CIL Semester Project: *Sentiment Analysis of Twitter Data*

Thomas Nilsson, Niko Leskinen, Anna Schmidt-Rohr, Selena Pepic
Team: *Grad Student Descent*
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—Using sentiment analysis to evaluate the vast amount of data available on online microblogging sites such as Twitter, is very valuable when gauging public opinion, assessing consumer opinions and monitoring brand reputation [1]. In this work, we focus on the binary classification of tweets as having positive or negative sentiment. We implement a few baseline models, as well as multiple variations of Recurrent Neural Network (RNN) based models. Our results demonstrate that minimal data pre-processing a bi-directional RNN based model using a pooling layer and the self-attention mechanism outperforms all other implemented individual classifiers. Furthermore, using an ensemble of the various RNN models we achieve an average accuracy of 88.43% on the two Kaggle test sets.**

I. Introduction

Sentiment analysis is the study of computationally identifying and categorizing opinions and attitudes that are expressed in natural language. The reduced time and cost of computationally determining the sentiment of texts makes the use of sentiment analysis beneficial for different fields such as business, politics, and finance. Some applications that have been studied include: determining brand reputation [2], analyzing how voters opinions on political parties and politicians [3], detecting the trend of prices of commodities [4] and assessing financial risk [5].

Twitter, is a social networking site, where short messages (limited to 280 characters) called tweets can be shared with other Twitter users. With over 330 million active monthly users as of the first quarter of 2019 [6], Twitter supplies an unequaled public collection of opinions about any topic of interest. The tweet length restriction forces the users to stay focused on the message they wish to convey and thus tweets make good candidates for the task of sentiment analysis. However, the informal and specialized language used in tweets poses a challenge.

This paper studies the use of machine learning techniques for the binary sentiment classification task of labeling tweets as positive or negative. We present multiple Recurrent Neural Network based models which incorporate various layers and mechanisms and lastly compare their performances to classical machine learning models such as Logistic Regression and Naive Bayes. We also implement an ensemble network of five RNN models, which reaches a competitive accuracy of 88.4%.

II. Data

A. The Twitter Dataset

The given dataset consists of 2.5M tweets in total and contains an equal number of tweets from the positive and negative class. Tweets that originally included a ':)' smiley were considered to be positive tweets and tweets that included a ':(' smiley were considered negative. These smileys were removed from the tweets and used as their label. The given data had already undergone some pre-processing beforehand: all text was converted to lowercase and URLs and usernames were replaced by the tags `<url>` and `<user>`, respectively.

B. Text Pre-processing

Inspired by [7], we experimented with various forms of text pre-processing such as (1) removing digits and punctuation, (2) marking smileys, elongated words and punctuation repetitions (3) removing stopwords, (4) stemming words, (5) applying rudimentary negation detection, and (6) removing duplicate tweets. However, these data-cleaning techniques unlike the papers suggest did not significantly improve performance, and in fact, led to a performance decrease in most cases. Thus, we chose to use dataset without applying such pre-processing.

C. Word Embeddings

Common for many of the models is the use of word embeddings, which is a way of representing words numerically using high dimensional vectors. Example implementations of word embeddings are Google's Word2Vec [8] and Stanford's GloVe [9]. In this report we will consider the GloVe technique and compare the embeddings of the implementation from the CIL course with the pre-trained embeddings available from Stanford [1]. For the purpose of comparability, we use the highest available dimension of the Stanford embeddings, which are 200 dimensions, for both embedding implementations. For the deep learning models, which also rely on word embeddings, we used the pre-trained embeddings, as these have been trained on 2.7 billion tweets and thus improve the classification accuracy of the models by a substantial margin.

[1] https://nlp.stanford.edu/projects/glove/

D. Neural Network Data

The deep neural network models require the input data to be numerical and of the same size. Therefore, standard natural language pre-processing techniques were applied. The vocabulary was limited to the 20,000 most used words, as these makeup over 95% of the words contained in the complete training set. Words that were not included in the vocabulary were replaced with a common <unk> token. The original Stanford embedding matrix contains 1.2M rows, thus by the implemented vocabulary reduction, the embedding matrix is reduced to $\frac{1}{60}$th of the original size. The tweets were clipped at length 40 and shorter tweets were left-padded to length 40 with the <pad> token. In total, tweets with length ≤ 40 make up 99.95% of the dataset and thus not a lot of data is lost through this compromise.

