Tweet Sentiment Classification

Aly Elbindary, André Schakkal, Peter Harmouch

CS-433 - Project 2 - EPFL, Switzerland

***Abstract*—Twitter serves as a dynamic platform for individuals to freely and vividly articulate their thoughts. Consequently, there exists a considerable interest in efficiently gauging the sentiment conveyed in Tweets without the need for laborious manual inspection. This research project endeavors to address this challenge by focusing on the classification of tweets into positive or negative sentiment categories. To achieve this objective, a comprehensive approach is employed, leveraging multiple Natural Language Processing (NLP) techniques. The methods employed include logistic regression, word embeddings, Term Frequency-Inverse Document Frequency (TF-IDF), and the renowned transformers architecture.**

I. Introduction

This project aims to automate sentiment analysis using machine learning and NLP techniques. The project explores word embeddings with GloVe [1], TF-IDF representation, and advanced Transformer models like BERT [2]. Each method undergoes preprocessing, hyperparameter search, and fine-tuning processes.

The objective is to compare the effectiveness of these approaches in sentiment analysis on Tweets, highlighting their advantages and challenges. Specifically, this project focuses on comparing state-of-the-art transformer models with established baselines like GloVe-based word embeddings [1] and TF-IDF representation, providing benchmarks for evaluating advanced models.

II. Word embeddings

The initial approach for this project involves training word embeddings to develop an effective representation of words within our dataset. We begin with preprocessing the dataset. Following this, we train word embeddings to effectively represent words. These embeddings are then used to convert each Tweet into a numerical vector. The final step involves employing a linear classifier to categorize the Tweets as either positive (assigned label 1) or negative (assigned label -1).

A. Preprocessing

1) Removal of Duplicate Tweets: Duplicate Tweets were identified and removed. Eliminating duplicates ensured that the model was trained on a diverse range of distinct instances to avoid bias. Furthermore, by ensuring that the training and testing datasets are distinct, the model's performance evaluation becomes more indicative of its true predictive capabilities.

2) Tokenization: The text segmentation process divides content into tokens by two criteria tailored for English Tweets: (1) white spaces between words, and (2) punctuation marks attached to words.

3) Creation of Word Dictionary and Exclusion of Infrequent Words: The word dictionary was compiled from the preprocessed dataset, listing unique words and their frequencies. To refine the dataset for more effective analysis, words occurring fewer than five times were excluded. This approach focused on reducing noise from rare words and enhancing the relevance of the vocabulary for sentiment analysis.

4) Lemmatization and Stop Word Removal: Hyperparameter Search: In our efforts to enhance model accuracy using preprocessing, we conducted a hyperparameter search focusing on lemmatization and stop word removal. Lemmatization reduces words to their base form and was implemented in three ways: no lemmatization (data unchanged), full lemmatization (applied to all words including adjectives and adverbs), and partial lemmatization (only nouns and verbs). For stop word removal, we experimented with removal thresholds: 0% (no removal), 0.01%, and 0.1%, where the threshold indicates the proportion of the most common words removed from the text.

This search followed our initial experiments with GloVe embeddings, where we had not applied any preprocessing to obtain a model that effectively balances accuracy and computational load. The specifics of this will be elaborated in the next subsection. Our approach for the hyperparameter search involved using that model, dividing the data into training and testing sets, and then iterating through the different combinations of lemmatization methods and stop word removal thresholds. We measured the accuracy for each combination to determine the most effective approach. As detailed in appendix B2, we found that partial lemmatization (applied only to nouns and verbs) combined with a 0% threshold for stop word removal yielded the best results. This result makes sense as adjectives and adverbs are crucial for sentiment analysis, and stopwords can offer insights into the writing style of Tweets, aiding in sentiment determination.

B. Training GloVe Embeddings

1) Building coocurence matrix: The co-occurrence matrix, crucial for our GloVe word embeddings, records word pair frequencies throughout the dataset. We counted each word pair's co-occurrences within Tweets, capturing broad word relationships vital for the GloVe model, irrespective of their proximity.

2) Global Vectors for Word Representation (GloVe): The Global Vectors for Word Representation (GloVe) technique [1] was employed to generate word embeddings. The GloVe model learns word embeddings by factorizing the co-occurrence matrix, essentially aiming to encode the probability ratios of word co-occurrences in the embedding space. The resulting word vectors show meaningful linear substructures, making them effective for various natural language processing tasks and in this case for binary sentiment classification.

