SUBJECT:**DEEP LEARNING**

PROFESSOR: **THEODOROS GIANNAKOPOULOS**

TEXT SENTIMENT FROM TWITTER

NAME	STUDENT IDENTIFICATION NUMBER
Andrianakou Stavroula	2022202404018
Kapelou-Lampazevits Ioustina	2022202404003
Kyriakidis Giorgos	2022202404022

Exploratory data analysis

This part of the project aims to give an outlook of the data so that they will be more understandable.

The selected dataset is this:

https://www.kaggle.com/datasets/jp797498e/twitter-entity-sentiment-analysis/data.

The task is to judge the sentiments from different messages(tweets/texts).

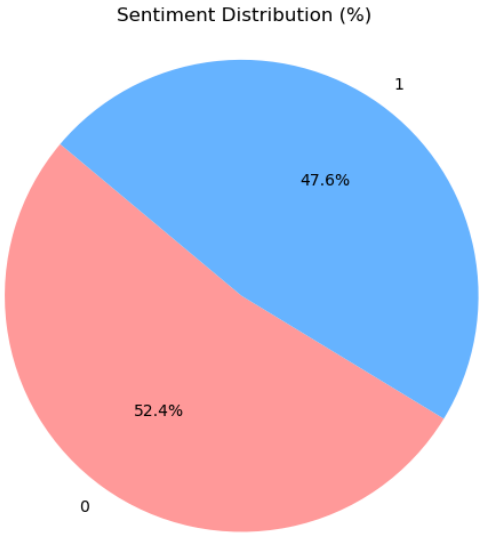
The first thing to do is to load the dataset. Then, we get to see the dataset, how many rows and columns it has, the first 5 rows and the information about the dataset such as the type of the variables.Data preprocessing must be implemented so the data have a proper format for the neural networks. The data did not have any headers so we inserted the one that we thought where appropriate. Then we dropped the data that had 0 values the sum of that was only 686 and it is a small amount compared to the dataset that is approximately 70000 instances. We dropped all the duplicate instances meaning the one that had the exact same values in every attribute with another instance. Their amount again is small only 2340. Then we decided to keep only the instances with the label Positive and Negative and as a result we had 41410 instances After that we remove the values “Neutral” and “Irrelevant” from the variable ‘sentiment’. We then map the Positive and Negative values with the numbers 1 and 0 respectively. We then get a look at the frequencies that each of the two remaining values of the variable ‘sentiment’ holds as it is seen below:

sentiment

0 21698

1 19712

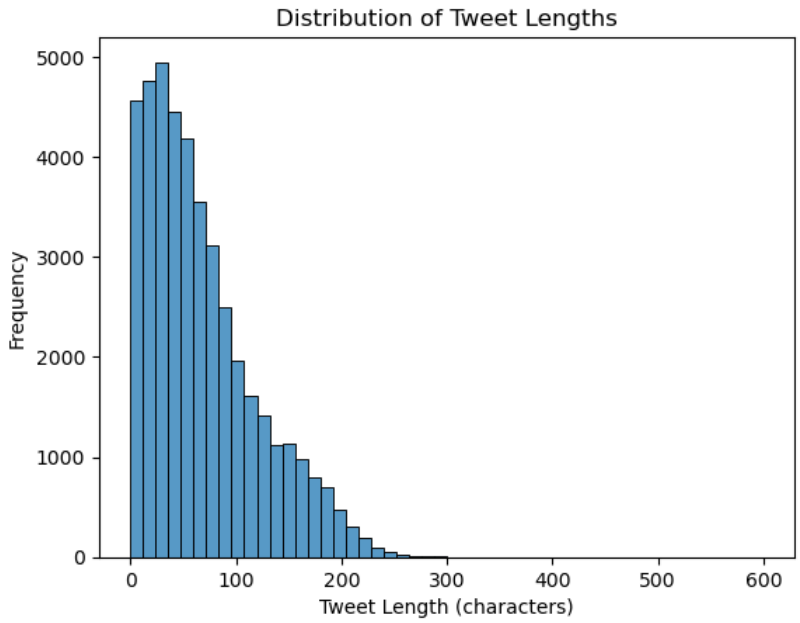
We create a pie chart that shows the percentage that is held from each value



Comments: As we can see from the diagram above, 47.6% of tweets hold a 'positive sentiment' while 52.4% of tweets hold a 'negative sentiment'.

We then see the top 10 entities from the dataset and we group it with the variable 'sentiment' to see how many negative or positive values there are for each entity.

