Building a Sentiment Classifier for Twitter

Ayman Boustati A.BOUSTATI@WARWICK.AC.UK
Centre for Complexity Science
University of Warwick
Coventry, CV4 7AL, UK

Abstract

This report details the steps for building two text classifiers for sub-tasks A and B of task 10 of SemEval 2015 (Rosenthal et al. (2015)). The first classifier that we will present is an extension to the WarwickDCS text classifier that placed fourth in task 10A of SemEval 2015 (Townsend et al. (2015)), **Contextual Polarity Disambiguation**. The second classifier that we will present for task 10B, **Message Polarity Classification**, is a combination of the KLUE classifier that ranked fifth in task 10B of SemEval 2013 (Proisl et al. (2013)), and an adaption of our classifier that we used for task 10A. Both of these classifiers perform competitively on the two tasks when measured on the development dataset. The first classifier achieve an F1-Score of 84.96% on task 10A, while the second achieves an F1-Score of %62.26 on task 10B.

1. Introduction

Sentiment analysis has been studied extensively for the past two decades. However, the increasing popularity of social media has added a new layer of complexity to the problem of automatic sentiment detection. Posts on websites like Facebook and Twitter are very different from traditional printed media like newspapers and magazines. On social media, people to not conform to rigid sets of rules when writing; hence, the structure of social media is inherently different from what has been studied before. For instance, on Twitter, tweets are often make extensive use of slang, are riddled with spelling and grammatical mistakes, and are filled with non-traditional words such as emoticons, user mentions and hashtags. This kind of structure presents many challenges to automatic sentiment detection. hence, in this report, we will introduce two classifiers that can detect the sentiment in Twitter posts, more commonly known as "tweets". Our classifiers will tackle two tasks, **Contextual Polarity Disambiguation** and **Message Polarity Classification**, corresponding to the two sub-tasks, A and B, of task 10 of SemEval 2015, respectively. We will start the report by detailing the methodology we followed when creating the classifiers, followed by a detailed description of each of the classifiers. Finally, we will conclude with remarks on the performance of our classifiers and discuss possible improvements.

2. Methodology

In this section, we will describe the general methodology we used to build our classifiers. First, we will describe the utilisation of the provided datasets for building our classifiers, then we will move on to describe the tools that we used to process the tweets in the datasets and extract features from them. Finally, we will describe the machine learning tools that we used to create the classifiers.

2.1 The Data

We were provided with eight datasets in total. Four of which correspond to sub-task A and the other four to sub-task B. For each sub-task, we were provided with a labelled dataset for training, a labelled dataset for development, an unlabelled dataset for development and an unlabelled dataset for testing. We used the training datasets to build our classifiers and tune their parameters and the labelled development datasets to test the performance of our classifiers and select the best models. Finally, we used the training dataset to obtain predictions for submission. The unlabelled development dataset remained unused.

2.2 Preprocessing and Feature Extraction Tools

For data preprocessing we used multiple tools. Some were bespoke tools written by us specifically for the task, others, that we will mention now, were obtained by other means. The first tool was a script provided by Adam Tsakalidis that uses regular expressions to replace URL's and user mentions in tweets (Townsend et al. (2015)). The second tool was a tokeniser (`twokenize`) that is build specifically to tokenise tweets (Owoputi et al. (2013)). For feature extractions, we used three lexicons that are freely available online, two of which are Twitter specific: the Sentiment140 Lexicon and the NRC Hashtag Sentiment Lexicon (Kiritchenko et al. (2014); Mohammad et al.; Zhu et al. (2014)). The other is a general lexicon: AFINN-111 (Nielsen (2011)). Futhermore, we also used word embeddings that were pre-trained on tweets (Tang et al. (2014)).

2.3 Model Building Tools

To build the classifiers, we used the scikit-learn machine learning library for Python. The library contains a variety of machine learning tools that can be used to transform the data and build a classifier.