3) GloVe Hyperparanmeters: The Glove algorithm that was employed in this project contained 5 hyperparameters. A hyperparameter search was conducted to determine the optimal configuration for training the embeddings. The 5 hyperparameters are the embeddings dimension, the maximum co-occurrence frequency, the scaling of co-occurrence counts, the number of epochs and the learning rate. They are presented in more detail in appendix B2.

Given the computationally intensive nature of training word embeddings, iterating over numerous values for each of the five hyperparameters was not feasible. Therefore, we initially set hyperparameters to extreme values based on intuition, then performed a targeted grid search in logarithmic space focusing on the most impactful hyperparameters. This approach balanced thorough optimization with computational feasibility.

Table IIIa displays the accuracy achieved using the default hyperparameters for GloVe embeddings, as specified in the code from the homework template. This model achieved an approximate accuracy of 59%. Table IIIb illustrates the top GloVe model resulting from the hyperparameter search. The best model achieved an accuracy of 79%. The hyperparameters corresponding to these top-performing models were selected for our final word embeddings model. Note that in the figures below Eta is the learning rate, Alpha is the scaling of co-occurrence counts and NMax is the maximum co-occurrence frequency.

Lemmatization type	Stop-words threshold	Embeddings dimension	Alpha	Eta	Final epoch	NMax	Accuracy
none	0	20	0.75	1e-3	10	100	0.5878

(a) Accuracy of the Default GloVe Embeddings Model

Lemmatization type	Stop-words threshold	Embeddings dimension	Alpha	Eta	Final epoch	NMax	Accuracy
partial	0	2000	0.85	1e-4	3	5000	0.7902

(b) Top Performing Model from Hyperparameter Search

Table I: Performance of various GloVe models

C. Representing Tweets

Once we have obtained the embedding for each word in our dictionary, we can use them to calculate a representation for each Tweet. The selected method involves computing the average of the embeddings of all words within the Tweet. The resulting vectors, representing Tweets, are used as an input to the classifier.

D. Training Classifier

1) Choice of Classifier: Logistic Regression was the primary classifier chosen for this task due to its simplicity and effectiveness in binary classification tasks.

2) Penalty Tuning: A hyperparameter search was conducted on the regularization coefficient λ. The type of regularization was chosen to be L2 since it provided the best results on the best model. We tested different λ values using 5-fold cross-validation. The different results are presented in appendix B2, the best accuracy was achieved with $\lambda = 10$.

E. Comments on our Best Word Embeddings Model

In summary, our final model encompasses nine hyperparameters: two for preprocessing, five associated with GloVe, and two for logistic regression. After fine-tuning these parameters, we achieved an accuracy of approximately 79%. This represents the optimal performance that we were able to obtain with GloVe embeddings. In the following sections, we will explore more advanced methods that promise to yield higher accuracies.

III. Bag-of-words TF-IDF

For this second method, although we knew that transformers models would outperform these models, for the sake of exploration and comparison, we decided to use a different text-encoding method along with the standard regression method that we used in section II: We encoded our Tweets using the TF-IDF method, which is a post-processing procedure of the Bag of Words matrix (BoW). We decided to explore the TF-IDF representation to establish an insightful second baseline for comparison purposes.

A. TF-IDF Matrix

The BoW matrix is a specific representation of text datasets. Firstly, the dataset is tokenized, and each unique token form the vocabulary for our dataset. Then, each Tweet is considered as a separate document. Finally, each row within the BoW matrix represents a document, and each column represents a token. Thus an entry (i, j) of this matrix contains the frequency of the token of column number j within document number i. This means that the BoW matrix ends up being a sparse matrix that captures the term frequency (TF) of each token for every document.

Afterwards, the inverse document frequency (IDF) for each individual token is computed : the document frequency of a particular token is the number of documents that contain the corresponding token, divided by the total number of documents. Thus, TF-IDF post-processing is equivalent to multiplying every column of the BoW matrix with its corresponding IDF constant.

