Mining the Web for Sympathy: The Pussy Riot Case

Anders Westling*, Joel Brynielsson*†, Tove Gustavi*†
*KTH Royal Institute of Technology, SE-100 44 Stockholm, Sweden
†FOI Swedish Defence Research Agency, SE-164 90 Stockholm, Sweden
Email: {andew, joel, gustavi}@kth.se

***Abstract*—With social media services becoming more and more popular, there now exists a constant stream of opinions publicly available on the Internet. In crisis situations, analysis of social media data can improve situation awareness and help authorities to provide better assistance to the affected population. The large amount of activity on social media services makes manual analysis infeasible. Thus, an automatic system that can assess the situation is desirable.**

In this paper we present the results of training machine learning classifiers to being able to label tweets with one of the sentiment labels positive, neutral, and negative. The classifiers were evaluated on a set of Russian tweets that were collected immediately after the much debated verdict in the 2012 trial against members of the Russian punk rock collective Pussy Riot. The aim for the classification process was to label the tweets in the dataset according to the author's sentiment towards the defendants in the trial. The results show that the obtained classifiers do not accurately and reliably classify individual tweets with sufficient certainty. However, the classifiers do show promising results on an aggregate level, performing significantly better than a majority class baseline classifier would.

***Index Terms*—Crisis management; Pussy Riot; Twitter; text mining; sentiment analysis.**

I. Introduction

Thanks to the Internet, people from all around the world are now able to share their opinions with each other wherever they are and whenever they want. This means that the public opinion on many subjects is openly available to those who want it. With social media services like Facebook, YouTube, and Twitter, users can share their opinions with their friends and the public at whatever time they feel like it. By finding and analyzing such messages related to a subject, the general opinion about something can be inferred.

Since it is easy to write and post social media texts, especially with smartphones becoming increasingly popular, there exists a constant stream of opinions. During major events such as crisis situations or sports events, thousands of messages can be expected to be posted every hour related to the event. In late 2010, a wave of demonstrations and local protests against the regimes in several Arab countries began. The events have become known as the Arab Spring [1]. During these events, Twitter became a popular tool for communication between protesters. The protesters used social media to organize protests and to voice their opinions to the rest of the world. Twitter has also been heavily used to organize help and to spread information both during and after natural disasters such as earthquakes [2].

By analyzing messages on social media as they are being posted, one can make real-time assessments of a population's reactions during an event. The information obtained from such analysis could for example be useful for crisis management during a disaster. Sentiment analysis could then be used to monitor how the affected people are feeling and how they are responding to the help and the information they get [3], [4], [5], [6], [7]. The analysis can provide valuable information regarding what kind of help that would be the most useful at the moment, and what areas to focus on next. However, as the number of messages increase, it becomes more and more difficult for humans to analyze these messages at a sufficient speed, making an automated process necessary.

In this paper, we consider a dataset consisting of tweets that were collected immediately after three members of the political punk group Pussy Riot were sentenced to prison [8]. The verdict resulted in much activity on Twitter. First, a subset of the collected tweets were manually classified as belonging to one of three categories: *positive* towards Pussy Riot (negative towards the sentence), *negative* towards Pussy Riot (positive towards the sentence), or *other*, representing the tweets that do not belong to one of the first two categories. These tweets were then used to train a number of machine learning classifiers to see if they could generalize and decide the correct sentiment of unseen texts.

The paper is organized as follows. In Section II, we present related work within the area of sentiment analysis. Section III describes the collected dataset and the process in which the training data was manually annotated. In Section IV we present the experiments conducted: the machine learning algorithms used, how the tweets were pre-processed before they were fed to the machine learning algorithms, and some of the results from evaluating the created classifiers using different algorithms and parameter settings. The results are discussed further in Section V. Finally, the paper is concluded with some thoughts regarding future work in Section VI.

