# Exploring the Effect of Input Features on the Sentiment Classification Task with BiLSTM Models

Anton Alexandrov, Moritz Dück, Mert Ertugrul, Lovro Rabuzin, Group: MAMLS
Department of Computer Science, ETH Zurich, Switzerland

***Abstract*—The task of extracting opinions and polarities from text data, known as sentiment analysis, can be motivated by the demand for gauging public opinion, assessing consumer opinions or monitoring brand reputation [1]. This paper uses tweets gathered from Twitter and aims to find a binary classifier that labels them as either positive or negative. The specific challenges of the data include constrained length, time-dependent slang, non-standard words, grammar and punctuation mistakes. To tackle this challenge, we compare the effect of different preprocessing techniques to normalise non-standard language. We construct a BiLSTM model that additionally uses statistical features, lexical features and topic distributions in hopes of providing a useful inductive bias to the model. Inspired by the success of transformer based models, we combine the augmented BiLSTM model with a BERT model. Experiments to combine the BiLSTM model with BERT suggest that the addition of the BiLSTM model doesn't increase the modelling power compared to the plain BERT model. Furthermore, adding auxiliary inputs to plain BiLSTM models and BERT models doesn't result in a performance increase for either model category. Hence, we show that augmenting model inputs with lexical and statistical features and information about topic distributions does not help the models in capturing the distribution of our dataset.**

## I. Introduction

With the rapid growth of social media platforms like Facebook, Twitter and Instagram over the past decade, the vast amount of content produced by the users of these platforms has become a valuable source of data for extracting useful information. Specifically, methods have appeared for gauging public opinion on specific products, businesses or individuals by automatically extracting the sentiment expressed in social media posts. Focusing on data extracted from Twitter in this paper, we will develop machine learning models trained to classify the sentiment of Tweets as either negative or positive.

## II. Preprocessing and Baselines

### A. Preprocessing

Texts originating from social media platforms, such as Twitter, tend to be highly non-standard, include spelling mistakes and intentional abnormalities such as keyboard mashes. This makes it more difficult to capture the semantics of the language. [2]

We will try different strategies to preprocess the dataset provided to us before starting to train a classifier. [3] mentions a summary of possible preprocessing steps for the twitter sentiment analysis task.

The number of options makes an exhaustive exploration of all possible combinations and their respective performance infeasible, so we identified the most commonly used and effective strategies to do selective ablation studies on their performance.

For our ablation study, we will use three baseline models to compare differences in the accuracy and F1-score in order to see which preprocessing steps can improve the performance on the task of sentiment classification.

The given dataset already has URLs and user mentions removed. In order to reason about the possible effect of normalising different tokens within a corpus, it is important to understand how often they appear within the data at hand. We summarized relevant statistics in Table III (Appendix). We will now present the steps applied during preprocessing.

*1) Contractions, repeated characters and hashtag expansion:* We remove repeated characters and expand contractions, as was suggested in [2]. We also split hashtags into only their constituent words.

*2) Emoticon and Laughter treatment:* Social media text often contains emotions expressed through emoticons, e.g. :), :/, or :-(, or through adding expressions of laughter in various forms, e.g. ”haha”. We observe 769 different variations of tokens expressing ”haha” but including further repetitions of ”h” and ”a”, e.g. ”hahahhahaa”, which will all be normalized to ”haha” in the process. As more than 4% of the tweets contain emoticons and 3% contain some form of ”haha” and these symbols can contain valuable information for determining the sentiment of a tweet, normalizing these occurrences could help to improve a models performance. To the best of our knowledge, performing this kind of normalization of laughter expressions has not been done before.

*3) Dictionary based approach:* We use the dictionary used in [2] for replacing slang and acronyms. The authors of the dictionary already removed ambiguous slang and acronyms.

*4) Sequence to sequence language model for text normalization:* A sequence to sequence model can capture semantic meaning at a word level and long-term contextual dependencies that help in disambiguation of multiple correction candidates. We use the model developed in [4] to replace slang and correct minor spelling mistakes in case of ambiguity.

