# Sentiment Analysis Report

Daniyar Kultayev
*dept. of Computer Science*
*Nazarbayev Univercity*
Astana, Kazkahstan
email Daniyar.Kultayev@nu.edu.kz

Kamalkhan Artykbayev
*dept. of Computer Science*
*Nazarbayev Univercity*
Astana, Kazkahstan
email Artykbayev.Kamalkhan@nu.edu.kz

***Abstract*—Artificial Neural Networks are gaining increasing popularity, and are applied in variety of different fields. Particularly, in the area of Natural Language Processing, neural networks have replaced previously used rule based approaches. Moreover, current neural networks are capable of producing impressive results. Namely, different variations of CNNs and LSTMs are having great success in area of Natural Language Processing, Sentiment Analysis. While the field of NLP have moved on to more challenging problems, such as, Aspect Level Sentiment Analysis, the quality of available Datasets also increased. Thus, it raises the question, whether success of current technique is based solely on the methods themselves, or Dataset has its own influence towards the final result.**

***Index Terms*—Machine Learning, Natural Language Processing, Natural Language Understanding, RNN, LSTM, CNN**

## I. Introduction

Artificial Neural Networks have played key role in solving number of important problems in the past, and still are widely used in variety of different fields. Particularly, in the area of Natural Language Processing(NLP), recent successes allowed us to have efficient Speech Recognition, Language Translation, or even Sentiment Analysis. Although, we are very far from Natural Language Understanding, still, Sentiment Analysis is considered as a key insight towards achieving it. Thus, most of the current works related to Natural Language Understanding are focused on Sentiment Analysis.

Moreover, recent advancements in Sentiment Analysis using neural networks surpassed previously used Rule Based Grammar Analysis. For instance, neural network based approaches are far superior in terms of speed and accuracy, while they are capable of constructing Syntax Structure of a given sentence. Besides, neural network based approaches are much more flexible, and require significantly less human intervention. Not only that, but also, some of the neural network approaches are unsupervised in terms of training. That is to suggest that, they only need training data to be able to produce results.

Additionally, current advancements in neural networks provided means of analyzing sentences with even better precision. Not only they are capable of accurately determine the sentiment of a given sentence, they are also capable of Aspect Level Sentiment Analysis.

Aspect Level Sentiment Analysis as grown its popularity in past years, due to the thoroughness of the approach. That is, determining Aspects or Targets of the sentence and evaluating their sentiment based on a given context. In contrast with Sentiment Analysis of the sentence, Aspect Level Sentiment Analysis provides much more deeper evaluation of a sentence. Also, Aspect Level Sentiment Analysis methods, resemble human approach of evaluating sentiment of the sentence.

Besides, most of currently used neural networks adopt the fact that they have to be flexible. Moreover, due to significant differences in language variations, they reject the notion of incorporating statically defined rules. Thus, training process have to be data-driven and unsupervised/semi-supervised.

Therefore, two of the most common neural networks that fit the purpose considered to be: CNN and different variations of RNNs.

## II. Related Work

While Sentiment Analysis dates back to 2002 [14], we are still far from Natural Language Understanding. Moreover, sentiment classification seemed inadequate in cases were structure contained polar sentiments. Thus, it lead to proposition of Aspect Sentiment Classification [6] as an alternative. Though, admittedly, ASC is much more sophisticated and resource demanding. Nevertheless, with social media influence growth, commercial interest for opinion mining, and sentiment analysis has increased dramatically. Thus, making Sentiment Analysis and ASC as hottest topics in NLP currently.

Therefore, there have been number of attempts towards both Sentiment Analysis and ASC. While RNNs[5] date back to 1997, ASC and Sentiment Analysis are heavily dominated by them till this day [1][11][16]. Moreover, development of LSTMs lead to attention based approaches, where the sentiment of the aspect in determined based on current context[11][16]. Additionally, it has been showen that further accuracy improvements can be achieved by using Bidirectional LSTMs[15] or if language syntax is incorporated into LSTM model using Recurrent Neural Network Grammar[2].

Alternatively, the other method that is gaining popularity for aspect and sentence sentiment classification is CNN[9][13][7].

