Text Sentiment Classification on Twitter Data

Judith Beestermöller, Katharina Börsig, Zuzana Frankovská and Katja Möhring
Team: Hey there, Twinkeldies
Department of Computer Science, ETH Zurich, Switzerland

Abstract. Recent results in text sentiment classification focus on computationally complex models, mainly convolutional and recurrent neural networks paired with attention mechanisms. State-of-the-art results are achieved with substantial computational power. Our model makes use of the same basic architectures, but keeps memory requirements small and computation fast. This is achieved by running various small models concurrently on a twitter data set and combining their results with an ensemble method to achieve high an overall accuracy of 88.42% rather than relying on a single complex architecture.

I. INTRODUCTION

The goal of sentiment analysis is to classify the opinion expressed in text, for example, to distinguish between positive or negative statements. In the last two decades sentiment analysis has become one of the fastest growing research areas in computer science and has found applications in social media [1], film reviews [2], and healthcare [3]. Deep neural networks have increasingly become the state of the art models for text sentiment analysis in recent years [4] with projects following a similar procedure: Initially, the text is represented as a mathematical object using text embedding algorithms such as FastText [5], GloVe [6], and Word2Vec [7]. Additionally, the text is often preprocessed prior to embedding, aiming to both remove noise and reduce irrelevant information [8]. The types of deep neural networks which have achieved highest performance in sentiment analysis are convolutional neural networks (CNN) [9] and recurrent neural networks (RNN) [10], with long-short term memory (LSTM) [11], bi-directional LSTM (Bi-LSTM) [12] and gated recurrent unit (GRU) [13] being the most popular layers. More recently, attention mechanisms have been used, with the aim to model dependencies independent of spatial distance [14, 15]. Such attention mechanisms are heavily used in machine translation tasks providing pre-trained models for language understanding [16].

This study aims to apply sentiment analysis to classify a dataset of twitter messages as either positive or negative by building upon the previously mentioned algorithms. We test different forms of text preprocessing and self-trained text embeddings and use these as inputs for various CNNs and RNNs, with and without attention layers. Finally, we combine the output of the different neural networks in different ensemble models to improve performance. Throughout this study, our focus has been on developing high performing algorithms without requiring extensive computational resources such as memory and runtime.

II. MODELS AND METHODS

A. Preprocessing and Padding

The supplied data was already preprocessed with usernames and hyperlinks being replaced by standardized tags. For further preprocessing, we considered the following techniques as stated by Magliani et al. [17] and Bao et al. [18]: the removal of numbers, punctuation, repeated letters, and stopwords, as well as stemming, lemmatization, and handling emojis. However, not all of these techniques are relevant for twitter data since tweets are short and often contain slang, abbreviations, typos, intentionally misspelled words, and excessive use of punctuation. Based on experiments using linear classification and Word2Vec embeddings (Section II B), we identified the highest performing techniques (Figure 2) and applied these in all further computations.

To improve computational efficiency, tweet lengths were standardized to 32 words. By this threshold, less than 1% of all tweets were shortened (Figure 1). For tweets shorter than 32 words, we increased their length by padding, considering two possibilities: pre-padding and post-padding. The experiments by Mahidhar and Reddy [19] showed that pre-padding performs better for LSTMs, whilst having no significant effect on CNNs. Therefore, we decided on this method.

B. Word Embedding

We self-trained three types of word embedding: FastText, Word2Vec, and a Keras integrated embedding layer. For comparison, each of them was built on the same preprocessed text file and where possible used the same parameters such as the embedding dimension. We choose an embedding dimension of 200 to balance runtime and model complexity. For the integrated embedding layer, we reduced the vocabulary to the most common 10,000 words, whereas FastText and Word2Vec embeddings considered only words which occurred at least 15 times in the given data set. These thresholds were chosen empirically in order to reduce irrelevant information.

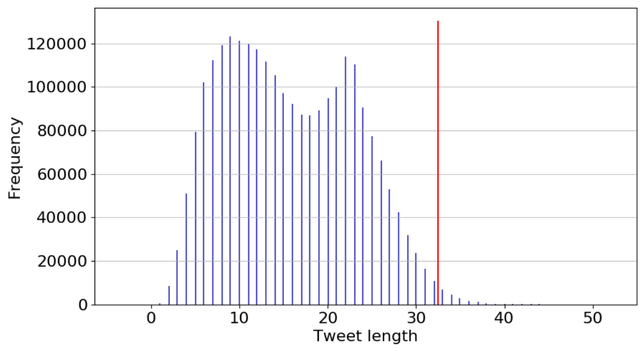

FIG. 1. Histogram showing the frequency of tweets as function of tweet length measured in words. The threshold value of 32 words (red line) was used for standardization of tweet length and encompasses 99% of the available data.