II. Related Work

Sentiment analysis is a form of text classification where the purpose is to determine whether a text expresses an emotion. Sentiment analysis can for instance be performed by looking at the words that are used in the text. Some word classes are considered more useful than others for expressing sentiment. For example, adjectives and adverbs are good indicators of subjectivity [9].

When using a discriminant word approach, a list of polarized words must be created. One popular lexical database for that purpose in English is WordNet [10]. WordNet can be used

978-1-4799-6364-5/14 $31.00 © 2014 IEEE
DOI 10.1109/JISIC.2014.27

to find synonyms and antonyms of words. One way to create the list is to select a few sentiment words such as "good" and "bad," and then find words that should be similarly valued by looking at the synonyms, the synonyms of the synonyms, and so on [11]. Another lexical database based on WordNet is SentiWordNet [12]. In SentiWordNet the words have already been given sentiment values. A similar resource is the MPQA subjectivity lexicon [13], which is a lexicon consisting of 8,221 tagged lemmas.

A machine learning-based approach was used successfully by Pang et al. [14] to perform sentiment analysis on positive and negative movie reviews. The reviews were represented by feature vectors consisting of n-grams (sequences of words present in the text) from the data. Other features that have been tested are parts of speech and word positions in the text [15]. Completely different representations have been proposed, such as keeping track of the distances between positive and negative words in a text. These distances are then placed into bins, using each bin as a feature to train with [16]. When performing sentiment analysis on foreign texts, automatic translation combined with SentiWordNet and machine learning has resulted in decent results [17].

Twitter is still relatively new, so the research done on sentiment analysis of tweets is still in its infancy. Tweets are characterized by its limited number of characters. The format forces authors to express themselves creatively, using abbreviations and making up new words. Text length, as well as the use of language, affects text analysis and makes the analysis of tweets different from that of analysis of text in general. Bermingham and Smeaton [18] compared classification of short and long texts. When using only positive and negative texts, they found that unigrams and Naïve Bayes reached 74.85% accuracy on tweets. Including neutral tweets reduced the accuracy to 61.3%. Kouloumpis et al. [19] performed sentiment analysis of tweets using training data based on common hashtags that should indicate the sentiment. They used the presence of these tags to create a training set. The features used were uni- and bigrams, words from the MPQA subjectivity lexicon, as well as the presence of specific emoticons and abbreviations. The accuracy gained on a separate test set ranged up to 74%. To improve the results slightly, they increased the size of the training set by including tweets that were classified based on the presence of polarized emoticons such as ":-)" and ":(."

III. Methodology

In this section we describe how the data was collected and how the manual annotation was performed in order to create a training set to be used with the machine learning algorithms.

A. Creating the Dataset

The dataset used was collected on August 17, 2012, when three members of the Russian punk rock collective Pussy Riot were condemned to prison [8]. The group is known for performing provocative music related to the politics in Russia, and the three members were convicted of hooliganism. A number of relevant Twitter hashtags were followed with a Python program using the Twitter Streaming API [20] to collect the tweets. Roughly 130,000 tweets were collected during a period of 24 hours.

After removing non-Russian tweets and retweets, 390 of the remaining tweets were randomly selected for manual classification. By choosing the tweets at random, the training set should be a representative sample of the full set of tweets.

The tweets were then classified into three classes: *positive* to Pussy Riot (i.e., negative to the sentence), *negative* to Pussy Riot, and *other*. *Other* includes neutral tweets, tweets where a sentiment could not be found or decided upon, and tweets that simply had unrelated content, including tweets that expressed a sentiment but were not related to Pussy Riot or to the verdict. Note that the problem of classifying texts based on their expressed sentiment *for* or *against* something is significantly more difficult than the problem of classifying texts based on expressed sentiment only. Since the classifier cannot follow links, classification was made without considering the content of external websites.

