Sentiment Analysis of Movie Reviews: A Comparison of Architectures

Fabian Schneider, *Technische Hochschule Nuremberg* and Lisa Branz, *Technische Hochschule Nuremberg*

***Abstract*—The field of sentiment analysis of real-world text data has received considerable attention in recent years with a number of different machine learning models developed for this purpose. In this paper, we implement and compare two different pre-existing approaches for sentiment analysis on the Large Movie Review dataset. We implement a Long Short-Term Memory (LSTM) architecture as well as a Convolutional Neural Network (CNN) architecture in order to build a machine learning model to classify sentiment. In our experiments, CNN outperformed LSTM in terms of accuracy as well as training time. We then proceed to build our own architecture for sentiment analysis by adapting the concept of Gated Linear Units (GLU) for sentiment classification and building a Gated Convolutional Neural Network (GCNN). In our experiments, the GCNN outperforms the LSTM architecture in regards to training and test accuracy as well as training time but cannot compare to the CNN architecture. Some limitations of our approach as well as directions for future research and improvement are discussed.**

***Index Terms*—Sentiment Analysis, Machine Learning, RNN, GLU, CNN, GCNN**

I. Introduction

THE area of sentiment analysis has been getting a lot of attention in recent years due to its wide variety of application fields. However, as with many other Natural Language Processing (NLP) tasks the automated extraction of valuable information from pieces of written unstructured or semi-structured text remains a challenge. Not only is real-world data often noisy, but gauging the polarity of a text document is not a trivial task due to the ambiguity of natural language. Nonetheless have machine learning approaches made considerable progress in recent years when it comes to sentiment classification. In this paper, we compare three machine learning approaches towards sentiment analysis with respect to performance. In the following section, we will give an overview over sentiment analysis as well as the machine learning techniques and architectures used for the task.

A. Sentiment Analysis

Sentiment analysis is a method for the automated extraction of expressions of positive or negative attitudes or specific emotions from written language[1]. It is an area of NLP, which explores how natural language text or speech can be understood and manipulated using computers to produce a valuable outcome [2]. While sentiment analysis can also be used on audio and video data [3], it is currently mostly used to detect sentiment in written text. In recent years sentiment analysis has been widely applied to capture individuals sentiment towards entities such as people, products or movies or to assess the overall sentiment expressed in a piece of text. Due to the wealth of user-generated data available on internet platforms such as IMDb they have quickly become a frequently used source when it comes to developing machine learning models for sentiment analysis. Several approaches towards sentiment analysis have been developed throughout recent years that can mostly be classified as supervised, unsupervised or hybrid methods. Supervised approaches such as Naive Bayes, Maximum Entropy and Support Vector Machines are based on machine learning techniques applied to train a classifier using a set of training data and a set of test data [4]. Unsupervised approaches are widely based on lexicons containing the polarity of a variety of words, word combinations or n-grams. A variety of sentiment analysis tools and algorithms are currently available using one or a combination of the approaches mentioned above. Depending on the purpose, classification can focus on analyzing sentiment on a coarse-grained level labelling a piece of text as positive, neutral or negative, or a more fine-grained level, capturing specific emotions such as joy, fear or disgust. In the present paper, we compare approaches for classifying sentiment on a coarse-grained level based on the labels included in the dataset in order to keep the task simple and comparable between architectures.

F. Schneider is with the Department of Computer Science at the Technische Hochschule Nuernberg Georg Simon Ohm.
E-Mail: schneiderfa78094@th-nuernberg.de
L. Branz is with the Department of Computer Science at the Technische Hochschule Nuernberg Georg Simon Ohm.
E-Mail: branzli56977@th-nuernberg.de

B. Machine Learning Techniques for Sentiment Analysis

In this paper, we compare several machine learning architectures that can be used to perform sentiment analysis. In the following section, we will give an overview over the machine learning techniques used in the approaches presented and compared in this study.