C. Baselines

For baseline computations, we used simple, established models for sentiment classification. The text preprocessing techniques and word embeddings are the same as those used in our more complex models, as described in Section II D. The first baseline is computed with a linear classifier and the other baselines use a basic neural network consisting of a fully connected layer with 1024 neurons followed by softmax layer.

D. Models

The more complex deep neural network models we trained for this study are described below. Overall, we trained ten different models consisting of convolutional neural networks as well as recurrent neural networks.

a. Convolutional neural networks (CNN)
Convolutional neural networks (CNN) have been proven to be successful in sentence classification [9]. In contrast to the typical visualization tasks of CNNs, text-based data is 1-dimensional and therefore, we used 1-dimensional convolutional layers. The filters of a convolutional layer extract data-dependent features considering sequences of specified size. In order to reduce the sensitivity of the features to the sequence position, we applied a pooling layer afterwards. We used two different convolutional architectures (Table I), one two layered CNN with an attention layer, and a single CNN layer stacked with a bidirectional LSTM in order to extract both short and long-term features.

b. Recurrent neural networks (RNN)
Recurrent neural networks (RNN) are known to be well-suited for Natural Language Processing (NLP) tasks [20]. By their recurrent architecture such models are able to establish relationships between words in sequences and can compute contextualized features for sentences. In this study, we used two different types of recurrent architectures consisting of LSTMs and GRUs (Table II).

	CNN + Att	**CNN + BiLSTM + Att**
emb	Integrated/W2V	Integrated/W2V
conv1	100 kernels, size 7	200 kernels, size 3
maxpool	size 2	size 2
conv2	64 kernels, size 7	-
bilstm	-	128 cells
att	yes	yes
dense (relu)	-	64 units
softmax	yes	yes

TABLE I. Architecture of the two CNN Models. One CNN combined with an attention layer (Att) and a CNN stacked with a bidirectional LSTM and an attention layer.

	LSTM	**BiGRU + Att**	**3-BiLSTM**
emb	Integrated/W2V	Integrated/W2V	Integrated/W2V
rnn1	200 cells	200 cells	100 cells
rnn2	100 cells	-	100 cells
rnn3	-	-	100 cells
attention	-	yes	-
dense (relu)	100 units	200 units	50 units
softmax	yes	yes	yes

TABLE II. Architecture of the three RNN Models. The first model uses only LSTMs, the second model uses a bidirectional GRU with an attention layer (Att) and the third model uses three layers of bidirectional LSTMs.

c. Attention
For some neural network models we included an attention layer as described by Bahdanau et al. [21]. The authors define attention as being part of the input encoding. The attention mechanism produces a context vector by performing a weighted average of all the encoders hidden states. The weight associated with a particular hidden state is determined by an alignment score between that particular hidden state and the previous hidden state. Thus, the input vector $X = (x_1, x_2 \ldots, x_n)$ of words is converted into a vector $Y = (y_1, y_2, \ldots, y_n)$, where y_i combines the information of x_i and its relationship to all other words in the input sequence, independent of the relative distances between the words. Attention is therefore able to assign different weights to different features of the input [15].

d. Classifier Ensemble Methods
In order to improve the performance of the classification, we explored different methods of combining the predictions of the ten models described in Section II D, as suggested by da Silva et al. [22]. In the first ensemble method, we applied a simple majority vote to the predictions of the models. In the second ensemble method, we computed a weighted majority vote, where we considered the probabilities of a tweet belonging to each class, as determined by each model. This takes into account the confidence each model has in its prediction. Other than these classic methods, we were inspired by Deriu et al. [23] to train a random forest classifier, with 100 trees and a maximum depth of 2 per forest. For our last ensemble method, we trained a simple neural network,

with two fully connected layers of 20 units, and a softmax layer.

