UNIVERSITY OF ATHENS
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

Deep Learning for NLP

Student name: *Christina Diamanti*
sdi: sdi1800046

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Spring Semester 2025*

Contents

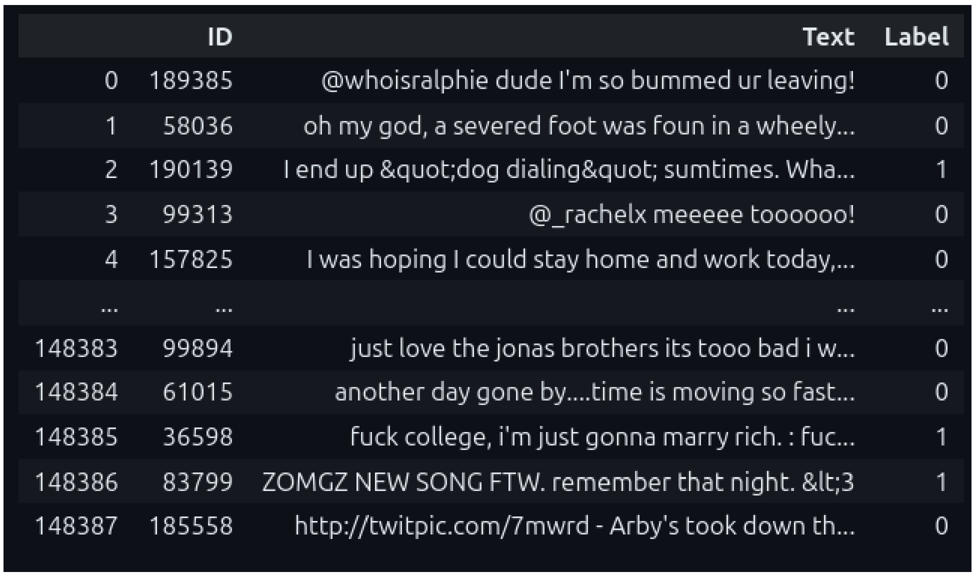
1. Abstract

This assignment is about developing a sentiment classifier using deep neural networks (DNN), with PyTorch machine learning framework and Word2Vec word embeddings as inputs, for English-language Twitter data. The dataset consists of tweets. Especialy, we have three csv files, one for training, one for validation and one for testing. Train and Validation set consists of three columns ID, Text and Label (to label every text as positive or negative - with 1 and 0), the Test set is used only for predictions and consists of two columns ID and Text.
The task involves Exploratory Data Analysis (EDA), text preprocessing, model training and evaluation using metrics such as accuracy, precision, recall, and F1-score, and model testing.
To achieve this, we preprocess the text by removing unnecessary elements and convert it into numerical features by training two different types of Word2Vec word embeddings, CBOW and Skip-Gram. Then we train the DNN model using both types of word embeddings and we choose the better one. Finaly we optimize the model by changing its architecture and hyperparameters, and evaluate its performance.

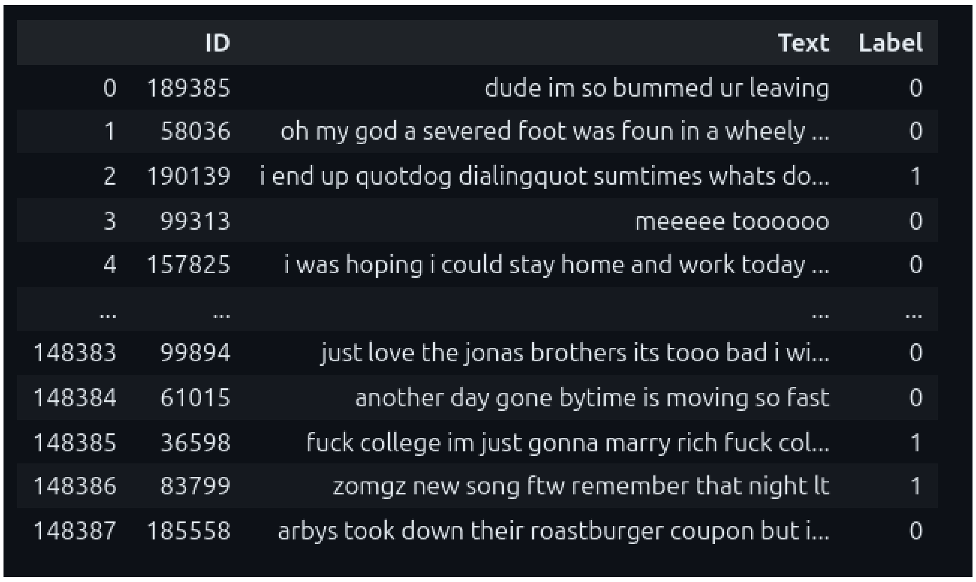
2. Data processing and analysis

2.1. Pre-processing

Pre-processing is the most important step to ensure that the data is clean and properly formatted for model training. First, unwanted symbols such as HTML, @mentions, numbers, and special characters were removed, and all text was converted to lowercase. After that, missing values (NaN) were dropped. You can see below the text (of train dataset) before cleaning and then the cleaned text column.