## III. proposed method

As it can be seen, LSTMs are far superior than anything while dealing with sequential data, and they are sophisticated compared to other approaches. Yet, at the same time, the quality of datasets that are currently used for training is also high. Thus, raising the question, if sophistication of the method is more influential than quality and size of training Dataset.

In order to draw that conclusion, it is suggested to pick same methods that are currently in use while they were used in the past for much simpler task of determining Sentiment for the hole sentence. Although, the only difference now would be the Datasets that would be used to train on. That is to suggest, currently available datasets are much higher in terms of quality, and if the quality of dataset plays any role in overall outcome, scaling back to simple sentiment analysis using above mentioned methods should vindicate the difference, if there is any.

Moreover, to make a case, two different datasets with different quality and size are going to be used for training.

As it was mentioned earlier, two most common neural networks that were used for sentence sentiment analysis and still used for aspect level sentiment analysis are, **CNN** and different variations of **RNNs**. In particular, performance of **CNN** and regular **LSTM** are going to be analyzed on different datasets after training both networks for 100 epoch.

### *A. CNN*

To begin with, **CNN** similar to [13] is going to be considered. That is, google word2vec [12] is going to applied for word transformation into embedding layer(48x300). Next, reshaped embedding layer is passed on into 3 parallel two dimensional constitutional layers(46-44x1x100). After that, each constitutional layer is connected to two dimensional pooling layer(1x1x100).finally, all 3 filtered features from all pools are concatenated(3x1x100) flattened(300) and passed through dropout layer

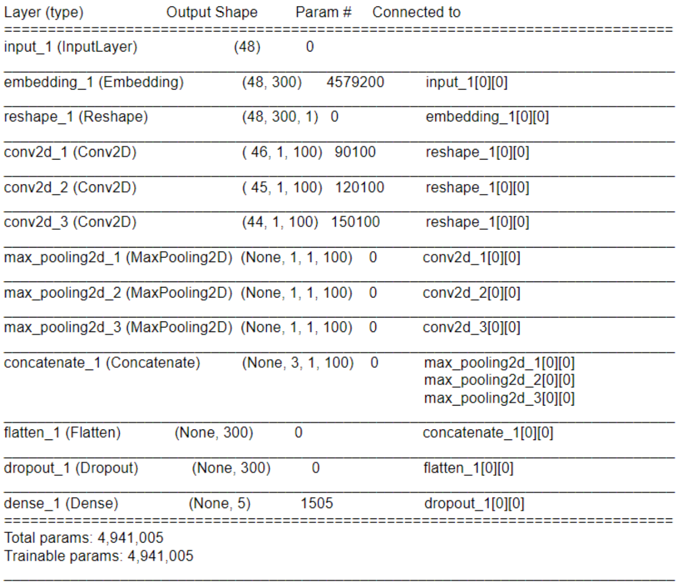
Also, due to its relatively simple and compact construction, CNN is considered not as demanding in terms of data volume and computational cost. Thus, it was expected, CNN should outperform LSTM given relatively small amount of data. Lastly, resulting CNN summary can be found on III-A, while the implementation code is available in submission folder,
**train\train_CNN_RT.py** for Rotten Tomatoes and
**train\train_CNN_TW.py** for Sentiment140 .

### *B. LSTM*

Alternatively, LSTM, have require much more data to train and computationally expensive. Although LSTMs training process is unsupervised/semi-supervised, thus, provides much more flexibility. In particular, Bidirectional LSTM, wich is going to be used, requires twice as much training and computational costs since it incorporates two layers. On the other hand, Bidirectional LSTM produces higher accuracy, since at each step, network has outputs of both layers, front and back sequence at its disposal.

Therefore, it was anticipated, that LSTM should perform better, given a larger dataset, even though it is much less annotated and is lower quality.