2.4 Pipeline

In building both of our classifiers, we use the following machine learning pipeline. First we start by preprocessing the tweets in all of our datasets, i.e. converting all letters to lowercase and substituting URL's and user mentions. We then move on to tokenising the tweets and obtaining additional features like polarity from lexicons and word embeddings. We then use a count vectoriser on the tweets to obtain the number of occurrences of the unigrams and bigrams in our vocabulary for each tweet. We aggregate all these features and feed the ones belonging to the training set to the classifier. We tune the parameters of the classifier by performing 10-fold cross-validation on the training set. We finally train the classifier on all of the training set using the optimal parameters. We use the development set to obtain an accuracy score and a macro F1-Score for our classifier. This enables use to compare different learning algorithms.

3. Sub-task A

In this section we will detail the steps we took to build our classifier for sub-task A. This classifier is based on the WarwickDCS entry in SemEval 2015 (Townsend et al. (2015)).

We will first elaborate on the data preprocessing and processing stage, followed by an explanation of the machine learning methods that we used and we will end with performance comparison.

3.1 Data Processing

The following data processing steps apply to all three datasets that we used for this sub-task: the training dataset, the labelled development dataset and the test dataset. We started off by importing the dataset into python. We followed that by randomising the dataset (training dataset only) and extracting the labels, the limits (begin and end) of the marked instances and the tweets. Working on the tweets, take for instance the following tweet from the training dataset with phrase limits (10, 12):

#OilersMobile: July 20: Sam Gagner | http://t.co/MisqEmgQ @gagns89 nice to have you there for at least another year!

We first converted all letters to lowercase:

#oilersmobile: july 20: sam gagner | http://t.co/misqemgq @gagns89 nice to have you there for at least another year!

We then replaced all URL's and user mentions with placeholders:

#oilersmobile: july 20: sam gagner | URLLINK USERMENTION nice to have you there for at least another year!

We then tokenised the tweets by splitting on white space:

[#oilersmobile:, july, 20:, sam, gagner, |, URLLINK, USERMENTION, nice, to, have, you, there, for, at, least, another, year!]

Having split on white space we were then able to extract the target phrases from the tokenised tweets. Furthermore, we decided to also extract the 4 tokens that occur before the start of the target phrase and the 4 tokens that follow the target phrase. Note, in case there were less than 4 tokens occuring before or after the target phrase, we only extract available tokens.

- Target tokens: [have, you, there]
- Pre-target tokens: [URLLINK, USERMENTION, nice, to]
- Post-target tokens: [for, at, least, another]

We also calculated the length of the target phrase to be used as a feature for out classifier:

Target length: 3

Having obtained the target phrase along with its precursors and predecessors, we then moved on to feature extraction. For this sub-task, we used 3 lexicons as well as a single word2vec embeddings dictionary. For each of the three lexicons, we obtained the polarity of the unigrams in each of the three parts of the tweet (pre-target, target, post-target) and averaged their values across the three parts. Hence, we ended up with nine features corresponding to each lexicon and tweet-part combination. Furthermore, we used a twitter-trained word embeddings dictionary on the unigrams in the three parts of the tweet to extract their vector representations. We then calculated element-wise averages for the vectors in the three parts of the tweet, one average vector for each part. We finally obtained bag-of-word (BOW) representations on the unigrams and bigrams of the tweet. We omitted any token that occurs less than 5 times or more than 80% of the time, in the whole corpus. Note that our BOW vocabulary consists only of words in the training dataset. We aggregated all these features to be fed into a classifier.

3.2 The Models

We decided to try three model on the feature set that we produced. The first of which is a Maximum Entropy classifier, more commonly known as a Logistic Regression Classifier. We also tried Support Vector Machine Classifiers with different kernels, as well as a Random Forrest Classifier. We used 10-fold cross-validation on the training set to tune the parameters of our classifiers. We used a macro F1-Score on the positive and negative classes as our performance metric. We then retrained our classifiers with the most optimal parameters on the whole training set, and remeasured their performance on the development dataset using the macro F1-score again. This served as our model selection criterion. We finally tried an ensemble of the three classifiers, and measured its performance against the development set, to compare it to the individual models. To create the ensemble, we obtained the classification probability of each class from the three models, calculated the unweighted mean of the probabilities across the three model and chose the class with the highest probability as our prediction. We used these predictions to calculate the macro F1-Score for the ensemble.