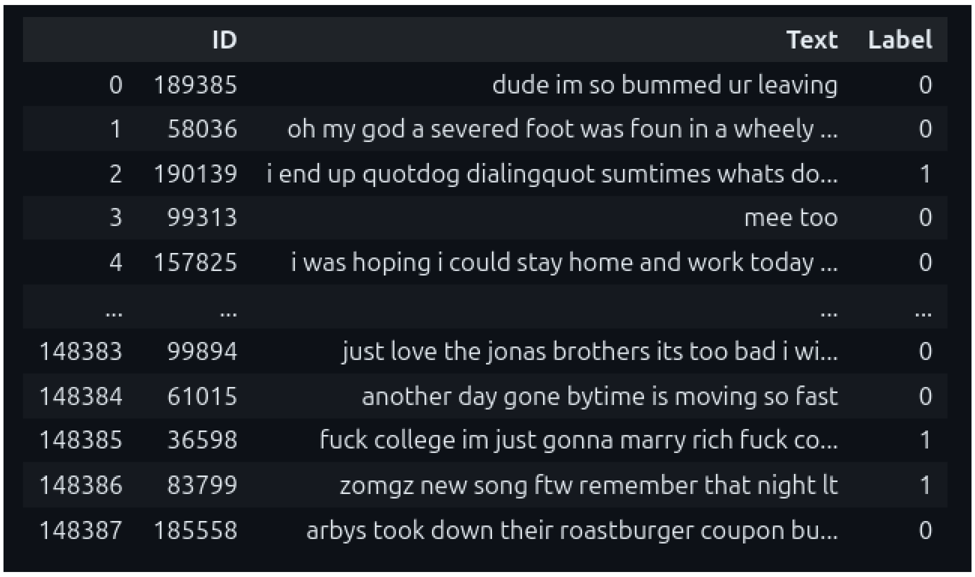


	ID	Text	Label
0	189385	@whoisralphie dude I'm so bummed ur leaving!	0
1	58036	oh my god, a severed foot was foun in a wheely...	0
2	190139	I end up &quot;dog dialing&quot; sumtimes. Wha...	1
3	99313	@_rachelx meeeee toooooo!	0
4	157825	I was hoping I could stay home and work today,...	0
...	...	...	...
148383	99894	just love the jonas brothers its tooo bad i w...	0
148384	61015	another day gone by....time is moving so fast...	0
148385	36598	fuck college, i'm just gonna marry rich. : fuc...	1
148386	83799	ZOMGZ NEW SONG FTW. remember that night. <3	1
148387	185558	http://twitpic.com/7mwrd - Arby's took down th...	0



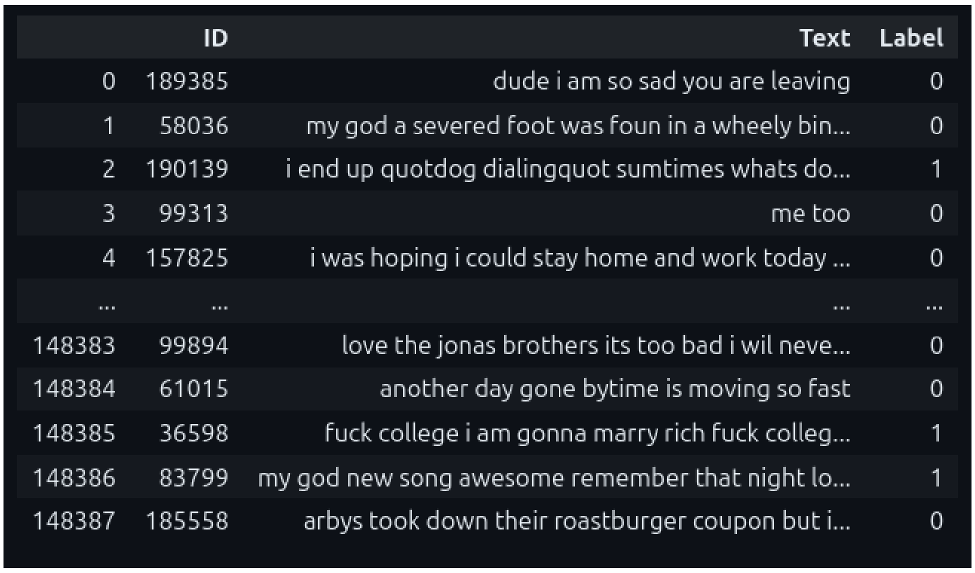
	ID	Text	Label
0	189385	dude im so bummed ur leaving	0
1	58036	oh my god a severed foot was foun in a wheely ...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	meeeee toooooo	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	just love the jonas brothers its tooo bad i wi...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college im just gonna marry rich fuck col...	1
148386	83799	zomgz new song ftw remember that night lt	1
148387	185558	arbys took down their roastburger coupon but i...	0

Seeing that many Tweets have repetitive characters in a row, (such as "meeeee tooooo" and "tooo bad" in the example above) we used a function to reduce these repetitive characters in two. Below you can see the same text column with reduced characters.



	ID	Text	Label
0	189385	dude im so bummed ur leaving	0
1	58036	oh my god a severed foot was foun in a wheely ...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	mee too	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	just love the jonas brothers its too bad i wi...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college im just gonna marry rich fuck co...	1
148386	83799	zomgz new song ftw remember that night lt	1
148387	185558	arbys took down their roastburger coupon bu...	0

The words "got", "oh" and "just" appear to be too frequent and meaningless for model training so we remove them. Also, we replace some slang words (such as "ur", "ily", "fb") with their original form. Below you can see the final cleaned text column of train dataset. We apply the same text processing in all three datasets.