B. Hyperparameter Search

We use the implemented TF-IDF encoder function contained within the `sklearn` library (`TfidfVectorizer`) [3], and then train a logistic regression model. We select the best model by performing a hyperparameter search over the different parameters of the encoder function, as well as the regularization parameter λ of the logistic regression model. The results of this hyperparameter search and an explanation for each of the hyperparameters can be found within the appendix B1. The optimal hyperparameters found are the following : $\lambda = 0.1$, $max\_df = 0.5$, and $min\_df = 1$.

C. Results

Selecting the best-performing hyperparameters, we achieve an accuracy of 84.5% on the `aicrowd` platform on the provided test-set. However, better results were obtained using a transformer state-of-the-art architecture, which will be explained in the following section.

IV. Transformers

After experimenting with previous techniques, the research adopts the Transformer architecture [4], leveraging its ability to integrate text representation and regression in a single step as opposed to the previous explained techniques.

A. Choice justification and advantages

As mentioned earlier, Transformers streamline the entire process into a single step, eliminating the need for separate phases of learning data representation. Unlike traditional approaches such as word embeddings (e.g., GloVe, word2vec), Transformers integrate a text representation feed-forward network directly into their architecture. This unique design ensures that the text representation layer is trained concurrently with the predictor, resulting in a more tailored and task-specific representation.

Additionally, Transformers introduce the concept of position embeddings, a crucial feature absent in earlier methods. Previous techniques treated Tweets as a vector, either through bag-of-words TF-IDF representation or by averaging word embeddings. However, these methods overlooked the positional aspects of words within the Tweet. Transformers address this limitation by including token positions through positional embeddings. This enhancement captures the sequential order of words, acknowledging that word positioning can influence sentence meaning.

A central advantage of Transformers lies in their attention mechanism [4]. This mechanism enables the model to discern which words should receive more attention, even when they do not appear side by side. Moreover, employing multiple attention heads allows the model to attend to various meanings and perspectives, enriching the understanding of sentence semantics. An interesting visualization of this attention mechanism can be found in appendix A.

To ensure optimal performance, a Transformer must grasp the semantics of input sentences. This entails having a sufficiently large model with an ample number of attention heads to capture nuanced meanings. Additionally, training on an extensive corpus of text and allowing sufficient training time are crucial, these are resource-intensive tasks.

Rather than embarking on the arduous task of implementing and training a Transformer from scratch, we opt to leverage a pretrained model. The advantage here lies in the pretrained model's prior exposure to vast text corpora, enabling it to capture complex text semantics effectively. The pretrained model requires fine-tuning only for the specific downstream task we are interested in—Tweet sentiment classification.

Among various Transformer architectures, we choose the Bidirectional Encoder Representations from Transformers (BERT) [2] due to its encoder model nature. Unlike decoder models such as GPT, encoder models are designed to process input sentences and create encoded representations, they are used in diverse downstream tasks, especially text classification.

To further boost model performance, we opt for a specific pretrained version of BERT known as BERTweet [5]. This model, built upon the BERT architecture, is pretrained on a corpus of 850 million English Tweets [5], equipping it with a profound understanding of Tweet semantics.

Summarizing the advantages of using a pretrained transformer, specifically BERTweet:

- Transformers learn task-specific word representations effectively.
- Consideration of word positioning enhances prediction accuracy.
- Attention mechanisms contribute to a nuanced understanding of semantics.
- Pretrained models are well-equipped to capture complex text semantics.
- BERT's encoder model is well-suited for text classification.
- BERTweet's training on a large corpus of Tweets enhances its understanding of Tweet semantics.

B. Method

1) Model architecture: The chosen model for this study is the BERTweet model, having the same architecture as BERT. As illustrated in [2], a special token, `[CLS]`, is appended at the beginning of every input sentence. The BERT model generates an output state (C, T_1, T_2, ... ,T_N) for each input token (`[CLS]`, Tok 1, Tok 2, ... ,Tok N), as shown in Figure 1. In typical classification tasks, the final hidden state corresponding to the `[CLS]` token serves as an aggregate sequence representation for classification.