B. Manual Annotation

The classifications were made by two analysts at the Swedish Defence Research Agency who are fluent in Russian and well-oriented in Russian politics. The dataset was split into two halves and the analysts got one subset each. First they manually went through and classified the tweets in their own subset. Then they went on to classify the tweets in the other person's subset independently of the first classification. If it was found that the analysts disagreed about the classification of a tweet, the person who classified it first made the final decision on the tweet's class. The analysts agreed with each other on 306 classifications, corresponding to 78% of the tweets. The result of the final classifications was that:

- 159 (41%) of the tweets were classified as *positive*,
- 59 (15%) of the tweets were classified as *negative*,
- 172 (44%) of the tweets were classified as *other*.

It should not be reasonable for an automatic classifier to perform better than the manual classifiers, so therefore an accuracy of 78% is the desirable end result for the training of a computer-based classifier. However, a lower limit for what is a reasonable performance should also be considered. The natural baseline for what one should at least be able to accomplish is the accuracy achieved when simply guessing the most common class. Based on the training data the baseline would therefore be 44% accuracy, which can always be obtained by classifying all tweets as *other*.

The distribution itself should be acceptable even if the *negative* category is much smaller than the other categories. However, since there are quite few tweets to begin with, the *negative* set might be too small, so it is expected to be more difficult to classify compared to the other two classes (see Section V for a further discussion). One thing worth mentioning here is that both of the manual annotators found the classification to be more difficult than they first expected, indicating that this is a difficult problem even for human experts: it was not always clear whom the author of the tweet

was in favor of, especially when using sarcasm and links to external websites to make their point.

IV. Experiments

The classifiers were trained to classify the tweets based on a selected set of features in the text. The feature set can be constructed in many different ways. In this work we have tested a number of representations, which will be described in this section along with the experimental results.

A. Classifiers Created

The tests were performed using the Weka open source library of machine learning algorithms [21]. The classifiers tested were four popular machine learning algorithms:

- Support Vector Machines: the SVM implementation in Weka uses a variant called sequential minimal optimization (or SMO) [22].
- Naïve Bayes: the version used was the Multinomial Naïve Bayes (NBM) classifier that is available in Weka.
- C4.5 (also known as J48 in Weka): J48 was used with its default configuration.
- k-Nearest Neighbors: kNN was configured to look at the 10 closest neighbors and have them vote on the class, with the vote weight being the inverse of the distance.

SVMs and Naïve Bayes are two of the most popular algorithms for sentiment analysis. C4.5 and kNN, on the other hand, were mostly included for the sake of comparison, but were not expected to perform as well. kNN also uses lazy learning, doing all the work when classifying each instance. This makes kNN non-viable in a real-time system as it would be too slow. That is also the reason why little time has been put into finding optimal parameter values for the kNN classifier.

Six parameters were considered when creating the classifiers, all of which affect what the feature sets look like:

- *Word presence*, i.e., whether to use word frequency or just word presence in the feature vectors.
- *Maximum n-gram size*, i.e., up to how large the n-grams could be. Values tried were 1, 2, and 3.
- *Minimum term frequency*, i.e., the minimum number of times an n-gram had to occur in the training tweets to be considered. The values tried were the integers in the range of 1 to 5.
- *Stemming*, i.e., whether the stemmer was used or not.
- *Stop terms*, i.e., the number of stop terms removed. The tested values ranged from 0 to 30, increasing in steps of 5.
- *Information gain features*, i.e., how many of the most informative n-grams calculated by information gain that were used. This value began at 25 and increased in steps of 25 until all features remained.

The parameters are described in more detail below. All combinations of these parameter values, as well as the classifiers, were tested. Finding an optimal classifier by fine-tuning parameter values like this is likely to result in a higher accuracy than can be considered realistic, being very specific for this particular dataset (so called *over-training*). To compensate for this, the reader should keep in mind that the results for the best classifiers, as presented in this paper, are probably a little better than they would have been if the classifiers had been evaluated on another dataset. We now return to the parameters considered in the classification process.

Word presence refers to whether the representation of a text should contain the number of times a word occurs in the text or if only the presence of the word should be considered.