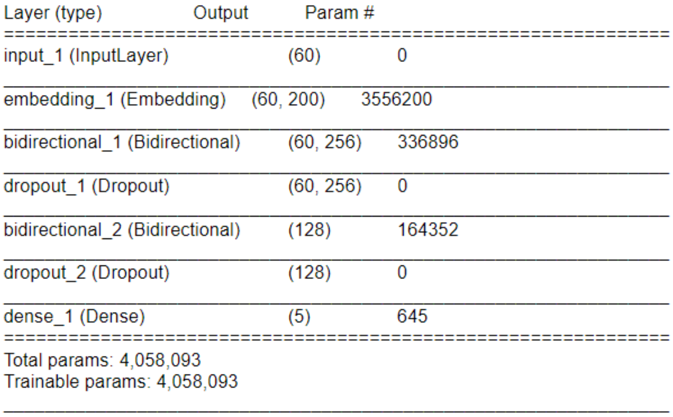
First, all trained data is fed to Keras Tokenizer to convert them from words to numeric representation. Next received tokens are provided to Embedding Layer(60x200). After that, Embedding Layer is fed to first Layer of Bidirectional LSTM(60x256). Finally, first Bidirectional LSTM is supplied into other directional Layer of Bidirectional LSTM(128).



```
Layer (type)                    Output Shape       Param #    Connected to
==========================================================================================
input_1 (InputLayer)                (48)            0
__________________________________________________________________________________________
embedding_1 (Embedding)             (48, 300)       4579200    input_1[0][0]
__________________________________________________________________________________________
reshape_1 (Reshape)                 (48, 300, 1)    0          embedding_1[0][0]
__________________________________________________________________________________________
conv2d_1 (Conv2D)                   ( 46, 1, 100)   90100      reshape_1[0][0]
__________________________________________________________________________________________
conv2d_2 (Conv2D)                   ( 45, 1, 100)   120100     reshape_1[0][0]
__________________________________________________________________________________________
conv2d_3 (Conv2D)                   (44, 1, 100)    150100     reshape_1[0][0]
__________________________________________________________________________________________
max_pooling2d_1 (MaxPooling2D)  (None, 1, 1, 100)   0          conv2d_1[0][0]
__________________________________________________________________________________________
max_pooling2d_2 (MaxPooling2D)  (None, 1, 1, 100)   0          conv2d_2[0][0]
__________________________________________________________________________________________
max_pooling2d_3 (MaxPooling2D)  (None, 1, 1, 100)   0          conv2d_3[0][0]
__________________________________________________________________________________________
concatenate_1 (Concatenate)         (None, 3, 1, 100)   0      max_pooling2d_1[0][0]
                                                               max_pooling2d_2[0][0]
                                                               max_pooling2d_3[0][0]
__________________________________________________________________________________________
flatten_1 (Flatten)           (None, 300)        0             concatenate_1[0][0]
__________________________________________________________________________________________
dropout_1 (Dropout)              (None, 300)        0          flatten_1[0][0]
__________________________________________________________________________________________
dense_1 (Dense)                  (None, 5)        1505         dropout_1[0][0]
==========================================================================================
Total params: 4,941,005
Trainable params: 4,941,005
__________________________________________________________________________________________
```

III-A

Fig. 1. CNN summary



```
Layer (type)                 Output        Param #
=================================================================
input_1 (InputLayer)               (60)          0
_________________________________________________________________
embedding_1 (Embedding)     (60, 200)      3556200
_________________________________________________________________
bidirectional_1 (Bidirectional)    (60, 256)     336896
_________________________________________________________________
dropout_1 (Dropout)                (60, 256)     0
_________________________________________________________________
bidirectional_2 (Bidirectional)    (128)         164352
_________________________________________________________________
dropout_2 (Dropout)                (128)         0
_________________________________________________________________
dense_1 (Dense)                    (5)           645
=================================================================
Total params: 4,058,093
Trainable params: 4,058,093
_________________________________________________________________
```

III-B

Fig. 2. LSTM summary

Additionally, summary of LSTM can be found at III-B, while code for LSTM is available in submition folder,
**train\train_LSTM_RT.py** for Rotten Tomatoes and
**train\train_LSTM_TW.py** for Sentiment140 .