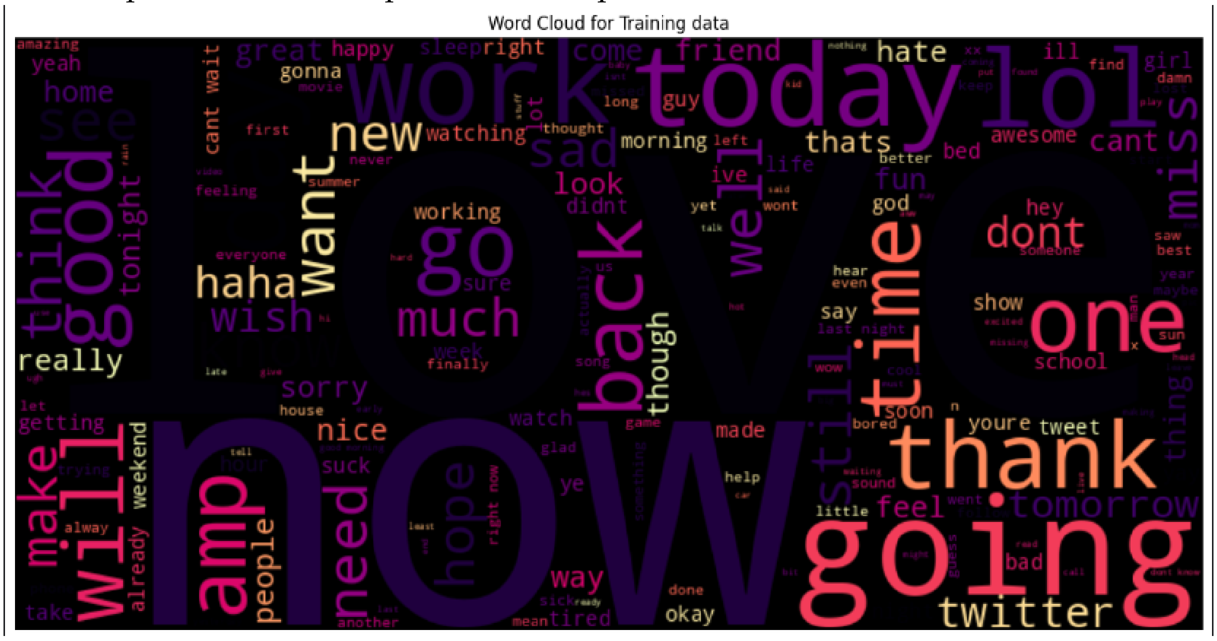


	ID	Text	Label
0	189385	dude i am so sad you are leaving	0
1	58036	my god a severed foot was foun in a wheely bin...	0
2	190139	i end up quotdog dialingquot sumtimes whats do...	1
3	99313	me too	0
4	157825	i was hoping i could stay home and work today ...	0
...	...	...	...
148383	99894	love the jonas brothers its too bad i wil neve...	0
148384	61015	another day gone bytime is moving so fast	0
148385	36598	fuck college i am gonna marry rich fuck colleg...	1
148386	83799	my god new song awesome remember that night lo...	1
148387	185558	arbys took down their roastburger coupon but i...	0

The removal of Stopwords (such as a, an, the, of, in, for) did not ease the model training process, on the contrary, it made model training more difficult and led to a reduction in accuracy and overfitting. For this reason, this technique was removed from the final project. Below, in the Trial Table you will see this difference in results.

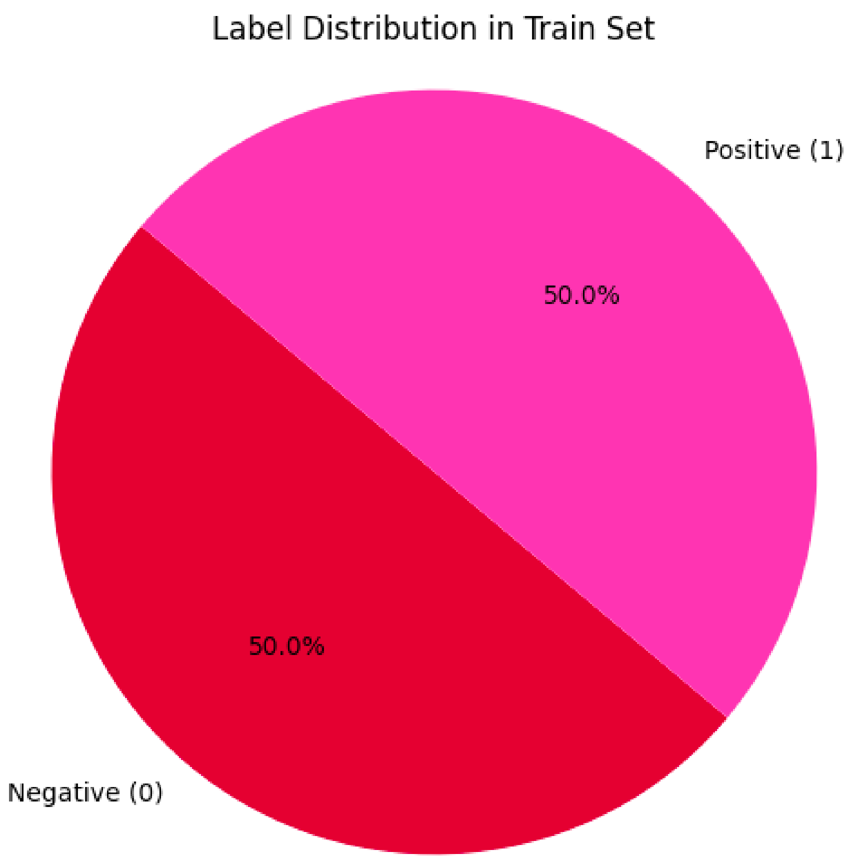
2.2. Analysis

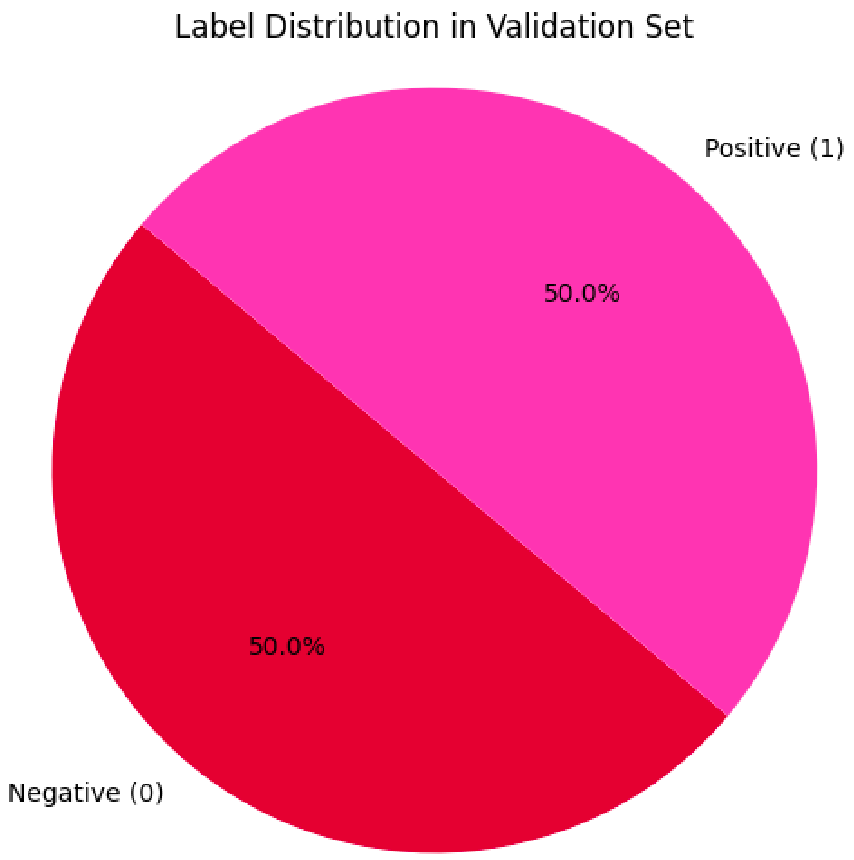
To get a clearer picture of our data we tried to visualize it by using wordclouds and also using pie charts to depict the number of labels per dataset. We also made a plot of the Top 10 most frequent words per label. Below you can see the wordcloud of every dataset, the pie charts and the plots for the Top 10 words.