3.3 Performance Measurement

The following table details the performance of each of our three classifiers, as well as the ensemble classifier.

Performance			
Model	10-Fold CV F1-Score on training data	F1-Score on development data	Accuracy Score on development data
Logisitic Regression	86.01%	84.81%	83.41%
SVM (Linear)	86.13%	84.05%	82.33%
Random Forrest	84.80%	84.56%	83.19%
Ensemble	N/A	84.96%	83.51%

As the table suggests, the ensemble classifier is the best performing one out of the bunch; therefore, we decided to use it on the testing data for our submission. Beside having the best performance metrics, one of the advantages of the ensemble classifier is its robustness. This gave us further incentive to use it for our submission.

4. Sub-task B

In this section we will describe the classifier we created for sub-task B. This classifier is a combination of an adapted version of the classifier we used in part A (which we will refer to here as Classifier A) and the KLUE classifier that placed fifth in SemEval 2013 (Proisl et al. (2013)). We will start the section by describing the adjustments we made to Classifier A that enables it to work on sub-task be. We will then move on to detailing our implementation of the KLUE classifier, through the data preprocessing step and the model fitting step. Finally, we will present the performance of our hybrid classifier.

4.1 Adjustments to Classifier A

We made slight adjustments to Classifier A in order for it to be applicable to sub-task B. Since we are not supplied with target phrase boundaries for sub-task B, we are unable to extract pre-target, target and post-target features. Instead, we split the tweet into rough thirds (Beginning, middle and end) and extract the same features of Classifier A from these thirds. We will call this new adjusted classifier, Classifier B. Note that, since we do not have a target phrase, we cannot calculate a target length. Instead, we use the length of the tweet as a substitute to this feature.

4.2 Data Processing

This subsection describes the data preprocessing steps we took in our implementation of the KLUE classifier. As before, these steps apply to all the three datasets, training, labelled development and testing. We started off by loading the datasets into Python and randomising the training dataset. We then tokenised the tweets using the twitter tokeniser, `twokenize` (Owoputi et al. (2013)). Then we converted all tokens to lowercase and replaced URL's and user mentions with place holders. We also replaced the most used emoticons with place holders indicating whether it conveys positive sentiment and negative sentiment. We then replaced hashtag signs with the prefix (HT_) to allow us to remove all non-alphanumeric characters. We then moved on to remove non-alphanumeric symbols. Having preprocessed the tweets, we were now able to extract feature based on there structure and from lexicons and word embeddings. We first calculated the number of tokens in each tweet as one of the features, as well as the number of positive and the number of negative emoticons in each tweet. We used he AFINN-111 lexicon to find the polarity of the tokens in each tweet. Based on the polarity of the tokens, we calculated the mean polarity of the tweet, as well as the number of positive words and the number of negative words in the tweet. We then mapped the tokens in the tweets to their word embeddings, and calculated their element-wise average. Finally, w obtained the BOW representations of the tweets based on the unigrams and the bigrams in each tweet. We omitted all tokens that occur less than 6 times or more than 80% of the time. Note that, as before, we constructed the vocabulary of our BOW from the training set only.

4.3 The Models

While the original KLUE classifier used a Maximum Entropy (MaxEnt) classifier, we decided to try different options and compare them to MaxEnt. Hence, we also tried a linear SVM and a Random Forrest classifiers. As in sub-task A, we performed 10-fold cross-validation on the training set to tune the parameters of our classifiers, using the macro F1-Score as our performance metric and we tested the final performance on the development set. We also build a hybrid ensemble classifier using the two best performing models from the KLUE classifier and the two best performing models from classifier B.