Maximum n-gram size is relevant to consider since pairs or triples of words can have their own meaning (such as "not good"). We test whether extending the feature vectors with pairs (2-grams) and triples (3-grams) of words will improve the classification.

Minimum term frequency refers to how many times an n-gram must occur in the training data to be included in the feature vectors. Many words will only occur a few times in the training set, and very little statistical support for their sentiment value exist. These words should be removed to reduce noise when training the classifiers.

Stemming is a linguistic technique that can also be used to reduce the number of words in the feature set. The process reduces a word into a base form, or stem [23]. The purpose of this is to treat all inflections of a word in the same way. For example, "stemming," "stemmer," and "stems" could all be reduced into "stem." The Snowball [24] stemmer for Russian was used to perform the stemming.

To reduce dimensionality further the most common n-grams in the training data can be removed from the feature set since they can be assumed to be function words with little subjective meaning. These terms are referred to as stop terms. An alternative to removing the most frequently occurring words is to use an external list of function words to remove. This was tested at an early stage, but did not result in a noticeable improvement of the results. Also, the quality of the list could not be assessed. Therefore it was decided that the most common n-grams should be removed.

Since the feature vectors can become quite large and there are often many features that have little or no impact on the classification process, feature reduction techniques can be used to create smaller and hopefully more effective feature vectors. Classical feature reduction techniques used for machine learning include information gain, chi-square and mutual information which are all viable options [11]. In this work, information gain was chosen as the feature selection algorithm to be used.

10-fold cross-validation was used to evaluate the classifiers. Creation and reduction of the feature set was performed on the training set of each fold.

B. Experimental Results

The four classifiers were compared when using the different parameter settings discussed above. Since there are many possible combinations, only a few of the results will be presented below. In addition, some observations regarding the parameters' influence on the results will be discussed. Naïve

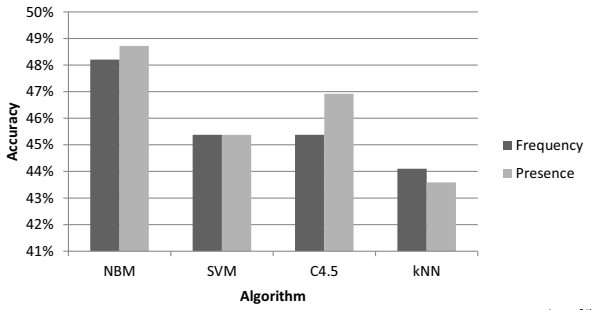

Fig. 1. A comparison of the difference in accuracy between using term frequency and term presence. The features used were unigrams with no feature selections made.

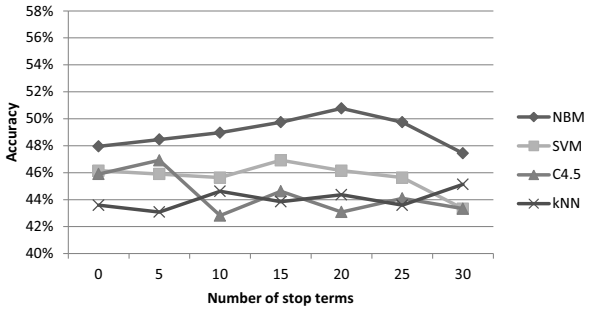

Fig. 2. A comparison of the difference in accuracy between different numbers of stop words. Unigrams were used with word presence and a minimum frequency of two.

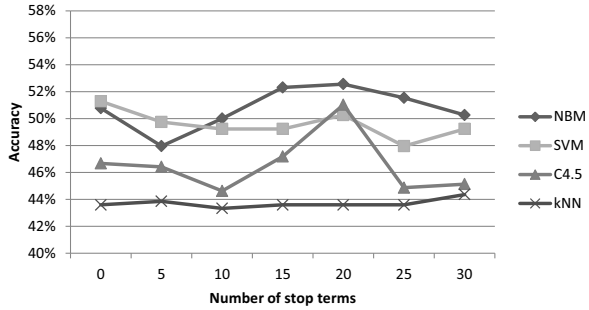

Fig. 3. The graph shows how the stemmer improves the accuracies in the test shown in Fig. 2.