1) Recurrent Neural Networks: RNNs are characterized by the existence of feedback loops from neurons of one layer to other neurons of the same or a preceding layer. This structure is usually used to discover temporally coded information in data. RNNs are dedicated sequence models that maintain a vector of hidden activations that are propagated through time [5]. This type of architecture is particularly suitable for machine translation and language modeling tasks and has therefore gained popularity in these fields of application in recent years [5]. The basic principle is the existence of a hidden state that can represent all previously seen information

in the sequence. As basic RNNs are comparatively hard to train, the use of LSTM [6] and Gated Recurrent Unit (GRU) [7] architectures is more common.

2) Convolutional Neural Networks: While a conventional neural network, e.g. in the form of a multi-layer perceptron (MLP), requires a vector as input, a CNN is capable of processing input in the form of a matrix. Conventional neural networks consist of fully or partially connected neurons arranged in multiple layers. These structures reach their limits when it comes to image or language processing. The CNN consists of different layers and is based on the principle of a partially locally connected feedforward net. The convolutional layer is followed by one or several pooling layers. Since the pooling layer and the convolutional layer are locally connected subnets, the number of connections in these layers remains limited even with large input quantities. They are followed by a fully-connected layer. CNN architectures have been applied to sequences for years, with fields of application including speech recognition, Natural Language Processing (NLP) tasks and, in recent years, sentence and document classification, machine translation and language modeling [5].

II. Related Work

In 2014, Kim presented an approach using CNNs for sentence-level classification tasks such as sentiment analysis [8]. In his work, CNNs are built on top of Google's word2vec vectors [9]. By conducting unsupervised pre-training of word vectors, a CNN with one convolution layer achieved remarkable results. For the basic variant of the model, a filter is developed that will produce a new feature when applied to a window of a given number of words in a sentence. A feature map is then created by applying the filter to each possible window of words in a given sentence. By conducting max-pooling, the maximum value can be determined, yielding the most important feature while naturally dealing with varying sentence lengths. The described model contains a number of filters forming a separate layer, to which dropout is added for regulization. This layer passes its data on to the following fully-connected softmax layer which then yields the eventual classification as probabilities across labels. The second approach included in this comparison of architectures is the LSTM architecture developed by Hochreiter and Schmidhuber [6], which is a technique for significantly increasing the performance of RNNs. An LSTM unit commonly consists of a cell, a multiplicative input gate protecting memory contents from perturbation by irrelevant inputs, a multiplicative output gate protecting other units from perturbations and a forget gate, controlling the extend to which a value remains in the cell i.e., is forgotten. This structure allows the implementation of basic memory functionality as the cell remembers values over time intervals and the gates regulate the flow of information into and out of the cell. LSTM architectures are suitable for tasks such as sentiment analysis as they can not only process single data points but are also suited for processing sequential data. The authors use a network consisting of one input layer, one hidden layer and one output layer, with the hidden layer containing the memory cells and their respective gate units. The approach developed in the paper at hand is based on the work of Dauphin et al. [10] and was adapted for sentiment analysis tasks. The authors present a language modeling approach using Gated Convolutional Networks. The finite context approach is realized through stacked convolutions, which allow parallelization over sequential tokens. While RNNs typically use recurrent connections, the authors instead make use of gated temporal convolutions. Although the context size is finite, the architecture achieves high performance on language modeling tasks, demonstrating that infinite context size is not necessary in order to achieve state-of-the-art results. The authors develop Gated Linear Units (GLUs), a simplified gating mechanism for non-deterministic gates that mitigates the vanishing gradient problem by coupling linear units to the gates. Using this approach, the non-linear capabilities of the layer can be retained while allowing the gradient to propagate through the linear unit without scaling. In comparison to LSTM-style gating mechanisms, GLUs allow for faster convergence to better perplexities. GLUs are not to be confused with Gated Recurrent Units (GRUs) [11]. Although they also aim at solving the vanishing gradient problem, GRUs exhibit structural similarities with LSTM with the main difference being the lack of an output gate in GRUs.

III. Data Sample

The data sample used for all experiments conducted in this paper was the Large Movie Review Dataset for binary sentiment classification [12]. The dataset consists of 50000 movie reviews from the movie rating platform IMDb annotated with the polarities "positive" and "negative", containing a balanced number of reviews from both categories. The dataset contains a maximum of 30 reviews per movie and considers only highly polarized reviews. The sample comes pre-split into a training and test set containing 50% of the data points each. The dataset has been widely used in a number of previous studies (e.g., [13], [14], [15]) as a base for training models for sentiment analysis. This allows for a comparison between different architectures for classifying sentiment in text.