### B. Baseline Models

Three baseline models with different levels of complexity are introduced to examine the complexity required to tackle our task. The following sections introduce the methods used for our baseline classifiers.

*1) Linear SVM Baseline with TF-IDF:* For the first baseline, a simple way of encoding tweet information (TF-IDF) is combined with a simple linear classifier (SVM). *Term frequency-inverse document frequency (TF-IDF)* can be broken down into two parts: *TF (term frequency)* and *IDF (inverse document frequency)*. Term frequency refers to the frequency of a particular token relative to the whole document. Inverse document frequency refers to how common a token is amongst the corpus. Using these numerical values, we can use a linear *Support Vector Machine (SVM)* to classify whole sentences.

*2) MLP Baseline with GloVe:* For the second baseline, we increase the complexity of how tweets are embedded to feature vectors, and opt for a more complex non-linear classifier with higher capacity than the first baseline. We use 200-dimensional pretrained word embeddings generated through *GloVe*[5], a technique for training word embeddings based on global word-word co-occurrence statistics[6]. We average the embeddings for each word in the Tweet to obtain a fixed length input vector for a *Multilayer Perceptron (MLP)* classifier.

*3) BiLSTM Baseline:* The *bidirectional long short-term memory (BiLSTM)* model is composed of two long short-term memory cells, one receiving the input sequence sequentially in the order provided, the other one working on the reverse sequence. The LSTM model is a well-established architecture in the family of *recurrent neural networks (RNNs)*, greatly improving on training stability compared to initial models of that type. The BiLSTM sequentially processes word embedding vectors, and can be used to form sentence classifiers as in [7]. In the scope of this baseline, initial runs were carried out with third party GloVe embeddings pretrained on other Twitter datasets[5]. Later, to have a pipeline that is not influenced by external data and to be able to see the effects of different variations of our preprocessed datasets on model performance, embeddings that are trained from scratch on our dataset were preferred. We expect that the embedding vocabulary would adapt according to the preprocessing choices when trained from scratch rather than using purely pretrained embeddings. Since fine tuning the pretrained GloVe model is computationally inefficient, Fasttext embedding model is used as an alternative[8].

An initial base model with a minimal classifier head is used to observe performance changes when training with different preprocessing configurations applied to our data. Then, with the most preprocessed dataset, we optimize the model architecture. Training hyperparameters, number of one dimensional convolutional layers, number of fully connected layers are selected by searching in a multi step procedure due to the prohibitive size of the parameter space.

## III. Models and Methods

As the BiLSTM is the best performing baseline model (refer to IV-A), we experiment with extensions to the model through changing input features.

### A. Adding Statistical features

Different tokens of interest identified in the preprocessing section can also be used as explicit features when converted to counts on a sentence level, as suggested in [9]. Specifically, we use the frequency of ”haha” tokens, emoticons, numbers, combinations of numbers and letters, contractions and slang words and concatenate these counts to the output of the first fully-connected layer in the classifier head.

### B. Using Lexical features

Sentiment lexicons contain lists of lexical features labeled as either positive or negative. These human generated dictionaries can even be used on their own for classification [10], which has been the case for earlier sentiment models. They may also help to augment models [11]. We use the VADER [10] lexicon for augmenting our BiLSTM models. We treat the sentiment score of each word according to the VADER dictionary as an extension of the word embedding we use for the model. In order to incentivise the model to capture this information, we append the sentiment score to the embedding multiple times. A similar approach is followed in relevant papers for this reason [11].

### C. Adding Topic Modelling

Topic modelling can be done with Latent Dirichlet allocation in an unsupervised manner. In our case, it can be used as a feature engineering method to create topic distribution features for sentiment classification, as also carried out in [12] which we choose to use as an auxiliary model input. One would expect tweets within a well defined tweet topic to produce similar sentiment consistently. For example, theoretically, if political tweets were captured as a topic, we could hypothesise that they could skew towards negative sentiment. Therefore, when combined with another model, using LDA can be seen as an intermediate learning step that learns an auxiliary function and ideally provides perspective to the main model about clustering behavior in token distributions of tweets.