E. Experimental setup

We were given a Twitter dataset that was split into a full training dataset (2,500,000 tweets), small training dataset (200,000 tweets), and a test dataset (10,000 unlabelled tweets). The training sets were split evenly into positive and negative tweets. We constructed five validation splits from the full training dataset, with each validation test set containing 500,000 randomly ordered tweets. The self-trained word embeddings were computed on the full training dataset and the test set. Individual models were trained on the full training set and all cross-validation training datasets. The ensemble models that needed to be trained (Neural Network and Random Forest) were trained using the predictions of the individual models on the cross validation test dataset.

All models were trained for two epochs using a batch size of 32, and an Adam optimizer with learning rate 0.001. We trained our models mostly on our own laptops and used the ETH Leonhard cluster to perform some preliminary experiments. For the implementation of our models we used standard python libraries including Gensim [24], Keras [25], and Scikit-learn [26].

III. RESULTS

A. Preprocessing

Figure 2 shows that removing numbers, stemming, lemmatization as well as handling emojis individually improves accuracy whilst other preprocessing techniques including removing punctuation lower accuracy. Combining all techniques that improve the accuracy into one ensemble technique achieves the highest accuracy. Hence, we decided to apply this form of preprocessing for all of our further models.

B. Baselines & Word Embeddings

The baseline model validation accuracies are stated in Table III. The results indicate that simple neural networks outperform the linear classifier. Furthermore, the integrated trainable embedding followed by the word2vec embedding gives highest performance for a simple neural network. The FastText embedding with a score of 81.13% for the simple neural network is only slightly worse than Word2Vec (81.89%), albeit requires substantially more computational resources including runtime and memory. Thus, we refrained from using the FastText embedding in further models.

C. Deep Learning Models

The cross-validation accuracies of our individual models are shown in Table IV. Each model was computed both

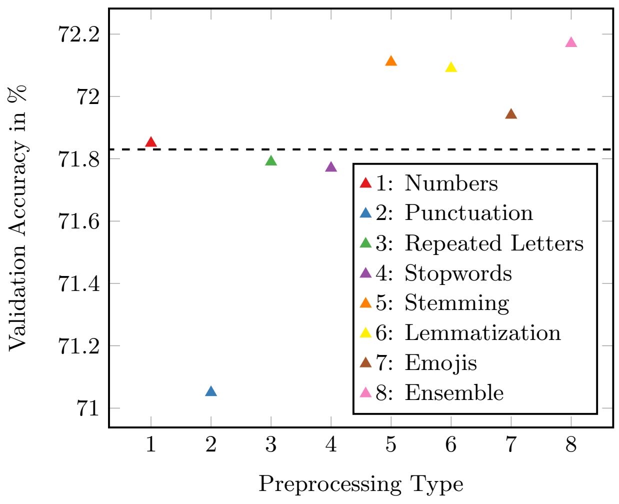

FIG. 2. Validation accuracies of different preprocessing techniques compared with no preprocessing (dashed line). The "Ensemble" technique (8) consists of the combined preprocessing techniques which improve validation results relative to no preprocessing, namely removing numbers, stemming, lemmatization and handling emojis.

Model	Validation Accuracy in %
Word2Vec + Lin. Class.	79.47 ± 0.09
FastText + NN	81.13 ± 0.105
Word2Vec + NN	81.89 ± 0.52
Integrated + NN	**85.53** ± 0.070

TABLE III. Average validation accuracies of Baseline models for various word embeddings.

with an integrated and a Word2Vec embedding layer. All scores are very similar, lying in a range of 86.47% - 87.64%. The best performing model uses a bidirectional GRU with a Word2Vec embedding layer (BiGRU + Att). Its accuracy is 2.11% higher compared to the highest baseline score of 85.53 % (Table III).

Model	Validation Accuracy in %	
	Integrated	Word2Vec
CNN + Att	86.88 ± 0.063	86.82 ± 0.034
LSTM	87.53 ± 0.025	86.47 ± 0.074
BiGRU + Att	87.52 ± 0.036	**87.64 ± 0.057**
3-BiLSTM	87.51 ± 0.063	87.33 ± 0.101
CNN + LSTM + Att	87.264 ± 0.062	87.202 ± 0.037

TABLE IV. Average validation accuracies of individual models using both integrated and Word2Vec embedding layers.