Bayes proved to be the best algorithm, reaching an accuracy of almost 55.4%. This was followed by the SVM at 53.3%. That these two algorithms would perform the best was expected since they are known to be good algorithms for sentiment analysis. Even so, all four algorithms achieved over 50% accuracy, exceeding our 44% baseline accuracy. In Table I the best settings for each classifier algorithm and the achieved accuracy can be seen.

TABLE I
THE PARAMETER SETUP FOR THE MOST ACCURATE VERSION OF EACH ALGORITHM. "COUNT" IS YES IF FREQUENCY WAS USED FOR FEATURE VALUES. "SIZE" IS THE MAXIMUM SIZE OF THE N-GRAMS. "FREQ." IS HOW MANY TIMES AN N-GRAM HAD TO OCCUR IN THE TRAINING DATA. "STEM" IS IF STEMMING WAS USED. "STOP" IS HOW MANY STOP TERMS THAT WERE REMOVED. "IG" IS HOW MANY OF THE MOST INFORMATIVE FEATURES FOUND USING INFORMATION GAIN THAT WERE USED.

Alg.	Accuracy	Count	Size	Freq.	Stem	Stop	IG
NBM	55.38%	No	1	2	Yes	20	150
SVM	53.33%	Yes	1	2	Yes	15	125
C4.5	51.03%	No	1	2	Yes	20	All
kNN	50.51%	Yes	1	4	Yes	15	25

Regarding whether to count the number of occurrences of a word in the text versus just looking at the presence, no version was clearly superior although word presence was used with the most accurate classifier (see Fig. 1).

Increasing the maximum sizes of the n-grams was hoped to improve results by providing knowledge not available when solely looking at isolated words. Surprisingly, it turned out that including larger n-grams than unigrams did not make a large difference. While good results were still achieved when including bigrams, the best classifiers only used unigrams for all four classifier algorithms.

Removing rare words did help to some extent, although removing all words occurring fewer than four times as described by Pang et al. [14] did not result in the best results except for the kNN algorithm. Removing n-grams occurring just once in the training data was the most effective for the remaining classifier algorithms.

Removal of the most common n-grams helped. In the experiments, removal of the 15–20 most common words gave the best results, while removing more lowered the accuracy (see Fig. 2). The Snowball stemmer also proved to be quite useful, improving results by up to several percentage units as can be seen in Fig. 3.

Removing the n-grams with the lowest information gain helped improve the results even further: the SVM and Naïve Bayes performed the best when removing roughly half of the n-grams, while kNN worked well with only 25 n-grams, and C4.5 was not very affected when using information gain.

In Table II the confusion matrix for the best classifier can be found. The matrix shows how the classifier performed on the individual classes. It can be observed that the classifier is very good at classifying the *other* tweets correctly, having a recall of 77%. Classification of the *positive* tweets had a recall of 48%. The *negative* class had a low recall of 14%.

TABLE II
THE CONFUSION MATRIX FOR THE BEST VERSION OF THE NAÏVE BAYES CLASSIFIER. THE MATRIX SHOWS HOW THE CLASSIFIER PREDICTIONS WERE DISTRIBUTED AND HOW WELL THEY COINCIDE WITH THE ACTUAL CLASS VALUE.

	Predicted class		
Actual class	Positive	Negative	Other
Positive	76	1	82
Negative	20	8	31
Other	32	8	132

V. Discussion

The results show that all of the classifiers reached over 50% accuracy, i.e., better than the baseline accuracy of 44%. The best classifier achieved 55% accuracy, i.e., far from the goal of 78%. Still, the experiments show that the algorithms have indeed been able to learn characteristic sentiment-related features using as little as 390 tweets for training. Nevertheless, the size of the available dataset for training is believed to be the main reason why the accuracy was not higher. Achieving a low accuracy with a training set consisting of only a few hundred tweets is hardly surprising, and especially so considering the complexity and variety of human language. Using a larger dataset for training consisting of at least a thousand tweets as has been reported on in similar work [6], [18], [19], should improve the accuracy. This would, however, require more manual work for annotating the training data.