### *C. Dataset*

While considering two datasets, key factors for selections were the fact that, datasets should be opposite and have clear distinctions. Namely, one of the datasets had to be relatively small, small to medium size, while preserving higher quality of sentiment to train on. Conversely, the other dataset should have been direct oposite of the first, so that, later on it would have been easier to make direct comparison. Thus, Rotten Tomatoes [8] and Sentiment140 [3] where selected for those intents.

| PhraseID | SentenceID | Sentence | Sentiment |
|---|---|---|---|
| 106 | 3 | a hard time sitting through this one | 1 |
| 107 | 3 | a hard time | 1 |
| 108 | 3 | hard time | 1 |
| 109 | 3 | hard | 2 |

TABLE I
SAMPLE FROM DATASET ROTTEN TOMATOES

I

| Sentence | Sentiment |
|---|---|
| don't let the sun catch you crying - oh my.. so cool | pos |
| the floor just bit my face. | neg |
| definitely smiling | pos |

TABLE II
EXERPT OF DATASET SENTIMENT140

II

*1) Rotten Tomatoes [8]:* Rotten Tomatoes , is a dataset, that consists of movie reviews from RottenTomatoes website. Movie reviews primarily are designed to express the sentiment towards the certain aspects of the movies.

As it was mentioned, one of the datasets had to be relatively small and should also have higher quality data, thus, Rotten Tomatoes [8] was considered for the following reasons:

1) Small-Medium size(156000x4 size)
2) Dataset preserves higher quality Sentiment
3) 5 Way Sentiment

Cross-referencing items 1 and 2.

Claim 2 comes from the fact that dataset provides sentiment for each individual element of the sentence. Additionally, example of the sentence from dataset with sentiment for each phrase of the sentence is provided in table I.

Also, 5 way sentiment encoding is following : 0 - negative, 1 - somewhat negative, 2 - neutral, 3 - somewhat positive, 4 - positive.

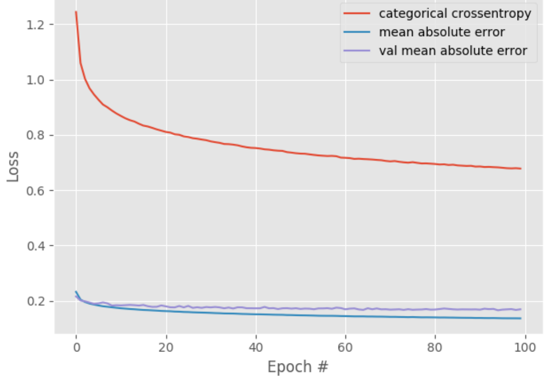
*2) Sentiment140 [3]:* Sentiment140 [3] is a dataset that was collected from twitter, by traversing trough large number of twitter posts. Though, twitter post are considered to have sentiments to certain extent.

Conversely, Sentiment140 [3], has direct opposite properties compared to Rotten Tomatoes [8]. First, as it can be seen from II, sentiment is only provided for the hole sentence. Moreover, sentiment is only binary. Second, size of Sentiment140 [3] dataset is much bigger in contrast with Rotten Tomatoes [8], although, much less annotated.

## IV. RESULTS

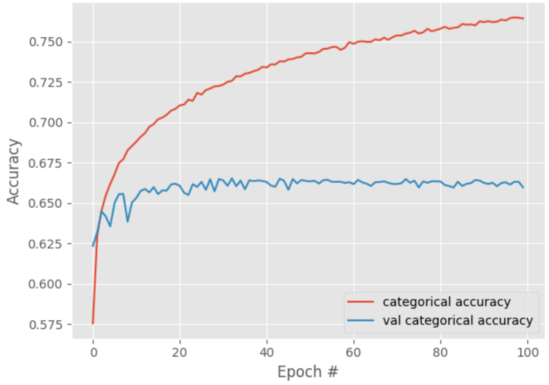
Categorical Accuracy, Categorical Cross-entropy and Mean Absolute Error(MAE) were used as key metrics for network performance measurements. Also, all of the measurements were obtained while performing tests on different data of the same dataset. For instance, CNN trained on Rotten Tomatoes was tested on separate data from same dataset, namely, Rotten Tomatoes , that was not used for training

