

# CIL 2022 Project 2 - Sentiment Analysis

Huang Daoji\*, Jorel Elmiger\*, Gabriel Gavrilas\*, and Mihhail Sokolov\*

\*Group: SentimentalGuys, Department of Computer Science, ETH Zurich, Switzerland

**Abstract**—State-of-the-art models in Natural Language Processing (NLP) today rely on preprocessing and tokenization. While these methods have been proven effective, tokenization-free approaches have emerged in recent years. We study the case of Twitter sentiment analysis to showcase the performance of tokenization-free models and compare them to state-of-the-art models and different preprocessing methods.

## I. INTRODUCTION

Twitter sentiment classification is a well-studied problem with the goal of classifying the sentiment of tweets. This particular subproblem of sentiment classification is interesting in that it can give real-world information about people’s sentiments on certain topics. In this project, we focus on binary sentiment classification, where a given text is labeled as either positive or negative.

In recent years large transformer-based language models[1] have revolutionized the field of NLP, and sentiment classification is no exception. State-of-the-art models are based on popular transformer architectures and pre-trained models such as BERT[2] and GPT-3[3]. For Twitter sentiment analysis specialized pre-trained models were published (e.g. TweetBert[4], TimeLM[5]).

A common approach to solving NLP problems is performing a significant amount of preprocessing on text. For transformer-based models used in sentiment analysis, this is no different. A common preprocessing step is tokenization, where the text is split into sub-segments, usually words or subwords, which get encoded to be used by the transformer. Although this approach has proven very successful, it comes with disadvantages, such as having a hard time accounting for spelling mistakes and noisy data, which is especially problematic for real-world data similar to tweets and chat messages. To mitigate these problems, tokenization-free approaches like Google’s ByT5[6] were introduced, working directly on UTF-8 bytes and removing the need for tokenization. The resulting models typically are more robust to spelling mistakes and noisy data but incur longer training times and worse performance for longer text sequences.

We work with provided Twitter sentiment data, in the following referred to as “Twitter dataset”. It is split into two sets: the training set with labels and the test set without labels.

Our contributions in this work are four-fold:

- We investigate several preprocessing methods on the Twitter dataset.

- We train multiple baseline models on the Twitter dataset, starting with simple baselines to fine-tuning and optimizing a state-of-the-art DistilBERT.
- We fine-tune a pre-trained ByT5 model on the Twitter dataset.
- We compare the ByT5 results to the baselines and preprocessing methods and interpret our findings.

## II. MODELS AND METHODS

### A. Baselines

We compare our tokenization-free approach to various tokenization-based methods, including:

- CountVectorizer approach: in this model, each tweet is represented as the frequency of top-5k words, and a logistic regressor is built upon such feature representations. It can be seen as a fixed one-hot embedding for each of the top-5k words.
- Word2Vec[7] mean vector: in this model, we load the word2vec word embedding trained on our Twitter dataset and take the mean word embedding as feature representation for each tweet. From these features, we train a similar logistic regression.
- Word2Vec LSTM: to motivate later deep learning approaches, we train an LSTM[8] with input being fixed word embeddings of each tweet’s words as a sequence.
- Word2embd LSTM: similar to the previous model, but here the word embedding is being constantly trained instead of the fixed ones from word2vec.

### B. Data Preprocessing

In this section, we explore possible data preprocessing methods to avoid overfitting and provide an additional regularization term for large language model training, including:

- Choice of stop-words, where in the processing step, some common and possibly meaningless words are filtered out to enforce the model to focus more on meaningful ones.
- Data augmentations, including: alternating clause order, substituting words by their synonyms, and translating to another language and then translating back. We notice that the size of our training data is relatively small compared to the number of parameters in language models, and augmented data can essentially avoid overfitting and help the model better generalize to unseen test sentences.

Some examples of our data preprocessing are shown in Table I. We found that, although such methods work for models trained from scratch, they do not generalize to pre-trained models with an inferior cross-validation result. We hypothesize that the pre-trained models are not trained with such preprocessing techniques.

Method	Example tweet
Original	i told my mom that i might come home coz i don't have warm things here , now she's all excited ! ! fml
Stop words	told mom might come home coz don't warm things , she's excited ! ! fml
Clause order	now she's all excited ! ! fml , i told my mom that i might come home coz i don't have warm things here
Synonyms	i told my mom that i might come home coz i don't have warm things here , now she's all <b>wild</b> ! ! fml
Translation	I told my mother that I might come home because I have <b>no</b> warm things here, now she is all excited! ! FML

Table I  
EXAMPLE OF DATA PREPROCESSING METHODS.