IV. Method

In our work we compare two commonly used language models for sentiment analysis as well as a third recently introduced model from [10]. In order to obtain comparable results, all three models are implemented in their most basic form. The language models used in this work consist of blocks of computations, which are then often stacked on top of each other to form a deep neural network of their respective specific architecture. In order to ensure comparability in this work, we only used models consisting of one layer of each of the three architectures. This might not yield the state-of-the-art performance for the specific architectures and for the task given this dataset, but allows for an insightful comparison of their basic performance. In this paper we try to adopt the language model introduced by [10] which replaces the recurrent connections typically used in RNNs, and adapt it to the task of sentiment classification of the Large Movie Review Dataset [12]. RNNs f produce a continuous representation of

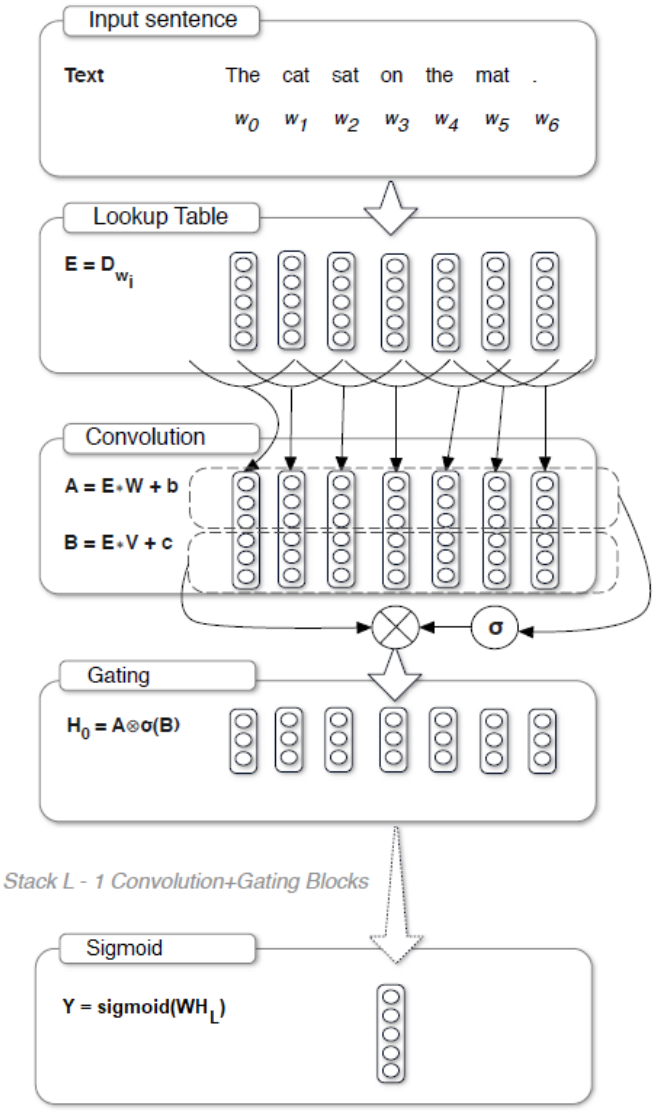

Fig. 1. The GCNN architecture as described in [10] adapted for sentiment analysis. The input sentence is converted into integer numbers and passed to the lookup table which obtains the individual embeddings for the words. They are fed into a convolution with bottleneck structure (see 5 for details). In our implementation we use normal convolutions, meaning that the filter is centered over the center word. Consider a filter with size $k = 5$ centered on the word **sat**, then the words **The**, **cat**, **sat**, **on** and **the** would be convolved. In contrast, the authors of [10] use causal convolutions, which means zero padding the sequence with $k - 1$ zeros so the filters right border convolves exactly the center word and its $k - 1$ preceding words. The convolved features are then modulated through the gating mechanism introduced by [10]. See figure 2 for more details.

a word and its preceding words through a recurrent function $h_i = f(h_{i-1}; w_{i-1}$ which is an inherently sequential process that cannot be parallelized over i. The proposed method of [10] convolves the inputs with a function f to obtain $H = f * w$. Because H depends only on the current context and has no temporal dependencies, it is easier to parallelize over the individual words of a sentence than its recurrent counterparts. This process will compute each context as a function of a center word w_i and its surrounding words $[w_{i-seq_len}; w_{i+seq_len}]$.