Fig. 3. Categorical Crossentropy and MAE. CNN-RT



3

Fig. 4. Categorical Accuracy. CNN-RT



4

### A. CNN

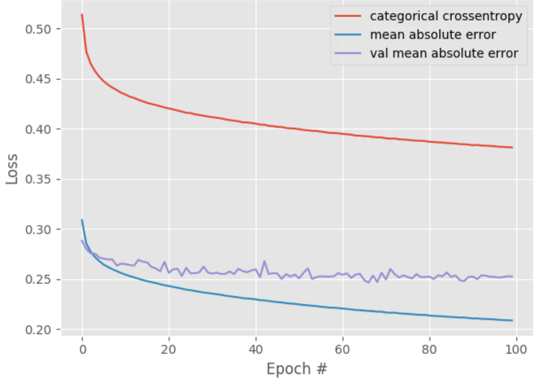
*1) CNN trained on Rotten Tomatoes :* the graph 4 shows, loss function CNN that was trained on Rotten Tomatoes , gradually descends reaching 0,7% by 100 epoch. Meanwhile, Categorical accuracy of CNN trained on Rotten Tomatoes , steadily grows, reaching approximately 0.77% accuracy by the same time.

*2) CNN trained on Sentiment140 :* Figures 5 , 6 suggest, that CNN trained on Sentiment140 descend from a much higher Categorical Cross-entropy value of about 0.55 and descends much slower up to only 0.38 by 100 epoch, compared to CNN trained on Rotten Tomatoes . Though, training accuracy of CNN trained on Sentiment140 , is similar to CNN trained on Rotten Tomatoes .

Overall performance of CNN trained using both datasets can be found in table IV As it can be seen from the table IV, overall accuracy of CNN trained on Rotten Tomatoes , even though, the size of the dataset is relatively small, is much higher, reaching around 0.7934%, while CNN trained on Sentiment140 only produced 0.6406 %.

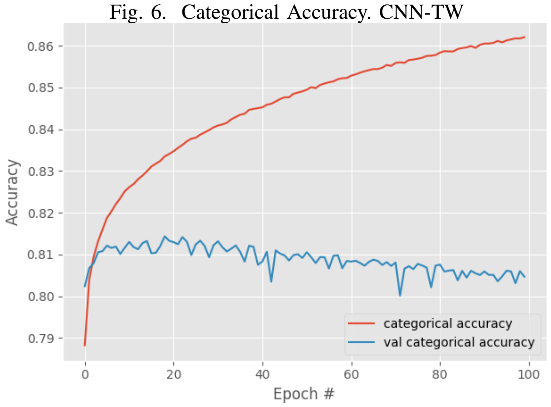
while scripts for training CNNs on Rotten Tomatoes and Sentiment140 can be found at
**train\train_CNN_RT.py**,
**train\train_CNN_TW.py**. Besides, resulting trained model can be found at [10] under **\model**.

Fig. 5. Categorical Crossentropy and MAE. CNN-TW



5

Fig. 6. Categorical Accuracy. CNN-TW



6

## B. LSTM

*1) LSTM trained on Rotten Tomatoes :* figures 7 and 7 illustrate, that Categorical Cross-entropy constantly declines through the hole period, starting from around 0.80% at 2 epoch, reaching about 0.18% by 100 epoch. At the same time, Categorical Accuracy steadily increases from about 0.65%, and flattening out at about 0.92% by 100 epoch.

*2) LSTM trained on Sentiment140 :* Pictures 9 and 10 show that LSTM trained on Rotten Tomatoes reached almost 0 Categorical Cross-entropy and MAE and about 0,98% training accuracy by 100 epoch. That is to suggest, that LSTM trained on Sentiment140 over-fitted. Although , the size of the Sentiment140 dataset is quite high, it is believed, due to poor quality of the dataset, LSTM was not able to capture sentence structures for further sentiment analysis, thus, resulting in an over-fitted model.