4.4 Performance Measurement

The following table details the performance of the individual, as well as, the ensemble classifiers from both Classifier B and KLUE.

Performance			
Model	10-Fold CV F1-Score on training data	F1-Score on development data	Accuracy Score on development data
Classifier B - Logisitic Regression	61.79%	59.59%	67.03%
Classifier B - SVM (Linear)	62.05%	61.81%	67.03%
Classifier B - Random Forrest	42.70%	44.37%	59.17%
KLUE - Logisitic Regression	63.02%	61.79%	67.98%
KLUE - SVM (Linear)	63.14%	62.44%	68.12%
KLUE - Random Forrest	54.87%	55.48%	65.28%
Ensemble: Classifier B (Log Reg + SVM) + KLUE (Log Reg + SVM)	N/A	62.26%	68.56%

Although the table suggests that the ensemble classifier is not the best performing model, we decided to choose this model for our submission as it is likely to be the most robust out of all the other models. Since, it is performance is marginally smaller that the best performing model, our reasoning is justifiable.

5. Final Remarks

5.1 Performance

Although upon testing the performance of the classifiers seem to be very high, this might differ slightly on the testing set. Hence, we chose the two classifiers knowing that their performance might not be as powerful on the testing set.

5.2 Possible Improvements

We understand that our classifiers are by no means perfect and there is much more room for improvement. In the future, we can experiment on using further lexical and word embedding features. We can also try to augment our training datasets with tweets that have been slightly deformed (for instance, deleting words with a small probability or adding new words). Finally, we can try to use more advanced classifiers such as Gaussian Processes, Convolutional Neural Nets, and Long Short Term Memory Recurrence Neural Nets.

References

Svetlana Kiritchenko, Xiaodan Zhu, and Saif M Mohammad. Sentiment analysis of short informal texts. *Journal of Artificial Intelligence Research*, 50:723–762, 2014. URL `http://saifmohammad.com/WebDocs/NRC-Sentiment-JAIR-2014.pdf`.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. URL `http://saifmohammad.com/WebDocs/sentimentMKZ.pdf`.

Finnårup Nielsen. A new anew: Evaluation of a word list for sentiment analysis in microblogs. 2011. URL `http://arxiv.org/pdf/1103.2903v1.pdf`.

Olutobi Owoputi, Brendan O 'connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. Improved part-of-speech tagging for online conversational text with word clusters. 2013. URL `http://www.cs.cmu.edu//TweetNLP/owoputi+etal.naacl13.pdf`.

Thomas Proisl, Paul Greiner, Stefan Evert, and Besim Kabashi. Klue: Simple and robust methods for polarity classification. volume 2, pages 395–401. Association for Computational Linguistics, 2013. URL `http://www.stefan-evert.de/PUB/ProislEtc2013.pdf`.

Sara Rosenthal, Saif M Mohammad, Preslav Nakov, Alan Ritter, Svetlana Kiritchenko, and Veselin Stoyanov. Semeval-2015 task 10: Sentiment analysis in twitter. pages 451–463. Association for Computational Linguistics, 2015. URL `http://www.aclweb.org/anthology/S15-2078`.

Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification *. pages 1555–1565. Association for Computational Linguistics, 2014. URL `http://anthology.aclweb.org/P/P14/P14-1146.pdf`.

Richard Townsend, Adam Tsakalidis, Yiwei Zhou, Bo Wang, Maria Liakata, Arkaitz Zubiaga, Alexandra Cristea, and Rob Procter. Warwickdcs: From phrase-based to target-specific sentiment recognition. pages 657–663. Association for Computational Linguistics, 2015. URL `http://www.zubiaga.org/publications/files/townsend-semeval2015.pdf`.

Xiaodan Zhu, Svetlana Kiritchenko, and Saif M Mohammad. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. 2014. URL `http://saifmohammad.com/WebDocs/SemEval2014-Task9.pdf`.