### C. DistilBERT (state-of-the-art models)

DistilBERT[9] is a "smaller, faster, cheaper and lighter" variant of the BERT[2] model, retaining approximately 97% of BERT's language understanding performance with 40% fewer parameters. As a BERT derivate, it employs bidirectional training of Transformers to achieve a deeper level of language understanding. However, compared to BERT, it is cutting the number of layers by 2, resulting in a much faster training phase - around 60% - a very important factor for us, since we have very limited resources. For the same reason, we use smaller datasets for a lot of the experimentation with preprocessing and hyperparameters, as a single epoch on the full dataset can take upwards of 8 hours even when utilizing a GPU. DistilBERT is available as a pre-trained model in the Hugging Face[10] library<sup>1</sup>.

DistilBERT is provided for cased or uncased input texts. Since proper capitalization is a rare treat on Twitter, we use the uncased version. In a first step, preprocessing on the raw input can be performed. We tried removing stop words but found the performance to be ever so slightly better when they are included. Then, the input gets tokenized with a vocabulary of roughly 30000 words. Special tokens are added, e.g. to denote the start of a sequence and to separate sentences. This can be done using a Hugging Face tokenizer for this particular model.

At this point, we can set the hyperparameters of our model. We primarily want to configure the input length, since DistilBERT defaults to 512 tokens - while Twitter limits each tweet to 280 characters. After inspecting the

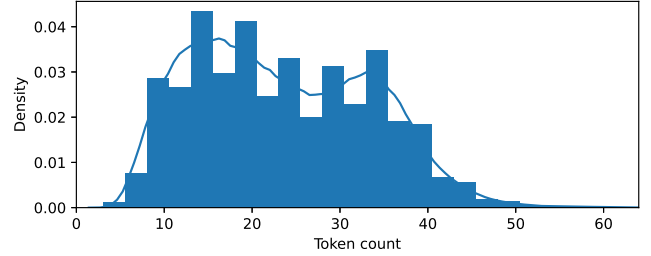


Figure 1. Frequency of the number of tokens in the full Twitter dataset.

lengths of our tokenized dataset sequences, we concluded that perhaps as few as 48 tokens may be enough (see Fig. 1). While a couple of sequences get truncated, most of them we would consider "noise" and not contributing to the classification effort anyway, e.g. tweets which mostly consist of what we assume is ASCII art which was scrambled by the data collection into escaped characters. We tried out several different lengths around 48 without noticing much of a difference in the achieved accuracy, but more testing would be necessary to fully determine the impact of the chosen length. To make sure, we settled with 64 tokens to easily capture all non-distorted tweets.

To finally perform a sequence classification task, we then have the option to either directly use the functionality provided by Hugging Face, or we can choose to put our own classifier "head" after the pre-trained BERT model, perhaps by using a softmax after gradually reducing the hidden layer size. This would require and enable us to at first freeze the model and train the head to map DistilBERT's output to the proper label, leaving the pre-trained language understanding fully intact, and then later possibly fine-tune the full model.

### D. Tokenization-free Approach

Our tokenization-free approach is based on Google's ByT5[6] model. ByT5 is based on Google's mT5[11], which is a multi-language version of the famous T5[12] text-to-text transfer transformer model. It shares the characteristics of T5, consisting of a large transformer model pre-trained for transfer learning on a large general dataset, the input and the output of the model both being in text format. ByT5's architecture differs from mT5 and T5 in that it is fed UTF-8 bytes directly without tokenization of the data. Another difference is the unequal number of layers for the encoder and decoder, having a larger encoder to account for the additional information stored in the network.

The advantages of ByT5 are a smaller token vocabulary, leading to smaller total model size, especially for small models, and improved robustness to spelling errors and noise.

We fine-tune ByT5 on the provided data to perform sentiment analysis. We load the pre-trained model from

<sup>1</sup>[https://huggingface.co/docs/transformers/model\\_doc/distilbert](https://huggingface.co/docs/transformers/model_doc/distilbert)

HuggingFace[10]<sup>2</sup> only using the smallest pre-trained model ByT5-Small, with 300 million parameters, due to computational resource limitations. We train the model to output "negative</s>" and "positive</s>" for negative and positive tweet sentiment respectively.

Despite the approach being tokenization-free, we utilize a tokenizer that splits the data into single characters, transforms it into the required format and performs padding for simpler data handling during training. We further perform parameter optimization to optimize training time. In particular, inspired by the findings of the previous subsection, we fix the maximum sequence length of the inputs to the model to 300 bytes. We report more detailed training settings and results in section III.

### III. RESULTS

#### A. Computational Requirements

We conducted all our experiments on the Leonhard/Euler cluster on a single NVIDIA GeForce RTX 2080 Ti GPU.

#### B. Results

*Baselines:* We report the public test accuracy from the considered baselines in Table II, W2V and W2E are short for Word2Vec and Word2Embd, respectively. For Word2Vec, we used the Gensim package to train a 25-dimensional word embedding on our Twitter training dataset. For a fair comparison, in each of the following LSTM experiments, all the word embeddings are also set to a length of 25.

