# CIS-positive: Combining Convolutional Neural Networks and SVMs for Sentiment Analysis in Twitter

**Sebastian Ebert** and **Ngoc Thang Vu** and **Hinrich Schütze**
Center for Information and Language Processing
University of Munich, Germany
{ebert|thangvu}@cis.lmu.de, inquiries@cislmu.org

## Abstract

This paper describes our automatic sentiment analysis system – CIS-positive – for SemEval 2015 Task 10 "Sentiment Analysis in Twitter", subtask B "Message Polarity Classification". In this system, we propose to normalize the Twitter data in a way that maximizes the coverage of sentiment lexicons and minimizes distracting elements. Furthermore, we integrate the output of Convolutional Neural Networks into Support Vector Machines for the polarity classification. Our system achieves a macro $F_1$ score of the positive and negative class of 59.57 on the SemEval 2015 test data.

## 1 Introduction

On the Internet, text containing different forms of sentiment appears everywhere. Mining this information supports many types of interest groups. Companies, for instance, are interested in user feedback about the advantages and drawbacks of their products. Users want to read short reviews or ratings of hotels they want to book for their next vacation. Politicians try to predict the outcome of the next presidential election. An automatic sentiment analysis system can support all these different requirements. One source of these types of information covering many domains and topics is the social networking service Twitter. Its popularity and the users' productivity in creating new text makes it an interesting research topic. However, Twitter introduces specific challenges as we will see next.

In general, automatic sentiment analysis is challenging due to many different factors, such as ambiguous word senses, context dependency, sarcasm, etc. Specific properties of Twitter text make this task even more challenging. The limit of 140 character per message leads to countless acronyms and abbreviations. Moreover, the vast majority of tweets is of informal character and contains intentional miss-spellings and wrong use of grammar. Hence, the out-of-vocabulary (OOV) rate of Twitter text is rather high, which leads to information loss.

One of the SemEval 2015 shared tasks – Task 10: Sentiment Analysis in Twitter – addresses these challenges (Rosenthal et al., 2015). We participated in Subtask B the "Message Polarity Classification" task. The goal is to predict the polarity of a given tweet into *positive*, *negative*, or *neutral*. The task organizers provided tweet IDs and corresponding labels to have a common ground for training polarity classification systems. More information about the task, its other subtasks as well as information about how the data was selected can be found in (Rosenthal et al., 2015).

In this paper, we present our sentiment analysis system for SemEval 2015 - Task 10. Our system addresses the above mentioned challenges in two ways. First, we normalize the text to maximize the coverage of sentiment lexicons and minimize distracting elements such as user names or URLs. Second, we combine deep Convolutional Neural Networks (CNN) and support vector machines (SVM) for a better overall classification. The motivation of using CNNs is to extract not only local features but also context to predict sentiment. Integrating CNN output into an SVM improves classification.

## 2 Data Preprocessing

Twitter texts are challenging and differ from other domains in some specific properties. Due to the 140

*Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 527–532,
Denver, Colorado, June 4-5, 2015. ©2015 Association for Computational Linguistics

characters limit of tweet length, users make heavy use of abbreviations and acronyms. This leads to a high OOV rate and makes tasks like tokenizing, part-of-speech (POS) tagging, and lexicon search more difficult. Furthermore, special tokens such as user mentions (e.g., “@isar”), urls, hashtags (e.g., “#happy”), and punctuation sequences like “!?!?” are often utilized. Therefore, normalization of all tweets is necessary to facilitate later polarity classification. Our text preprocessing pipeline can be described as follows: Tweets are first tokenized and POS tagged with the CMU tokenizer and tagger (Owoputi et al., 2013). This tagger is specialized for Twitter and therefore superior to other general domain taggers. Afterwards, all user mentions are replaced by “<user>” and all urls by “<web>”, because they do not provide any cues of polarity. We do not replace hashtags, because they often contain valuable information such as topics or even sentiment.

Punctuation sequences like “!?!?” can act as exaggeration or other polarity modifier. However, the sheer amount of possible sequences increases the OOV rate dramatically. Therefore, all sequences of punctuations are replaced by a list of distinct punctuations in this sequence (e.g., “!?!?” is replaced by “[!?]”). That reduces the OOV rate and still keeps most of the information.

