TwitterHawk: A Feature Bucket Approach to Sentiment Analysis

William Boag, Peter Potash, Anna Rumshisky
Dept. of Computer Science
University of Massachusetts Lowell
198 Riverside St, Lowell, MA 01854, USA
{wboag,ppotash,arum}@cs.uml.edu

Abstract

This paper describes TwitterHawk, a system for sentiment analysis of tweets which participated in the SemEval-2015 Task 10, Subtasks A through D. The system performed competitively, most notably placing 1^{st} in topic-based sentiment classification (Subtask C) and ranking 4^{th} out of 40 in identifying the sentiment of sarcastic tweets. Our submissions in all four subtasks used a supervised learning approach to perform three-way classification to assign positive, negative, or neutral labels. Our system development efforts focused on text pre-processing and feature engineering, with a particular focus on handling negation, integrating sentiment lexicons, parsing hashtags, and handling expressive word modifications and emoticons. Two separate classifiers were developed for phrase-level and tweet-level sentiment classification. Our success in aforementioned tasks came in part from leveraging the Subtask B data and building a single tweet-level classifier for Subtasks B, C and D.

1 Introduction

In recent years, microblogging has developed into a resource for quickly and easily gathering data about how people feel about different topics. Sites such as Twitter allow for real-time communication of sentiment, thus providing unprecedented insight into how well-received products, events, and people are in the public's eye. But working with this new genre is challenging. Twitter imposes a 140-character limit on messages, which causes users to use novel abbreviations and often disregard standard sentence structures.

For the past three years, the International Workshop on Semantic Evaluation (SemEval) has been hosting a task dedicated to sentiment analysis of Twitter data. This year, our team participated in four subtasks of the challenge: Contextual Polarity Disambiguation (phrase-level), B: Message Polarity Classification (tweet-level), C: Topic-Based Message Polarity Classification (topic-based), and D: Detecting Trends Towards a Topic (trending sentiment). For a more thorough description of the tasks, see Rosenthal et al. (2015). Our system placed 1^{st} out of 7 submissions for topic-based sentiment prediction (Subtask C), 3^{rd} out of 6 submissions for detecting trends toward a topic (Subtask D), 10^{th} out of 40 submissions for tweet-level sentiment prediction (Subtask B), and 5^{th} out of 11 for phrase-level prediction (Subtask A). Our system also ranked 4^{th} out of 40 submissions in identifying the sentiment of sarcastic tweets.

Most systems that participated in this task over the past two years have relied on basic machine learning classifiers with a strong focus on developing robust and comprehensive feature set. The top system for Subtask A in both 2013 and 2014 from NRC Canada (Mohammad et al., 2013; Zhu et al., 2014) used a simple linear SVM while putting great effort into creating and incorporating sentiment lexicons as well as carefully handling negation contexts. Other teams addressed imbalances in data distributions, but still mainly focused on feature engineering, including an improved spelling correction, POS tagging, and word sense disambiguation (Miura et al., 2014). The second place submission for the 2014 Task B competition also used a neural network

Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), pages 640–646,
Denver, Colorado, June 4-5, 2015.

setup to learn sentiment-specific word embedding features along with state-of-the-art hand-crafted features (Tang et al., 2014).

Our goal in developing TwitterHawk was to build on the success of feature-driven approaches established as state-of-the-art in the two previous years of SemEval Twitter Sentiment Analysis competitions. We therefore focused on identifying and incorporating the strongest features used by the best systems, most notably, sentiment lexicons that showed good performance in ablation studies. We also performed multiple rounds of pre-processing which included tokenization, spelling correction, hashtag segmentation, wordshape replacement of URLs, as well as handling negated contexts. Our main insight for Task C involved leveraging additional training data, since the provided training data was quite small (489 examples between training and dev). Although not annotated with respect to a particular topic, we found that message-level sentiment data (Subtask B) generalized better to topic-level sentiment tracking than span-level data (Subtask A). We therefore used Subtask B data to train a more robust model for topic-level sentiment detection.

The rest of this paper is organized as follows. In Section 2, we discuss text preprocessing and normalization, describe the two classifiers we created for different subtasks, and present the features used by each model. We report system results in Section 3, and discuss system performance and future directions in Section 4.

2 System Description

We built a system to compete in four subtasks of SemEval Task 10 (Rosenthal et al., 2015). Subtasks A-C were concerned with classification of Twitter data as either positive, negative, or neutral. Subtask A involved phrase-level (usually 1-4 tokens) sentiment analysis. Subtask B dealt with classification of the entire tweet. Subtask C involved classifying a tweet's sentiment towards a given topic. Subtask D summarized the results of Subtask C by analyzing the sentiment expressed towards a topic by a group of tweets (as opposed to the single tweet classification for Subtask C).