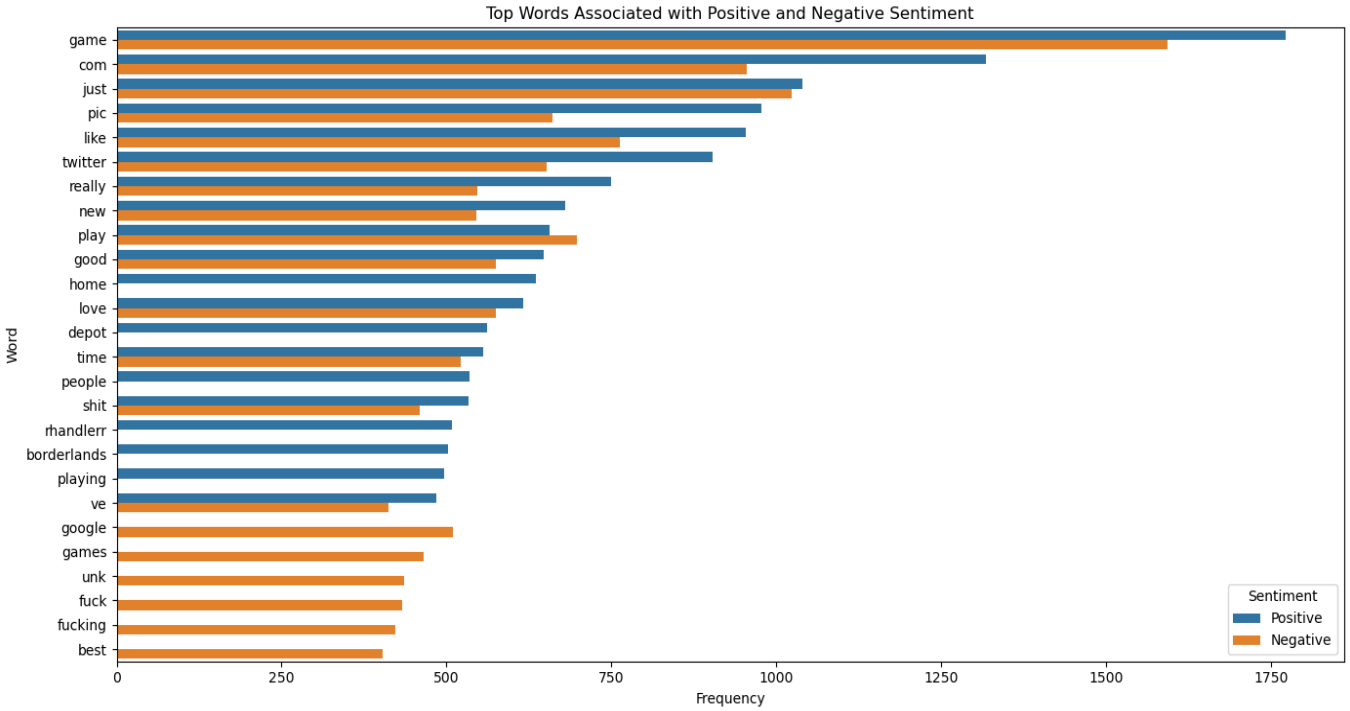
After that we move onto normalizing text, removing noise, tokenizing and filtering out unhelpful words and preserving semantic meaning(like emojis!). We apply the clean_text function to every entry in the text column of our df and we create a new column called cleaned_text with the cleaned version of each text. We create a new column called text_length to see how many characters there exist for each tweet/text and we create a histogram of these values.



Comments: As we can see, 4.500 tweets contain 0-10 characters, 4.700 tweets contain 10-20 charactersand almost 5.000 tweets contain 20-30 characters. After that, as the characters length grow, the number of tweets decreases

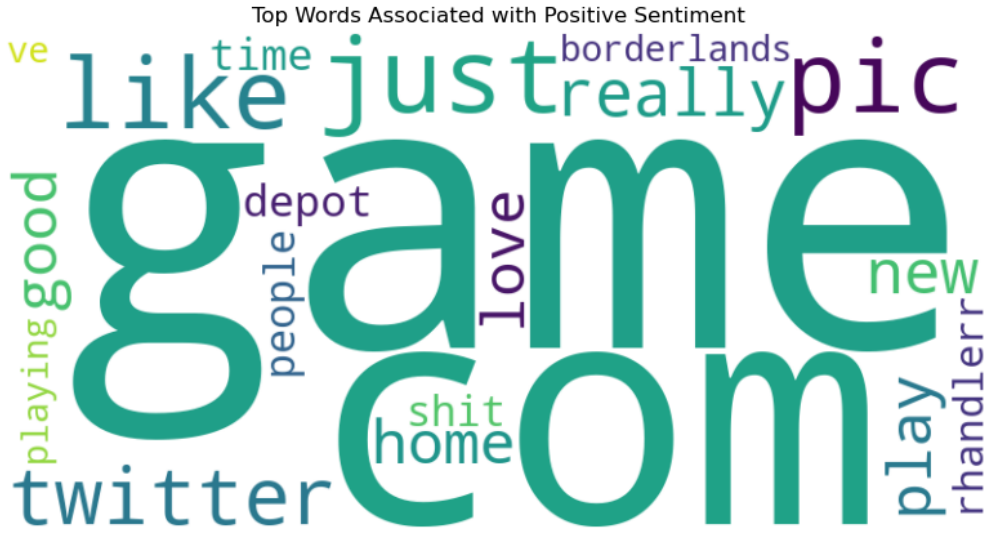
We then drop the columns ‘tweet_id’, ‘entity’, ‘text’ and ‘text_length as we don’t need them for the analysis. We convert the cleaned text (cleaned_text) into a list of word tokens and store it in a new column ‘tokens’. We download the 100-dimensional GloVe word vectors trained on Wikipedia + Gigaword. Each word (e.g., 'king') is mapped to a 100-dimensional vector. We print the first 10 values of the 'king' word vector as an example. We then convert words to vectors and we apply the function to all tokenized texts to generate a new column of vector representations. We ensure that all vectors are exactly 100 elements long. This is mostly a safeguard — average vectors should already be size 100, but this guarantees compatibility with models expecting a fixed length and we print the result of the vectors.