D. Classifier Ensemble

Table V shows the mean validation accuracies when combining the ten individual models as stated in Table IV using various ensemble methods. For the individual models integrated embedding layers perform better, whilst the accuracy of the ensemble methods is increased when both models with integrated embedding and models with Word2Vec are combined. The highest performing ensem-

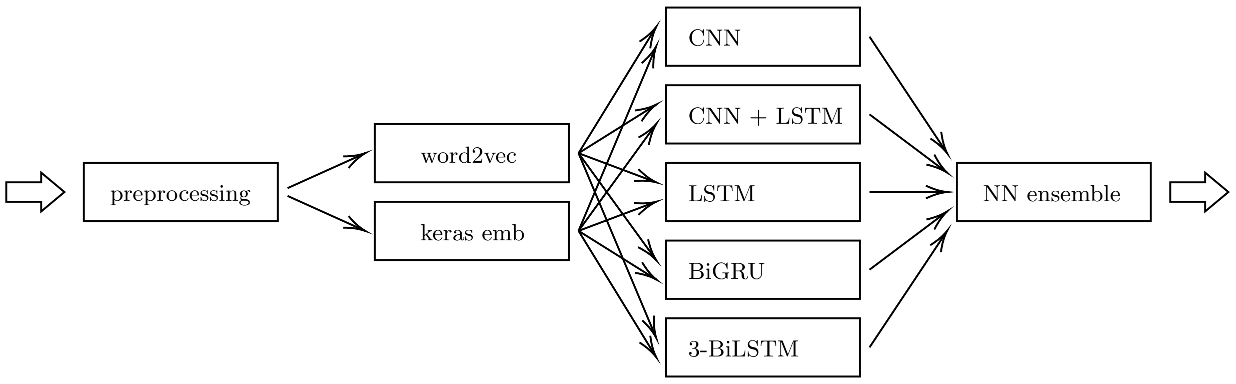

FIG. 3. Architecture of final ensemble method.

ble method is a neural network with fully connected layers. By combining the models, we were able to gain an additional 0.78% in accuracy. The structure of our best performing model is shown in Figure 3.

Ensemble Method	Validation Accuracy in %
Majority Vote	88.28 ± 0.036
Weighted Majority Vote	88.36 ± 0.024
Random Forest	88.17 ± 0.017
NN	**88.42** ± 0.022

TABLE V. Average validation accuracies of combined models using various ensemble methods.

E. Computational Resources

We trained all individual models in parallel on 20 CPUs, the final ensemble model was computed in less than 24 hours. For none of the models did the training exceed 8 GB of memory.

IV. DISCUSSION

This project aimed to develop high performing classification algorithms on a twitter dataset for sentiment analysis using reasonable computational resources. In general, we used well-established approaches including the Word2Vec embedding, CNN and RNN architectures, and combined these in a novel way. We tested various deep learning architectures by combining commonly used layers such as LSTMs and Convolutional layers. These architectures performed significantly better than a simple neural network with only fully connected layers.

Results indicate that all five individual models, including CNN and RNN architectures, perform similarly in terms of accuracy. Nonetheless, our novel ensemble solution was able to achieve a higher score than all individual models and by extension the baseline models. This suggests that CNNs and RNNs are exhibiting different feature importances which although they do not show differences in individual model accuracies, they do offer advantage when training with an ensemble method. Another key result of our study is that a more complex ensemble method using neural networks outperforms standard majority voting.

By using well-supported libraries including Gensim, Keras, and Scikit-learn, our code became less prone to implementation mistakes and also optimized for time and memory complexity. We further adjusted the dimensions, hyperparameters and architectures to reduce computational resources which allowed the final model to train in less than 24 hours. This runtime is significantly lower than other state-of-the art results [14].

We are aware of pre-trained models from the literature with higher accuracies than ours including BERT [16] and ELMo [27]. However, they require substantially more computational resources than our models in both memory and runtime, in part due to their significantly larger dimension size of 1024.

V. CONCLUSION

We analyzed a twitter dataset containing 2.5 million tweets to perform text sentiment classification (two classes). We used Word2Vec and Keras self-trained word embeddings, multiple deep learning models with CNN, LSTM, GRU and attention layers, and used their individual results to train a neural network. The individual model accuracies were significantly outperformed by our novel ensemble method. The best standalone model performed 2.11% better than a classical neural network (85.53%) and combining multiple architecturally complex, but small-sized models together further pushed accuracy to 88.42%. Overall we were able to develop a high performing classification algorithm using relatively small computational resources.