Regarding precision it is not surprising that the low amount of negative tweets in the dataset resulted in a low precision for this class. The best classifiers probably learned that more or less ignoring the class proved more efficient than taking it into consideration, i.e., trading a low precision on *negative* tweets for higher precisions on the other two classes.

Another issue to be considered is the nature of tweets. To begin with tweets are short, meaning that there is not much information to learn from when deciding on a sentiment class. On the plus side they should not contain mixed sentiments as there can only be room for one or possibly two emotional statements within one single tweet. Another potential problem is related to the informal language. Using slang and misspellings makes classification more difficult, and with tweets being informal and short, people tend to be creative with regard to how they phrase themselves.

The nature of the considered classification problem is also a challenge in itself. Compared to the more "classical" sentiment analysis problem where just the general sentiment of a text is considered, the classification problem in this work should be more difficult since here it is not enough to just decide that a positive tweet is positive, but it is also essential to know what the tweet is positive towards. How much more difficult this actually makes learning is hard to tell, but it is safe to say that it should only make things more difficult. This means that more advanced natural language processing techniques might prove helpful for determining the subject of a statement.

A few words also needs to be said about the parameters used when creating the feature set. In terms of word presence versus word frequency, we found no clear winner. Most likely, this parameter makes little difference since tweets are so short to begin with. In most cases the words only occur once anyway.

Increasing the size of the n-grams did not result in clear differences. Looking at Table I, all algorithms performed at their best with only unigrams. Adding larger n-grams is probably both helpful and harmful. Some features improve classification while most of them are just noisy. In the end, using just unigrams supplemented with feature selection techniques seems to be enough, which is also in agreement with similar work [14].

Stemming turned out to be quite helpful. With so few words being used, the process of stemming should be especially useful since turning inflected words into a single term helps making sure that important words are not removed for being infrequent. One thing to consider is therefore whether stemming is as useful when using more training data.

Regarding the removal of infrequent words, no clear general optimal values could be found although the right value did improve the accuracy for all classifier instances. Removing words that only occur once makes perfect sense as there is close to no statistical support for them being helpful. Having a higher threshold than two probably removed too many words, even if many of them were unnecessary. This is a consequence of the dataset being small.

Removing the most frequent words seemed to be helpful to some extent. In the future one could consider using a list of function words to be removed, but this would require that the quality of the list can first be verified by a person having a thorough knowledge about the Russian language.

Information gain worked in a similar way compared to the other feature removal techniques: the best result did become better. One thing worth noting according to Table I is that C4.5 did not make use of the feature removal, which is most likely due to C4.5 using information gain internally when creating the decision tree.

VI. Conclusions and Future Work

We have compared four popular machine learning algorithms as well as investigating a number of different representations of text features for the purpose of performing sentiment analysis of tweets written during a crisis-related situation. After manually annotating 390 randomly selected tweets, we trained and evaluated classifiers to see how high accuracy that could be achieved. We found that despite a small amount of training data, using the right feature selection techniques we were able to reach an accuracy of 55% using the Naïve Bayes classifier. This is significantly better than the considered baseline of 44%, but far from the goal accuracy of 78%. Even if the accuracy is high enough to provide sentiment knowledge on an aggregate level, further inspection showed that the classification of *negative* tweets (the smallest subset in the training data) was too poor to be of any use.

