Classification Techniques for Stance Detection

Sai Krishna Karthikeya Dulla, Vinita Chakilam and Nilima Sahoo

***Abstract*—With the increase in awareness of social media platforms and the current scenario of exponentially increasing interactions between people in online forums like Twitter, it has become essential for organizations to understand what people think and identify the stance of their customers towards a target product or service of their own. Here, in this paper, we focus on the process of detecting the stance from tweets. It elaborates on the process of extracting features that contribute significantly to find the stance of the persons comments. Multiple techniques are explored and implemented to predict the users stance towards the target product in every tweet. Given a tweet and a corresponding target about which the tweet talks about, a stance detection system is built that determines whether the tweet is in favor or against or neither. Multiple stance detection models and techniques have been built and the model with the best accuracy for each of the given targets has been discussed and compared with other models.**

***Keywords*—Stance detection - TIMBL Lexicon Parsing N grams Text Classification - Random Forest**

I. Introduction

The goal of stance detection is to determine the opinion of the author for a text towards a given target. The opinion could be in favor of the target or against or neither. The target under study can be a person, place, product, organization, policy, newly formulated laws, etc. It can be viewed as a sub task of opinion mining and it stands next to the sentiment analysis. This has a wide range of applications from sorting out the product reviews to identifying the fake news.

In this paper we are going to discuss about the stance detection of tweets. The training data containing 2914 tweets from twitter were used for training the model. It includes stance towards five different targets: *Atheism*, *Climate Change is a Real Concern*, *Feminist Movement*, *Hillary Clinton*, and *Legalization of Abortion*. For each of these targets, different classification models were built and were tested against the respective targets from the test data. The TiMBL models build using features from Bag of Words, MPQA subjectivity lexicon[3] and character n-grams gave good results for the targets *Atheism* (75.54%) and *Legalization of Abortion* (69.28%). For the target *Climate Change is a Real Concern*, TiMBL model with just Bag of Words feature set gave best accuracy of 73.37%. For the remaining two targets *Feminist Movement* (72.04%), *Hillary Clinton* (65.1%) Random Forest with feature set from Bag of Words, MPQA subjectivity lexicon and character n-grams gave the best accuracy.

II. Data Overview

The provided train and test data files have tweets along with the target information for each tweet and also the Stance as well (which we are considering as class labels). Train file contains a total of 2914 tweets distributed across five targets, namely *Atheism*, *Climate Change is a Real Concern*, *Feminist Movement*, *Hillary Clinton*, and *Legalization of Abortion*. Test file contains 1956 tweets, out of which 707 tweets corresponding to an unseen target $DonaldTrump$, for the current task we are not using these additional tweets in test data. The fig.1 shows the distribution of different class (Stance) labels across all the targets. Except for the target $ClimateChangeisaRealConcern$, rest of all the targets have ”AGAINST” as the majority class.

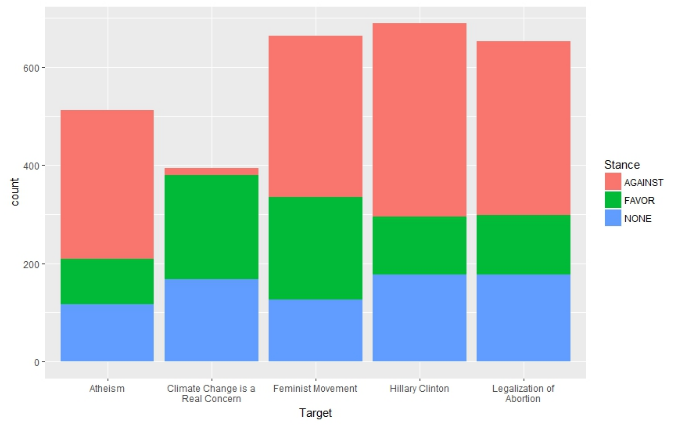


Fig. 1. Frequency of Stance in each target

Data Pre-Processing

The features were created using different approaches, which includes:

- word uni-grams created from Bag of Words that contain the individual words that are nouns, verbs and adjectives.
- features using MPQA subjectivity lexicon
- character 7-grams retrieved from the train tweets

Tokenizing: Tokenizing is a way to split a string into its desired constituent parts which is fundamental to any NLP tasks. This is very tricky for our task, we could not expect a tweet to be in proper structure, because of the language used being a common day to day language and also the character limit constrains the user to tweet in a well defined structure. The tokenizer that we built will be replacing all the numbers, prices, URLs with constant strings because these components can have any number of variants or values which just increases our feature space. However, it does not separate hash-tags ”#” from the word, and similarly the ”@” and the textual part of the mentions were retained as a single entity. After this cleaning part is done the tweet is split into individual words.

