Text Classification Project - CIL BULLS Team

Computational Intelligence Lab (Spring 2022)

Matteo Omenetti †§, Quynh Anh Nguyen*†§, Pitcho Oscar†§ and Francesco Di Stefano†§
† *Department of Computer Science, ETH Zurich, Switzerland*
* *Department of Computer Science, University of Milan, Italy*
Email: {momenetti, quynguyen, opitcho, fdistefano}@ethz.ch

Abstract—**The project aims to classify each tweet as positive or negative according to the sentiment expressed by the latter. This task differs from general sentiment analysis due to the complexity introduced by the presence of slang words and misspellings in 2.5 M tweets. We explored different methods starting from classical machine learning approaches, including SVM, and Randomforest on tf-idf word representation, to state-of-the-art methods using pretrained models and deep neural architecture with transfer learning ability. Our focus was to investigate existing classical and State-of-the-Ar methods to identify the respective models'e underlying strengths and weaknesses. Besides, we proposed TransformerLogReg, a new method that uses Logistic regression train on the prediction distribution resulting from different pretrained models. TransformerLogReg results in the best performance with 91.08% in the public test score, outperforming other implemented models. Besides, several valuable insights from other model experiments are also observed.**

I. Introduction

This report is part of the project for the course "Computational Intelligence Lab" hosted at ETH Zürich. The project's scope was to perform sentiment classification on a dataset made up of 2.5M tweets. Each tweet had to be classified as either having a positive or negative sentiment. The code to train and test all of the models described in this report can be found under "src/". All of the models have been trained (in a reasonable amount of time) using the GPUs offered by Google COLAB-Pro.

II. Related Work

Text classification is a classical task in NLP, yet it is non-trivial due to its wild application in diverse domains. Although deep neural models nowadays yield state-of-the-art performance, in many NLP tasks, shallow machine learning models still play an essential role in classifying corpus. In the research of Basarslan et al. [1], to analyze the sentiment of social media posts, TF-IDF and W2V[2], followed by machine learning methods, including Support Vector Machines [3] and Artificial Neural Networks [4], are experimented. Artificial Neural Network had the best accuracy performance. Palangi et al. [5] extensively investigated the use of RNNs, particularly LSTMS [6], for information extraction. The RNN [7] generates an embedding that puts similar sentences close together and diverges sentences far apart. Their results show that the models learn to pick up on key context words and attenuate stop-words importance when creating embeddings out of the sentences. Moreover, their results produce and embedding for further downstream applications. Regarding other non canonical models for tackling sentimental text analysis tasks, Transformers models based on the Attention mechanism represent the state-of-the-art. For example, BERT [8], RoBERTa [9], ELECTRA [10], or XLNet [11] are models built based on Transformers architecture that outperformed all other methods across multiple NLP tasks. In particular, the XLNet model of Z. Yang et al. achieved 96.21%, and 96.86% accuracy in binary sentiment analysis in both the IDBMs [12] and in the Stanford Sentiment Treebank dataset [11], which established the current state of the art in the sentiment classification task.

III. Dataset

The training dataset consists of 2.5M tweets that had to be classified as either having a positive or negative sentiment. The classes are balanced since each class has the same amount of examples, e.g. 1.25M tweets. An additional dataset of 10K tweets without labels are also provided for testing purposes. All tweets are tokenized, words are separated by a single white space, user mentions are replaced with <user> and links are replaced with <url>.

A. Exploration Data Analysis

Initial analysis shows that the average length of positive and negative tweets are 84 and 67 characters, respectively. Then, the most frequent words for each class were computed. The two classes share the majority of the most frequent words. Thus, models that do not consider context, for instance, TF-IDF, would not be enough to reach a good accuracy in this particular scenario.

B. Preprocessing

Text data is one of the most unstructured forms of available data, it contains noise in various forms. Two distinct preprocessing approaches were applied to the data fed to "standard" machine learning models and to transformers.

§Equal contribution

Preprocessing for standard models: Standard NLP word representation approaches, e.g. TF-IDF and W2V, are more affected by noise and less context-aware than transformers mechanism. Thus, we pre-process the data as the following steps: *Lowercasing*, *Tokenization*, *Hashtags removal*, *Stop-word removal*, *Lemmatization*, *Numbers removal*, *user and url removal*.