The two tokens <unk> and <pad> were added to the vocabulary, with the vectors for these two tokens being initialized at random, drawn from a normal distribution.

III. Sentiment Models

A. Baseline Models

1) Logistic Regression with Glove Embeddings: This model is one of the simplest baseline models for the sentiment analysis task and has no context of grammatical structure. Each tweet is represented as an average of the embeddings of all the words in the tweet, which exist in the vocabulary. Words that are not contained in the vocabulary are disregarded. On the resulting tweet embedding, we perform logistic regression.

Both the self-trained embeddings as well as the pre-trained Stanford embeddings were considered for this model.

2) Naive Bayes Classifier: We implemented several variations of Bernoulli and Multinomial Naive Bayes classifiers taking into account unigrams, bigrams and trigrams. We experimented with different performance optimizing techniques such as additive smoothing and the term frequency-inverse document frequency (tfidf). However, we reached the best test accuracy among Naive Bayes variants with a simple model, where we treated each tweet as a bag of trigrams and used add-one smoothing. We also padded the beginning and the end of the tweet with <bos> and <eos> tokens to give them greater contextual value. For example the tweet *longer words are really breakin my back now* would be turned into the following set of 3-tuples:

3-tuple		
<bos>	<bos>	longer
<bos>	longer	words
longer	words	are
	⋮	
my	back	now
back	now	<eos>
now	<eos>	<eos>

After that, each 3-tuple is treated as a word in the usual Naive Bayes classifier [10].

B. Neural Network Models

While traditional machine learning approaches rely on manually constructed features that the researcher or engineer deems important, deep learning approaches learn important features of the dataset, without having been explicitly programmed to do so. Deep learning approaches are very effective at learning language features and have been found to be superior to manual machine learning approaches for language tasks [11], [12].

The neural network models considered in this report make use of the GloVe word embeddings, in order to process each word. The corresponding word vectors representing each word are trained using backpropagation, which is why the number of entries in the matrix matters, since each entry is adjusted during training. The advantage of using pre-trained embeddings is that models then converge towards the optimal parameter values in much fewer epochs and because they were trained with a much larger dataset, the representations capture the semantic relationships more precisely. The embedding matrix consists of $20,000$ vectors, each having a dimension of 200, meaning the total number of entries is 4 million.

An advanced form of a neural network is the *Recurrent Neural Network (RNN)*, a type of neural network which excels at learning from sequential data such as natural language. An RNN is made up of cells containing an input, output and hidden state for each time step, time step being the length of the sequence being traversed. There are two commonly used architectures for RNNs, the Long Short-Term Memory (LSTM) [13] and the Gated Recurrent Unit (GRU) presented by Cho et al. [14] in 2014. Both of these architectures help to remove information no longer needed from the sequence being analyzed and emphasizing information likely to be needed later on, and in essence learning to pay *attention* to the important parts of the sequence. They accomplish this through the use of gating units, which control the flow of information into and out of the computational units in the network layers. These gated units have their own set of weight-parameters which are applied sequentially on the to the context. The GRU is computationally less demanding and has shown similar performance as the LSTM on machine translation tasks, where long-term dependencies within sequences play a central role in capturing the semantic meaning [15].

Common for RNN models is the use of the GRU cell with cell-size 256, it was also tried to run the best with cell-size 512 but this resulted in lower accuracy.

1) Basic RNN: Inspired by the model proposed in Jurafsky, Page 186, [16], this model uses a single RNN layer and a dense output layer and is a very basic network for analyzing sequential data.

2) Bi-directional RNNs: Combining RNNs is also possible, such as stacking many RNNs in layers, and feeding the output of one layer into a new layer. Another option is *bi-directional RNNs* (bi-RNN's), which traverse the sequence

from left to right and afterwards from right to left. By applying this method, network models have been capable of understanding deeper grammatical structure and dependencies such as referring back to things mentioned earlier in the sentence [17].

We also implement dropout regularization to reduce overfitting and thus improve the generalization of the our bi-RNNs. Dropout causes individual nodes to be dropped out of the network with probability $1 - p$ in each training stage. Through experimentation we found that the best results were achieved when we added dropout with $p = 0.3$ only directly after the Embedding layer. All our bidirectional-RNN models include this identical dropout mechanism.

Neural network models that use of dimension reduction layers such as pooling, convolution or attention have been employed with great success for similar tasks [18].