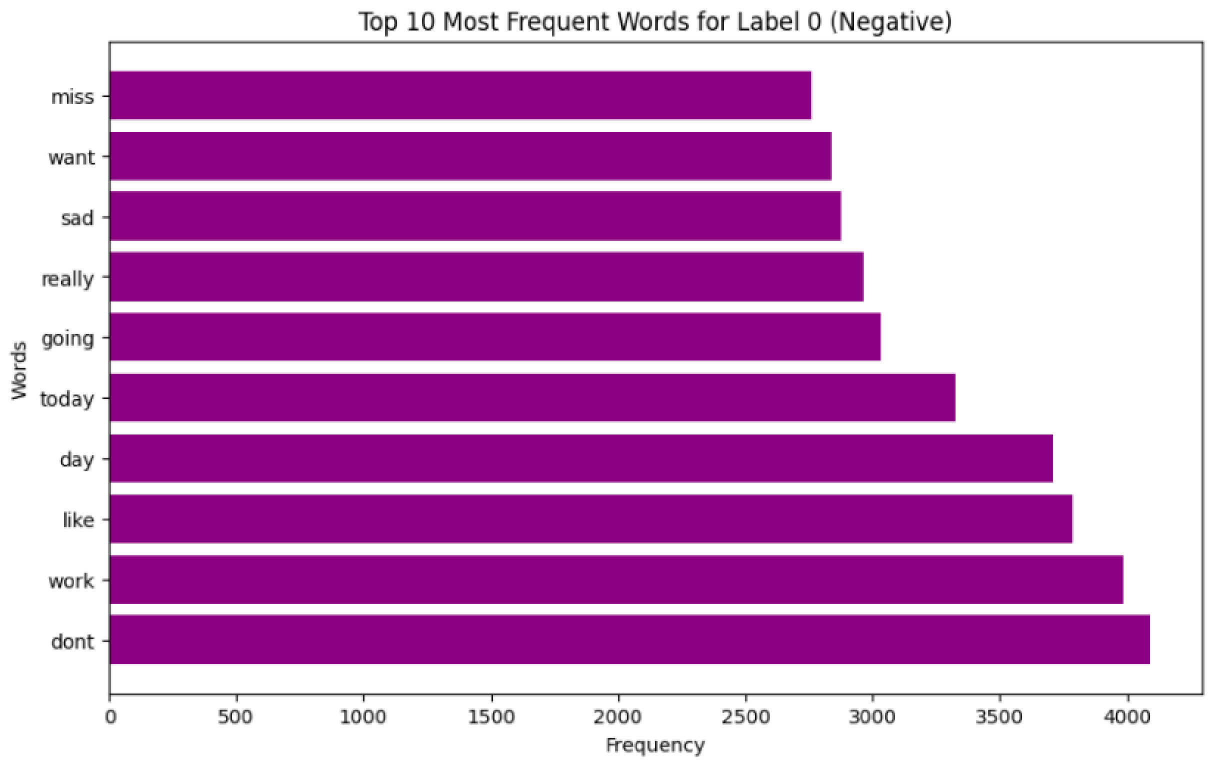


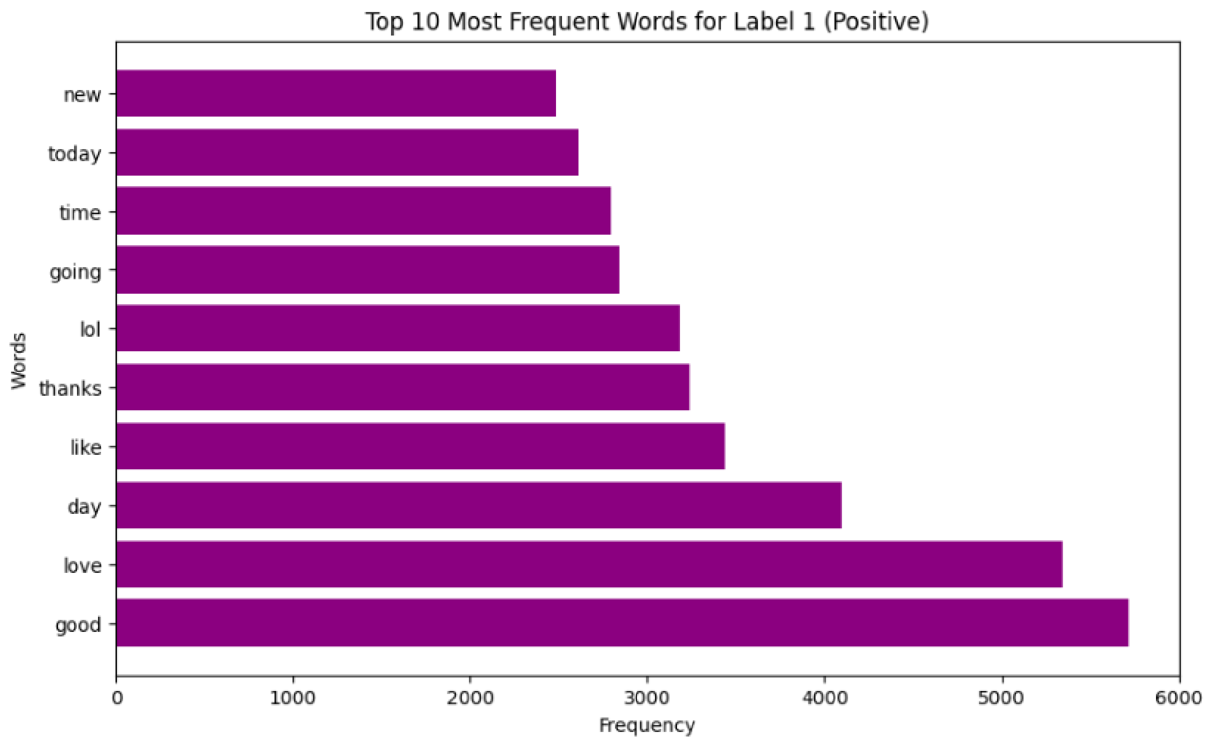
Word Cloud for Validation data

Word Cloud for Tetsing data


Label Distribution in Train Set
Positive (1)
50.0%
50.0%
Negative (0)


Label Distribution in Validation Set
Positive (1)
50.0%
50.0%
Negative (0)





2.3. Data partitioning for train, test and validation

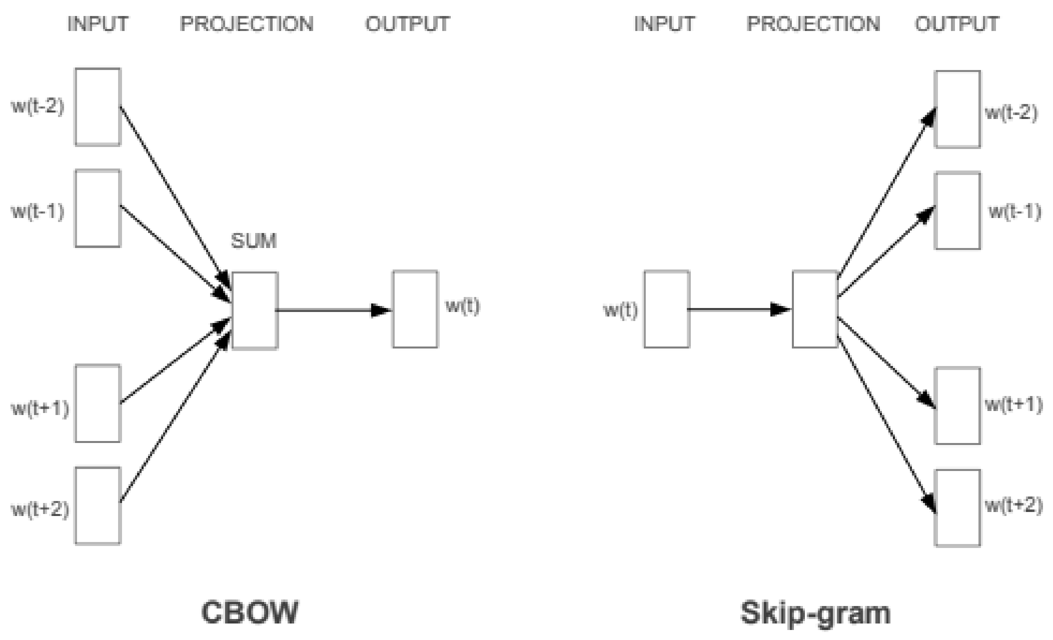
The datasets were already split into 3 parts, Train set was used to train the DNN model, Validation set was used to evaluate the predictions of the model based on metrics such as accuracy, precision, recall, and F1-score and Test set was used to make predictions with our trained model and save them in a csv file.

2.4. Vectorization

In this implementation, two different methods of the Word2Vec word embeddings were used and trained with our dataset. The CBOW (Continuous Bag of Words) model and the Skip-Gram model were trained and then used in DNN model training to determine which one achieves the best accuracy. The most important difference between the CBOW and Skip-gram models in Word2Vec is the way they make predictions.

CBOW (Continuous Bag of Words) tries to guess the target word by looking at its surrounding context words within a given window size (in this implementation window=5). In contrast, the Skip-gram model does the opposite. It uses a single target word to predict the words that appear in its surrounding context. CBOW is usually faster and works better with large amounts of text. However, it might not work so well with rare words (in this dataset there are many slang and rare words). Skip-gram is a bit slower but does a better job when dealing with less common words or smaller datasets.

Bellow you can see an image showing the difference between the way CBOW and Skip-Gram make predictions.



3. Algorithms and Experiments

3.1. Experiments

In this assignment we build a feedforward DNN classifier with three layers and one output neuron.

First, we trained our DNN model using CBOW word embeddings, save the metrics and then trained it again using Skip-Gram word embeddings and compared the results (We have word embedding vectors with vector_size = 100 for both methods). Below are two tables with trials, the first one has the experiment with CBOW and the second one with Skip-Gram (trial 1 in both cases). Then, based on the metrics results, we chose to use the Skip-Gram word embeddings to train and optimize our model, because it seems that these word embeddings handled more correctly the rare words that appear in the twitter text and helped DNN model to make better predictions.