To perform classification, the output of the `[CLS]` token is inputted into a feed-forward classification layer. This layer has an input size equal to the dimension of the final hidden

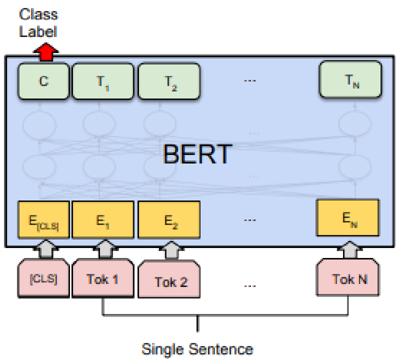


Figure 1: BERT Architecture

state corresponding to the `[CLS]` (768 in this case) and an output size equal to the number of classes for classification (2 for Positive and Negative sentiment).

2) Fine-tuning: Utilizing the `transformers` library [6], we implement the fine-tuning process following the steps outlined in the BERTweet paper [5]. After adding the feed-forward classification layer, the method involves:

- Normalizing Tweets using a predefined function[1]. Tweet normalization encompasses basic operations like replacing "cannot" with "can not" and substituting quotations with apostrophes.
- Tokenizing Tweets using BERTweet's tokenizer, available in the `transformers` library. This tokenizer is the "TweetTokenizer" from the NLTK toolkit [7].
- Training the model on three different seeds of the data with an AdamW optimizer [8] using a fixed learning rate, employing cross-entropy loss, as detailed in [5].
- Selecting optimal model weights for each seed.
- Employing an ensemble of the three best models, selecting the class predicted by the majority among these models.

3) Hyperparameter search: The sole hyperparameter in the chosen model is the learning rate. Despite the recommendation of a learning rate of 1e-5 in [5], a hyperparameter search is conducted on a logarithmic scale. The sample dataset, with an 80:20 train-test ratio, is employed for each iteration, comprising 10 epochs. The best-performing model for each iteration is selected.

The results of this hyperparameter search are presented in appendix B3, the peak test accuracy was observed for a learning rate of 1e-6. Consequently, this learning rate was chosen for training on the complete dataset.

C. Results

Upon training the model with the optimal learning rate on the full dataset and evaluating it on the provided test set, a promising classification accuracy of 92.1% was achieved on the `aicrowd` platform.

[1]`https://github.com/VinAIResearch/BERTweet/blob/master/TweetNormalizer.py`

V. Comparative Analysis of Models

This section provides an examination of the models experimented within previous sections, offering a comparison of their performance.

To evaluate these models, we conducted a 5-fold cross-validation, employing the most effective model identified for each technique. The results are presented in Figure 2. The transition from GloVe to TF-IDF with logistic regression reveals a good improvement in performance. However, the most interesting observation arises from the substantial leap in accuracy when transitioning from traditional techniques (GloVe, TF-IDF) to the BERT Transformer model. Additionally, we can see an accuracy boost achieved by utilizing a pre-trained BERT model fine-tuned specifically for Twitter data (BERTweet).

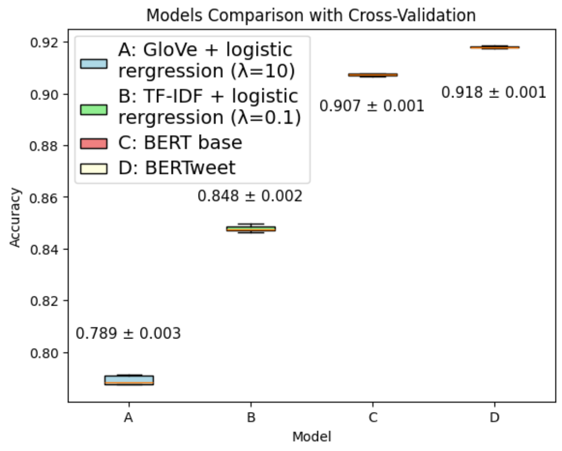


Figure 2: Models Comparison with Cross-Validation

Furthermore, by employing the best model for each technique, we report the highest accuracy achieved on `aicrowd` for the provided test set, as illustrated in Table II. Notably, the best performance is attained through an ensemble approach applied to the top three performing BERTweet models, resulting in a promising accuracy of 0.921.