Method	CountVectorizer	W2V mean embedding
Test Accuracy	79.34	68.72

---

Method	W2V LSTM	W2E LSTM
Test Accuracy	72.74	83.12

Table II  
TEST ACCURACY OF BASELINES.

We see here that embedding methods affect the performance of the model a lot, as the one hot embedding in CountVectorizer outperforms Word2Vec embedding regardless of the model built on top of, while the best result is obtained by W2E LSTM where the word embedding is specifically trained on the sentiment analysis task.

*Data Prepossessings:* We first present the effect of various choices of stop-words in Table III, where the "None" refers to the W2E LSTM method, and based on this model, we apply different stop-word choices, including

- Top-5k: only top 5k words will remain in preprocessed tweets
- NLTK<sup>3</sup>: stop word list from NLTK package

<sup>2</sup>[https://huggingface.co/docs/transformers/model\\_doc/byt5](https://huggingface.co/docs/transformers/model_doc/byt5)

<sup>3</sup><https://www.nltk.org/>

- NLTK+: we also manually include some meaningless words that are outside of NLTK list, like "< user >", and "< url >".

All models are trained on all training data with Adam optimizer for 100 epochs.

Method	None	Top-5k	NLTK	NLTK+
Test Accuracy	83.12	82.78	84.06	82.34

Table III  
TEST ACCURACY OF DIFFERENT CHOICE OF STOP-WORDS.

We see from the results that the original NLTK stop-word list performs the best, while our addition to the list harms the performance. Top-5k also does not beat the baseline model without any preprocessing, hinting at some low-frequency words containing useful information for later classification. However, by comparing it to the CountVectorizer baseline, we see the sequential prediction with trainable latent embedding works better than the sum of fixed one-hot embeddings, under the same stop-word strategy.

We now compare different data augmentation methods in Table IV. The augmentations are only applied to training data, with the input being 50% original and 50% augmented. All models are trained for the same number of epochs. We use synonyms taken from wordnet<sup>4</sup>, and for translation, the target language is German. NLTK stop-words are applied after these augmentations as it is shown to perform the best. Alternating clause order and changing synonyms does not beat the baseline as they will possibly lose some semantic information, while translation works better. We hypothesize that by translating to other languages, the augmented data preserves the semantics while exploring dynamic grammar variations.

Method	None	Clause order	Translation	Synonyms
Test Accuracy	84.06	83.98	84.62	83.40

Table IV  
TEST ACCURACY OF DIFFERENT CHOICE OF DATA AUGMENTATIONS.

*DistilBERT:* Many paths can be explored when trying to optimize DistilBERT for a specific downstream task. To get a rough overview of the impact of different parameters on the accuracy, we first conducted a broad grid search on a smaller dataset of just 50000 tweets. The tested parameters include the used DistilBERT variant, the dropout rate in the model, whether to remove stopwords, the learning rate, and the maximum length of the input token sequence. The

<sup>4</sup><https://wordnet.princeton.edu/>

custom head used was a three-layer sequential network of linear modules of sizes  $768 - 256 - 32 - 2$  with *ReLU* activation and 0.2 dropout between them. With a relatively large learning rate of  $5 \times 10^{-5}$ , we ran each setup for a total of 2 epochs, keeping the DistilBERT model frozen in the first one. Some of the 48 results are listed in Table V. Generally speaking, we confirmed our expectation that the uncased model outperforms the cased variant, and further found that the default dropout rate of 0.1 works fine, that it doesn't matter much if stopwords are removed or not, and that a maximum token length of 48 is sufficient, perhaps even 32 - at least for these particular 50000 samples.

Model	Dropout	Max tokens	Accuracy
distilbert-base-uncased	0.2	48	84.14
distilbert-base-uncased	0.1	32	84.02
distilbert-base-uncased	0.1	48	84.0
distilbert-base-uncased	0.2	64	83.9
distilbert-base-uncased	0.1	64	83.82
distilbert-base-uncased	0.2	32	83.78
distilbert-base-cased	0.2	64	83.78
distilbert-base-cased	0.1	48	83.75
distilbert-base-cased	0.1	32	83.69
distilbert-base-cased	0.1	64	83.65
distilbert-base-cased	0.2	48	83.55
distilbert-base-cased	0.2	32	83.39

Table V  
TEST ACCURACY OF DIFFERENT APPROACHES TO DISTILBERT OPTIMIZATION, SORTED BY PERFORMANCE. (SELECTED 12 OF THE TOTAL 48 TESTED CONFIGURATIONS)

Aside from the obvious hyperparameters like learning rate, number of epochs, and dropout rate, we also have the option to train our own head. Finding the best parameters for such a head is intensive in time and computational resources, and after trying a couple of architectures and training approaches, we were consistently lagging 1% – 2% of accuracy behind Hugging Face's provided classification functionality, thus we decided to simply use that.