### D. Using Transformer model embeddings

The BERT language model [13] introduced in 2018 showed state of the art performance in many NLP tasks. The Transformer architecture allows to process an input sequence all at once, using an attention mechanism to determine a weighting of relevance of the input tokens. This allows to capture long-ranging dependencies more easily than in a classic LSTM. Trained on unlabeled data, the base model can be used for different tasks, one of them being classification by adding a classification layer on top of the Transformer output.

Bert language models use a specific kind of tokenizer that inserts a [CLS] token at the beginning of a sentence and a [SEP] token at the end. Tokens within the sentence are split into constituent subwords. The standard way of using Transformer outputs for classification is feeding the output corresponding to the [CLS] token to a series of fully-connected layers.

Since the overall architecture can also be regarded as a way to create very elaborate word embeddings, we experiment with feeding the full output of BERT to our BiLSTM model, inspired by [11]. The approach we used here differs from the standard approach to classification using BERT by the fact that we feed the embeddings of all of the tokens into our BiLSTM model in the hopes of our BiLSTM model being able to capture additional information from the sequence of embeddings. In this way, we regard BERT as a replacement of the GloVe or Fasttext embedding approaches used previously.

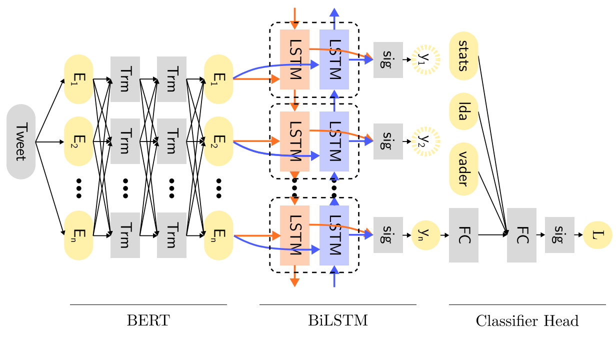

Figure 1. Diagram of BERT + BiLSTM model

We experiment with BERT-based models using only fully-connected layers in their classification head as well as all versions of the BiLSTM we experimented with previously. The main difference in the two approaches is that in BiLSTM-based models without BERT, lexical features are concatenated with the embeddings at the input of the BiLSTM. For BERT-based models with BiLSTM, since the model does not work with individual word embeddings, the lexical features extracted from the entire tweet are concatenated with an intermediate representation of the data in the classification head of the model, after the sequential section of the architecture is over.

As backbone of the BERT-based models, we use DistilBERT, a smaller version of BERT presented in [14]. We use this version of BERT because it requires less computational resources and training time. We also experiment with using RoBERTa as a model backbone for select model configurations. RoBERTa is a robustly pretrained version of the BERTbase model presented in [15]. The models using BERT were fine-tuned end-to-end with the learning rate linearly increasing until it reached $2 \cdot 10^{-5}$ for the first 10% of the training steps and linearly decaying to zero afterwards as suggested in [16]. Training is done using the version of the dataset with all of the preprocessing methods applied.

## IV. Results

For performance measurement, accuracy and F1-Score metrics were used. Results were recorded for a held out 10% of the original training set which represents our validation set.

### A. *Baselines and Preprocessing*

The effects of preprocessing on the performance of the baseline models can be seen in Table I. Each column represents a different set of preprocessing methods. ”s2s” refers to II-A4, dictionary to II-A3 The columns until ”all standard methods” are standard preprocessing steps applied on their own while the column ”all standard methods” combines previous steps as in II-A1. The columns to the right starting with ”+” are each cumulatively added preprocessing steps, meaning that each of these columns include all steps given in the columns to its left.