Convolution applies a kernel filter, which enables the extraction of local features unlike (bi)-RNN's that capture long-term dependencies. Because tweets are very short adding a convolutional layer on top of our (bi)-RNN seemed promising.

We also use pooling, to avoiding the output being oversensitive to the location of features in the input. We achieve the best results when we concatenate the output of the two most popular ways for summarizing the presence of features in patches: (1) max pooling, which returns the most activated one and (2) average pooling, which summarizes the average presence of the feature [19].

Attention mechanisms are used to identify which words in a sentence that are most relevant. Self-attention is a recent variant of an attention mechanism, which has shown promising results in sentiment analysis tasks [20]. On a high-level it can be explained as allowing each word in a sequence to pay attention to other words in a sequence independently of their positions, therefore focusing more on the important segments.

3) Ensemble of (Bi-directional) RNNs: Previous research [21], [22] indicates that instead of using the single neural network with the highest validation accuracy for classification, performance improves if one uses an ensemble network that combines multiple neural networks. Thus, we implemented an ensemble of the five implemented neural networks (Basic RNN, Bi-RNN with a convolutional layer, Bi-RNN with a pooling layer, Bi-RNN with an attention mechanism and Bi-RNN with and attention and Pooling layer), where each neural network was trained on the same dataset, and the predicted values from the base models where combined using majority voting.

C. Implementation and Training

All models were implemented in Python3 and the deep learning models were written using Keras [23], due to its concise syntax and ease of use. The models were trained on the Leonhard cluster from ETHZ using the Nvidia GTX 1080 GPU using a batch size of 256 and a learning rate of 0.001.

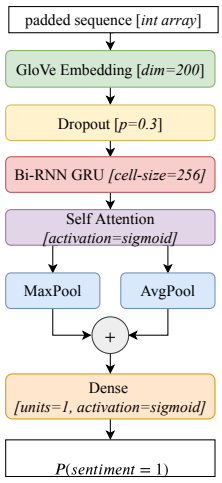


Figure 1. The architecture of the highest performing Recurrent Neural Network-based model which uses self-attention as well as max- and mean-pooling before the final dense layer.

In order to evaluate parameters and compare models without relying on the online Kaggle test set, for which the number of daily submissions was limited, the given dataset was split into a training- and validation set, using a 90:10 ratio. To ensure consistent results, the splitting was done using indices such that all models trained and evaluated on exactly the same tweets.

All deep learning models were trained for 3 epochs with the default learning rate of 0.001, using the Adam optimizer. Using smaller learning rates and training for more epochs, while increasing the training accuracy, did not result in an increase of the validation accuracy.

IV. Results

In this section, we present our results by comparing our proposed deep-learning models with the baselines. Our empirical results are summarized in Table I. For each model, we report the accuracy obtained on the validation set and on the public and private Kaggle data splits, as well as the number of parameters of the model.

As shown in Table I, all our (bi)-RNN models clearly outperform the baselines. The best baseline model, the Naive Bayes model, achieves an accuracy of 81.66%, while the worst deep-model achieves an accuracy of 87.68%. The highest accuracy, 88.54%, was achieved with the *Bi-RNN-Ensemble* model. However, the *Bi-RNN-Ensemble* does not outperform the other implemented deep-learning models by a large margin. Even the most simple implemented deep-learning model, an RNN with 4.35M parameters, achieves an accuracy that is only less than 1% worse than the accuracy of the *Bi-RNN-Ensemble*.

Model	Params	Val	Test (public)	Test (private)	Test (mean)
Logistic-Regression-CIL-GloVe	201	70.66	66.86	71.66	69.26
Logistic-Regression-Stanford-GloVe	201	77.82	76.88	77.10	76.99
Naive-Bayes (Bag-of-trigrams)	-	83.40	81.62	81.66	81.64
Basic-RNN-256	4.35M	88.10	87.74	87.68	87.71
Bi-RNN-256-Convolution	4.77M	88.26	87.28	87.44	87.36
Bi-RNN-256-Pooling	4.70M	88.24	87.62	88.14	87.88
Bi-RNN-256-Attention	4.75M	88.20	88.16	87.76	87.96
Bi-RNN-256-Attention-Pooling	4.74M	88.23	88.22	88.48	88.35
Bi-RNN-512-Attention-Pooling	6.26M	88.27	88.12	87.96	88.04
RNN-Ensemble	24.83M	88.73	88.32	88.54	**88.43**

Table I
ACCURACIES (%) ON THE VALIDATION AND TEST SETS AND NUMBER OF PARAMETERS OF ALL MODELS. IT CAN BE OBSERVED THAT THE PERFORMANCE OF THE *RNN-Ensemble* MODEL IS SLIGHTLY BETTER THAN THE *Bi-RNN-256-Attention-Pooling*.