For future work, the most important thing to test is how much the classification can be improved using more training data, and in particular how capable the different classifier algorithms are when it comes to making use of such additional training data. It is reasonable to assume that more training data will lead to a better classifier, and especially so considering how small the set of negative tweets were. Also, according to machine learning theory it can be expected that for a certain amount of additional training data the (non-biased) SVM classifier will start outperforming the (biased) Naïve Bayes classifier [25]. Other methods for improving classification can be tested as well. Using part of speech tagging to only include nouns, adjectives, verbs, and adverbs as features would be a

good way to eliminate noisy words. It could also be used to separate homographs and help decide the subject of a polarized statement.

There are many different scenarios where an automatic tweet classifier can be of use, and it would be of interest to try some other classifications to see how they compare to the one presented in this paper. An example of this could be to find different emotions in tweets written during a natural disaster such as a tsunami or an earthquake. Such an example differs from this work since the target of an emotion would not matter, which should result in an easier problem.

Since one of the biggest problems was acquiring a decently sized training dataset, future work could include looking at alternatives to having experts classify tweets manually. Having analysts spend time doing this can be seen as a waste of time, and if a large amount of tweets is required it will not be feasible. One idea would be to crowdsource the classification using a service like Amazon Mechanical Turk [26]. A large number of tweets could be distributed to people all over the world for obtaining manual classifications in exchange for a sum of money. This would require that the instructions for the classification task are crystal clear so that the whole crowd classifies the tweets in the exact same way.

Another idea presented by Kouloumpis et al. [19] is to automatically classify tweets based on some criteria. The presence of emoticons is suggested, but inspection of the collected tweets shows that there barely exists any of the more common emoticons in the complete dataset, and even if emoticons did exist it probably would not help since the target of the sentiment is important. In other situations it might be possible to create a training set consisting of tweets related to a sentiment by looking for specific words. These words could be found by identifying a couple of seed words ("happy," "angry," etc.) and then expand the set by looking for synonyms using, e.g., WordNet. Hopefully the tweets found this way would be general enough to teach a classifier how to identify texts with similar sentiment not using these words. A test set would still have to be created through manual classification, though, but it would not have to be very large compared to the training set which might make such a method feasible anyway.

ACKNOWLEDGMENTS

The authors would like to thank Ulrik Franke and Carolina Vendil Pallin for annotating the dataset, and Fredrik Johansson for insightful methodological discussions.

REFERENCES

[1] G. Blight, S. Pulham, and P. Torpey. Arab spring: an interactive timeline of Middle East protests. The Guardian. [Online]. Available: http://www.theguardian.com/world/interactive/2011/mar/22/middle-east-protest-interactive-timeline

[2] S. Sternberg. Japan crisis showcases social media's muscle. USA TODAY. [Online]. Available: http://www.usatoday.com/tech/news/2011-04-12-1Ajapansocialmedia12_CV_N.htm

[3] H. Artman, J. Brynielsson, B. J. E. Johansson, and J. Trnka, "Dialogical emergency management and strategic awareness in emergency communication," in *Proceedings of the Eighth International Conference on Information Systems for Crisis Response and Management (ISCRAM 2011)*, Lisbon, Portugal, May 2011.

[4] S. Nilsson, J. Brynielsson, M. Granåsen, C. Hellgren, S. Lindquist, M. Lundin, M. Narganes Quijano, and J. Trnka, "Making use of new media for pan-European crisis communication," in *Proceedings of the Ninth International Conference on Information Systems for Crisis Response and Management (ISCRAM 2012)*, Vancouver, Canada, Apr. 2012.

[5] F. Johansson, J. Brynielsson, and M. Narganes Quijano, "Estimating citizen alertness in crises using social media monitoring and analysis," in *Proceedings of the 2012 European Intelligence and Security Informatics Conference (EISIC 2012)*, Odense, Denmark, Aug. 2012, pp. 189–196.

[6] J. Brynielsson, F. Johansson, and A. Westling, "Learning to classify emotional content in crisis-related tweets," in *Proceedings of the 11th IEEE International Conference on Intelligence and Security Informatics (ISI 2013)*, Seattle, Washington, Jun. 2013, pp. 33–38.