### B. *Model Extensions*

The performance of each final model including arrangements of BiLSTM, DistilBERT and RoBERTa as well as their extensions are given in Table II. The identifier ”Lex” represents the extension presented in III-B, ”LDA” represents III-C and ”Stats” represents III-A.

## V. Discussion

### A. *Baselines and Preprocessing*

We see that there are no significant differences in classification accuracy for any baseline classifier across differently preprocessed versions of the dataset.

[3] did a comparative analysis on preprocessing methods and were able to find significant performance differences (more than 2% difference in classification accuracy) when using TF-IDF embeddings paired with Logistic Regression, Bernoulli Naïve Bayes and Linear SVC classifiers. [17] offer similar results.

On the contrary, our comparison does not highlight any significant differences.

There was no consistent performance difference when changing embedding models across different variations of the Bidirectional LSTM based models, despite the fact that GloVe and Fasttext embedding models are built on different principles. We discern that the amount of useful semantic information they captured when applied to our dataset was

| Model | Preprocessing Method: | none | repeated chars | contractions | hashtags | all standard methods | + s2s | +dictionary | + emoticons + haha |
|---|---|---|---|---|---|---|---|---|---|
| TF-IDF + SVM | Accuracy | 0.8206 | 0.8208 | 0.8216 | 0.8197 | 0.8206 | 0.8190 | 0.8185 | 0.8182 |
| | F1-Score | 0.8247 | 0.8250 | 0.8257 | 0.8239 | 0.8249 | 0.8234 | 0.8231 | 0.8226 |
| Glove + MLP | Accuracy | 0.8383 | 0.8396 | 0.8421 | 0.8404 | 0.8445 | 0.8441 | 0.8449 | 0.8435 |
| | F1-Score | 0.8420 | 0.8442 | 0.8454 | 0.8433 | 0.8483 | 0.8466 | 0.8489 | 0.8476 |
| Fasttext + BiLSTM | Accuracy | 0.8713 | 0.8714 | 0.8710 | 0.8741 | 0.8713 | **0.8752** | 0.8745 | 0.8737 |
| | F1-Score | 0.8731 | 0.8726 | 0.8732 | **0.8752** | 0.8722 | 0.8761 | 0.8735 | 0.8724 |

Table I
COMPARISON OF BASELINE PERFORMANCE WITH DIFFERENT PREPROCESSING METHODS

on a similar level, perhaps bounded by the limitations of the data distribution.

The paramater search for BiLSTM based models demonstrated that the presence of convolutional layers and increasing classifier capacity resulted in minor improvements. Most variations of the model displayed similar performance within a margin of 87$\pm$1% . Therefore, the simplest classifier head (2 fully connected layers) was chosen while retaining the same accuracy.

### B. *Model Extensions*

The accuracy of the model combining BERT with BiLSTM does not improve over a BERT model with a simpler classification head, however it is still of interest to understand the contributions to the performance in the combined setting.

Looking at the classification agreement on the test data between a BERT model on its own, the BiLSTM-BERT model and a BiLSTM model on its own, we observe that while the two models including BERT agree on 95.39% of the predictions, the BiLSTM model shares only 89.49% and 89.46% of the predictions with the BiLSTM-BERT and BERT model respectively. This indicates that BERT has a larger influence on the behavior of the combined model than the BiLSTM. Given the much higher model capacity and parameter number, this outcome is reasonable.

Overall, both the BiLSTM extensions and Bert extensions with auxiliary inputs perform the same as the non-extended versions, while there is a consistent performance gap between BiLSTM and BERT based models. We deduce that the non-extended models already capture most of the information we aim to introduce by extension with auxiliary inputs. It could be that Lexicon information is not informative for the specific word distribution of our tweet dataset, or the same information is already extracted by neural networks easily. The statistical indicator vector extension, on the other hand, may be providing indicators that are independent of sentiment (e.g contractions may not correlate with sentiment) or their format as an input may be difficult to interpret or learn weights for.

