GU-MLT-LT: Sentiment Analysis of Short Messages using Linguistic Features and Stochastic Gradient Descent

Tobias Günther
University of Gothenburg
Olof Wijksgatan 6
41255 Göteborg, Sweden
email@tobias.io

Lenz Furrer
University of Zurich
Binzmühlestrasse 14
8050 Zürich, Switzerland
lenz.furrer@gmail.com

Abstract

This paper describes the details of our system submitted to the SemEval-2013 shared task on sentiment analysis in Twitter. Our approach to predicting the sentiment of Tweets and SMS is based on supervised machine learning techniques and task-specific feature engineering. We used a linear classifier trained by stochastic gradient descent with hinge loss and elastic net regularization to make our predictions, which were ranked first or second in three of the four experimental conditions of the shared task. Furthermore, our system makes use of social media specific text preprocessing and linguistically motivated features, such as word stems, word clusters and negation handling.

1 Introduction

Sentiment analysis, also known as opinion mining, is a research field in the area of text mining and natural language processing, which investigates the automated detection of opinions in language. In written text, an opinion is a person's attitude towards some topic, pronounced by verbal (e.g. choice of words, rhetorical figures) or non-verbal means (e.g. emoticons, emphatic spelling). More formally, Liu (2012) defines an opinion as the quintuple $(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$ where "e_i is the name of an entity, a_{ij} is an aspect of e_i, s_{ijkl} is the sentiment on aspect a_{ij} of entity e_i, h_k is the opinion holder, and t_l is the time when the opinion is expressed by h_k. The sentiment s_{ijkl} is positive, negative, or neutral, or expressed with different strength/intensity levels [...]. When an opinion is on the entity itself as a whole, the special aspect GENERAL is used to denote it. [...] e_i and a_{ij} together represent the opinion target" (Liu, 2012).

With the massively growing importance of social media in everyday life, being able to automatically find and classify attitudes in written text allows for estimating the mood of a large group of people, e.g. towards a certain event, service, product, matter of fact or the like. As working with the very short and informal texts typical for social networks poses challenges not encountered in more traditional text genres, the International Workshop on Semantic Evaluation (SemEval) 2013 has a shared task on sentiment analysis in microblogging texts, which is detailed in Wilson et al. (2013). The task requires sentiment analysis of Twitter[1] and SMS messages and comprises two subtasks, one of which deals with determining the sentiment of a given message fragment depending on its context (Task A) and one on overall message polarity classification (Task B).

We treat both tasks as document-level sentiment classification tasks, which we define according to Liu (2012) as determining the opinion $(\_, \text{GENERAL}, s, \_, \_)$ of a given message, where $s \in \{\text{positive}, \text{negative}, \text{neutral}\}$ and "the entity e, opinion holder h, and time of opinion t are assumed known or irrelevant" (Liu, 2012). For Task A we only consider the marked fraction of the message to be given.

This introduction is followed by sections discussing related work (2), details of our system (3), experiments (4) and results and conclusion (5).

[1] a popular microblogging service on the Internet, see http://twitter.com

*Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 328–332, Atlanta, Georgia, June 14-15, 2013.

2 Related Work

Previous approaches to sentiment analysis of microblogging texts make use of a wide range of features, including unigrams, n-grams, part-of-speech tags and polarity values from (usually hand-crafted) sentiment lexicons. O'Connor et al. (2010) examine tweets concerned with the 2009 US presidential elections, relying solely on the occurrence of words from a sentiment lexicon. Nielsen (2011) investigates the impact of including internet slang and obscene language when building a sentiment lexicon. Barbosa and Feng (2010) make use of three different sentiment detection websites to label Twitter data, while Davidov et al. (2010), Kouloumpis et al. (2011) and Pak and Paroubek (2010) use Twitter hashtags and emoticons as labels. Speriosu et al. (2011) propagate information from seed labels along a linked structure that includes Twitter's follower graph, and Saif et al. (2012) specifically address the data-sparsity problem by using semantic smoothing and topic extraction.

3 System Description

In this section we present the details of our sentiment analysis system, which was implemented in the Python programming language and is publicly available online.[2] We used the same preprocessing, feature extraction and learning algorithm for both subtasks, only the hyperparameters of the machine learning algorithm were adjusted to the respective dataset.