Mohammad et al. (2013) showed that sentiment lexicons are crucial for achieving good polarity classification. Unfortunately, miss-spellings and elongated surface forms of sentiment-bearing tokens, such as “cooooolllll”, lead to lower coverage of all sentiment lexicons. Since elongated words often convey sentiment (Brody and Diakopoulos, 2011), we carefully normalize them in the following way. First, all elongated words are identified by searching for tokens that contain a sequence of at least three equal characters. Afterwards, for each elongated word a candidate set is created by removing the repeated character one by one until only one occurrence is left. If a word contains several repeated character sequences, all combinations are taken as candidates. For instance, the candidate set of the word “cooolll” will be {coolll, colll, cooll, coll, cool, col}. We then search every candidate in a sentiment lexicon to find the correct canonical form of the elongated word. If there is more than one match, the shortest match is taken. Since several sentiment lexicons with different qualities exist, we apply a sequential approach. We search the canonical form of the elongated word in one lexicon. If it does not exist, the next lexicon in the sequence is searched. The sequence of sentiment lexicons is sorted based on the reliability of the lexicon. Manually created lexicons precede automatically created lexicons. In this paper, the ordering is as follows: MPQA subjectivity cues lexicon (Wilson et al., 2005), Opinion lexicon (Hu and Liu, 2004), NRCC Emotion lexicon (Mohammad and Turney, 2013), sentiment 140, and Hashtag lexicon (both in (Mohammad et al., 2013)). As a result, a mapping from elongated words to their canonical form is found and used to normalize the corpus. Lowercasing finalizes the preprocessing step.

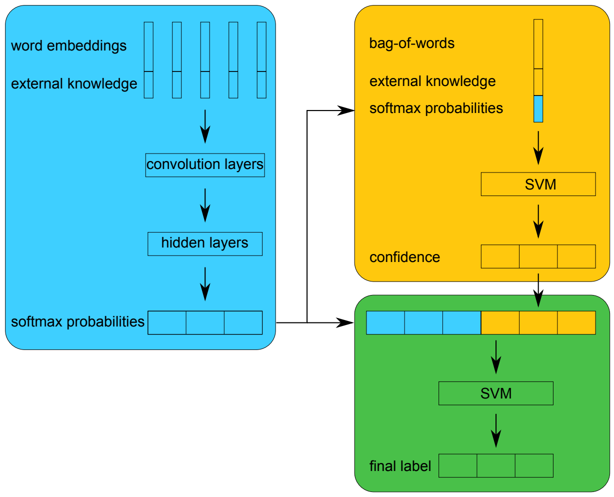

Figure 1: System architecture

## 3 Model

The system architecture consists of three main components and is depicted in Figure 1. The first component is a CNN (left part in the figure), which makes use of the sequence of all words in a tweet. The second component is an SVM classifier which uses several linguistic features and the CNN’s output as input (top right part in Figure 1). Finally, to combine the polarity prediction of the CNN and the SVM we use another SVM on top to receive the final polarity label (bottom right part in Figure 1). In this section all components are described in detail.

### 3.1 CNN

The intuition of using a CNN for sentence modeling is to have a model that is able to capture sequential phenomenon and considers words in their contexts. In a bag-of-words approach the word *not*, indicating negation, is not set into relation to the words it negates. An n-gram approach might tackle this problem to some extent, but long distance effects are still not captured. Furthermore, a bag-of-words model suffers from sparsity. A CNN is a neural network that can handle sequences by performing a mathematical convolution operation with a filter matrix and the input. The goal is to conflate the input sequence into a meaningful representation by finding salient features that indicate polarity. More formally, the words in the model are represented by two matrices. First, $P \in \mathbb{R}^{d_p \times V}$ denotes a matrix of low dimensional word representations, so called *word embeddings*. $d_p$, the size of the embeddings, is usually set to 50-300, depending on the task. $V$ denotes the size of the vocabulary. The matrix $P$ is learned during model training. It is initialized either randomly or with a pretrained matrix, as we will describe later. In addition to $P$, we introduce another matrix $Q \in \mathbb{R}^{d_q \times V}$ which contains external word features. In this case, $d_q$ is the number of features per word. This approach allows us to add as much external knowledge into the training process as needed. The features are precomputed and not embedded into any embeddings space, i.e. $Q$ is fixed during training. A description of all features is given later in this section.