We trained two classifiers one for phrase-level classification and one for tweet-level sentiment classification. We use the phrase-level classifier for Subtask A and we use the tweet-level classifier for Subtasks B and C. Subtask D did not require a separate classifier since it effectively just summarized the output of Subtask C. We experimented to determine whether data from Subtasks A or B generalized for C, and we found that the Subtask B model performed best at predicting for Subtask C.

2.1 Preprocessing and Normalization

Prior to feature extraction, we perform several preprocessing steps, including tokenization, spell correction, hashtag segmentation, and normalization.

Tokenization and POS-tagging Standard word tokenizers are trained on datasets from the Wall Street Journal, and consequently do not perform well on Twitter data. Some of these issues come from shorter and ill-formed sentences, unintentional misspellings, creative use of language, and abbreviations. We use ARK Tweet NLP toolkit for natural language processing in social media (Owoputi et al., 2013; Gimpel et al., 2011) for tokenization and part-of-speech tagging. An additional tokenization pass is used to split compound words that may have been mis-tokenized. This includes splitting hyphenated phrases such as 'first-place' or punctuation that was not detached from its leading text such as 'horay!!!'.

Spell Correction Twitter's informal nature and limited character space often cause tweets to contain spelling errors and abbreviations. To address this issue, we developed a spell correction module that corrects errors and expands abbreviations. Spell correction is performed in two passes. The first pass identifies the words with alternative spellings common in social media text. The second pass uses a general-purpose spell correction package from PyEnchant library.[1]

If a word w is misspelled, we check if it is one of four special forms we define:

1. **non-prose** - w is hashtag, URL, user mention, number, emoticon, or proper noun.
2. **abbreviation** - w is in our custom hand-built list that contains abbreviations as well as some common misspellings.
3. **elongated word** - w is an elongated word, such as 'heyyyy'. We define 'elongated' as repeating the same character 3 or more times in a row.

[1] http://pythonhosted.org/pyenchant/

4. **colloquial** - w matches a regex for identifying common online phrases such as 'haha' or 'lol'. We use a regex rather than a closed list for elongated phrases where more than one character is repeated in order. This allows, for 'haha' and 'hahaha', for example, to be normalized to the the same form.

Non-prose forms are handled in the tweet normalization phase (see sec 2.1). For abbreviations, we look up the expanded form in our hand-crafted list. For elongated words, we reduce all elongated substrings so that the substring's characters only occur twice. For example, this cuts both 'heeeeyyyy' and 'heeeyyyyyyyyy' down to 'heeyy'. Finally, colloquials are normalized to the shortened form (e.g., 'hahaha' becomes 'haha'). If w is not a special form, we feed it into PyEnchant library's candidate generation tool. We then filter out all candidates whose edit distance is greater than 2, and select the top candidate from PyEnchant.

Hashtag Segmentation Hashtags are often used in tweets to summarize the key ideas of the message. For instance, consider the text: We're going bowling #WeLoveBowling. Although the text "We're going bowling" does not carry any sentiment on its own, the positive sentiment of the message is expressed by the hashtag.

Similarly to spell correction, we define a general algorithm for hashtag segmentation, as well as several special cases. If hashtag h is not a special form, we reduce all characters to lowercase and then use a greedy segmentation algorithm which scans the hashtag from left to right, identifying the longest matching dictionary word. We split off the first word and repeat the process until the entire string is scanned. The algorithm does not backtrack at a dead end, but rather removes the leading character and continues. We use a trie structure to insure the efficiency of longest-prefix queries.

We identify three special cases for a hashtag h:

1. **manually segmented** - h is in our custom hand-built list of hashtags not handled correctly by the general algorithm;
2. **acronym** - h is all capitals;
3. **camel case** - h is written in CamelCase, checked with a regex.

For hashtags that are in the manually segmented list, we use the segmentation that we identified as correct. If h is an acronym, we do not segment it. Finally, for CamelCase, we treat the capitalization as indicating the segment boundaries.

Normalization and Negation During the normalization phrase, all tokens are lowercased. Next, we replace URLs, user mentions, and numbers with generic URL, USER, and NUMBER tokens, respectively. The remaining tokens are stemmed using NLTKs Snowball stemmer (Bird et al., 2009).

We also process negation contexts following the strategy used by Pang et al. (2002). We define a negation context to be a text span that begins with a negation word (such as 'no') and ends with a punctuation mark, hashtag, user mention, or URL. The suffix _neg is appended to all words inside of a negation context. We use the list of negation words from Potts (2011).