3.1 Preprocessing

Tokenization of the messages was done using a simple regular expression, which matches either URLs, alphanumeric character sequences (plus apostrophe) or non-alphanumeric non-whitespace character sequences. This way punctuation sequences like emoticons are preserved, while still being separated from words in case of missing whitespace. The same happens to hashtags, so "#liiike:)" gets separated into the three tokens `#`, `liiike` and `:)`, which can then be processed separately or as n-grams. While this strategy performed reasonably well for us, more sophisticated tokenizers for social media messages that handle more special cases like emoticons including letters are thinkable.

To address the large variety in spelling typical for social networks we store three different variants of each token:

a) The *raw* token found in the message

b) A *normalized* version, in which all characters are converted to lowercase and all digits to 0

c) A *collapsed* version, in which all adjacent duplicate characters are removed from the normalized version, if it is not present in an English word list. That way "school" stays "school", but "liiike" gets converted to "like".

3.2 Features

We explored a wide variety of linguistic and lexical features. In our final submission we used the following set of features for each message:

- The **normalized tokens** [boolean]
- The **stems** of the collapsed tokens, which were computed using the Porter stemming algorithm (Porter, 1980) implemented in the Python Natural Language Toolkit (Bird et al., 2009). [boolean]
- The **word cluster** IDs of raw, normalized and collapsed tokens. The clusters were obtained via unsupervised Brown clustering (Brown et al., 1992) of 56,345,753 Tweets by Owoputi et al. (2013) and are available on the web.[3] [boolean]
- The accumulated (summed) positive and accumulated negative **SentiWordNet** scores (Baccianella et al., 2010) of all synsets matching the collapsed token strings. [continuous]

Furthermore, the normalized tokens and stems were marked with a special **negation** prefix, if they occurred after a negation word or word cluster of negation words. If a punctuation token occurred before the end of the message the marking was discontinued at that point.

[2] `http://tobias.io/semevaltweet`

[3] `http://www.ark.cs.cmu.edu/TweetNLP`

3.3 Machine Learning Methods

For the classification of the messages into the positive, negative and neutral classes we use three linear models, which were trained in an one-vs.-all manner. At prediction time we simply choose the label with the highest score. All training was done using the open-source machine learning toolkit *scikit-learn*,[4] which provides a consistent Python API to fast implementations of various machine learning algorithms (Pedregosa et al., 2011).

The linear models were trained using *stochastic gradient descent* (SGD), which is a gradient descent optimization method that minimizes a given loss function. The term "stochastic" refers to the fact that the weights of the model are updated for each training example, which is an approximation of batch gradient descent, in which all training examples are considered to make a single step. This way SGD is very fast to train, which was important to us to be able to rapidly evaluate different feature combinations and hyperparameter settings using cross-validation.

Algorithm 1 Stochastic gradient descent with hinge loss and elastic net regularization

1: $t \leftarrow 1/(\eta\,\alpha)$
2: $u \leftarrow 0$
3: Initialize w_j and q_j with 0 for all j
4: **for** *epoch* to N_{ITER} **do**
5: **for** i to N_{SAMPLES} **do**
6: $s \leftarrow w^T x^{(i)}$
7: $\eta \leftarrow 1/(\alpha\,t)$
8: $c \leftarrow \text{CLASSWEIGHT}(y^{(i)})$
9: $u \leftarrow u + ((1-\rho)\,\eta\,\alpha)$
10: **for** j to N_{FEATURES} **do**
11: $\frac{\partial \ell}{\partial w_j} \leftarrow \begin{cases} -y^{(i)} x_j^{(i)} & \text{if } y^{(i)} s < 1 \\ 0 & \text{otherwise} \end{cases}$
12: $w_j \leftarrow (1-\rho\,\eta\,\alpha)\,w_j - \eta\,c\,\frac{\partial \ell}{\partial w_j}$
13: $z \leftarrow w_j$
14: **if** $w_j > 0$ **then**
15: $w_j \leftarrow \max(0, w_j - (u + q_j))$
16: **else if** $w_j < 0$ **then**
17: $w_j \leftarrow \min(0, w_j + (u - q_j))$
18: $q_j \leftarrow q_j + (w_j - z)$
19: $t \leftarrow t + 1$

[4]Version 0.13.1, `http://scikit-learn.org`

Hyperparameter	Task A	Task B
N_{ITER}	1000	1000
$\text{CLASSWEIGHT}(y^{(i)})$	1	auto[5]
α	0.0001	0.001
ρ	0.15	0.15

Table 1: Hyperparameters used for final model training

The loss function we used was *hinge loss*, which is a large-margin loss function known for its use in support vector machines. To avoid overfitting the training set we employed *elastic net regularization*, which is a combination of L1 and L2 regularization.