Model	**Accuracy**
GloVe + logistic rergression (λ=10)	0.787
TF-IDF + logistic rergression (λ=0.1)	0.845
BERT base	0.908
BERTweet	0.918
Ensembling with BERTweet	**0.921**

Table II: Performance on `aicrowd` test set

VI. Conclusion

In conclusion, this project compared traditional methods and advanced transformer models for sentiment analysis on Tweets. While traditional approaches showed good performance, the real breakthrough came with BERTweet, achieving a promising 92.1% accuracy, which highlights the effectiveness of transformer models in Tweet sentiment classification.

VII. Ethical Risk

It is important to consider ethics when developing any Machine Learning model, especially in cases where the model makes decisions related to individuals. When it comes to sentiment classification, this concern is very prevalent: the dataset on which we train our model contains over two million tweets that portray a certain image of online discourse.

Machine Learning models are made to recognize patterns and utilize them in order to make better inferences, however even if these patterns do exist and may approve the model's performance, their identification may lead to decision-making that reflect the same discriminatory aspects that we as humans might have. In our context, the ethical risks would be that our model may be reflecting certain problems within its decision-making process, such as the amplification of certain stereotypes that may be harmful to certain minorities (i.e. falsely associating negative characteristics to certain groups).

Thus, there is an ethical risk of obtaining a model that includes certain biases when making predictions, which come from the patterns that it recognizes from the training dataset. Fairness through unawareness may not be sufficient to solve these problems : intentionally discarding sensitive attributes (such as the stereotypes) may only lead to aggravating them. Thus in order to avoid these ethical risks, the model would need to be regularly re-evaluated, and multiple fairness and transparency procedures would need to be implemented during the development phase of the model. However, it's crucial to acknowledge that this undertaking is challenging and demands careful consideration. It's important to note that successfully implementing such measures within the scope of this project would have required a dedicated effort.

The identification of the ethical risks discussed above was done upon reviewing the course lectures [9] and relevant literature on the ethics of sentiment analysis [10][11].

References

[1] J. Pennington, R. Socher, and C. Manning, "GloVe: Global Vectors for Word Representation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, A. Moschitti, B. Pang, and W. Daelemans, Eds. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. [Online]. Available: https://aclanthology.org/D14-1162

[2] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," May 2019, arXiv:1810.04805 [cs]. [Online]. Available: http://arxiv.org/abs/1810.04805

[3] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, A. Müller, J. Nothman, G. Louppe, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and Duchesnay, "Scikit-learn: Machine Learning in Python," Jun. 2018, arXiv:1201.0490 [cs]. [Online]. Available: http://arxiv.org/abs/1201.0490

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention Is All You Need," Jun. 2017, arXiv:1706.03762 [cs] version: 1. [Online]. Available: http://arxiv.org/abs/1706.03762

[5] D. Q. Nguyen, T. Vu, and A. Tuan Nguyen, "BERTweet: A pre-trained language model for English Tweets," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Q. Liu and D. Schlangen, Eds. Online: Association for Computational Linguistics, Oct. 2020, pp. 9–14. [Online]. Available: https://aclanthology.org/2020.emnlp-demos.2

[6] 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," Jul. 2020, arXiv:1910.03771 [cs]. [Online]. Available: http://arxiv.org/abs/1910.03771

[7] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. "O'Reilly Media, Inc.", Jun. 2009, google-Books-ID: KGIbfiiP1i4C.

[8] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," Jan. 2019, arXiv:1711.05101 [cs, math]. [Online]. Available: http://arxiv.org/abs/1711.05101

[9] N. Flammarion and M. Jaggi, "Machine Learning CS-433," 2023. [Online]. Available: https://www.epfl.ch/labs/mlo/machine-learning-cs-433/

[10] S. M. Mohammad, "Ethics Sheet for Automatic Emotion Recognition and Sentiment Analysis," Mar. 2022, arXiv:2109.08256 [cs]. [Online]. Available: http://arxiv.org/abs/2109.08256