Preprocessing for Transformers: In order to handle a large corpus, Transformers-based models are much more resistant to noise and they are able to learn long-term dependencies. At first, the same preprocessing steps described above were applied. However, preprocessing step slightly decreased the performance of the model. Thus, we decided to use the data without any preprocessing step except *Lowercasing* for Transformers-related models.

IV. Methods

In this section, we will briefly describe different approaches that we exploited, varying from statistical methods to deep neural architectures to tackle the text classification task.

A. Machine Learning methods with TF-IDF

For classical machine learning techniques, scaled word-level TF-IDF features vectors were adopted. In figure 2 tSNE reduction was applied to show the TF-IDF embeddings of the sentences in the development set obtained from a 80/20 split f the training set. This visualization will be useful to justify some of the results of our models. Both linear and non-linear models were applied to the embeddings to observe the differences in their performance. As a linear model, a Support Vector Machine was chosen, while for the non-linear one, a Random Forest.

B. MLP with Word2Vec

Words are represented as vectors in an embedding space by word2vec [2], a neural model architecture . The length of each embedding vector is a hyper-parameter that must be chosen. First, this length was set to 300, which means that every word is represented using a vector with 300 entries. However, we noticed that even with a size of 220, the classifier achieved almost the same results, therefore using vectors of size 300 was not justifiable from a performance and complexity standpoint. Some experiments were run on the trained word2vec to see if it could grasp the semantic meaning of the words in the dataset. When asking the model for the most similar words to 'lol' the model output is: 'lmao', 'tho', 'lmfao', 'haha'. Finally, the 10 most similar words to "lol" were plotted along with 8 random words in two dimensions using t-SNE. As figure 1 shows the 8 random words are further away in the graph compared to the 10 most similar words. This is exactly what an embedding model tries to achieve, to have similar words physically close together in the learned embedding space. The sentence embedding of each training sample was obtained by averaging each word vector in the sentence. Finally, the obtained sentence embeddings were classified using an MLP.

C. Recurrent Neural Networks

The used RNN approach builds on top of section IV-B work. Whereas the word2vec method aggregates the feature vectors of each word with a sum, the RNN method aims to leverage positional information by composing the different word embeddings into a sequence through concatenation as described in figure 4a. For performance considerations, a batch size of 64 was used, with padding of sequences to ensure all vectors in the same batch are the same size. Conceptually this means a batch corresponds to a sequence of 64 matrices of varying width depending on the number of words in the corresponding tweet. When composing the hidden states for our final result, we discard the hidden states corresponding to placeholders as these could introduce noise in the predictions, the process is described in figure 4b. In this case, the tweet would be part of a batch where the max sequence length is 6, and all shorter elements are padded to be the same size. The RNN architecture uses a hidden state of dimension 300. Experiments showed that any further increase in the hidden state dimensionality had a negligible or even negative impact on the overall model accuracy.

Finally, this vector was decoded using *the same fully connected layers as in the W2V architecture*. This enables us to compare the embedding qualitatively directly through W2V summation and the embedding by RNN, which should (in theory) be able to leverage separate token information and their order within the sequence. For training, a typical training loop with a batch size of 64 and adam optimizer was used. Training data is shuffled between every epoch to ensure there is no risk of picking up on patterns in the dataset order. In addition, we used a validation set to avoid overfitting. Again to ensure comparability of results, the split between train and test data is identical across different models. In the end, the model could execute 10 epochs in 2.5 hours, thanks to GPU acceleration.

D. Transformers

BERT: BERT makes use of Transformer, an attention mechanism that learns contextual relations between words in a text. Since training a BERT model from scratch is too computationally expensive and requires too much data, a pre-trained model [13] from the HuggingFace library [14] was fine-tuned. We experimented by setting the sequence max_length hyper-parameter to 30 tokens to make sure not to lose any information. The first pretrained model we experimented with is the Huggingface *BERT-base-uncased*, i.e. a transformer model pretrained on a large corpus of English data without human labeling. We implemented models with different pretrained BERT and RoBERTa models, including *BERT-base-uncased*, i.e., a transformer model pretrained on a large corpus of English data without human labeling and *cardiffnlp/twitter-roberta-base-sentiment-latest*, model trained on 58M tweets and fine-tuned for sentiment analysis with the TweetEval benchmark.

Ensemble: Finally, having now different well-performing models we decided to try an ensemble approach. We summed up all of the output probabilities of the models and then we took the maximum.