A simplified version of the SGD learning procedure implemented in *scikit-learn* is shown in Algorithm 1, where w is the weight vector of the model, $x^{(i)}$ the feature vector of sample i, $y^{(i)} \in \{-1, +1\}$ the ground truth label of sample i, η the learning rate, α the regularization factor and ρ the elastic net mixing parameter. Be aware that we did not pick samples at random or shuffle the data, which is crucial in case of training data which is sorted by classes. The initial learning rate is set heuristically and updated following Shalev-Shwartz et al. (2007).[6] The way of applying the L1 penalty (lines 13 to 18) is published as "cumulative L1 penalty" in Tsuruoka et al. (2009). The final settings for the hyperparameters were determined by running a cross-validated grid search on the combined training and development sets and can be found in Table 1.

4 Experiments

For our experiments and the final model training we used the combined training and development set of the shared task. For Task A we removed messages labeled "objective" prior to training, while we merged them into the "neutral" class for Task B. This left us with 9419 training samples (5855 positive, 457 neutral, 3107 negative) for Task A and 10368 training samples (3855 positive, 4889 neutral, 1624 negative) for Task B. As the shared task was evaluated on average F_1 of the positive and negative class, disregarding the neutral class, we also provide our results in these measures in the following.

[5]inversely proportional to class frequency

[6]This is achieved by choosing "optimal" as setting for the learning rate for *scikit-learn's SGDClassifier*.

	Negative		Positive		Avg.
	Prec	**Rec**	**Prec**	**Rec**	**F_1**
ALL	53.86	62.68	77.88	68.95	65.54
-stem	-0.38	**-1.10**	-0.07	-0.08	-0.385
-wc	**-0.74**	-0.30	+0.13	**-2.05**	**-0.835**
-swn	-0.15	-0.73	-0.27	+0.10	-0.23
-neg	+0.04	-0.92	**-1.06**	+0.44	-0.30
bow	-4.03	-7.01	-0.44	-3.68	-3.83

Table 2: Feature ablation study (Task B)

During the process of preparing our submission we used 10-fold cross-validation to evaluate different combinations of features, machine learning algorithms and their hyperparameter settings. While we found the features described in section 3.2 to be useful, we did not find further improvement by using n-grams and part-of-speech tags, despite using the Twitter-specific part-of-speech tagger by Owoputi et al. (2013). Table 2 shows a cross-validated ablation study on the features, removing one group of features at a time to see their contribution to the model. Using only normalized tokens is referred to as bag-of-words (bow). One can see that word clusters are the most important for our model, causing the highest overall loss in F_1 performance when being removed. Nevertheless, all other features contribute to the performance of the model as well.

Further improvement can be made by carefully picking a machine learning algorithm and tuning its hyperparameters. For this task we found linear models to perform better than other classification methods such as naive bayes, decision tree / random forest and k-nearest neighbor. Figure 1 shows that models trained with the method described in section 3.3 (marked SGD) clearly outperforms models trained with the popular perceptron algorithm (which could be described as stochastic gradient descent with zero-one loss, no regularization and constant learning rate, marked PER) with increasing training set size. The values were obtained by training on different portions of the training set of Task B and testing on the previously unseen Task B Twitter test set (3813 samples). Starting from a certain amount of available training data, the choice of the training algorithm becomes even more important than the choice of features.