POS Tagging: We are using the Parts-of-Speech (POS) tagger from Stanford CoreNLP[1] (a natural language software created by Stanford) in python. The tokens for each tweet that are extracted using the tokenizer are inputted to the POS-tagger which gives a list of tuples containing all the words and their respective parts of speech tags. This POS tagger uses Penn Treebank tagset, so this tagset can be used if we want to filter the words with specific tags.

Bag of Words: The list of POS tagged word set is used to create two separate Bag of Words, one with the words corresponding to all the the POS tags and the second with the words only with tags nouns, verbs or adjectives. The two separate Bag of Word models are used to compare the performance and significance of feature selection in our classification task.

Feature Engineering

Word Vectors: The Bag of Words created previously are used to form the word feature vectors which will be used for the classification task. Each word in the list will be used as a feature, for each tweet a cell in the word vector will contain a 1 if the word is present in the tweet and 0 if not. The size of feature vector for both the models is represented in the table below.

TABLE I. COUNT OF FEATURES USED FOR CREATING THE WORD VECTORS FOR EACH TARGET

Target	All Words	Filtered Words
Atheism	2621	1941
Climate	2438	1840
Feminism	3214	2509
Hillary	3166	2947
Abortion	2947	2259

MPQA Subjectivity Lexicon: MPQA subjectivity lexicon was used in the process of feature engineering. Subjectivity Lexicon are words that typically indicate expression of subjective opinion or judgment, that have been tagged with a prior polarity (i.e. the sentiment they convey in the absence of any context - neutral, positive, negative or both) to identify text passages that are likely to contain subjective expression. This lexicon contains a clue file which was developed by Riloff and Wiebe, 2003 as a part of their work in subjectivity expressions. The clues in the file were provided in the following format:

type=strongsubj len=1 word1=abuse pos1=verb stemmed1=y priorpolarity=negative

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). Word1 denotes the token or stem of the clue, whereas pos1 denotes the part-of-speech of the corresponding word. We created few new features according to the Lexicon file. We gave a value (either 0, 1 or -1), according to the words availability in the Tweet (Bag of Words).
For the words in the clue file which are present in the Tweet, a score has been given to them as per the prior polarity value. (Positive: +1, Negative: -1, Neutral: 0, Both:0). These are assigned as the new features and same thing is repeated for both test and train file. The results are explained in the Experiment and Results section.

Extracted features using N-grams : We used N-grams[4] to create new features in our analysis process. N-grams are a set of co-occurring words in sequence. Here we have used character n-grams to extract features. Character n-grams are nothing but a sequence of characters. These n-grams are then extracted from this sequence and added to the already created bag of words and a model is trained.

We extracted 7-grams for our models. For each Tweets, we created a bag-of-words using 7-gram and these words are appended in the original words we got from the tweet. If the words extracted from n-gram is present in the original tweet, we gave a score 1 and if it is not present to assigned a score of 0 to it. Then the extracted features were modelled using TIMBL and tuned afterwards for better accuracy.

MALT Parser: MALTParser is one of the state-of-the-art parser used for stance detection. MALT is nothing but Models and Algorithms for Language Technology and MALTParser is a system for data-driven dependency parsing, which can be used to induce a parsing model from treebank data and to parse new data using an induced model. For this project, we have made use of the pre-trained English MALTParser which gives us a decent and robust dependency parsing output needed. The input of MALTParser is different from other parsers and it is supposed to be in CONLL format (Conference on Natural Language Learning). The input CONLL format is in the form of the below example of a word of a sentence:

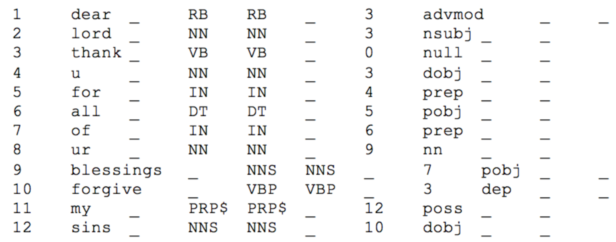
Index Word _ _ PoSTag PoSTag _

The given train and test data sets are in .xlsx format and the input and needs to be converted to CONLL input format. Below are the steps we performed to convert the given train and test data sets:

- Using the Stanford tagger, we extracted the POS tags for all the words of all the tweets of the test and train data. We made sure that the indexing is done separately for each tweet in the POS tagged output file.