As mentioned in II-A, irregular language with variations of words, slang and misspellings is a major obstacle for tweet NLP. The fact that BERT combines sub-word tokenization with a high-capacity, attention-based model would mean that it is more robust to our word distribution than word embedding models, as it can divide and combine sub words to both reduce the out of vocabulary problem and interpret misspelled words. These properties explain partially why it outperforms BiLSTMs and does not benefit from extensions for our dataset.

| Model | Accuracy | F1 Score |
|---|---|---|
| BiLSTM | 0.8755 | 0.8765 |
| DistilBERT + MLP | 0.9001 | 0.9001 |
| RoBERTa + MLP | 0.9121 | 0.9121 |
| BiLSTM + Lex | 0.8757 | 0.8766 |
| BiLSTM + Lex + LDA | 0.8752 | 0.8774 |
| BiLSTM + Lex + LDA + Stats | 0.8754 | 0.8764 |
| DistilBERT + BiLSTM | 0.9005 | 0.9005 |
| DistilBERT + BiLSTM + Lex | 0.9001 | 0.9001 |
| DistilBERT + BiLSTM + Lex + LDA | 0.9001 | 0.9001 |
| DistilBERT + BiLSTM + Lex + LDA + Stats | 0.9001 | 0.9000 |
| RoBERTa + BiLSTM + Lex + LDA + Stats | 0.9078 | 0.9077 |

Table II
MODEL PERFORMANCE COMPARISON

## VI. SUMMARY

We trained three baseline models — namely an SVM, MLP and BiLSTM model — on several preprocessed versions of the dataset and different word embeddings, where the BiLSTM trained on Fasttext embeddings has proven to be the best performer of our initial models. In search of improvements, we added statistical features, lexical features and topic distributions to the input of our BiLSTM model, which showed no performance improvement for any of the combinations. Lastly, we experimented with using the output of a fine-tuned BERT language model as input to the BiLSTM model, which resulted in noticeable improvements in performance compared to a BiLSTM with a Fasttext embedding. Yet, comparing the results to the performance of BERT used in combination with a simple classification layer on its own, we can conclude that in the combined model, the results are dominated by BERT, leaving the additional complexity of combining the models unjustified. As in the case for plain BiLSTM models, adding statistical features, lexical features and topic distributions to the input of models based on BERT did not result in a performance increase. We conclude that augmenting BERT-based classifiers with BiLSTMs or the additional input features does not help BERT in capturing the idiosyncrasies of our dataset.

## References

[1] R. Feldman, "Techniques and applications for sentiment analysis," *Communications of the ACM*, vol. 56, no. 4, pp. 82–89, 2013.

[2] A. Sarker, "A customizable pipeline for social media text normalization," *Social Network Analysis and Mining*, vol. 7, no. 1, pp. 1–13, 2017.

[3] S. Symeonidis, D. Effrosynidis, and A. Arampatzis, "A comparative evaluation of pre-processing techniques and their interactions for twitter sentiment analysis," *Expert Systems with Applications*, vol. 110, pp. 298–310, 2018.

[4] I. Lourentzou, K. Manghnani, and C. Zhai, "Adapting sequence to sequence models for text normalization in social media," in *Proceedings of the international AAAI conference on web and social media*, vol. 13, 2019, pp. 335–345.

[5] S. University, "Twitter pre-trained word vectors," Jun. 2019. [Online]. Available: https://doi.org/10.5281/zenodo.3237458

[6] 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

[7] B. Jang, M. Kim, G. Harerimana, S.-u. Kang, and J. W. Kim, "Bi-lstm model to increase accuracy in text classification: Combining word2vec cnn and attention mechanism," *Applied Sciences*, vol. 10, no. 17, 2020. [Online]. Available: https://www.mdpi.com/2076-3417/10/17/5841

[8] U. Naseem, I. Razzak, S. K. Khan, and M. Prasad, "A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models," *CoRR*, vol. abs/2010.15036, 2020. [Online]. Available: https://arxiv.org/abs/2010.15036

[9] J. Carvalho and A. Plastino, "On the evaluation and combination of state-of-the-art features in twitter sentiment analysis," *Artificial Intelligence Review*, vol. 54, no. 3, pp. 1887–1936, 2021.

