observed that this might be since most of the training tweets of Atheism we in FAVOR of the target which caused the classifier to have biased results.

Table 2. TiMBL results for all words as features

Target	Accuracy	
	Default Settings	Parameter Tuning
Atheism	71.49%	72.85%
Climate Change	25.88%	57.06%
Hillary Clinton	17.91%	58.45%
Feminist Movement	29.72%	61.89%
Abortion	58.72%	69.40%

Table 3. TiMBL results for words tagged as verbs, nous and adjectives as features

Target	Accuracy	
	Default Settings	Parameter Tuning
Atheism	72.40%	72.40%
Climate Change	29.41%	67.65%
Hillary Clinton	32.10%	58.45%
Feminist Movement	26.57%	56.29%
Abortion	32.38%	64.77%

3.1.2 Using weighted features based on MPQA Subjectivity lexicon

Using the feature vectors created using the MPQA Subjectivity lexicon for classification, we obtained results as shown in table 4. This weighted feature model gave the best results so far with high accuracies even with out any parameter tuning on Timbl. Using the same parameter changes we had done for the bag of words, we tuned the classifier. There was no drastic change in accuracies like we observed in the previous case but almost all the targets had improvements in accuracy. The best results observed were again for Atheism, but this time climate change also had a considerable improvement in accuracy. If we compare the results for Hillary and climate change from the previous section under default setting to the results here under default settings, we see that there is a huge difference. This may be due to the strong language used in tweets related to these topics. The subjectivity lexicon added a considerable weight to the features that were strongly subjective making it easier for the classifier, giving better results.

3.1.3 Using character n-grams of n=7 as features

The feature set created using the 7-gram string of characters was too large due to the number of tweets present in the training set. To overcome this issue we have recorded the number of occurrences of each of the 7-gram and only took those which occur more than 10 times in the training data. This has reduced the size of the feature set by a considerable amount. the results can be observed in table 5. The accuracies were not noticeably different from our previous results. The accuracies

Table 4. TiMBL results for words engineered using MPQA as features

Target	Accuracy	
	Default Settings	Parameter Tuning
Atheism	71.04%	72.85%
Climate Change	70.59%	72.94%
Hillary Clinton	54.39%	58.45%
Feminist Movement	52.45%	59.44%
Abortion	59.43%	61.57%

Table 5. TiMBL results using character 7-grams as features

Target	Accuracy	
	Default Settings	Parameter Tuning
Atheism	52.04%	71.95%
Climate Change	55.88%	67.65%
Hillary Clinton	59.46%	62.16%
Feminist Movement	48.95%	54.90%
Abortion	43.06%	46.26%

on average were better than the bag of words approach and slightly similar to the results from that of the MPQA feature set. Best results again were for the Atheism tweets closely followed by tweets about climate change.

3.1.4 Using dependency triplets as features

Using all the words and their POS tags that we generated using nltk, we modelled an input file in the CoNLL format like shown earlier. The output of the MaltParser was of the similar format and contained the dependency edge labels and the dependency heads. Here's an example of how the output file looks like for the input that we have given -

1	Hillary	_	NNP	NNP	_	5	nsubj	_	_
2	is	_	VBZ	VBZ	_	5	cop	_	_
3	our	_	PRP$	PRP$	_	5	poss	_	_
4	best	_	JJS	JJS	_	5	amod	_	_
5	choice	_	NN	NN	_	0	null	_	_

After extracting the dependency triplets, we created triplets for the test file as well and created feature vectors for train and test by comparing the triplets to the triplets from the tweets and assigning 1 when its present and 0 otherwise. After generating the feature vectors, they were fed to the classifier and the results can be observed in table 6. Like we did in previous cases, we modified the parameters and the results were not very convincing. The best results were for Atheism and legalization of abortion performed the worst.

3.2 Detecting stance using Random forest classifier with various feature sets

We repeated the same experiments by using Random Forest instead of TIMBL for classification. We used the feature files created for each target and ran the Random Forest using maximum depth of each tree as 3 and the number of trees were set to 100 [3]. These were the hyper parameters used. Since the features were all number and there were no categorical features, we did not have to do any more feature engineering

Table 6. TiMBL results using dependency triplets as features

Target	Accuracy	
	Default Settings	Parameter Tuning
Atheism	50.10%	54.43%
Climate Change	40.39%	44.65%
Hillary Clinton	44.76%	45.13%
Feminist Movement	39.62%	41.54%
Abortion	36.53%	46.14%

on it. The results for each feature file and for each target are shown in Table 7.

4. Summary and Conclusions

From this project we learned various methods to create features for classification. We also realized the importance of POS tagging and using them as features to get better results for most of the cases. We also realized the importance of hyper parameters for TIMBL and Random Forest. This was evident from the drastic result change after providing the hyper parameters for TIMBL. Surprisingly, we got almost the same accuracy for all features files when using Random Forest. This is because our feature file was very sparse with many zeros and feature importance turned out to be almost the same across each variable in a feature file.

Acknowledgments

This work was submitted as a term paper in Computational Linguistics Class (Fall-2017) at Indiana University. We would like to extend our thanks and express our sincere gratitude to our Prof.Sandra Kubler for the continuous support and guidance through out the period of this course.