2.2 Machine Learning

For the phrase-level sentiment classification, we trained a linear Support Vector Machine (SVM) using scikit-learn's LinearSVC (Pedregosa et al., 2011) on the Subtask A training data, which contained 4832 positive examples, 2549 negative, and 384 neutral. The regularization parameter was set to C=0.05, using a grid search over the development data (648 positive, 430 negative, 57 neutral). To account for the imbalance of label distributions, we used sklearn's 'auto' class weight adjustment which applies a weight inversely proportional to a given class's frequency in the training data to the numeric prediction of each class label.

The tweet-level model was trained using scikit-learn's SGDClassifier with the hinge loss function and a learning rate of 0.001. The main difference between the learning algorithms of our classifiers was the regularization term of the loss function. While the phrase-level classifier uses the default SVM regularization, the tweet-level classifier uses an 'elasticnet' penalty with l1 ratio of .85. These parameter values were chosen following Gunther (2014) from last year's SemEval and verified in cross validation. We also used the 'auto' class weight for this task because the training label distribution was 3640 positive, 1458 negative, and 4586 neutral. We

used scikit-learn's *norm_mat* function to normalize the data matrix so that each column vector is normalized to unit length.

2.3 Features

Our system used two kinds of features: *basic text features* and *lexicon features*. We describe the two feature classes below. There was a substantial overlap between the features used for the phrase-level classifier and those used for the tweet-level classifier, with some additional features used at the phrase level.

Basic Text Features Basic text features include the features derived from the text representation, including token-level unigram features, hashtag segmentation, character-level analysis, and wordshape normalization. For a given text span, basic text features included

- presence or absence of: raw bag-of-words (BOW) unigrams, normalized/stemmed BOW unigrams, stemmed segmented hashtag BOW, user mentions, URLs, hashtags;
- number of question marks and number of exclamation points;
- number of positive, negative, and neutral emoticons; emoticons were extracted from the training data and manually tagged as positive, negative or neutral; [2]
- whether the text span contained an elongated word (see Section 2.1, special form 3).

The above features were derived from the annotated text span in both phrase-level and tweet-level analysis. For the phrase-level analysis, these were supplemented with the following:

- normalized BOW unigram features derived from 3 tokens preceding the target phrase;
- normalized BOW unigram features derived from 3 tokens following the target phrase;
- length 2, 3, and 4 character prefixes and suffixes for each token in the target phrase;
- whether the phrase was in all caps;
- whether phrase contained only stop words;
- whether a phrase contained only punctuation;
- whether the phrase contained a word whose length is eight or more;
- whether the phrase contained an elongated word (cf. Section 2.1).

There were a few other differences in the way each classifier handled some of the features. The phrase-level classifier changed the feature value from 1 to 2 for elongated unigrams. In the tweet-level classifier, we ignored unigrams with proper noun and preposition part-of-speech tags. Negation contexts were also handled differently. For the phrase-level classifier, a negated word was treated as a separate feature, whereas for the tweet-level classifier, negation changed the feature value from 1 to -1.

Lexicon Features We used several Twitter-specific and general-purpose lexicons. The lexicons fell into one of two categories: those that provided a numeric score (usually, -5 to 5) score and those that sorted phrases into categories. For a given lexicon, categories could correspond to a particular emotion, to a strong or weak positive or negative sentiment, or to automatically derived word clusters.

We used the features derived from the following lexicons: AFINN (Nielsen, 2011), Opinion Lexicon (Hu and Liu, 2004), Brown Clusters (Gimpel et al., 2011), Hashtag Emotion (Mohammad, 2012), Sentiment140 (Mohammad et al., 2013), Hashtag Sentiment (Mohammad et al., 2013), Subjectivity (Wilson et al., 2005), and General Inquirer (Stone et al., 1966). Features are derived separately for each lexicon. General Inquirer and Hashtag Emotion were excluded from the tweet-level analysis since they did not improve system performance in cross-validation. We also experimented with features derived from WordNet (Fellbaum, 1998), but these failed to improve performance for either task in ablation studies. See Section 3.1 for ablation results.

The features for the lexicons that provided a numeric score included:

- the average sentiment score for the text span;
- the total number of positively scored words in the span;
- the maximum score (or zero if no words had a sentiment score);
- the score of the last positively scored word;

[2] http://text-machine.cs.uml.edu/twitterhawk/emoticons.txt

	Opinion	Hashtag Sentiment	Sentiment140	Subjectivity	AFINN	Hashtag Emotion	Brown Clusters	General Inquirer
Phrase-level	✓	✓	✓	✓	✓		✓	
Tweet-level	✓	✓	✓	✓	✓	✓		✓

Table 1: Which lexicons we used for each classifier.