We used these insights to design a couple of models to train on our full Twitter dataset. The best results we achieved can be found in Table VI. Here, we always used *distilbert-base-uncased*, never removed stopwords and the DistilBERT dropout rate was left at 0.1. Custom classifier head A was a set of linear modules of sizes  $768 - 256 - 32 - 2$  with *ReLU* activation and 0.2 dropout, while custom head B was a set of linear modules of sizes  $3072 - 512 - 2$  with *ReLU* activation and 0.25 dropout.

*Tokenization-free Approach:* We trained the ByT5\_small model first on 10% of the training data and then on the whole Twitter training dataset for one epoch. We used a version of the adam optimizer with *adam\_epsilon* set to  $1 \times 10^{-8}$ . We set the learning rate to  $3 \times 10^{-5}$  and used a batch size of 2. We used gradient accumulation over 16 steps. We ran our training on 4 CPU

Model	Head	LR	# tokens	Epochs	Accuracy
uncased	default	$5 \times 10^{-6}$	64	3	89.52
uncased	custom A	$3 \times 10^{-5}$	48	5 (3 frozen)	88.24
uncased	custom B	$5 \times 10^{-5}$	32	2	88.2

Table VI  
TEST ACCURACY OF DIFFERENT APPROACHES TO DISTILBERT OPTIMIZATION. SORTED BY PERFORMANCE.

Method	Accuracy
ByT5_small 10% dataset	85.6
ByT5_small full dataset	87.0

Table VII  
TEST ACCURACY OF BYT5\_SMALL.

nodes with each 100GB of memory. The results are shown in Table VII.

#### IV. DISCUSSION

We achieved the highest test accuracy with our optimized DistilBERT model. ByT5 achieves accuracy values close to DistilBERT, outperforming all other baselines. Surprisingly also simpler baselines like the LSTM achieve results only slightly worse. These results show the general feasibility of the tokenization-free approach. It is able to match existing state-of-the-art architectures but doesn't outperform them. A disadvantage of the tokenization-free approach is the large computational overhead with up to 3 times longer training and higher memory load compared to DistilBERT. We hypothesize training for more epochs might lead to even better accuracy values matching or outperforming DistilBERT.

#### V. SUMMARY

We implement several baselines for tweet sentiment classification. Our best-performing model is an optimized and fine-tuned DistilBERT. We compared these baselines with our tokenization-free approach. The tokenization-free ByT5 model achieves similar but slightly worse results than DistilBERT, despite the proposed advantages. On the other hand, training the tokenization-free ByT5 model requires much more resources compared to the other approaches explored above. We conclude that tokenization-free sentiment analysis achieves comparable results to state-of-the-art models, and has the potential for future improvement.

## REFERENCES

- [1] 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>
- [2] 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>
- [3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020. [Online]. Available: <https://arxiv.org/abs/2005.14165>
- [4] M. M. A. Qudar and V. Mago, "Tweetbert: A pretrained language representation model for twitter text analysis," 2020. [Online]. Available: <https://arxiv.org/abs/2010.11091>
- [5] D. Loureiro, F. Barbieri, L. Neves, L. Espinosa Anke, and J. Camacho-collados, "TimeLMs: Diachronic language models from Twitter," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 251–260. [Online]. Available: <https://aclanthology.org/2022.acl-demo.25>
- [6] L. Xue, A. Barua, N. Constant, R. Al-Rfou, S. Narang, M. Kale, A. Roberts, and C. Raffel, "Byt5: Towards a token-free future with pre-trained byte-to-byte models," 2021. [Online]. Available: <https://arxiv.org/abs/2105.13626>
- [7] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013. [Online]. Available: <https://arxiv.org/abs/1301.3781>
- [8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, nov 1997. [Online]. Available: <https://doi.org/10.1162/neco.1997.9.8.1735>
- [9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," 2019. [Online]. Available: <https://arxiv.org/abs/1910.01108>
- [10] 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>
- [11] L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel, "mt5: A massively multilingual pre-trained text-to-text transformer," 2020. [Online]. Available: <https://arxiv.org/abs/2010.11934>
- [12] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2019. [Online]. Available: <https://arxiv.org/abs/1910.10683>



Eidgenössische Technische Hochschule Zürich  
Swiss Federal Institute of Technology Zurich

## Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

**Title of work** (in block letters):

**Authored by** (in block letters):

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

**Name(s):**

**First name(s):**

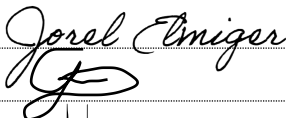
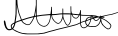


With my signature I confirm that

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

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

**Place, date**

**Signature(s)**

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