- The above POS tag file has then been converted to the tab separated conll format using a python script.

After successful conversion of .xlsx files to .conll formats, we executed the train data file on the pre-trained English 'engmalt' MALTParser using the pre-trained .mco and .jar files from MALT. After running this script, we got the desired parsed test and train data sets with head and labels. We repeated the same procedure for test data as well. Below is the screen shot of the output of a MALTParser trained data.



```
1     dear  _     RB    RB    _     3     advmod      _     _
2     lord  _     NN    NN    _     3     nsubj _     _
3     thank _     VB    VB    _     0     null  _     _
4     u     _     NN    NN    _     3     dobj  _     _
5     for   _     IN    IN    _     4     prep  _     _
6     all   _     DT    DT    _     5     pobj  _     _
7     of    _     IN    IN    _     6     prep  _     _
8     ur    _     NN    NN    _     9     nn    _     _
9     blessings   _     NNS   NNS   _     7     pobj  _     _
10    forgive     _     VBP   VBP   _     3     dep   _     _
11    my    _     PRP$  PRP$  _     12    poss  _     _
12    sins  _     NNS   NNS   _     10    dobj  _     _
```

Fig. 2. Example MALT Parser output

From the above MALTParser conll output files of test and train data, we extracted the dependency triplet of the word, its corresponding head word and label. From the training data output file, we concatenated the dependency triplet using an underscore separator in the form 'word_label_head' using R code. Here, the word will be the word from the tweet itself, the head will be the corresponding head word and label from the MALTParser output files. We repeated the above step for test data as well and got concatenated dependency triplets. Using the concatenated dependency triplets from training data as features, we gave scores to the concatenated dependency triplet words from test data. The scores are given depending on whether the word present in the tweet or not as 1s and 0s. This is the feature vector for each tweet.

III. Experiments and Results

The data sets constructed using different techniques mentioned earlier are used for the classification task. We are performing this task using TiMBL, a Memory Based Learning (MBL) system implemented using the classical k-NN approach for classification. In this section we will see some of the results of different approaches we tried for the classification task.

Bag of Words as Features

All words as features: The train and test files that are created using all the words are features are used as inputs to execute TiMBL for classification, these are the results with default parameters of TiMBL. Accuracies for targets Atheism (66.36%) and Hillary Clinton (49.5%) looks better than for the rest of the targets which were around 30% to 40%

Filtered words as features: The train and test files that are created using words which nouns, verbs and adjectives are used as input to execute TiMBL with the default parameters. We assumed that this should give a better performance than the earlier task because we are using the words (nouns, verbs and adjectives) that would play a major role in deciding the opinion of a text. However we see an increase in accuracy for targets Atheism, Climate and Feminism, there is a slight reduction in accuracy for Hillary and Abortion.

Parameter Tuning: After looking at the classification accuracy with default parameters of TiMBL, we tried to tune the hyper parameters of the model to increase the accuracy. We observed a significant increase in the accuracies for the targets that had very low accuracy with default parameters which shows the importance of parameter tuning for a machine learning algorithm. The algorithm IGTREE gave a good improvement when compared to the baseline values, and for the target Atheism TRIBL with and offset value of 5 gave much better result than the IGTREE.

On the other hand, the algorithm IB1 with tuned nearest neighbor count and feature weight adjustments outperformed all the remaining options. For the target Atheism, the *Cosine* distance metric produced a better result out of all the distance metrics. For the remaining target *Overlap* was the best distance metric.

MPQA Subjectivity Lexicon

The feature extracted using the MQPA lexicon were added to the data set containing Bag of Words vector. This new data was used classify using TIMBL with default settings and IGTREE setting for each target variable. We also have tuned the k-nearest neighbour algorithm with various k-values and optimized k-values are taken. This resulted in a significant improvement in the accuracy compared to the previous runs. The accuracy for Abortion target variable was the best when k=30.(67.5%). For Atheism the optimized parameters are k=15 and w= 2(Weighting factor using Information Gain) and the accuracy for it is 72.73%. For Feminism, the accuracy rate was highest(63.87%) when k=22 and when it was weighed using Information Gain. For Climate the tuned accuracy value is 72.78% with K=34.