Withholding	**F-score**
– (Full System)	63.76
Opinion Lexicon	63.70
Hashtag Sentiment	63.49
Sentiment140	63.22
Hashtag Emotion (HE)	**63.77**
Brown Clusters	63.01
Subjectivity Lexicon	63.49
AFINN Lexicon	63.43
General Inquirer (GI)	**63.94**
WordNet (WN)	**65.49**
WN, GI, HE	**66.38**

Table 2: Ablation results for lexicons features in tweet-level classification.

- three most influential (most positive or most negative) scores for the text span; this was only used by the phrase-level system.

The features derived from lexicons that provided categories for words and phrases included the number of words that belonged to each category.

For phrase-level analysis, the text span used for these features was the target phrase itself. For the tweet-level analysis, the text span covered the whole tweet. Table 1 shows which lexicons we used when building each classifier.

3 Results

In this section, we describe the experiments we conducted during system development, as well as the official SemEval Task 10 results.

The scores reported throughout this section are calculated as the average of the positive and negative class F-measure (Nakov et al., 2013); the neutral label classification does not directly affect the score.

3.1 System Development Experiments

Both phrase-level and tweet-level systems were tuned in 10-fold cross-validation using the 2013 training, dev, and test data (Nakov et al., 2013). We used fixed data folds in order to compare different runs. Feature ablation studies, parameter tuning, and comparison of different pre-processing steps were performed using this setup.

We conducted ablation studies for lexicon features using tweet-level evaluation. Table 2 shows ablation results obtained in 10-fold cross-validation. The figures are bolded if withholding the features derived from a given lexicon produced a higher score. Note that these experiments were conducted using a Linear SVM classifier with a limited subset of basic text features.

Our best cross-validation results using the configuration described in sections 2.2 and 2.3 above were 87.12 average F-measure for phrase-level analysis (Subtask A), and 68.50 for tweet-level analysis (Subtask B).

For topic-level sentiment detection in Subtask C, we investigated three different approaches: (1) using our phrase-level classifier "as is", (2) training our phrase level classifier only on phrases that resembled topics[3], and (3) using our tweet-level classifier "as is". We found that our phrase-level classifiers did not perform well (F-scores in the 35-38 range), which could be explained by the fact that the Subtask A data was annotated so that the target phrases actually carried sentiment (e.g., the phrase "good luck"), whereas the Subtask C assumption was that the topic itself had no sentiment and that the topics context determined the expressed sentiment. For example, in the tweet "Gotta go see Flight tomorrow Denzel is the greatest actor ever", positive sentiment is carried by the phrase "the greatest actor ever", rather than the token "Denzel" (corresponding to the topic). It is therefore not surprising that our tweet-level classifier achieved an F-score of 54.90, since tweet-level analysis is better able to capture long-range dependencies between sentiment-carrying expressions and the target topic. Consequently, we

[3]We kept the phrases comprised by 0-or-1-determiner followed by 0-or-more-adjectives, followed by a noun.

	Phrase-level	Tweet-level
Live Journal 2014	83.97	70.17
SMS 2013	86.64	62.12
Twitter 2013	82.87	68.44
Twitter 2014	84.05	70.64
Twitter 2014 Sarcasm	85.62	56.02
Twitter 2015	**82.32**	**61.99**

Table 3: Official results

used the tweet-level classifier in our submission for Subtask C.

3.2 Official Results

Our official results for phrase-level and tweet-level tasks on the *2014 progress tests* are given in Table 3. The models were trained on the 2013 training data.

In the *official 2015 ranking*, our system performed competitively in each task. For subtask A (phrase-level), we placed 5^{th} with an F-score of 82.32, compared to the winning teams F-score of 84.79. For subtask B, we placed 10^{th} out of 40 submissions, with an F-score of 61.99, compared to the top team's 64.84. Our classifier for Subtask C won 1^{st} place with an F-score of 50.51, leading the second place entry of 45.48 by over 5 points. Finally, for Subtask D, we came in 3^{rd} place, with an average absolute difference of .214 on a 0 to 1 regression, as compared to the gold standard (Rosenthal et al., 2015).

Our system also ranked 4^{th} out of 40 submissions in identifying the message-level sentiment of sarcastic tweets in 2014 data, with an F-score of 56.02, as compared to the winning team's F-score of 59.11.