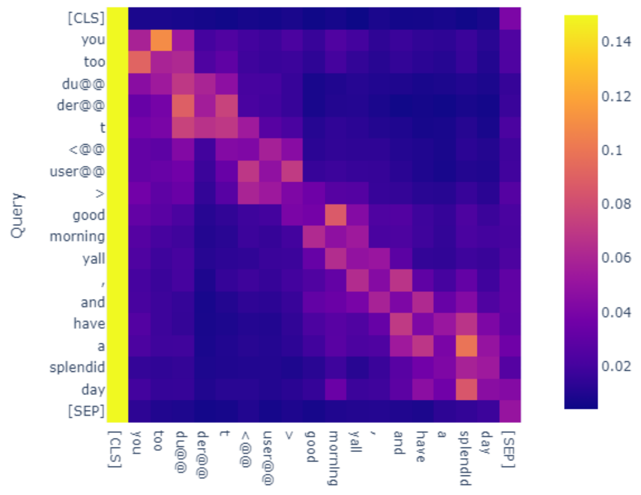
[11] K. Karoo and V. Chitte, "Ethical Considerations in Sentiment Analysis: Navigating the Complex Landscape," *International Research Journal of Modernization in Engineering Technology and Science*, Dec. 2023. [Online]. Available: https://www.irjmets.com/uploadedfiles/paper//issue_11_november_2023/46811/final/fin_irjmets1701254801.pdf

Appendix

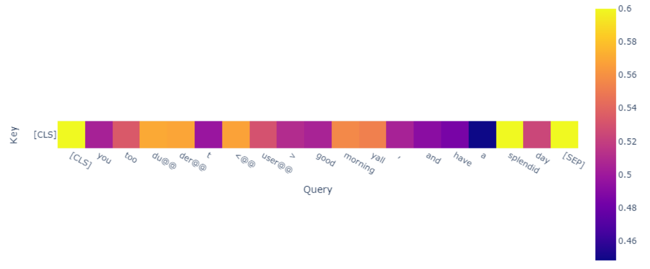
A. Transformers Visualizations

To gain deeper insights into our model's performance, we conducted attention map visualizations. Given that each layer has multiple attention heads, we averaged all attention maps over layers and heads. Taking the example tweet "you too dudert <user>good morning yall, and have a splendid day" correctly classified as positive, we generated its average attention map, as shown in Figure 3a. Notably, word pairs like ("you","too"), ("good","morning"), and ("a","splendid") exhibit high attention, aligning with the positive sentiment conveyed. Additionally, we explored attention between the

[CLS] token and other tokens, revealing the highest attention at the token "splendid," emphasizing the positivity of the message, as illustrated in Figure 3b.



(a) Attention map



(b) Attention between [CLS] and other tokens

Figure 3: Attention maps of "you too dudert <user>good morning yall , and have a splendid day" Tweet

Moreover, we visualized the classifier's performance on the [CLS] token. After splitting the data with an 80:20 train-test ratio, for each $[\mathrm{CLS}]_i$ in a tweet i of the testing data, we computed its corresponding [CLS] output C_i with a dimension of 768. These outputs were stored in a matrix

$$X = \begin{bmatrix} C_1 & C_2 & \dots & C_M \end{bmatrix}$$

with M being the number of tweets in the testing data. To facilitate visualization, Principal Components Analysis (PCA) was performed on the matrix X, retaining only 3 dimensions. The resulting matrix X' of dimensions $(3, M)$ was then plotted, as depicted in Figure 4.

Despite the significant dimensionality reduction, Figure 4 reveals a quasi-separation between the [CLS] outputs of positive and negative tweets. While the two clusters are not perfectly separable in three dimensions, the model successfully learned a representation of the classification token that

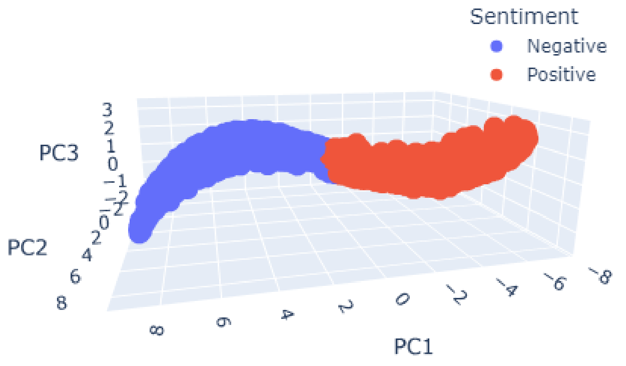


Figure 4: PCA of Classification Token of Testing Tweets

aids in distinguishing between positive and negative tweets. This underscores the model's efficacy in finding a meaningful representation of the classification token, contributing to its overall performance.