References

[1] Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015). "http://alt.qcri.org/semeval2015/cdrom/index.html"

[2] Malt Parser User guide for creating CoNLL file. "http://www.maltparser.org/userguide.html"

[3] Random Forest guide for classification. "http://scikit-learn.org/stable/modules/generated/RandomForest.htm"

Table 7. Results using Random Forest with various feature sets

Target	Accuracy				
	Allwords	Bag of Words	n-grams	MPQA	Dependency triplets
Atheism	72.73%	72.73%	71.36%	72.73%	71.36%
Climate Change	73.96%	73.37%	73.37%	72.78%	72.78%
Hillary Clinton	64.21%	64.21%	64.91%	64.21%	64.21%
Feminist Movement	58.31%	58.31%	58.31%	58.31%	58.31%
Abortion	67.5%	67.5%	67.5%	67.5%	67.5%

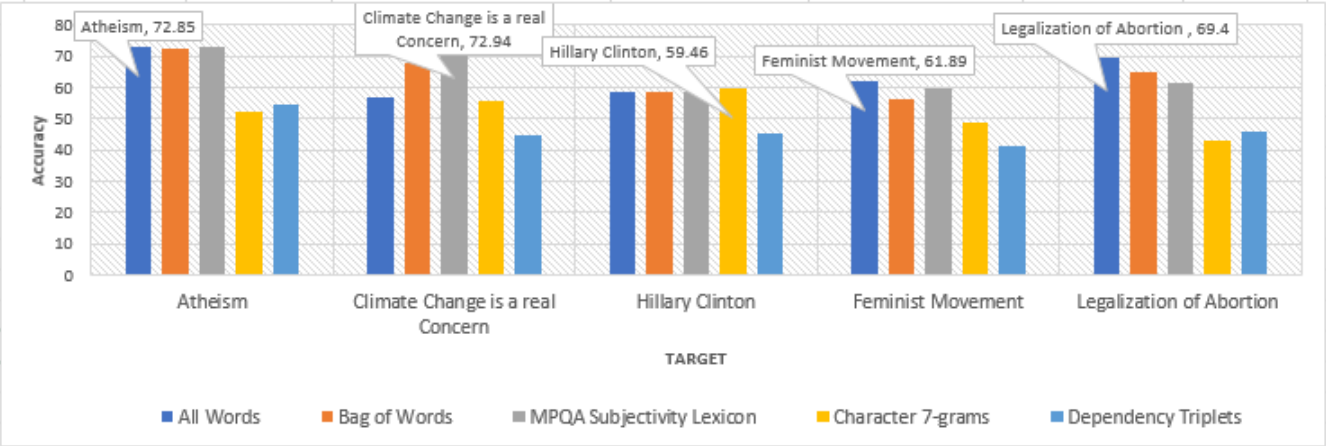

Figure 1. Accuracies across targets using various feature sets on TiMBL

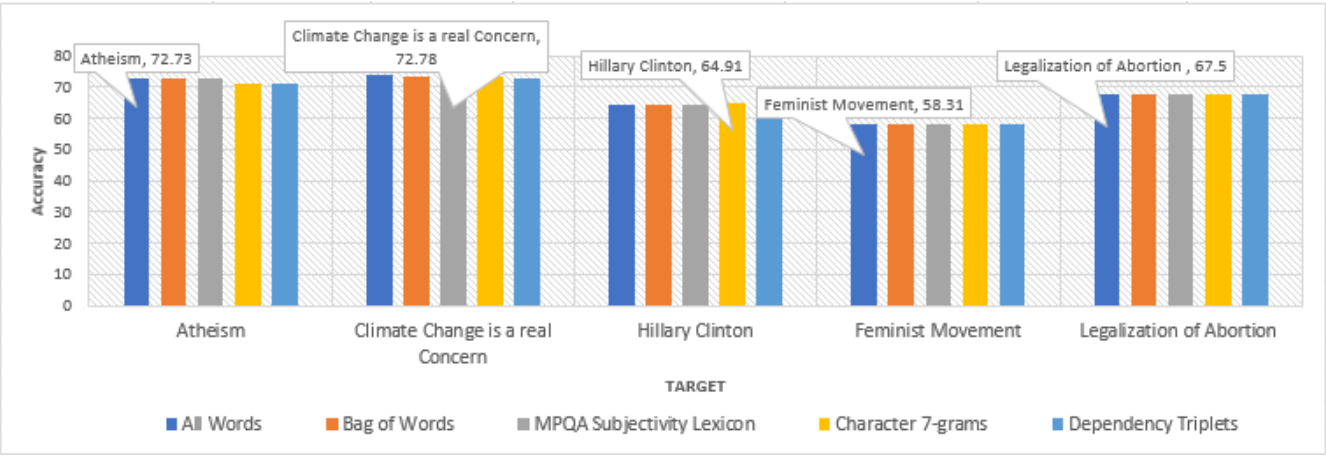

Figure 2. Accuracies across targets using various feature sets using Random Forest