E. TransformLogReg

We proposed the TransformLogReg method by applying the Logistic Regression model on the probability predictions resulting from different Transformers-based models. The model consists of three main stages including fine-tuning Transformer-based models, fine-tuning the Logistic Regression model, and predicting the final result. In this approach, we split train/val/test datasets with the corresponding size of $[2000000, 500000, 10000]$ tweets.

Initially, different Transformers-based models are fine-tuned using training data. We initialized models with three different Transformers-based architecture models. We used pre-trained models from different Transformer-based architectures including BERT, roBERTa and XLNet as in table II. After training models with the training set and classifying tweets of the validation data, we expect to achieve the probability distribution of each tweet from the validation set. These probabilities distribution predicted on the validation set are then concatenated and used to train the Logistic model in the second stage.

After that, the Logistic Regression model is trained using data resulting from the first step. Shape of each data point used to trained LogReg model is formed as $[p_{10}, p_{11}, p_{20}, p_{21}, p_{30}, p_{31}, gold]$, in which p_{i0}, p_{i1} are probabilities resulting from i^{th} Transformers-based model and $gold$ is the ground true label of the validation data.

In the final stage, the probability distributions resulting from fined-tuned Transformers models of testing data are concatenated. We used this data to feed the tuned Logistic models and achieve the final prediction.

V. Results and Discussion

The evaluations of the proposed approaches are taken into consideration in this section. Since the dataset classes are not imbalanced, the F$_1$ score and accuracy metrics are identical. We will sole use the validation accuracy and public test score, i.e. the evaluation considering half of the testing dataset which is published in Kaggle Leaderboard, as evaluation metrics.

A. Machine Learning method with TF-IDF

Linear models: Concerning the linear approach with TF-IDF embeddings, the application of SVM results in an accuracy of 66.9% on the validation set and 69.5% on the public test set. Figure 3a shows the confusion matrix related to the results of the SVM model on the validation set.

Non-linear models: To exploit the TF-IDF embeddings, we experimented with the Random Forest model, obtaining an F1 macro and an accuracy of about 81.7% on the development set and 80% on the public test set. Figure 3b shows the confusion matrix for the Random Forest model on the validation set.

Although the results obtained are not comparable with state-of-the-art deep neural models, we can observe the improvement in the performance of the Random Forest compared to the SVM. Our explanation for this relies on the observation of the TF-IDF embeddings of the validation dataset, shown in figure 2. In the space induced by the TF-IDF, the embeddings of positive and negative samples are not linearly separable, justifying the bad performance of the SVM. This remains a conjecture since we applied the tSNE reduction on high dimensional features vector with 440343 dimensions leading to information loss in the related visualization.

Model	**Val acc**	**Public test acc**
TF-IDF + SVM	0.6693	0.6954
TF-IDF + RandomForest	0.8175	0.8028
W2V + MLP	0.820	0.820
W2V + RNN + MLP	0.824	0.765
Transformers-based model	0.916	0.909
TransformerLogReg model	**0.9170**	**0.9108**

Table I: The best validation and test public scores of each approach.

B. MLP with Word2Vec

The chosen classifier is a standard MLP with 600k parameters. Different kinds of MLP architectures were used, bigger and smaller. Not huge differences were noticed due to the size of the architecture. We finally settled in with the one which was able to provide the best ratio performance/size. The MLP model was able to achieve an accuracy of 82% both on the validation and public test set. The training happened in a reasonable amount of time thanks to GPU [15] acceleration.

C. RNN

RNN yields an optimal accuracy of 82.4% on the validation set after 5 epochs of training under the cross-entropy loss. The overall performance of the model was only marginally improved compared to the results obtained through the MLP with the W2Vec method reported in the subsection V-B. Potential reasons that could explain these results are mostly linked to two following features: the nature of the data and the linear characteristics of the word2vec representation mechanism. Word2Vec provides an embedding with linear properties, e.g. when adding two positive words we find a point close to other positive words. Concretely this means that for positive tweets, summing up the word embedding leads to a vector associated with a

Adapting pretrained Model	Val acc	Test acc
xlnet-base-cased [16]	0.762	0.729
BERT-base-uncased [17]	0.90	0.891
bertweet-base-sentiment-analysis [18]	0.916	0.909
cardiffnlp/twitter-roberta-base-sentiment [19]	0.907	0.892

Table II: Transformers-based models evaluation

positive region. Our experience seems to show this general bag-of-words approach, thanks to the additive properties of the embedding, appears to be sufficient to capture the general tone of the tweet compared to the RNN approach, which leads to a 0.4% increase in accuracy.