Both components are concatenated into a lookup table $LT = \begin{bmatrix} P \\ Q \end{bmatrix}$, where each column corresponds to the entire representation of a certain word in the vocabulary. Given a sentence of $n$ words $w_1$ to $w_n$, the model concatenates all $n$ word representation to the input of the CNN

$$S = \begin{bmatrix} | & | & | \\ LT_{:,w_1} & \cdots & LT_{:,w_n} \\ | & | & | \end{bmatrix}.$$

A one dimensional convolution is a mathematical operation that slides a filter $\mathbf{m} \in \mathbb{R}^{1 \times m}$ over a vector and computes a dot product at every position. The length of the filter $m$ specifies how many elements the filter spans. Applying this concept to a two dimensional input leads to a convolution matrix where the elements are computed by

$$C_{i,j} = \mathbf{m}^T S_{i,j:j+m-1},$$

where $i$ is the $i$th row in $S$ and $j$ is the start index of the convolution.

$A$, the output of the convolution layer is computed by an element-wise addition of a bias term (one bias per row) and an element-wise non-linearity: $A = f(C + \mathbf{b})$. As non-linear function we use a rectified linear unit: $f(x) = max(0, x)$. This non-linearity proved to be a crucial part in object recognition (Jarrett et al., 2009), machine translation (Vaswani et al., 2013), and speech recognition (Zeiler et al., 2013).

Our model uses two layers of convolution. The concatenation of all rows of the second convolution layer output is the input to a sequence of three fully connected hidden layers. A hidden layer transforms the input vector $\mathbf{x}$ into $\mathbf{z} = f(W\mathbf{x} + b)$, where $W$ is a weight matrix that is learned during training and $b$ is a bias. In order to convert the final hidden layer output $\mathbf{z}$ into a probability distribution over polarity labels $\mathbf{o} \in \mathbb{R}^3$, the softmax function is used: $\mathbf{o}_i = \frac{\exp(\mathbf{z}_i)}{\sum_j \exp(\mathbf{z}_j)}$.

**Pretraining of Word Embeddings** The standard way of initializing the word embeddings matrix $P$ is by sampling from a uniform distribution. Since there is only a small amount of training data available, word representations cannot be learned from scratch before the model would overfit. Therefore, instead of initializing the word embeddings matrix randomly, we precompute word embeddings with the word2vec toolkit on a large amount of Twitter text data.[1] We first downloaded about 60 million tweets from the unlabeled Twitter Events data set (McMinn et al., 2013). This corpus is normalized as described in Section 2. We then select $V$ words, comprising all the words of the SemEval training data, words from the sentiment lexicons, and the most frequent words of the Twitter Events data set. Finally a continuous bag-of-words model (Mikolov et al., 2013) with 50 dimensional vectors is trained and used to initialize $P$.

[1]`https://code.google.com/p/word2vec/`

**Word Features** In addition to the word embeddings the CNN receives additional external features (matrix $Q$). These features are the following:

**binary sentiment indicators** binary features that indicate the polarity of a token in a sentiment lexicon. The lexicons for this feature are MPQA (Wilson et al., 2005), Opinion lexicon (Hu and Liu, 2004) and NRCC Emotion lexicon (Mohammad and Turney, 2013).

**sentiment scores** the sentiment 140 lexicon and the Hashtag lexicon (Mohammad et al., 2013) both provide a score for each token instead of just a label. We directly use these scores. Both lexicons also contain scores for bigrams and skip ngrams. In such a case each word of an ngram receives the score of the entire ngram.

**binary negation** following the procedure of Christopher Potts' Sentiment Symposium tutorial[2] we mark each token between a negation word and the next punctuation as negated.

### 3.2 SVM 1

Since training the CNN for many epochs (entire runs over the whole dataset) always led to overfitting, we decided to use a second classifier, an SVM. Following Mohammad et al. (2013) we use the following features:

**binary bag-of-words** binary bag-of-words features of uni- and bigrams, as well as character trigrams. In contrast to (Mohammad et al., 2013) our system does not use trigrams or character ngrams of higher order, because it degraded the performance on the validation set.

**sentiment features** for every tweet and every lexicon we add the following features: number of tokens in the tweet that occur in the lexicon, sum of all sentiment scores in the tweet, maximum sentiment score, and the sentiment score of the last token in the tweet.

**CNN output** to inform the SVM about the CNN's classification decision and certainty, we add the softmax output of the CNN as an additional feature.