	Basic RNN	Bi-RNN Conv	Bi-RNN Pool	Bi-RNN Att	Bi-RNN Att-Pool-512	Ensemble	Log-Reg CIL-GloVe	Log-Reg SU-GloVe	Naive Bayes
Simple-RNN	-	94.09	94.33	94.13	93.97	95.50	70.70	80.85	83.59
Bi-RNN-Convolution		-	93.96	93.86	93.82	97.27	70.99	80.70	84.24
Bi-RNN-Pooling			-	94.20	94.74	96.69	71.45	80.94	85.82
Bi-RNN-Attention				-	94.26	95.13	71.15	80.96	84.02
Bi-RNN-512-Attention-Pooling					-	95.31	71.33	80.86	85.10
Bi-RNN-Ensemble						-	71.04	81.01	84.89
Logistic-Regression-CIL-GloVe							-	72.67	74.71
Logistic-Regression-Standford-GloVe								-	79.04
Naive Bayes									-

Table II
PAIRWISE CLASSIFICATION AGREEMENT (%) BETWEEN MODELS
MODELS ARE DISPLAYED IN THE SAME ORDER IN HEADER COLUMN AND ROW. (ABBREVIATIONS ARE USED DUE TO SPACE LIMITATIONS)

V. DISCUSSION

Our results show that recurrent neural network models clearly outperform the baseline models, which are based on the simpler *bag of words* assumption. This indicates that the words used in the tweet are very important but word ordering also matters quite a lot, which is further confirmed by the fact that the Naive Bayes model improves from 72% to 79% to 81%, when setting the length of the n-grams to 1, 2 and 3, respectively. As for the RNN models, the basic model without dropouts, pooling or attention layers performs quite well, given that it has the lowest parameter count out of any of the deep learning models, meaning that parameter count for this task is not a significant indicator of potential precision.

The *Bi-RNN-Ensemble* network showed a slight improvement over the best single RNN model for the average test accuracy, and the *Bi-RNN-256-Attention-Pooling* model shows the best trade-off between model complexity and performance. Although it has a marginally lower performance than the ensembled networks, it only utilizes $\frac{1}{5}$th of the parameters of the *Bi-RNN-Ensemble* model. This lack of improvement may be due to the fact that classification errors of the individual base models tend to coincide [24]: For the models used in our ensemble each pairwise classification agreement percentage was above 93.8% [Table II]. However, while independence between individual classifiers is typically viewed as an asset when creating ensembles, for some tasks it has also been shown that an ensemble consisting of dependent classifiers, can offer a significant improvement in accuracy over the individual classifiers as well as reducing variance of the outputs [25]. Thus, an ensemble approach for this task is justified and could be developed further, preferably with models of a different neural architecture.

An interesting observation we made is that adding and moving dropout layers had a large impact on the performance of our neural networks. Therefore, in future work, experimenting further with dropout and perhaps implementing *dropConnect* [26], where weights in the network are set to zero instead of activations, might further improve performance.

Furthermore, the performance of our classifiers may also be limited by the given training data: We found that a number of seemingly neutral words (eg. ”complete, ”frame”, ”white”, ”edition”) appeared almost exclusively in the training data of the negative tweets. This suggests that improvements in data acquisition could improve the performance of our models.

REFERENCES

[1] R. Feldman, "Techniques and applications for sentiment analysis," *Commun. ACM*, vol. 56, p. 8289, 04 2013.

[2] N. Azizah Vidya, M. I. Fanany, and I. Budi, "Twitter sentiment to analyze net brand reputation of mobile phone providers," 11 2015.

[3] S. Sharma and N. P. Shetty, "Determining the popularity of political parties using twitter sentiment analysis," in *Information and Decision Sciences*, S. C. Satapathy, J. M. R. Tavares, V. Bhateja, and J. R. Mohanty, Eds. Singapore: Springer Singapore, 2018, pp. 21–29.