REFERENCES

[1] Bernard J Jansen et al. "Twitter power: Tweets as electronic word of mouth". In: *Journal of the American society for information science and technology* 60.11 (2009), pp. 2169–2188.

[2] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. "Thumbs up? Sentiment classification using machine learning techniques". In: *arXiv preprint cs/0205070* (2002).

[3] Vinod Nair and Geoffrey E Hinton. "Rectified linear units improve restricted boltzmann machines". In: *ICML*. 2010.

[4] Mika V Mäntylä, Daniel Graziotin, and Miikka Kuutila. "The evolution of sentiment analysis—A review of research topics, venues, and top cited papers". In: *Computer Science Review* 27 (2018), pp. 16–32.

[5] Armand Joulin et al. *Bag of tricks for efficient text classification*. 2016. arXiv: `1607.01759`.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[7] Tomas Mikolov et al. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: `1301.3781`.

[8] Jose Camacho-Collados and Mohammad Taher Pilehvar. "On the role of text preprocessing in neural network architectures: An evaluation study on text categorization and sentiment analysis". In: (2017). arXiv: `1707.01780`.

[9] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (2014). arXiv: `1408.5882`.

[10] Kai Sheng Tai, Richard Socher, and Christopher D Manning. "Improved semantic representations from tree-structured long short-term memory networks". In: (2015). arXiv: `1503.00075`.

[11] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[12] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks". In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[13] Kyunghyun Cho et al. *On the properties of neural machine translation: Encoder-decoder approaches*. 2014. arXiv: `1409.1259`.

[14] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems*. 2017, pp. 5998–6008.

[15] Artaches Ambartsoumian and Fred Popowich. "Self-Attention: A Better Building Block for Sentiment Analysis Neural Network Classifiers". In: *Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*. Brussels, Belgium: Association for Computational Linguistics, Oct. 2018, pp. 130–139. DOI: `10.18653/v1/W18-6219`.

[16] Jacob Devlin et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2018. arXiv: `1810.04805`.

[17] Federico Magliani et al. *A Comparison between Preprocessing Techniques for Sentiment Analysis in Twitter*. 2016.

[18] Yanwei Bao et al. "The Role of Pre-processing in Twitter Sentiment Analysis". In: *Intelligent Computing Methodologies*. Springer International Publishing, 2014, pp. 615–624.

[19] Dwarampudi Mahidhar and N V Subba Reddy. *Effects of padding on LSTMs and CNNs*. 2019. arXiv: `1903.07288`.

[20] Wenpeng Yin et al. "Comparative Study of CNN and RNN for Natural Language Processing". In: *CoRR* abs/1702.01923 (2017). arXiv: `1702.01923`. URL: `http://arxiv.org/abs/1702.01923`.

[21] Yoshua Bengio Dzmitry Bahdanau Kyunghyun Cho. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *ICLR 2015* (2015). arXiv: `1409.0473v7`.

[22] Nadia da Silva, Eduardo Hruschka, and Estevam Hruschka. "Tweet Sentiment Analysis with Classifier Ensembles". In: *Decision Support Systems* 66 (2014).

[23] Jan Deriu et al. "SwissCheese: Sentiment Classification Using an Ensemble of Convolutional Neural Networks with Distant Supervision". In: *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*. The Association for Computer Linguistics, 2016, pp. 1124–1128.

[24] Radim Řehůřek and Petr Sojka. "Software Framework for Topic Modelling with Large Corpora". English. In: *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*. `http://is.muni.cz/publication/884893/en`. Valletta, Malta: ELRA, May 2010, pp. 45–50.

[25] François Chollet et al. *Keras*. `https://keras.io`. 2015.

[26] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[27] Matthew E. Peters et al. "Deep contextualized word representations". In: *CoRR* abs/1802.05365 (2018). arXiv: `1802.05365`. URL: `http://arxiv.org/abs/1802.05365`.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Text Sentiment Cassification on Twitter Data

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Möhring	Katja
Börsig	Katharina
Frankovska	Zuzana
Beestermöller	Judith

With my signature I confirm that
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zürich, 28.7.2020	[illegible]
	[illegible]
	Frankovska
	Beestermöller

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.