B. Hyperparameter Searches

1) TF-IDF Hyperparameters: Explanation of the hyperparameters :

- **λ** : regularization strength for the logistic regression model ;
- ***max_df*** : tokens that have a document frequency strictly higher than this threshold are omitted from the study (corpus-specific stop words) ;
- ***min_df*** : tokens that have a document frequency strictly lower than this threshold are omitted from the study (also called "cut-off" in the literature).

The results of the hyperparameter search can be seen in figures 5 and 6 below. For figure 5, the lineplots for $max\_df = 0.7$ and $max\_df = 0.8$ are covered by the $max\_df = 1.0$ lineplot. Since all of the plots are very similar in figure 5, this means that changing max_df hyperparameter doesn't yield significant differences when it comes to accuracy.

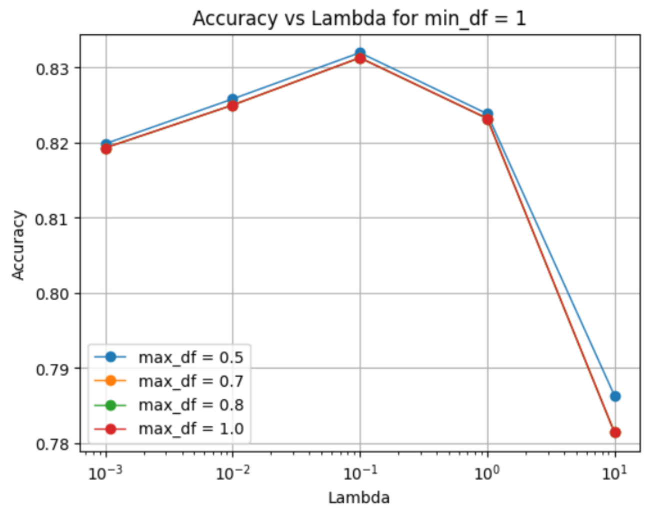


Figure 5: Accuracy vs λ for different values of max_df and fixed at optimal min_df value

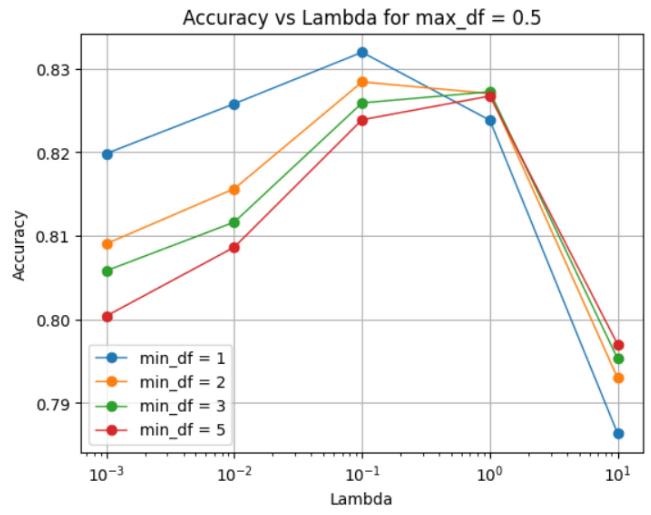


Figure 6: Accuracy vs λ for different values of min_df and fixed at optimal max_df value

2) GloVe Hyperparameters: We present in figure 7 the results of the hyperparameter search done on Lemmatization and stopwords.

Moreover, we present in figure 8, the hyperparameter search that was performed on the regularization coefficient λ of the logistic regression.

Furthermore, we explain below the meaning of other hyperparameters in the GloVe model:

- **embedding_dim**: This is the dimensionality of the embedding vectors, determining the size of the vector space for word representations. A lower dimension may not capture as much information, affecting performance, while a higher dimension can create more nuanced representations but increases computational complexity and the risk of overfitting.