In the first trial (trial 1 - Table 2), we removed stopwords from text, then, in the second trial (trial 2 - Table 2) we kept stopwords in text, and the model made much better predictions as shown in the table below. In the third trial (trial 3 - Table 2) we continued experimenting in text preprocessing, so we tried to remove some frequent words that are meaningless and don't give real emotions (words such as "oh", "just", "got") and the results were positive.
Then, to increase the accuracy, experiments and changes were made using hyperparameters. In particular, increasing epochs to 15, 20 and 25 did not change anything, probably because the model has already learned the best possible from the data and more epochs do not offer any further improvement. We end up keeping epochs = 10 and then we tried to change optimizers and see the difference. We tried SGD (that was the default we used in trial 1 and 2), Adam and AdamW, the results were not very different between Adam and AdamW, so we chose AdamW optimizer as it gave slightly better results (trial 4). Then we add the hyperparameter weight_decay=1e-5 (for L2 regularization) to reduce over-fitting and chose as learning rate lr=0.001 (trial 5). In trial 6 we change the architecture of the DNN model by adding BatchNorm1d in every layer to make the training process more stable, reduce over-fitting and stabilize training and evaluation loss. We also add Leaky ReLU to avoid dead neurons (trial 7).

Changes in batch size were made, in particular, we tried to increase the size $\rightarrow batch\_size = 64$, but that decreased accuracy (trial 8) so we tried smaller sizes $\rightarrow batch\_size = 32$ and $\rightarrow batch\_size = 16$. The last one gave quite the same results as batch_size = 32 but needed much more training time, so we kept batch_size = 32.

The next experiment involve changes in architecture of DNN, especially in the dimensions of the layers. At first we had (input_dim, 60) for first layer (with input_dim = 100), (60,60) for second layer and (60,60) for third layer. Then we tried different dimensions between layers, (input_dim, 100) in the first layer (helps model have more space to save information), (64,32) in the second layer and (32,16) in the third. The results were possitive since accuracy increased as it seems in the table below (trial 9).

Finally, we tried to increase the dimension of word embeddings vector, at first we used vector_size = 300 and input_dim = 300 (trial 10). We observed that this increment of vector dimensions helped the model learn better, as is seems that with this change vectors had more information about the relationships between words. In the end, we increased more the size and we had vector_size = 500 and input_dim = 500. We also adapt the architecture of DNN by setting the dimensions of the first layer in (input_dim, 192), the second layer in (192, 64) and the third layer in (64, 32), the results are below in trial 11.

3.1.1. Table of trials. Below there is a table that contains the trials described.

Trial	Precision	Recall	F1-score	Accuracy
1	0.6974	0.8273	0.7568	73.43%

Table 1: Trials with CBOW word embeddings

Trial	Precision	Recall	F1-score	Accuracy
1	0.7474	0.7669	0.7570	75.39%
2	0.7975	0.7113	0.7520	76.54%
3	0.7655	0.7737	0.7696	76.84%
4	0.7712	0.7673	0.7693	76.99%
5	0.7945	0.7305	0.7612	77.09%
6	0.7872	0.7457	0.7659	77.21%
7	0.7601	0.7993	0.7792	77.36%
8	0.7572	0.8044	0.7801	77.33%
9	0.7809	0.7639	0.7723	77.49%
10	0.7792	0.7700	0.7746	77.60%
11	0.7936	0.7598	0.7764	78.12%

Table 2: Trials with Skip-Gram word embeddings

3.2. Hyper-parameter tuning

The optimization of hyperparameters (we used manual tuning) had a limited effect on the performance of DNN model, with accuracy ranging between 76% – 77%. This shows that the model did not suffer heavily from over-fitting or under-fitting, and that its performance is more influenced by the nature of the data than by tuning.The hyperparameters we use in Word2Vec training are: **vector_size=500, window=5, min_count=2, sg=1, seed=42** for Skip-Gram word embeddings. Window = 5 gives a wider view of the words around the keyword and also with min_count = 2 does not take into account words that appear in the text less than 2 times. As mentioned above, the size of the word embedding vector seriously affected the performance of the model, since with a larger vector we have better results in both training and evaluation mode.

3.3. Evaluation

To evaluate the predictions of the model, we used val_dataset.csv and the metrics Accuracy, Precision, Recall and F1-score. The accuracy of the model reached 78.12%, while Precision (0.7936) and Recall (0.7598) show that the model maintains a good balance between false positives and false negatives.

F1-score (0.7764),

$$F1 = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

as a harmonic mean of Precision and Recall, confirms the stable performance of the model.

Below there are plots and diagrams of ROC curve, Learning Curve and Confusion matrix.

3.3.1. ROC curve. The Receiver Operating Characteristic (ROC) curve depicts the relationship between True Positive Rate (TPR) and False Positive Rate (FPR). Area Under the Curve (AUC) measures the overall performance of the model, in the plot below, the ROC curve with AUC 0.86 shows that the model has good classification ability (since AUC > 0.8).