D. Transformers

BERT model: By fine-tuning the head of the model BERT-base-uncased, we achieved an accuracy of 89% on the validation dataset. This model outperforms the models described above including SVM, Random Forest for TF-IDF, MLP for w2v, and RNN approaches. We then proceeded to fine-tune the *cardiffnlp/twitter-roberta-base-sentiment-latest* model which result an accuracy of 90% on the validation set. Having a fairly large dataset makes fine-tuning computationally expensive. However, the accuracy gain compared to standard approaches is significant. This makes the use of BERT models, in this particular project, justifiable from a performance standpoint.

Ensembling: Although the gained accuracy with the ensembling approach is not statistically significant, i.e. the models' performances are increased 0.5%, this is a potential approach to take advantage of multiple Transformers-based models. However, there is a trade-off between performance improvement and computational cost and we should take into consideration what is the priority when designing the experiments.

E. TransformerLogReg

Table II shows the evaluations of pre-trained models with further training and the proposed TransformerLogReg approach. TransformerLogReg results in the best performance with 91.62% in the validation set and 91.08% in the public test score, outperforming all implemented models. However, the evaluation of TransformLogReg only shows a slight improvement compared to single models. The improvement in performance of the TransformerLogReq model proved the advantages of combining the Transformers models and classical machine learning approach as well as making the prediction much more robust because of the double training architecture design.

In the first stage, finetuned Berttweet-based model yields the best performance in three pretrained-model with 91.6% of validation accuracy and 90.9% on the public test score. Similar to models described in section V-D, we set the max length of each tokenized tweet to 40 tokens for *bertweet-base-sentiment-analysis* and *cardiffnlp/twitter-roberta-base-sentiment* model and 128 tokens for *xlnet-base-cased* model. Although XLNet currently achieved state-of-the-art performance in the sentiment classification task, it yields the lowest accuracy of the three models. There are two reasons that can explain why BERTtweet outperformed XLNet in our experiments. First, since the computational resources are limited, i.e. we used Google Colab Pro with 25GB RAM, it is not allowed to set the max length of XLNet to be longer than 128 tokens while XLNET required a much longer sequence because of its *SentencePiece* tokenization mechanism. Second, the text domain of pre-trained models could significantly reason affecting the models' performances. In particular, *bertweet-base-sentiment-analysis* pre-trained model used text from Twitter to train while *xlnet-base-cased* pre-trained model used Wikipedia and book corpus as training resources. Besides, Transformer-based models are extremely expensive in training time. 2M tweets are tokenized in approximately 30 minutes and each epoch was trained in 210 minutes for XLNet and 110 minutes for the other two models.

VI. Conclusion

This paper presented the findings of the CIL BULLS team in handling the Text Classification problem by adopting different approaches. TransformerLogReg results in the best performance with 91.08% in the public test score outperforming other implemented models. Although not all methods yield remarkable results, every single experiment did overcome the provided baseline model accuracy and produced valuable insights.

- The TF-IDF embeddings represented in vector space are not linearly separable, which is the cause for the poor performance of the SVM compared to the Random Forest model.
- The size of MLP architecture does not affect model performance. Through the RNN experiment, we observed that the general bag-of-words approach is sufficient to capture the general tone of the tweet.
- There is a trade-off between the performance improvement and computational cost when ensembling Transformer-based models.
- The performance of the TransformerLogReq model proved the advantage of combining the Transformers models and classical machine learning approaches.

In the future, we hope to improve the models' performances by combining Transformer-based models with other machine learning approaches besides Logistic Regression. Furthermore, the hypothesis in section V-C concerning the properties of the W2V embeddings through vector summation could be further investigated by running a clustering algorithm on the embedded tweets.

Acknowledgements

The authors thank prof. Gunnar Rätsch and the TAs of Computational Intelligence Lab Spring 2022 for organizing the competition, holding tutorials, and clarifying questions regarding the project.