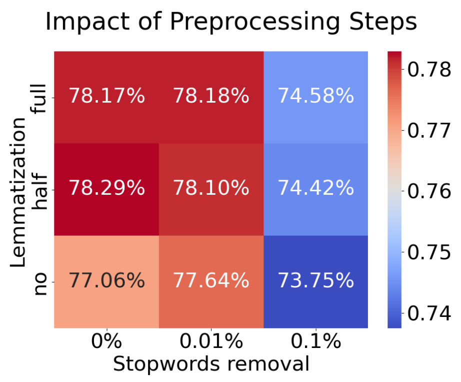


Figure 7: Impact of lemmatization method and stop words removal threshold on accuracy

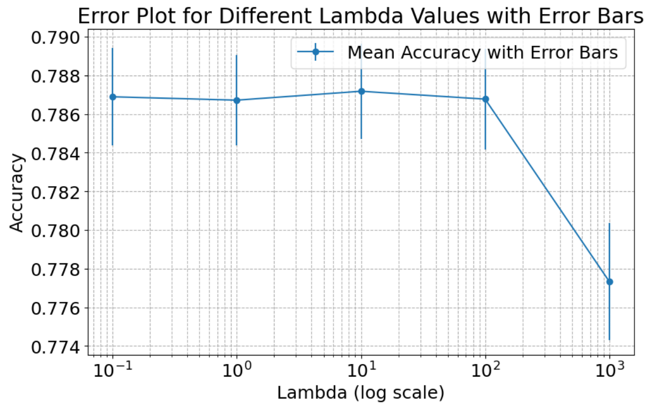


Figure 8: Accuracy vs λ with 5-fold cross validation

- **alpha**: A scaling factor affecting the weight of co-occurrence frequencies in the embeddings. When set low, it gives less importance to the frequency of word pairs, potentially undervaluing common associations. A higher setting increases the impact of frequency, which could overemphasize common word pairs and underrepresent rarer ones.
- **eta (learning rate)**: This rate dictates how much the embeddings adjust during each iteration. A lower learning rate leads to smaller updates and slower convergence but can aid in finer optimization. Conversely, a higher learning rate results in larger updates and faster convergence but may cause instability and risk overshooting optimal values.
- **epochs**: The total number of iterations through the entire dataset during training. Fewer epochs might lead to underfitting, as the model has less opportunity to learn, while more epochs allow for more extensive learning but can lead to overfitting, especially if the model begins to learn noise and specific patterns in the training data.
- **nmax**: This threshold for co-occurrence frequency influences the scaling of the weight of each co-occurrence. A lower threshold might ignore important

word associations that occur more frequently, whereas a higher threshold includes more word pairs but could dilute the influence of truly significant associations.

Lemma-tization type	Stop-words threshold	Embed-dings dimension	Alpha	Eta	Final epoch	NMax	Accu-racy
no _lem	0	20	0.75	1e-4	10	100	0.5878

(a) Accuracy of the Default GloVe Embeddings Model

Lemma-tization type	Stop-words threshold	Embed-dings dimension	Alpha	Eta	Final epoch	NMax	Accu-racy
half _lem	0	2000	0.85	1e-4	3	5000	0.7902
half _lem	0	2000	0.85	1e-4	7	5000	0.7896
half _lem	0	2000	0.75	1e-4	3	5000	0.7892
half _lem	0	2000	0.85	5e-5	5	6000	0.7892
half _lem	0	2000	0.85	1e-4	5	6000	0.7889

(b) Top Performing Models from Hyperparameter Search

Table III: Performance of various GloVe models

3) Transformers Hyperparameters: We present in figure 9 the results of the hyperparameter search done the learning rate.

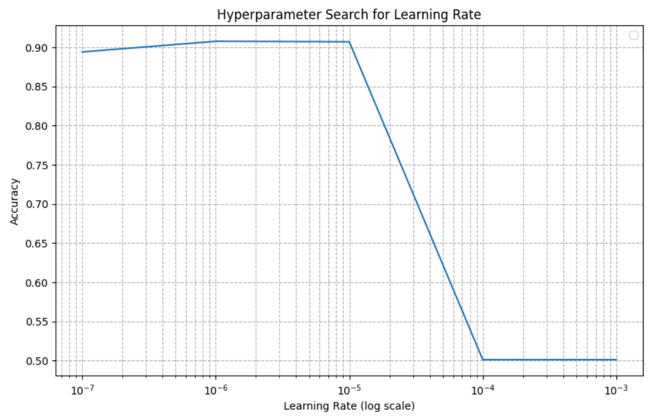


Figure 9: Learning rate hyperameter search