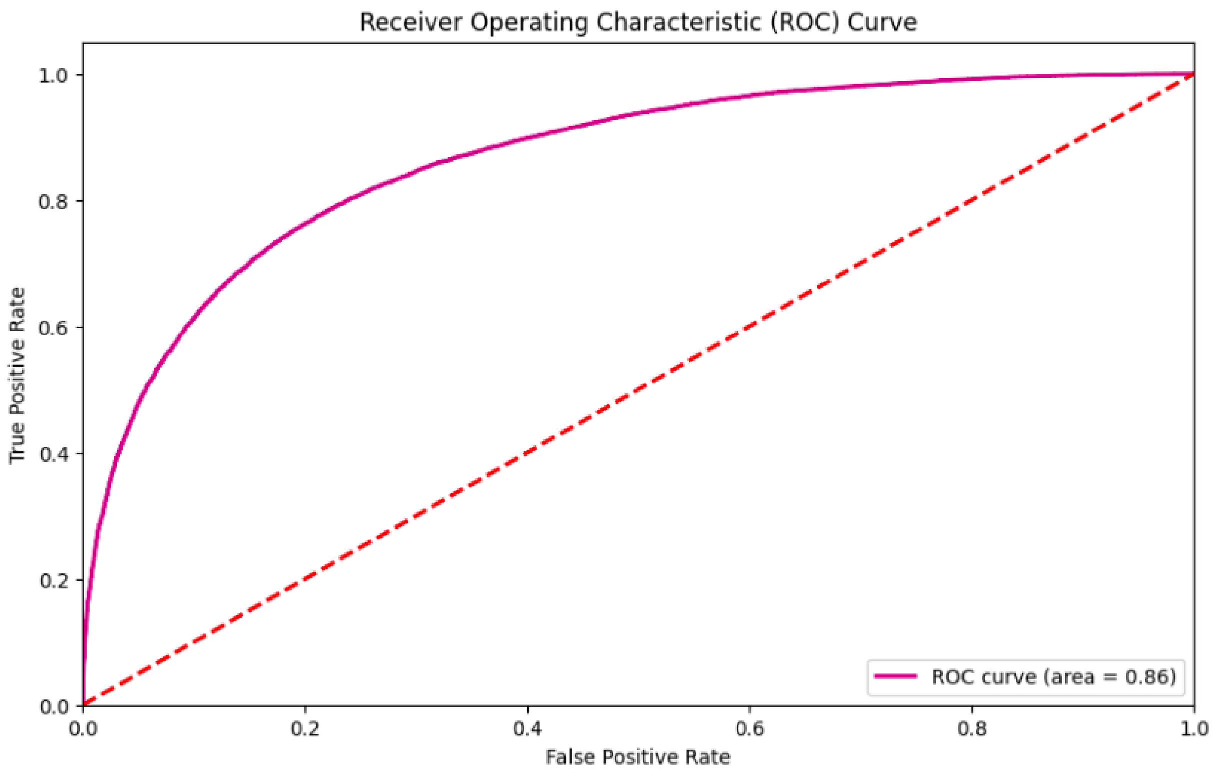


Figure 1: Trial 11 ROC curve

3.3.2. Learning Curve. The model was trained for 10 epochs. As shown in plot below, both the training loss and validation loss decrease steadily, which indicates that the model is learning effectively. Around epoch 6, the validation loss stops improving clearly and starts to vary slightly, which could mean that the model is reaching a limit in performance or starting to slightly over-fit.
In the accuracy plot, we see that training accuracy increases consistently. Validation accuracy also improves overall, but with some small changes from one epoch to the next after the 5th epoch. Despite this, the model keeps a good balance and shows no serious signs of over-fitting.

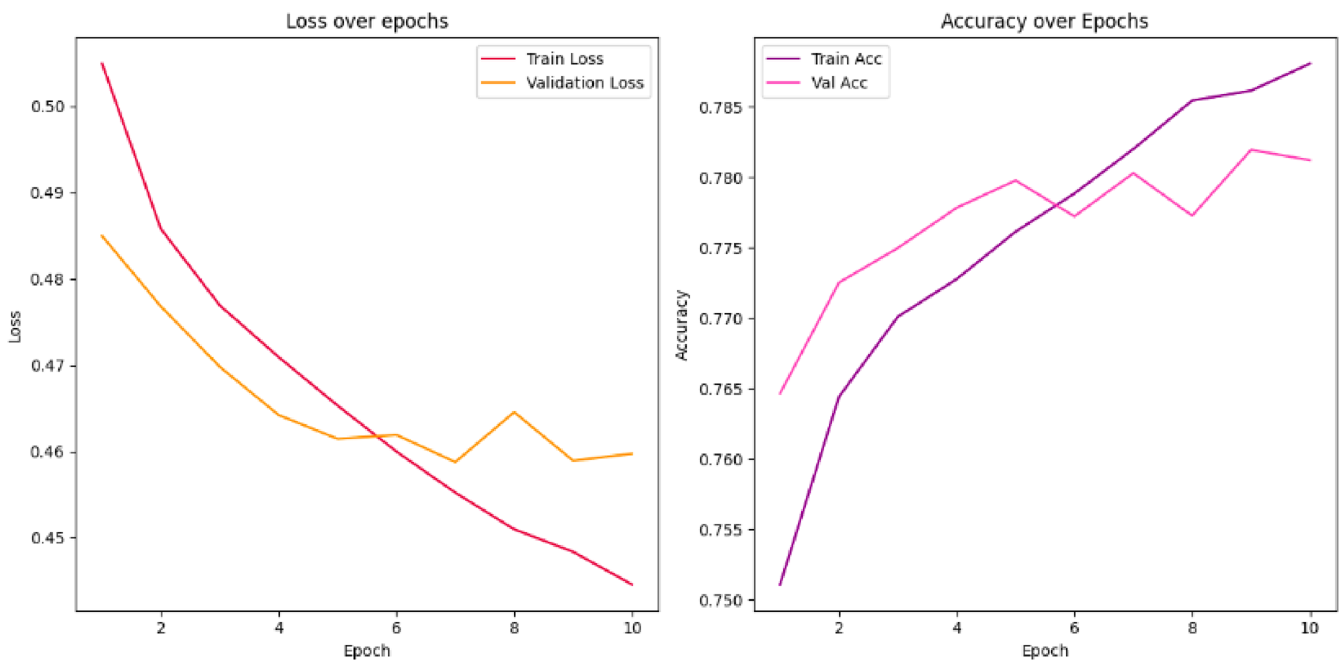


Figure 2: Trial 11 Learning curve - Loss and Accuracy

3.3.3. Confusion matrix. After data preprocessing and hyperparameter tuning of the Word2Vec and DNN model, we observe improvements in its performance. False Positives are 4.179, True Negatives 16.986, False Negatives 5.080 and True Positives 16.072.