[4] J. Smailović, M. Grčar, N. Lavrač, and M. Žnidaršič, "Predictive sentiment analysis of tweets: A stock market application," in *Human-Computer Interaction and Knowledge Discovery in Complex, Unstructured, Big Data*, A. Holzinger and G. Pasi, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 77–88.

[5] C.-J. Wang, M.-F. Tsai, T. Liu, and C.-T. Chang, "Financial sentiment analysis for risk prediction," in *Proceedings of the Sixth International Joint Conference on Natural Language Processing*. Nagoya, Japan: Asian Federation of Natural Language Processing, Oct. 2013, pp. 802–808. [Online]. Available: https://www.aclweb.org/anthology/I13-1097

[6] Twitter, Inc., "Selected company metrics and financials," April 2019. [Online]. Available: https://s22.q4cdn.com/826641620/files/doc_financials/2019/q1/Q1-2019-Selected-Company-Metrics-and-Financials.pdf

[7] G. Angiani, L. Ferrari, T. Fontanini, P. Fornacciari, E. Iotti, F. Magliani, and S. Manicardi, "A comparison between preprocessing techniques for sentiment analysis in twitter," 12 2016.

[8] T. Mikolov, K. Chen, G. S. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: http://arxiv.org/abs/1301.3781

[9] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[10] J. H. Martin and D. Jurafsky, *Speech and language processing: An introduction to natural language processing, computational linguistics, and speech recognition*. Pearson/Prentice Hall Upper Saddle River, 2009.

[11] Y. Goldberg, "A primer on neural network models for natural language processing (2015)," *CoRR abs/1510.00726*.

[12] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*, May 2017, pp. 350–354.

[13] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 1997.

[14] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *CoRR*, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[15] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014. [Online]. Available: http://arxiv.org/abs/1412.3555

[16] D. J. . J. H. Martin, *Speech and Language Processing*, draft of september 23, 2018 ed., 2018. [Online]. Available: https://web.stanford.edu/~jurafsky/slp3/

[17] A. Mousa and B. Schuller, "Contextual bidirectional long short-term memory recurrent neural network language models: A generative approach to sentiment analysis," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Valencia, Spain: Association for Computational Linguistics, Apr. 2017, pp. 1023–1032.

[18] L. N. Smith and N. Topin, "Deep convolutional neural network design patterns," *CoRR*, vol. abs/1611.00847, 2016. [Online]. Available: http://arxiv.org/abs/1611.00847

[19] C.-Y. Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," 09 2015.

[20] A. Ambartsoumian and F. Popowich, "Self-attention: A better building block for sentiment analysis neural network classifiers," 2018.

[21] L. K. Hansen and P. Salamon, "Neural network ensembles," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 10, pp. 993–1001, Oct. 1990. [Online]. Available: http://dx.doi.org/10.1109/34.58871

[22] D. Ito and T. Wadayama, "Sparse signal recovery for binary compressed sensing by majority voting neural networks," *CoRR*, vol. abs/1610.09463, 2016. [Online]. Available: http://arxiv.org/abs/1610.09463

[23] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[24] C. Ju, A. Bibaut, and M. van der Laan, "The relative performance of ensemble methods with deep convolutional neural networks for image classification," *Journal of Applied Statistics*, vol. 45, no. 15, pp. 2800–2818, 2018. [Online]. Available: https://doi.org/10.1080/02664763.2018.1441383

[25] L. Kuncheva, C. Whitaker, C. Shipp, and R. Duin, "Limits on the majority vote accuracy in classifier fusion," *Pattern Analysis & Applications*, vol. 6, no. 1, pp. 22–31, Apr 2003. [Online]. Available: https://doi.org/10.1007/s10044-002-0173-7

[26] L. Wan, M. Zeiler, S. Zhang, Y. L. Cun, and R. Fergus, "Regularization of neural networks using dropconnect," in *Proceedings of the 30th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, S. Dasgupta and D. McAllester, Eds., vol. 28, no. 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1058–1066. [Online]. Available: http://proceedings.mlr.press/v28/wan13.html

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):

CIL Semester Project: Sentiment Analysis of Twitter Data

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Nilsson	Thomas
Leskinen	Niko
Schmidt-Rohr	Anna
Pepic	Selena

With my signature I confirm that

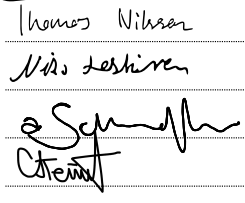
- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

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

Place, date

July 5h 2019, Zürich

Signature(s)



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