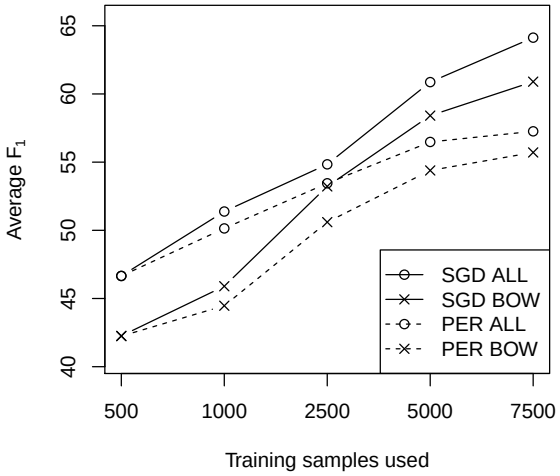

Figure 1: Effect of training set size on different classifiers

5 Results and Conclusion

The results of our submission for the four hidden test sets of the shared task can be found in Table 3. Given the relatively small deviation from the results of the cross-validation on combined training and development set and the good ranks obtained in the shared task system ranking, we conclude that the method for sentiment analysis in microblogging messages presented in this paper yields competitive results.

We showed that the performance for this task can be improved by using linguistically motivated features as well as carefully choosing a learning algorithm and its hyperparameter settings.

Task	**Prec**	**Rec**	**F_1 (Rank)**
A SMS	86.09	91.01	88.37 (1)
A Twitter	85.06	85.43	85.19 (7)
B SMS	55.83	72.55	62.15 (2)
B Twitter	70.21	61.49	65.27 (2)

Table 3: Final results of our submission

Acknowledgments

We would like to thank the organizers of the shared task for their effort, Peter Prettenhofer for his help with getting to the bottom of the SGD implementation in *scikit-learn* and Richard Johansson as well as the anonymous reviewers for their helpful comments on the paper.

References

Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. 2010. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC'10), Valletta, Malta*.

Luciano Barbosa and Junlan Feng. 2010. Robust Sentiment Detection on Twitter from Biased and Noisy Data. In *COLING (Posters)*, pages 36–44.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J Della Pietra, and Jenifer C Lai. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–479.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Enhanced Sentiment Learning Using Twitter Hashtags and Smileys. In *COLING (Posters)*, pages 241–249.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter Sentiment Analysis: The Good the Bad and the OMG! In *Fifth International AAAI Conference on Weblogs and Social Media*, ICWSM.

Bing Liu. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Finn Årup Nielsen. 2011. A new ANEW: Evaluation of a word list for sentiment analysis in microblogs. In *Proceedings of the ESWC2011 Workshop on 'Making Sense of Microposts': Big things come in small packages*, volume 718 of *CEUR Workshop Proceedings*, pages 93–98.

Brendan O'Connor, Ramnath Balasubramanyan, Bryan R. Routledge, and Noah A. Smith. 2010. From Tweets to Polls: Linking Text Sentiment to Public Opinion Time Series. In *Fourth International AAAI Conference on Weblogs and Social Media*, ICWSM.

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 *Proceedings of NAACL 2013*.

Alexander Pak and Patrick Paroubek. 2010. Twitter as a Corpus for Sentiment Analysis and Opinion Mining. In *Proceedings of the 7th conference on International Language Resources and Evaluation*, volume 2010, pages 1320–1326.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Martin F Porter. 1980. An algorithm for suffix stripping. *Program: electronic library and information systems*, 14(3):130–137.

Hassan Saif, Yulan He, and Harith Alani. 2012. Alleviating data sparsity for twitter sentiment analysis. In *Proceedings of the 2nd Workshop on Making Sense of Microposts*.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the 24th international conference on Machine learning*, pages 807–814. ACM.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. Twitter Polarity Classification with Label Propagation over Lexical Links and the Follower Graph. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, page 53–63.

Yoshimasa Tsuruoka, Jun'ichi Tsujii, and Sophia Ananiadou. 2009. Stochastic gradient descent training for l1-regularized log-linear models with cumulative penalty. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1*, pages 477–485. Association for Computational Linguistics.

Theresa Wilson, Zornitsa Kozareva, Preslav Nakov, Alan Ritter, Sara Rosenthal, and Veselin Stoyanov. 2013. Semeval-2013 task 2: Sentiment analysis in twitter. In *Proceedings of the 7th International Workshop on Semantic Evaluation*. Association for Computational Linguistics.