TABLE III. Accuracy table for extracted Features using MPQA Subjectivity Lexicon

Target	**Accuracy**
Atheism	72.73%
Climate	72.78%
Feminism	63.87%
Hillary	58.98%
Abortion	67.50%

N-grams Extracted Features

The features extracted from 7-gram procedure is added to the final BOW and then the new model was ran using TiMBL. The results for n-gram was quite astonishing. It increased the accuracy for Atheism, Abortion and Hillary target variables but, it significantly decreased the accuracy for Climate and Hillary target variables. We have tried varius tuning techniques such as varying the K values for k-nearest neighbor algorithm, using different weighing factors. The final tuned parameters are K=7 for Abortion & Atheism and K=3 for Climate, Feminism & Hillary. When experimented with any other algorithms, the accuracy percentage remained same or deprecated.

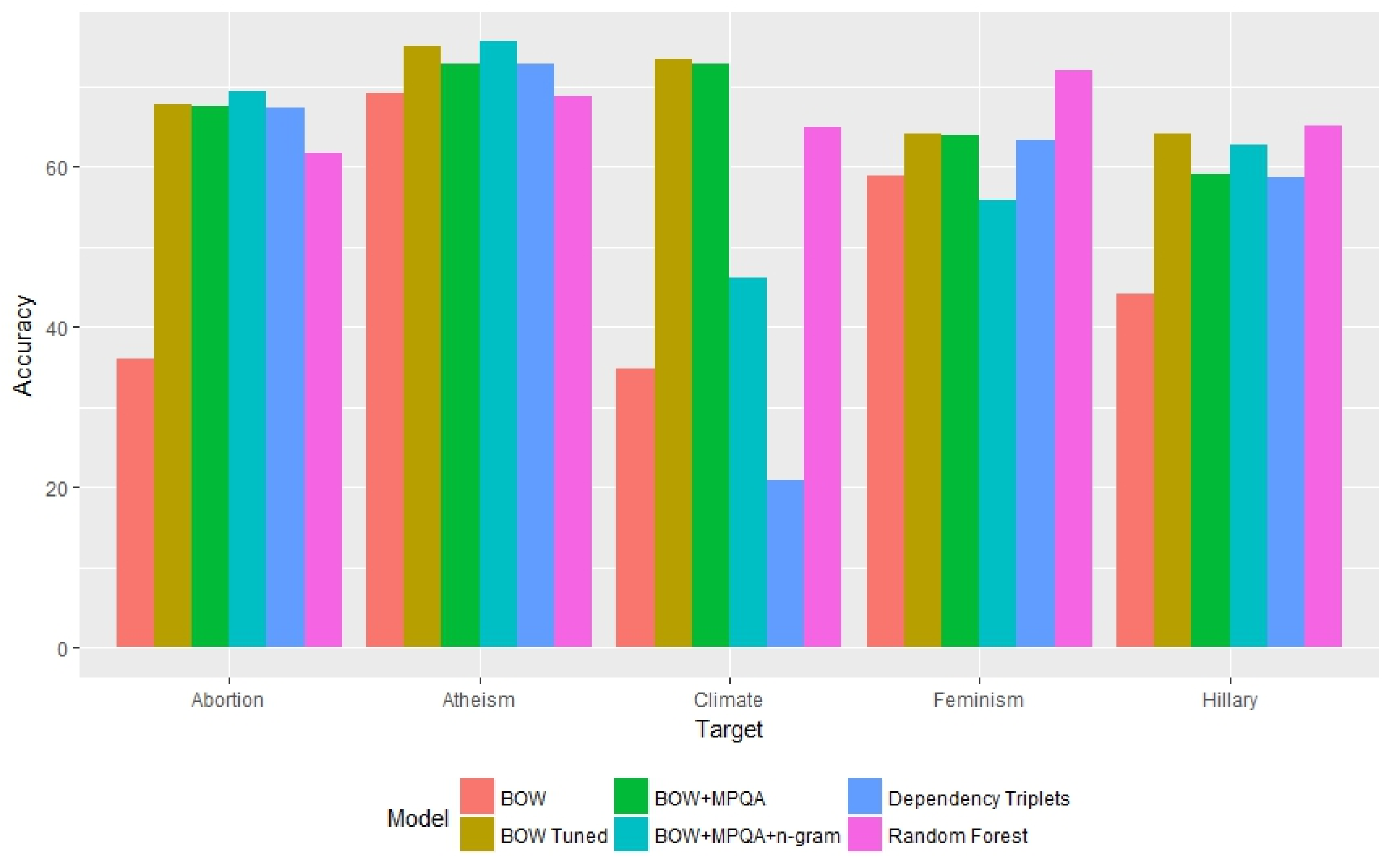


TABLE II. TiMBL Accuracy values for different Bag of Words models

Model	**Atheism**	**Climate**	**Feminism**	**Hillary**	**Abortion**
All words	66.36%	32.54%	32.63%	49.83%	40.35%
Filtered words	69.09%	34.91%	58.94%	44.06%	36.07%
Param Tuning	75.00%	73.37%	64.21%	64.06%	67.85%