[2]`http://sentiment.christopherpotts.net/lingstruc.html`

As linear SVM implementation we use LIBLINEAR (Fan et al., 2008).

### 3.3 SVM 2

Analyzing the CNN and SVM 1 predictions we found that both classifiers learn orthogonal features. Therefore, we introduce a second linear SVM into the classification pipeline, which combines the softmax probabilities of the CNN and the confidence scores of the first SVM. The output is the final predicted polarity label of our system.

## 4 Experiments

Twitter's terms of service do not allow to provide tweets as text. Instead, the participants of the SemEval 2015 task had to download the tweets using a list of user and tweet IDs. However, not all tweets are still available. After downloading, our training data comprises a total of 8394 tweets, 3133 of which are positive, 1237 negative, and 4023 neutral. The evaluation is done on two separate test sets. The first test set, the progress test set, was used as test set in previous years of SemEval 2013 (Nakov et al., 2013) and SemEval 2014 (Rosenthal et al., 2014). It consists of 3506 positive, 1541 negative, and 3940 neutral short text (a total of 8987). This set contains not only Twitter texts, but also SMS text messages, blog posts (LiveJournal), and tweets that are marked as sarcastic. The second test set, the SemEval 2015 test set, contains 2390 Twitter tweets, 1038 positive, 365 negative, and 987 neutral. Table 1 lists all test set sizes in detail. As evaluation measure the organizers chose to report the macro $F_1$ score of positive and negative examples, i.e., $F_{1,macro} = (F_{1,positive} + F_{1,negative})/2$.

The CNN is trained using minibatch stochastic gradient descent with a batch size of 200 examples. For learning rate adaptation we use AdaGrad (Duchi et al., 2011) with an initial learning rate of 0.001. $\ell_2$ with $\lambda = 0.001$ is utilized to avoid overfitting as much as possible. The embeddings size is set to 50. In the first convolution layer, we use 30 filters with a $m = 5$, which means it spans 5 words. The second convolution layer uses 10 filters with $m = 3$. The three hidden layers have sizes 200, 40, and 200. This choice of layer sizes with a bottleneck layer between two larger layers is frequently

Table 1: Test set sizes and results

| | #pos | #neg | #neu | $F_{1,positive}$ | $F_{1,negative}$ | $F_{1,neutral}$ | $F_{1,macro}$ |
|---|---|---|---|---|---|---|---|
| SemEval 2013 Twitter | 1572 | 601 | 1640 | 71.32 | 58.31 | 72.53 | 64.82 |
| SemEval 2013 SMS | 492 | 394 | 1207 | 66.94 | 63.34 | 80.33 | 65.14 |
| SemEval 2014 LiveJournal | 427 | 304 | 411 | 71.09 | 71.84 | 69.04 | 71.47 |
| SemEval 2014 Twitter | 982 | 202 | 669 | 73.63 | 58.47 | 67.14 | 66.05 |
| SemEval 2014 Twitter sarcasm | 33 | 40 | 13 | 60.00 | 38.46 | 53.33 | 49.23 |
| SemEval 2015 Twitter | 1038 | 365 | 987 | 65.32 | 53.82 | 68.06 | 59.57 |

used in automated speech recognition systems. For example Grézl et al. (2007) showed that using the bottleneck layer's output leads to lower word error rates than using hidden layer outputs. However, our experimental results show that using the output of the CNN softmax layer as input for the first SVM achieves slightly better performance than using the output of the bottle-neck layer.

For both linear SVMs we tune the $C$ parameter on the validation data.

**Results** The last line in Table 1 lists the $F_1$ performances of our system on the SemEval 2015 test set. The performance on negative examples is much worse than on positive or neutral examples. This is due to the small number of negative training examples. The macro $F_1$ score of 59.57 leads to rank 20 out of 40 participants in this year's SemEval. The fact that our system scores much better on LifeJournal and the SMS data in terms of $F_{1,negative}$ suggests that Twitter is an especially difficult medium for automated analysis.

The performance difference on Twitter from 2013 and 2014 compared to Twitter 2015 suggests that this year's Twitter data was different than in the years before. Our system scored similarly on Twitter from 2013 and 2014, but worse on 2015. Even worse results are achieved on the sarcasm data. However, the results should be taken with care, because this sub set is very small.