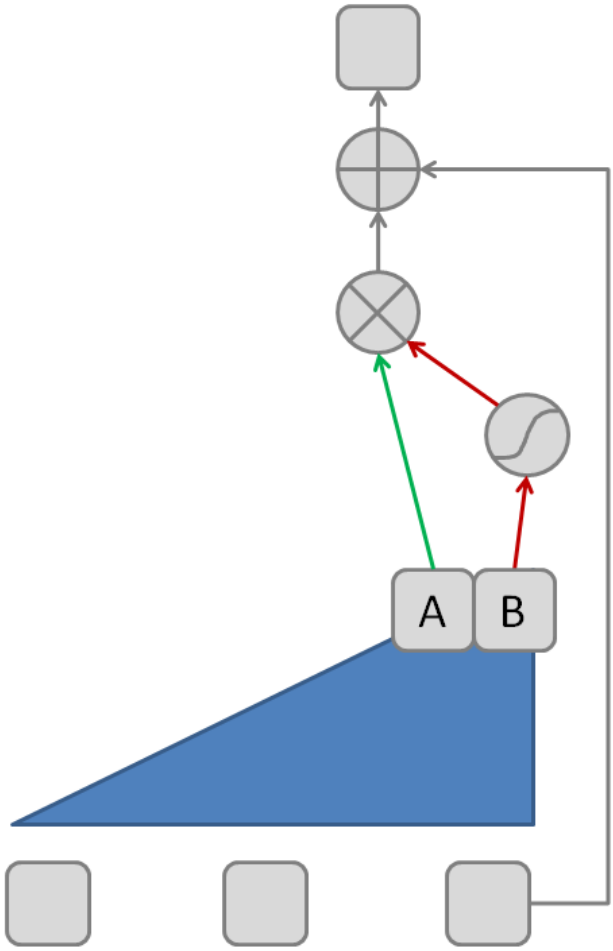

Fig. 2. GLU with residual skip connection. Figure adapted from [16].

Figure 1 illustrates the model architecture. Words are represented as dynamic vector embeddings, i.e., the embeddings are obtained during training through backpropagation into the embeddings. These embeddings are stored in a lookup table $D^{|V| \times e}$ where $|V|$ is the number of words in the vocabulary and e is the embedding size used for the vector representation of each word in the vocabulary. The input to our model are the words $w_0, \ldots, w_N$ which are represented as a sequence of word embeddings $E = [D_{w_0}, \ldots, D_{w_N}]$. The hidden layer h_0 is computed as $\mathrm{h}_0(X) = (X * W + b) \otimes (X * V + c)$ where m, n are the number of input and output feature maps, respectively. The size of the convolutional kernel is described as k, $X \in R^{N \times m}$ is the input of layer h_0 (the word embeddings), $W \in R^{k \times m \times n}$, $b \in R^n$, $V \in R^{k \times m \times n}$, $c \in R^n$ are the learned parameters, σ is the sigmoid function and $\otimes$ is the element-wise product between matrices.

The output of the layer h_0 is a linear projection $X * W + b$ which is added to the output of the gates $\sigma(X * V + c)$. The latter function will, due to the use of the sigmoid function, output values between 0 and 1. These outputs are then element-wise multiplied with the output of $X * W + b$ and, therefore, the values of V and c control the information outputted by the layer h_0. The authors of [10] dub this mechanism GLU. Figure 2 illustrates the concept. Although it is possible to stack