TABLE IV. Accuracy table for extracted Features using N-grams

Target	**Accuracy**
Atheism	75.54%
Climate	46.15%
Feminism	55.87%
Hillary	62.71%
Abortion	69.28%

MALT Parser

Using the features of training and testing data, we ran the same for accuracy's on TiMBL using default settings. We then repeated the above procedure for all targets and tabulated the results. Below is the table showing the accuracy's of stance detection for all targets.

Again, instead of dependency triplets, we extracted dependency head chains of lengths 3 and 4 from the above MALT Parser output files of test and train data using a similar R code. Python script also could be used instead of R code. The difference between the two kinds of head and label extraction would be as follows:

First method: Word + its Label + its immediate Head

Second method: Word + Immediate head + Its immediate head + its immediate head After forming dependency head chains in the above fashion, we repeated the above steps from 8 to 10 to calculate accuracy's of different targets. Below are the accuracy's that we have calculated for both the above mentioned methods:

TABLE V. TiMBL Accuracy Values for feature set from MALT Parser using Dependency head chains

Target	Dep. Triplet	Dep. Head Chains
Atheism	72.72	72.72%
Climate	20.92	20.92%
Feminism	63.22	15.47%
Hillary	58.64	58.64%
Abortion	67.36	67.36%

For the dependency triplet model, we can see that there is approximately equal accuracy's to the bag of words model for the targets of Atheism, Feminism, Hillary and Abortion except for the Climate change target where there is a drastic drop.

For the dependency head chains model, we observed that there are similar accuracy's to dependency triplet model probably due to the reason that TiMBL is learning from the features of train data, identifying the class which is predominant in there and assigning the same class to all tweets in test data. This way it predicts the majority class even with default parameter settings.

Random Forest

All the previous algorithms were executed using TiMBL, now we will try to do the classification task using Random Forests. The data files created for different tasks previously are used as inputs for the algorithm. First one being the Bag of Words, second is the Dependency triples extracted from the output of MALT parser, and the last one is the combined feature vector generated using the bag of words approach, MPQA and n-grams. it does not seem to be improving the accuracy's for any of the targets except for Feminism. For the target Feminism, Random Forest produced the best result among all other models with an accuracy of 72.04%

TABLE VI. Random Forest - Accuracy Values for different models

Target	BOW	Dep. Triples	N-grams
Atheism	72.01%	72.30%	68.80%
Climate	72.45%	73.00%	64.91%
Feminism	56.89%	62.34%	72.04%
Hillary	62.45%	59.89%	65.10%
Abortion	64.38%	67.34%	61.77%

IV. Conclusion

This paper describes the experiments that were carried out to narrow down the best approach to classify the stance for the given targets. Though we can't assign one model to be best for all the target variables, it is visibly clear from Figure 3, that tuned parameters and Bag of Words modelling outperformed all other models consistently. It is evident that, MPQA Subjectivity Lexicon and N-gram extraction of features improved the accuracy rate significantly. BOW with MPQA Subjectivity Lexicon and N-gram extraction performed best for Abortion and Atheism target variables. Where as for Climate, BOW with tuned parameters gave the best results. For rest of the two target variables i.e. for Feminism and Hillary, Random forest performed the best.

Acknowledgment

We would like to thank Professor Sandra Kuebler for the continued guidance, for providing necessary information regarding the project and support in completing the project.We would like to express our special gratitude and thanks to our friends and classmates for giving us such attention and time.

References

[1] Stanford CoreNLP : https://stanfordnlp.github.io/CoreNLP/

[2] MALTParser user guide : http://maltparser.org/userguide.html

[3] MPQA Subjectivity Lexicon: http://mpqa.cs.pitt.edu

[4] NLTK 3.2.5 Documentation: http://www.nltk.org/_modules/nltk/util.html