4 Discussion

Consistent with previous years' results, our system performed better on phrase-level data than on tweet-level data. We believe this is largely due to the skewed class distributions, as the majority baselines for Subtask A are much higher, and there are very few neutral labels. This is not the case for Subtask B, where the neutral labels outnumber positive labels. Also, the phrase-level text likely carries clearer sentiment, while the tweet-level analysis has to deal with conflicting sentiments across a message.

Note that hashtag segmentation strategy can be improved by using a language model to predict which segmentations are more likely, as well as evaluating the hashtag's distributional similarity to the rest of the tweet. A language model could also be used to improve the spell correction.

	Live Journal 2014	SMS 2013	Twitter 2013	Twitter 2014	Twitter Sarcasm 2014
nBow -spell	58.64	56.55	58.38	59.18	44.67
nBOW	58.87	57.22	59.19	60.27	46.76
nBOW +hashtag	58.94	57.81	60.09	61.38	53.00
nBOW +lexicon	70.65	62.08	68.46	67.86	52.89
nBOW +hashtag +lexicon	70.59	62.23	68.78	68.22	54.27

Table 4: Contribution of different features in tweet-level classification. *nBOW* stands for normalized bag-of-words features.

Our system's large margin of success at detecting topic-directed sentiment in Subtask C (over 5 points in F-score better than the 2^{nd} place team) likely comes from the fact that we leverage the large training data of Subtask B and the tweet-level model is able to capture long-range dependencies between sentiment-carrying expressions and the target topic.

We found that the most influential features for detecting sarcasm were normalized BOW unigrams, lexicon-based features, and unigrams from hashtag segmentation. Not surprisingly, lexicon features improved performance for all genres, including SMS, LiveJournal, and non-sarcastic tweets (see rows 2 and 4 in Table 4). The same was true of spelling correction (as shown in Table 4, row 1). Hashtag-based features, on the other hand, only yielded large improvements for the sarcastic tweets, as shown in the gain achieved by adding hashtag features to the normalized BOW unigrams (see rows 2 and 3 in Table 4). Note that the 6.24 point gain is only observed for sarcasm data; other genres showed the average improvement of about 0.67. We believe that hashtags were so effective at predicting sentiment for sarcasm, because sarcastic tweets facetiously emulate literal tweets at first but then express their true sentiment at the end by using a hashtag, e.g. "On the bright side we have school today... Tomorrow and the day after ! #killmenow".

Acknowledgments

@JonMadden @TaskOrganizers #thanks

References

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural language processing with Python*. O'Reilly Media, Inc.

Christiane Fellbaum. 1998. *WordNet*. Wiley Online Library.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A Smith. 2011. Part-of-speech tagging for twitter: Annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 42–47. Association for Computational Linguistics.

Tobias Günther, Jean Vancoppenolle, and Richard Johansson. 2014. Rtrgo: Enhancing the gu-mlt-lt system for sentiment analysis of short messages. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014) August 23-24, 2014 Dublin, Ireland*.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Yasuhide Miura, Shigeyuki Sakaki, Keigo Hattori, and Tomoko Ohkuma. 2014. Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data. *SemEval 2014*, page 628.

Saif M Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *arXiv preprint arXiv:1308.6242*.

Saif M Mohammad. 2012. # emotional tweets. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation*, pages 246–255. Association for Computational Linguistics.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 312–320, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Finn Årup Nielsen. 2011. A new anew: Evaluation of a word list for sentiment analysis in microblogs. *arXiv preprint arXiv:1103.2903*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *HLT-NAACL*, pages 380–390.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

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. *The Journal of Machine Learning Research*, 12:2825–2830.

Christopher Potts. 2011. Sentiment symposium tutorial. In *Sentiment Symposium Tutorial. Acknowledgments*.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. Semeval-2015 task 10: Sentiment analysis in twitter. In *Proceedings of the 9th International Workshop on Semantic Evaluation*, SemEval '2015, Denver, Colorado, June. Association for Computational Linguistics.

Philip J Stone, Dexter C Dunphy, and Marshall S Smith. 1966. The general inquirer: A computer approach to content analysis.

Duyu Tang, Furu Wei, Bing Qin, Ting Liu, and Ming Zhou. 2014. Coooolll: A deep learning system for twitter sentiment classification. *SemEval 2014*, page 208.

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

Xiaodan Zhu, Svetlana Kiritchenko, and Saif Mohammad. 2014. Nrc-canada-2014: Recent improvements in the sentiment analysis of tweets. In *Proceedings of the 8th International Workshop on Semantic Evaluation (SemEval 2014)*, pages 443–447, Dublin, Ireland, August. Association for Computational Linguistics and Dublin City University.