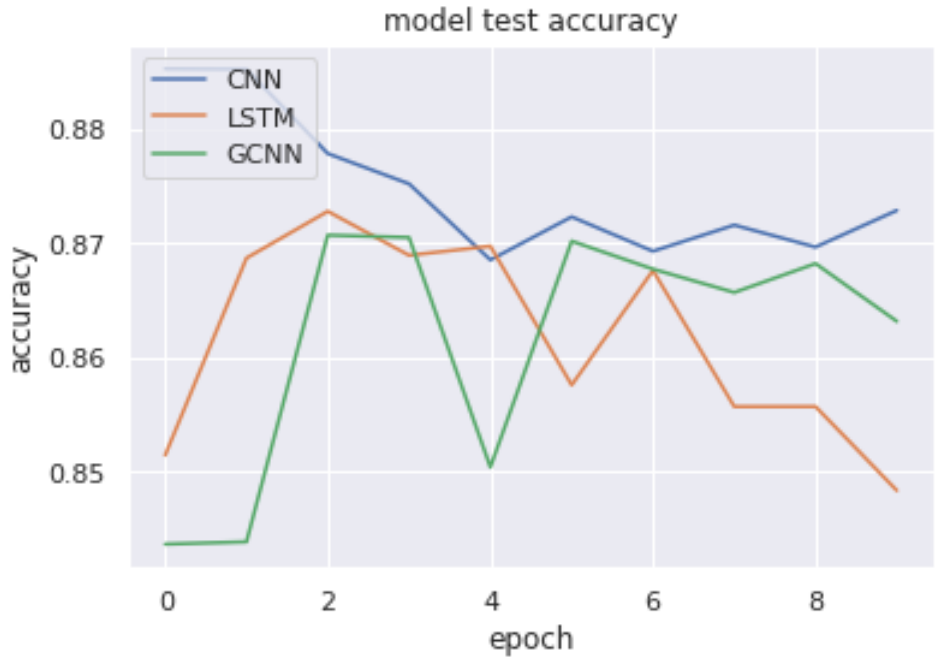

Fig. 3. Model test accuracy for all three architectures.

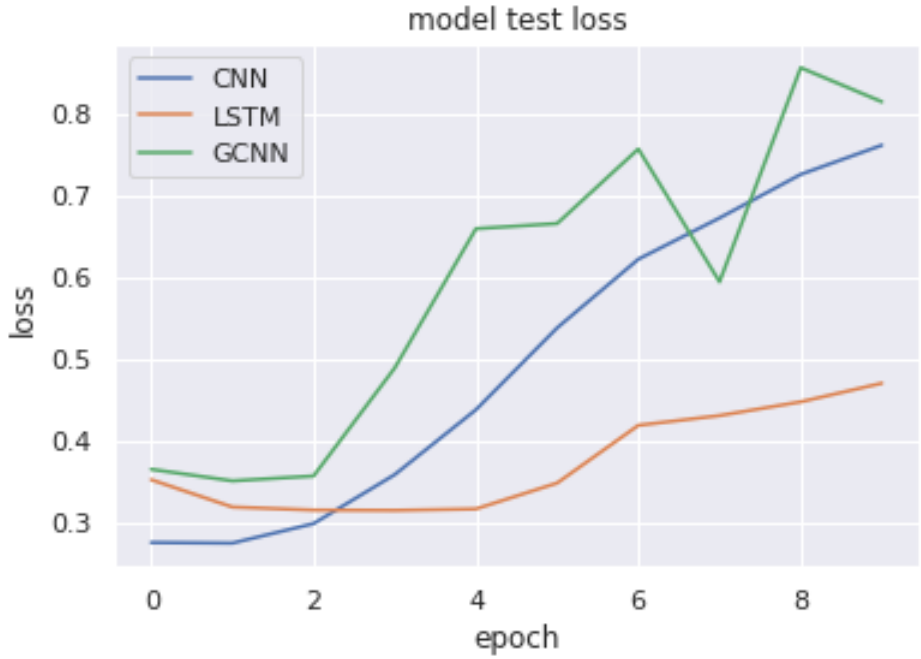

Fig. 4. Model test loss for all three architectures.