Online Resources

- Dataset
- Github Repository
- Kaggle Leaderboard

References

[1] M. S. Basarslan and F. Kayaalp, “Sentiment analysis with machine learning methods on social media,” *Gredos. Repositorio documental de la Universidad de Salamanca*, Sep 2020. [Online]. Available: https://gredos.usal.es/handle/10366/146100

[2] Y. Goldberg and O. Levy, “word2vec explained: deriving mikolov et al.’s negative-sampling word-embedding method,” 2014. [Online]. Available: https://arxiv.org/abs/1402.3722

[3] T. Evgeniou and M. Pontil, “Support vector machines: Theory and applications,” vol. 2049, pp. 249–257, 01 2001.

[4] V. Sharma, S. Rai, and A. Dev, “A comprehensive study of artificial neural networks,” *International Journal of Advanced research in computer science and software engineering*, vol. 2, no. 10, 2012.

[5] H. Palangi, L. Deng, Y. Shen, J. Gao, X. He, J. Chen, X. Song, and R. Ward, “Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval,” *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 24, no. 4, pp. 694–707, apr 2016. [Online]. Available: https://doi.org/10.1109%2Ftaslp.2016.2520371

[6] A. Sherstinsky, “Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network,” *Physica D: Nonlinear Phenomena*, vol. 404, p. 132306, mar 2020. [Online]. Available: https://doi.org/10.1016%2Fj.physd.2019.132306

[7] 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.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” 2018. [Online]. Available: https://arxiv.org/abs/1810.04805

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, V. Stoyanov, and et al., “Roberta: A robustly optimized bert pretraining approach,” *arXiv.org*, Jul 2019. [Online]. Available: https://arxiv.org/abs/1907.11692

[10] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, “Electra: Pre-training text encoders as discriminators rather than generators,” *arXiv preprint arXiv:2003.10555*, 2020.

[11] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” 2019. [Online]. Available: https://arxiv.org/abs/1906.08237

[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, Jun. 2011, pp. 142–150. [Online]. Available: https://aclanthology.org/P11-1015

[13] X. Han, Z. Zhang, N. Ding, Y. Gu, X. Liu, Y. Huo, J. Qiu, Y. Yao, A. Zhang, L. Zhang, W. Han, M. Huang, Q. Jin, Y. Lan, Y. Liu, Z. Liu, Z. Lu, X. Qiu, R. Song, J. Tang, J.-R. Wen, J. Yuan, W. X. Zhao, and J. Zhu, “Pre-trained models: Past, present and future,” 2021. [Online]. Available: https://arxiv.org/abs/2106.07139

[14] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, “Huggingface’s transformers: State-of-the-art natural language processing,” 2019. [Online]. Available: https://arxiv.org/abs/1910.03771

[15] Q. Huang, Z. Huang, P. Werstein, and M. Purvis, “Gpu as a general purpose computing resource,” in *2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, 2008, pp. 151–158.

[16] Z. Yang, Z. Dai, Y. Yang, J. G. Carbonell, R. Salakhutdinov, and Q. V. Le, “Xlnet: Generalized autoregressive pretraining for language understanding,” *CoRR*, vol. abs/1906.08237, 2019. [Online]. Available: http://arxiv.org/abs/1906.08237

[17] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” *CoRR*, vol. abs/1810.04805, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[18] J. M. Pérez, J. C. Giudici, and F. Luque, “pysentimiento: A python toolkit for sentiment analysis and socialnlp tasks,” 2021.

[19] F. Barbieri, J. Camacho-Collados, L. Espinosa Anke, and L. Neves, “TweetEval: Unified benchmark and comparative evaluation for tweet classification,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*. Online: Association for Computational Linguistics, Nov. 2020, pp. 1644–1650. [Online]. Available: https://aclanthology.org/2020.findings-emnlp.148

[20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[21] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” 2014. [Online]. Available: https://arxiv.org/abs/1412.6980

Appendix

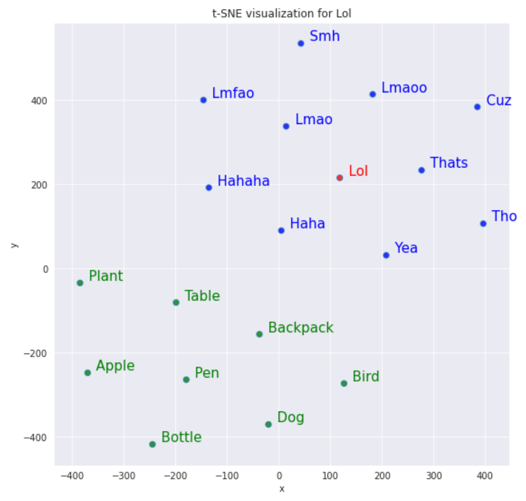


Figure 1: The 10 most similar words to ”lol” and 8 random words were plotted in two dimensions using t-SNE.