[7] J. Brynielsson, F. Johansson, C. Jonsson, and A. Westling, "Emotion classification of social media posts for estimating people's reactions to communicated alert messages during crises," *Security Informatics*, vol. 3, no. 7, 2014.

[8] M. Elder. Pussy Riot sentenced to two years in prison colony over anti-Putin protest. The Guardian. [Online]. Available: http://www.theguardian.com/music/2012/aug/17/pussy-riot-sentenced-prison-putin

[9] B. Liu, "Sentiment analysis and subjectivity," in *Handbook of Natural Language Processing*, 2nd ed., ser. Chapman & Hall/CRC Machine Learning & Pattern Recognition Series, N. Indurkhya and F. J. Damerau, Eds. Boca Raton, Florida: Taylor & Francis Group, 2010, ch. 26, pp. 627–666.

[10] WordNet. [Online]. Available: http://wordnet.princeton.edu/

[11] P. Koncz and J. Paralic, "An approach to feature selection for sentiment analysis," in *Proceedings of the 15th IEEE International Conference on Intelligent Engineering Systems (INES 2011)*, Poprad, Slovakia, Jun. 2011, pp. 357–362.

[12] SentiWordNet. [Online]. Available: http://sentiwordnet.isti.cnr.it/

[13] MPQA Subj. Lexicon. [Online]. Available: http://mpqa.cs.pitt.edu/

[14] B. Pang, L. Lee, and S. Vaithyanathan, "Thumbs up? Sentiment classification using machine learning techniques," in *Proceedings of the Seventh ACL Conference on Empirical Methods in Natural Language Processing (EMNLP-02)*, Philadelphia, Pennsylvania, Jul. 2002, pp. 79–86.

[15] S. Li, H. Zhang, W. Xu, G. Chen, and J. Guo, "Exploiting combined multi-level model for document sentiment analysis," in *Proceedings of the 20th International Conference on Pattern Recognition (ICPR 2010)*, Istanbul, Turkey, Aug. 2010, pp. 4141–4144.

[16] S. M. S. Hasan and D. A. Adjeroh, "Proximity-based sentiment analysis," in *Proceedings of the Fourth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2011)*, Stevens Point, Wisconsin, Aug. 2011, pp. 106–111.

[17] K. Denecke, "Using SentiWordNet for multilingual sentiment analysis," in *Proceedings of the 24th IEEE International Conference on Data Engineering Workshop (ICDEW 2008)*, Cancún, Mexico, Apr. 2008, pp. 507–512.

[18] A. Bermingham and A. F. Smeaton, "Classifying sentiment in microblogs: Is brevity an advantage?" in *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM 2010)*, Toronto, Canada, Oct. 2010, pp. 1833–1836.

[19] E. Kouloumpis, T. Wilson, and J. Moore, "Twitter sentiment analysis: The good the bad and the OMG!" in *Proceedings of the Fifth International AAAI Conference on Weblogs and Social Media*, Barcelona, Spain, Jul. 2011, pp. 538–541.

[20] Twitter API docs. [Online]. Available: https://dev.twitter.com/docs

[21] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The WEKA data mining software: An update," *ACM SIGKDD Explorations Newsletter*, vol. 11, no. 1, pp. 10–18, Jun. 2009.

[22] J. C. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods: Support Vector Learning*, B. Schölkopf, C. J. C. Burges, and A. J. Smola, Eds. Cambridge, Massachusetts: MIT Press, 1999, ch. 12, pp. 185–208.

[23] J. B. Lovins, "Development of a stemming algorithm," *Mechanical Translation and Computational Linguistics*, vol. 11, no. 1–2, pp. 22–31, Mar. and Jun. 1968.

[24] Snowball. [Online]. Available: http://snowball.tartarus.org/

[25] C. D. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. Cambridge, United Kingdom: Cambridge University Press, 2008, ch. 15, pp. 319–348.

[26] Amazon Mechanical Turk. [Online]. Available: https://www.mturk.com/