## 5 Related Work

One early work that used CNNs to model sentences was published by Collobert et al. (2011). They used one convolution layer followed by a max pooling layer to create a sentence representation. We extend their method by incorporating additional features focused on the polarity classification task. In contrast to their approach, we do not embed our external features, but make direct use of them.

Kalchbrenner et al. (2014) show that a CNN for modeling sentences can achieve competitive results in polarity classification. Among others, they introduce dynamic k-max pooling, a method that adapts max pooling to the length of an input sentence. Compared to their work we use a simpler architecture of the CNN without max-pooling, because this technique did not show any improvements in our experiments. Furthermore, we use the same filter for each dimension to reduce the number of parameters, whereas their model uses a different filter per dimension. Finally, our CNN model is combined with another classifier to produce the final polarity label.

Using an SVM for polarity classification is a common approach. One of the first polarity classification systems used bag-of-words features and an SVM to classify the polarity of movie reviews (Pang et al., 2002). The winning system of SemEval 2013 and SemEval 2014 also used an SVM with many different features (Mohammad et al., 2013). We implemented their most helpful features, which is bag-of-words and lexicon features and added the CNN output as an additional feature to improve the final performance.

## 6 Conclusion

This paper summarizes the features of our automatic sentiment analysis system – CIS-positive – for the SemEval 2015 shared task - Task 10, subtask B. We carefully normalize the Twitter data and integrate the output of convolutional neural networks into support vector machines for the polarity classification. Our system achieves a macro F-score of 59.57 on the SemEval 2015 test data. Among the 40 participants in this subtask our system reached rank 20 with a distance of 5.0 $F_1$ points to the winning system.

## Acknowledgments

This work was supported by DFG (grant SCHU 2246/10).

## References

Samuel Brody and Nicholas Diakopoulos. 2011. Cooooooooooooooollllllllllllll!!!!!!!!!!!!!! Using Word Lengthening to Detect Sentiment in Microblogs. In *EMNLP*.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural Language Processing (almost) from Scratch. *JMLR*, 12.

John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR*, 12.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A Library for Large Linear Classification. *JMLR*, 9.

Frantisek Grézl, Martin Karafiát, Stanislav Kontar, and Jan Cernocký. 2007. Probabilistic and Bottle-Neck Features for LVCSR of Meetings. In *ICASSP*.

Minqing Hu and Bing Liu. 2004. Mining and Summarizing Customer Reviews. In *KDD*.

Kevin Jarrett, Koray Kavukcuoglu, Marc'Aurelio Ranzato, and Yann LeCun. 2009. What is the Best Multi-Stage Architecture for Object Recognition? In *ICCV*.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *ACL*.

Andrew J. McMinn, Yashar Moshfeghi, and Joemon M. Jose. 2013. Building a large-scale corpus for evaluating event detection on twitter. In *CIKM*.

Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. In *ICLR*.

Saif M. Mohammad and Peter D. Turney. 2013. Crowdsourcing a Word-Emotion Association Lexicon. *Computational Intelligence*, 29(3).

Saif M. Mohammad, Svetlana Kiritchenko, and Xiaodan Zhu. 2013. NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. In *SemEval*.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In *SemEval*.

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 *NAACL HLT*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs Up?: Sentiment Classification Using Machine Learning Techniques. In *EMNLP*.

Sara Rosenthal, Alan Ritter, Preslav Nakov, and Veselin Stoyanov. 2014. SemEval-2014 Task 9: Sentiment Analysis in Twitter. In *SemEval*.

Sara Rosenthal, Preslav Nakov, Svetlana Kiritchenko, Saif M. Mohammad, Alan Ritter, and Veselin Stoyanov. 2015. SemEval-2015 Task 10: Sentiment Analysis in Twitter. In *SemEval*.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with Large-Scale Neural Language Models Improves Translation. In *EMNLP*.

Theresa Ann Wilson, Janyce Wiebe, and Paul Hoffmann. 2005. Recognizing Contextual Polarity in Phrase-Level Sentiment Analysis. In *HLT/EMNLP*.

Matthew D. Zeiler, Marc'Aurelio Ranzato, Rajat Monga, Mark Z. Mao, K. Yang, Quoc Le Viet, Patrick Nguyen, Andrew W. Senior, Vincent Vanhoucke, Jeffrey Dean, and Geoffrey E. Hinton. 2013. On rectified linear units for speech processing. In *ICASSP*.