[10] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proceedings of the international AAAI conference on web and social media*, vol. 8, no. 1, 2014, pp. 216–225.

[11] U. Naseem, I. Razzak, K. Musial, and M. Imran, "Transformer based deep intelligent contextual embedding for twitter sentiment analysis," *Future Generation Computer Systems*, vol. 113, pp. 58–69, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X2030306X

[12] M. F. A. Bashri and R. Kusumaningrum, "Sentiment analysis using latent dirichlet allocation and topic polarity wordcloud visualization," in *2017 5th International Conference on Information and Communication Technology (ICoIC7)*, 2017, pp. 1–5.

[13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[14] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[16] M. Mosbach, M. Andriushchenko, and D. Klakow, "On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines," *CoRR*, vol. abs/2006.04884, 2020. [Online]. Available: https://arxiv.org/abs/2006.04884

[17] Z. Jianqiang and G. Xiaolin, "Comparison research on text pre-processing methods on twitter sentiment analysis," *IEEE access*, vol. 5, pp. 2870–2879, 2017.

## VII. Appendix

| | | |
|---|---|---|
| Emoticons | 48 | 0.0081% |
| Contractions | 119 | 0.0200% |
| Variations of "haha" | 769 | 0.1297% |
| Slang words | 2581 | 0.4355% |
| Numbers and Letters | 48863 | 8.2460% |
| Numbers | 56974 | 9.6148% |
| Hashtags | 114931 | 19.3955% |
| Total Unique Tokens | 592563 | 100.00% |

| | | |
|---|---|---|
| Variations of "haha" | 99406 | 0.2526% |
| Emoticons | 124667 | 0.3168% |
| Numbers and Letters | 269854 | 0.6857% |
| Hashtags | 352485 | 0.8957% |
| Numbers | 788908 | 2.0046% |
| Contractions | 908051 | 2.3073% |
| Slang words | 1105080 | 2.8080% |
| Total Words | 39354956 | 100.00% |

| | | |
|---|---|---|
| Variations of "haha" | 94141 | 3.7656% |
| Emoticons | 112296 | 4.4918% |
| Numbers and Letters | 177652 | 7.1061% |
| Hashtags | 294729 | 11.7891% |
| Numbers | 453552 | 18.1421% |
| Contractions | 505657 | 20.2262% |
| Slang words | 734604 | 29.3841% |
| Total Tweets | 2500000 | 100.00% |

Table III
FREQUENCIES OF TOKEN TYPES IDENTIFIED FOR PREPROCESSING, IN INCREASING FREQUENCY. ALL TOKEN OF INTEREST APPEAR AT LEAST ONCE EVERY 30 TWEETS.

### A. *Experimental Setup*

*1) BERT-based models:* The final set of parameters used to carry out experiments are as follows:

Training Hyperparameters:

- Optimizer: AdamW
- Batch size: 64
- Learning rate: $2 \cdot 10^5$
- Number of training epochs: 3
- Dropout rate: 0.3

Model Design Hyperparameters:

- BiLSTM Cell Size: 128
- Number of 1D Convolutional Layers: 0
- Number of Fully Connected Layers: 2
- Fully-connected layer output dimensions: 32, 2

*2) BiLSTM-based models:* The final set of parameters used to carry out experiments are as follows:

Training Hyperparameters:

- Optimizer: Adam
- Batch Size: 500
- Learning Rate: $5 \cdot 10^4$
- Number of training epochs: 30
- Dropout Rate: 0.0

Model Design Hyperparameters:

- Embedding Type: Fasttext
- Embedding Dimension : 100
- BiLSTM Cell Size: 128
- Number of 1D Convolutional Layers: 0
- Number of Fully Connected Layers: 2
- Fully-connected layer output dimensions: 16, 1