multiple GLU layers, only one is used in the work at hand. The layer is wrapped in a residual block that adds the input of the GLU to the output. The GLU block has a bottleneck structure for computational efficiency.
To obtain the final output of the model, i.e., a prediction for positive or negative sentiment of a sentence, we use the sigmoid function which is then used to obtain the binary crossentropy as a optimization function.
Gating mechanisms are used to control the path through which the information in the network flows and are most prominently used in RNNs [6]. LSTMs can model long-term memory via a seperate cell controlled by input and forget gates. These gates enable the network to selectively remember or forget information, which is passed or blocked to flow further through the network. Without these gates, information could easily vanish through the transformations of each time step, which is known as the vanishing gradient problem. CNNs don't suffer from the vanishing gradient problem and [10] find that they do not require forget gates. For this reason CNNs could only benefit from output gates, which allow the network to control what information should be outputted by the layer. This is particularly useful for language modeling as it allows the model to select which word or feature is relevant for predicting the sentiment of the sentence.
Due to the large amount of data as well as past experience with long training periods, all experiments were conducted using Google Colaboratory [17] with a GPU as hardware accelerator.

V. Results

Figure 6 shows experimental results of all models as well as the epoch of best performance, validation loss measured as binary crossentropy loss on the test set, validation accuracy on the test set, the number of trainable parameters and the time to train the specific model for ten epochs. Figure 3 and Figure 4 show the accuracy and loss of all three models, respectively. It can be observed that the CNN achieves the overall best result in terms of accuracy and training time and already converges in the first epoch due to the large amount of training data for one epoch and the comparatively simple task at hand. Of all three models, the LSTM has the lowest number of trainable parameters (238,701), but requires, by far, the highest training time of 54.84 minutes due to its temporal dependencies. Although the CNN has a considerably higher number of trainable parameters (2,163,605), it only requires an impressive 0.33 minutes of training time. The main reason for the low training time of the CNN model is that this architecture has no temporal dependencies and is therefore highly parallelizable. The GCNN has, by far, the largest number of trainable parameters (80,410,833). Its training time of 11.0 minutes is, therefore, significantly longer than the training time of the CNN but still significantly shorter than the training time of the LSTM.
The LSTM neural network yields its highest performance in epoch five, achieving a validation accuracy 0.02 percent lower than the CNN's highest validation accuracy of 0.8852. With its overall best epoch being epoch 4, the GCNN outperforms the LSTM by a very narrow margin of 0.0005 percent in regards to test accuracy, but requires less training time, despite a much larger number of trainable parameters. The GCNN approach can, therefore, be considered more efficient than the LSTM approach without any loss of test accuracy.
In terms of loss it can be observed that all three models reach their best test loss already in epoch 1, which can be a result of the high amount of training data in combination with the relatively simple task of coarse-grained polarity classification.

VI. Limitations and future work

In this paper, we compare the performance of three different architectures for sentiment analysis of movie reviews. In the following section, we are going to discuss some limitations of the current study as well as give some directions for future research. Firstly, it has to be pointed out that a very rudimentary implementation of each architecture was chosen in this paper. While this increases comparability, it is important to note that higher performances can be achieved with more advanced implementations, as competitive results from Kaggle [18] suggest. Secondly, the GCNN approach might not be as well-suited to the task of sentiment analysis as other architectures. In this work, only a coarse-grained sentiment classification was conducted, capturing only the polarity of a review. As the complexity of tasks that can be solved by a neural network increases with the number of hidden layers and the task at hand was purposefully kept simple for comparability reasons, the GCNN architecture is already too complex for the task. The architecture might be better-suited for a more fine-grained analysis of sentiment, differentiating between different types of specific emotions such as joy, fear or sadness, as well as the intensity of the respective emotion. Moreover, the best results could so far be achieved with word2vec pretrained embeddings [8] [18]. While our comparison yields interesting insights about the performance of basic implementations of the three architectures, future research should take this advanced approach into account in order to further increase performance.

VII. Conclusion

In this paper, we compared three different architectures for sentiment analysis of movie review data in regard to their performance. We implemented a CNN, a LSTM and a GCNN in their most basic form for best comparability. To fairly study the different encoding capability of the inspected models, we trained from scratch, using neither pre-trained word embeddings nor complex tricks such as batch normalization in our setup. The models were trained on the large movie review dataset to predict the sentiment of a review, which could be either positive or negative.
From our experiments it can be observed that the CNN in its basic setup outperforms the other investigated models in terms of test accuracy, test loss and training time, which is mainly related to the high parallelizability of CNNs. The relatively new gated convolutional architecture performs slightly worse than the CNN with a higher number of trainable parameters, resulting also in a higher training time.