Summary performance of LSTM trained using both datasets can be found in table IV Numbers that can be found in table IV suggest, that, LSTM trained on Rotten Tomatoes produces tremendous results reaching nearly 0.94 % accuracy on tests. On the other hand, LSTM trained on Sentiment140 over-fitted, therefore, producing low test accuracy, that is only 54%

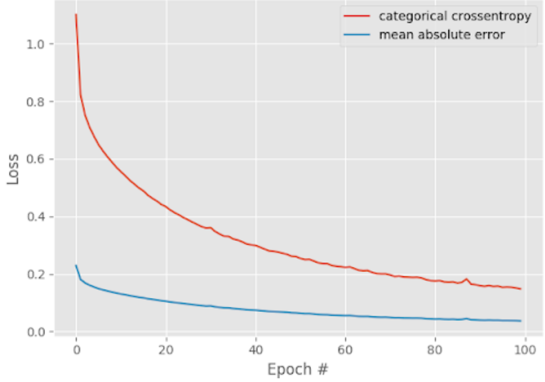
| Dataset | Accuracy | Categorical Cross Entropy | Mean Absolute Error | Test Accuracy |
|---|---|---|---|---|
| Rotten Tomatoes | 0.845 % | 0.5512 | 0.1009 | 0.7934 % |
| Sentiment140 | 0.879 % | 0.3845 | 0.2241 | 0.6406 % |

TABLE III
SUMMARY TABLE FOR CNN PERFORMANCE

III

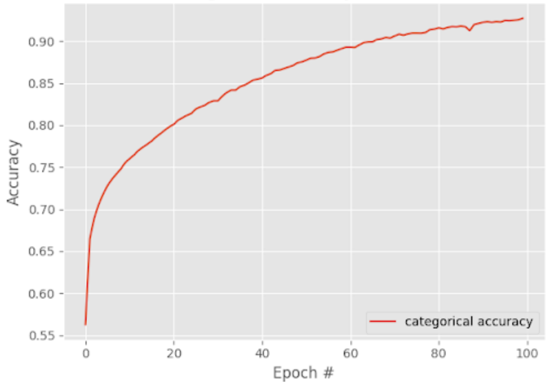
Similarly to CNN scripts for LSTMs training on Rotten Tomatoes and Sentiment140 can be found at **train\train_LSTM_RT.py**, **train\train_LSTM_TW.py**. Besides, resulting trained model can be found at [10] under **model**.

Fig. 7. Categorical Crossentropy and MAE. LSTM-RT



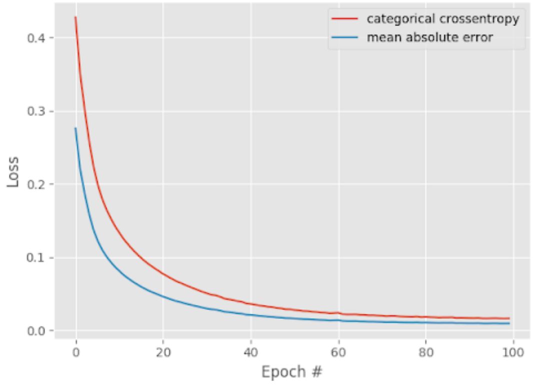
7

Fig. 8. Categorical Accuracy. LSTM-RT



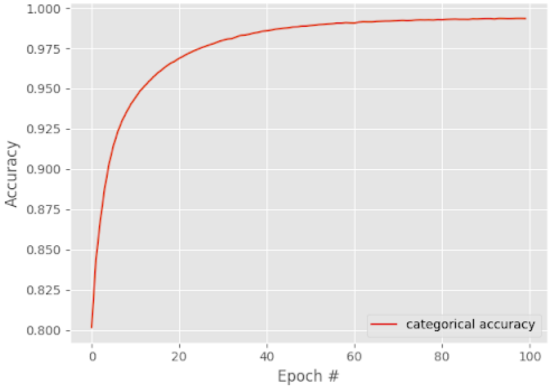
8

Fig. 9. Categorical Crossentropy and MAE. LSTM-TW



9

Fig. 10. Categorical Accuracy. LSTM-TW



10

| Dataset | Accuracy | Categorical Cross Entropy | Mean Absolute Error | Test Accuracy |
|---|---|---|---|---|
| Rotten Tomatoes | 0.945 % | 0.1464 | 0.0842 | 0.9389 % |
| Sentiment140 | 0.989 % | 0.0025 | 0.0012 | 0.5419 % |