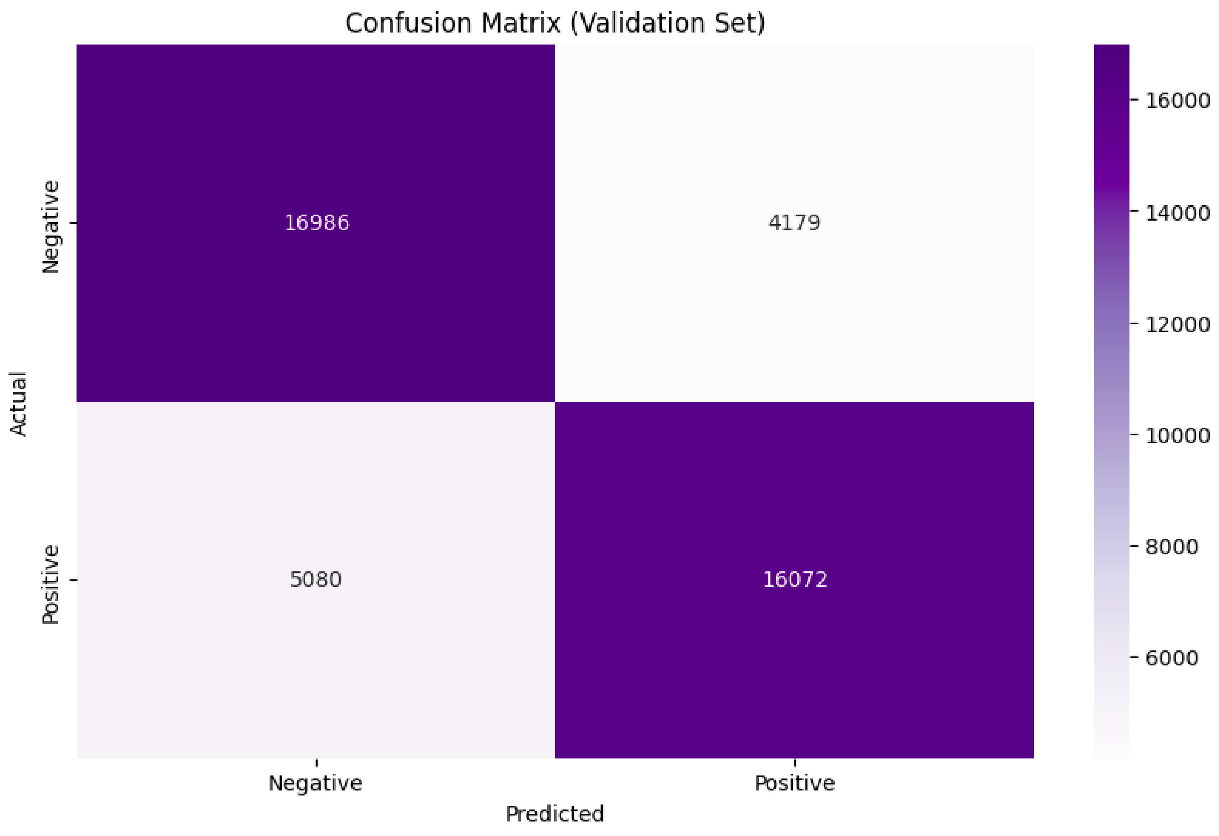


Figure 3: Trial 11 Confusion matrix

4. Results and Overall Analysis

4.1. Results Analysis

The results show that the best performance of the feedforward DNN model was achieved with three layers, with dimensions (input_dim, 192) for the first layer, (192, 64) for the second layer and (64, 32) for the third layer. Using AdamW optimizer with hyperparameters $lr = 0.001$ and $weight\_decay = 1e-5$. Also, we ended up in using Word2Vec Skip-Gram model for word embeddings, which we trained using our data and with hyperparameters: vector_size=500, window=5, min_count=2, sg=1, seed=42. We use batch_size=32 and epochs=10 for training.

The final accuracy of the model was 78.12%, which is satisfactory but not excellent. The F1-score (0.7764) indicates that the model has a good balance between recall and precision, which means that it does not overly favor one category over the other.

The ROC Curve showed that the AUC improved from 0.85 to 0.86 after applying optimizations, indicating that the model has good discriminatory power. However, the Learning Curve (Loss - Accuracy) shows that the model sometimes approaches over-fitting and tries to avoid it. In addition, the performance on the validation set remains more unstable than on the training set.

Overall, the model performs well, but there is certainly room for improvement. To reduce over-fitting we could try adding more data. Also, we could use a different vectorization method from Word2Vec.

4.1.1. Best trial. The best experiment (Trial 11) achieved a maximum accuracy of 78.12%, which was the highest of all the other trials.

The final metric values for the Best Trial (Trial 11) are:
Accuracy: 78.12%
Precision: 0.7936
Recall: 0.7598
F1-score: 0.7764

4.2. Comparison with the first project

The results of the first project - Logistic Regression with TF-IDF are:
Accuracy: 80.06%
Precision: 0.8064
Recall: 0.7911
F1-score: 0.7987

The results of the second project - DNN model with Word2Vec are:
Accuracy: 78.12%
Precision: 0.7936
Recall: 0.7598
F1-score: 0.7764

The comparison between the two models — Logistic Regression with TF-IDF features and Deep Neural Network (DNN) using Word2Vec word embeddings — reveals that the simpler Logistic Regression model outperformed the DNN in all evaluation metrics. Specifically, it achieved higher accuracy (80.06% vs. 78.12%), precision (0.8064 vs. 0.7936), recall (0.7911 vs. 0.7598), and F1-score (0.7987 vs. 0.7764).
This suggests that, in this particular task, the traditional TF-IDF representation combined with a linear classifier was more effective than the more complex DNN architecture with dense word embeddings. One likely reason is that the Word2Vec embeddings were trained solely on the project-specific dataset, which may not have been large or diverse enough to capture rich semantic relationships between words.

5. Bibliography

References

[1] Pytorch, 2025.

[2] Baeldung. Word embeddings: Cbow vs skip-gram, 2024.

[3] Shivam Baldha. Binary classification with pytorch, 2023.

[4] GeeksforGeeks. Auc-roc curve, 2024.

[5] Daniel Jurafsky and James H. Martin. Speech and language processing, chapter 6, 2024.

[6] Daniel Jurafsky and James H. Martin. Speech and language processing, chapter 7, 2024.

[7] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[8] Rishabh Singh. Learning word embeddings with cbow and skip-gram, 2024.

[9] V7 Labs. F1 score guide, 2024.