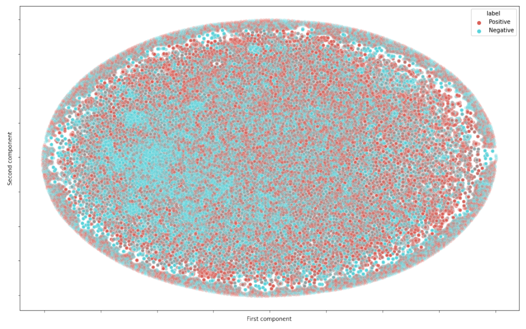
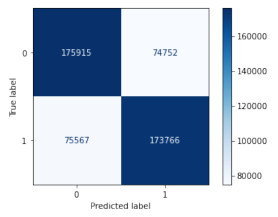
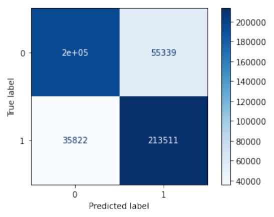


Figure 2: The TF-IDF embeddings for the sentences in the development set.

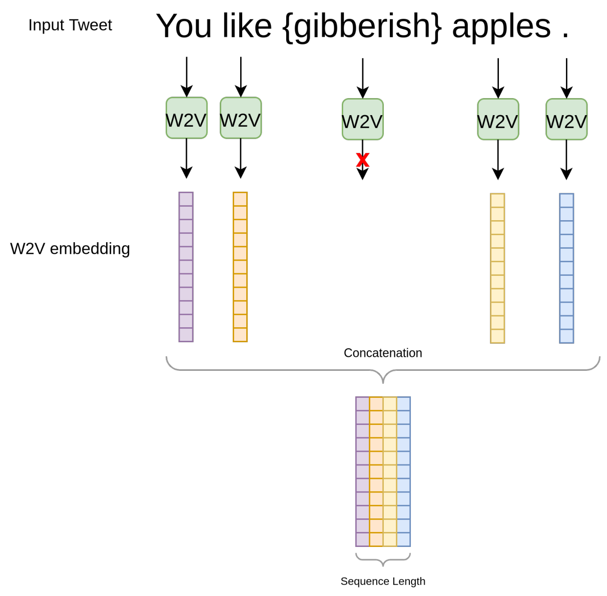


(a) Confusion matrix on the development set for SVM on TF-IDF embeddings.

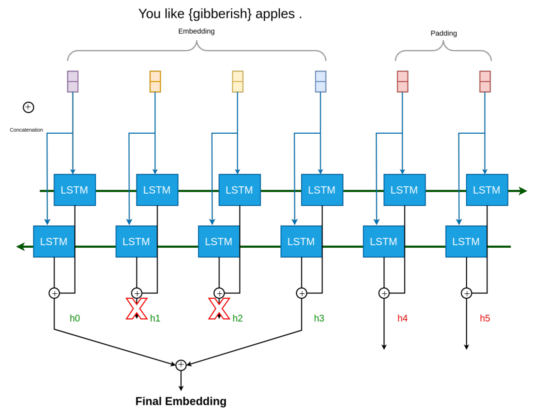


(b) Confusion matrix on the development set for the random forest on TF-IDF embeddings.

Figure 3: Confusion matrices



(a) The embedding process for the RNN Unit.



(b) Unrolling the BI-LSTM with a padded input.

Figure 4: The process used for the RNN model.

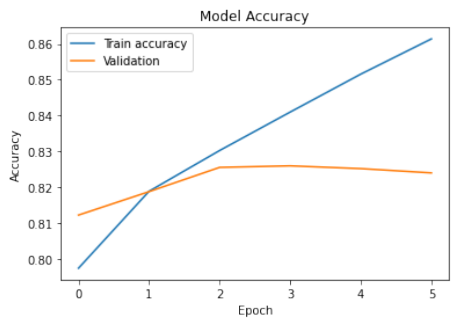


Figure 5: Validation and training accuracies for each epoch.