TABLE IV
SUMMARY TABLE FOR CNN PERFORMANCE

IV

### C. Demo

Finally, to illustrate the final results fo trained models, small demo scripts were written up so that they can predict sentiment for a give sentences. Moreover, sources for demo scripts are located in **demo_CNN_RT.py**, **demo_CNN_TW.py**, **demo_LSTM_RT.py**, **demo_LSTM_TW.py**,

## V. Conclusion

To conclude, as it can be seen from table V, dataset had a higher impact on overall accuracy than the sophistication of the method, although additional complexity allowed further development of the area, enabling Aspect Level Sentiment Analysis.

| Model | Dataset | Accuracy | Categorical Cross Entropy | Mean Absolute Error | Test Accuracy |
|---|---|---|---|---|---|
| CNN | Rotten Tomatoes | 0.845 % | 0.5512 | 0.1009 | 0.7934 % |
| CNN | Sentiment140 | 0.879 % | 0.3845 | 0.2241 | 0.6406 % |
| LSTM | Rotten Tomatoes | 0.945 % | 0.1464 | 0.0842 | 0.9389 % |
| LSTM | Sentiment140 | 0.989 % | 0.0025 | 0.0012 | 0.5419 % |

TABLE V
Summary Table

V

# References

[1] B. Bohnet, R. McDonald, G. Simões, D. Andor, E. Pitler, and J. Maynez. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1246.

[2] C. Dyer, A. Kuncoro, M. Ballesteros, and N. A. Smith. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1024. URL https://www.aclweb.org/anthology/N16-1024.

[3] A. Go. Sentiment140, 2009. URL http://cs.stanford.edu/people/alecmgo/trainingandtestdata.zip. data provided by Go et al. [4].

[4] A. Go, R. Bhayani, and L. Huang. Twitter sentiment classification using distant supervision. *Processing*, pages 1–6, 2009. URL http://www.stanford.edu/~alecmgo/papers/TwitterDistantSupervision09.pdf.

[5] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

[6] M. Hu and B. Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014073. URL http://doi.acm.org/10.1145/1014052.1014073.

[7] B. Huang and K. Carley. Parameterized convolutional neural networks for aspect level sentiment classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1091–1096, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D18-1136.

[8] Kaggle. Sentiment analysis on movie reviews, 2015. data retrieved from Kaggle, https://www.kaggle.com/c/sentiment-analysis-on-movie-reviews.

[9] Y. Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1181. URL https://www.aclweb.org/anthology/D14-1181.

[10] D. Kultayev and K. Artykbayev. Sentiment analysis report, 2019. URL https://drive.google.com/drive/folders/1V8fLDlbyIJ6atqQyZT153QA9AeGWyT_C. report drive repository for all elements.

[11] A. Kuncoro, C. Dyer, J. Hale, D. Yogatama, S. Clark, and P. Blunsom. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1132.

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

[13] X. Ouyang, P. Zhou, C. H. Li, and L. Liu. Sentiment analysis using convolutional neural network. In *2015 IEEE International Conference on Computer and Information Technology; Ubiquitous Computing and Communications; Dependable, Autonomic and Secure Computing; Pervasive Intelligence and Computing*, pages 2359–2364, Oct 2015. doi: 10.1109/CIT/IUCC/DASC/PICOM.2015.349.

[14] B. Pang, L. Lee, and S. Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*, pages 79–86. Association for Computational Linguistics, July 2002. doi: 10.3115/1118693.1118704. URL https://www.aclweb.org/anthology/W02-1011.

[15] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *Trans. Sig. Proc.*, 45(11):2673–2681, Nov. 1997. ISSN 1053-587X. doi: 10.1109/78.650093. URL http://dx.doi.org/10.1109/78.650093.

[16] S. Wang, S. Mazumder, B. Liu, M. Zhou, and Y. Chang. Target-sensitive memory networks for aspect sentiment classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 957–967, Melbourne, Australia, July 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P18-1088.