	Layer	Output Shape	Weights
Input	Input	(None, 500)	
	Embedding	(None, 500, 300)	
Convolution with Bottleneck	Downscale	(None, 500, 15)	(1, 300, 15)
	Convolution	(None, 500, 15)	(4, 15, 15)
	Upscale	(None, 500, 600)	(1, 15, 600)
Projection	Flatten	(None, 300 000)	
	Dense	(None, 256)	(300 000)
	Dense	(None, 1)	(256, 1)

Fig. 5. Configuration of the GCNN architecture adapted for sentiment analysis.

Type	Best Epoch	Validation Loss	Validation Accuracy	Trainable Parameters	Time in min.
CNN	1	0.2751	0.8852	2,163,605	0.33
LSTM	5	0.3163	0.8697	238,701	54.84
GCNN	4	0.4891	0.8705	80,410,833	11.0

Fig. 6. Experimental results of all models, epoch in which the best performance is measured, validation loss as binary crossentropy, validation accuracy (higher is better), the number of trainable parameters and the time to train for 10 epochs.

REFERENCES

[1] C. R. Fink, D. S. Chou, J. J. Kopecky, and A. J. Llorens, "Coarse- and fine-grained sentiment analysis of social media text," *Johns hopkins apl technical digest*, vol. 30, no. 1, pp. 22–30, 2011.

[2] G. G. Chowdhury, "Natural language processing," *Annual review of information science and technology*, vol. 37, no. 1, pp. 51–89, 2003.

[3] L.-P. Morency, R. Mihalcea, and P. Doshi, "Towards multimodal sentiment analysis: Harvesting opinions from the web," in *Proceedings of the 13th international conference on multimodal interfaces*. ACM, 2011, pp. 169–176.

[4] G. Vinodhini and R. Chandrasekaran, "Sentiment analysis and opinion mining: a survey," *International Journal*, vol. 2, no. 6, pp. 282–292, 2012.

[5] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[6] S. Hochreiter and J. Schmidhuber, "Lstm can solve hard long time lag problems," in *Advances in neural information processing systems*, 1997, pp. 473–479.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Gated feedback recurrent neural networks," in *International Conference on Machine Learning*, 2015, pp. 2067–2075.

[8] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[9] (2013) word2vec: Tool for computing continuous distributed representations of words. [Online]. Available: https://code.google.com/archive/p/word2vec/

[10] . A. F. Y. N. Dauphin, M. Auli, and D. Grangier. (2016) Language modeling with gated convolutional networks. [Online]. Available: http://arxiv.org/pdf/1612.08083v3

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[12] A. L. Maas, R. E. Daly, P. T. Pham, D. Huang, A. Y. Ng, and C. Potts, "Learning word vectors for sentiment analysis," in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Portland, Oregon, USA: Association for Computational Linguistics, June 2011, pp. 142–150. [Online]. Available: http://www.aclweb.org/anthology/P11-1015

[13] S. Wang and C. D. Manning, "Baselines and bigrams: Simple, good sentiment and topic classification," in *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*. Association for Computational Linguistics, 2012, pp. 90–94.

[14] V. Narayanan, I. Arora, and A. Bhatia, "Fast and accurate sentiment classification using an enhanced naive bayes model," in *International Conference on Intelligent Data Engineering and Automated Learning*. Springer, 2013, pp. 194–201.

[15] V. Kharde, P. Sonawane *et al.*, "Sentiment analysis of twitter data: a survey of techniques," *arXiv preprint arXiv:1601.06971*, 2016.

[16] D. Bressler. (2018) How to build a gated convolutional neural network (gcnn) for natural language processing (nlp). [Online]. Available: https://mc.ai/how-to-build-a-gated-convolutional-neural-network-gcnn-for-natural-language-processing-nlp/

[17] Google colaboratory. [Online]. Available: https://colab.research.google.com

[18] Bag of words meets bags of popcorn. [Online]. Available: https://www.kaggle.com/c/word2vec-nlp-tutorial