We then convert texts into word count features, separate words by positive/negative sentiment, identify top words most associated with each sentiment and visualize those words and their frequencies side-by side in a bar chart



Comments: As seen above, out of all the times the word ‘game’ appeared in a tweet, almost 1.750 times was connected to a ‘positive sentiment’ while 1.600 times was connected to a ‘negative sentiment’

We then create an image with the top words connected with the Positive and Negative sentiment. The words that are most connected with the respective sentiment are bigger than the words that are less connected with the sentiment. Additionally, we can see that the word game is connected with both positive and negative sentiment but this word is more frequent in the positive sentiment.



Data preprocessing

Now when it comes to the attribute text data cleaning where implemented. First of all emojis were spotted in the text so we converted the emoji face to text because the emojis are crucial and reflect the sentiment in a text if it is positive or negative therefor, we did not delete them but converted them to text.Next we removed from the text the https, @, www characters because they do not provide any information and are useless for the neural network. We removed the string punctuation (!"#$%&'()*+,-./:;<=>?@[\]^_`{|}~) and converted all the words to lowercase so the neural network understand that it is the same word either it is in lowercase or not. We removed the stop words because again they do not provide any information.Last but not least the clean text we received after the cleaning we converted it to tokens.

We had to convert each word to vector. For this assignment a pre-trained word2vec model was allowed to be used. We chose the glove-wiki-gigaword-100. We did it strategically because we wanted our neural network to generalize good se we decided that it should receive vectors from a general language that is used in Wikipedia. For this reason, we did not choose models created for tweets because the objective is to be able to use our network and on data from different domains, so the general language is the solution to this objective. For the simple forward neural network we just created the average for each sentence (tweet). For the CNN and RNN (simple RNN, bidirectional RNN, bidirectional LSTM) a different technique was used. For each word in each tweet, we created a vector we stacked them together and created a table(matrix) for each tweet. As a result we had

(41410, 50, 100)

Each input is a matrix of shape 50 words 100 dimension each. We did that because the CNN and RNN work sequentially meaning that they take into consideration each word at the time (the RNN have the ability to remember words because of their integrated memory). As a result, using the average would cancel the sequence of these two types of neural networks.

Neural network architectures

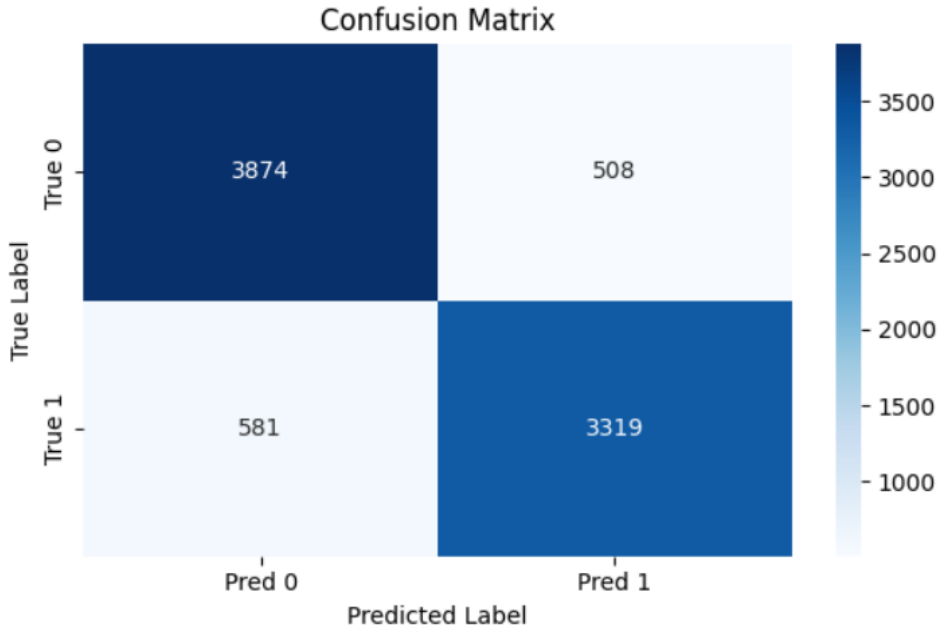
Simple forward neural network

First, we split the train dataset into 80 percent for train and 20 percent for validation. And we also have a hidden test dataset that will be used to test the neural network.

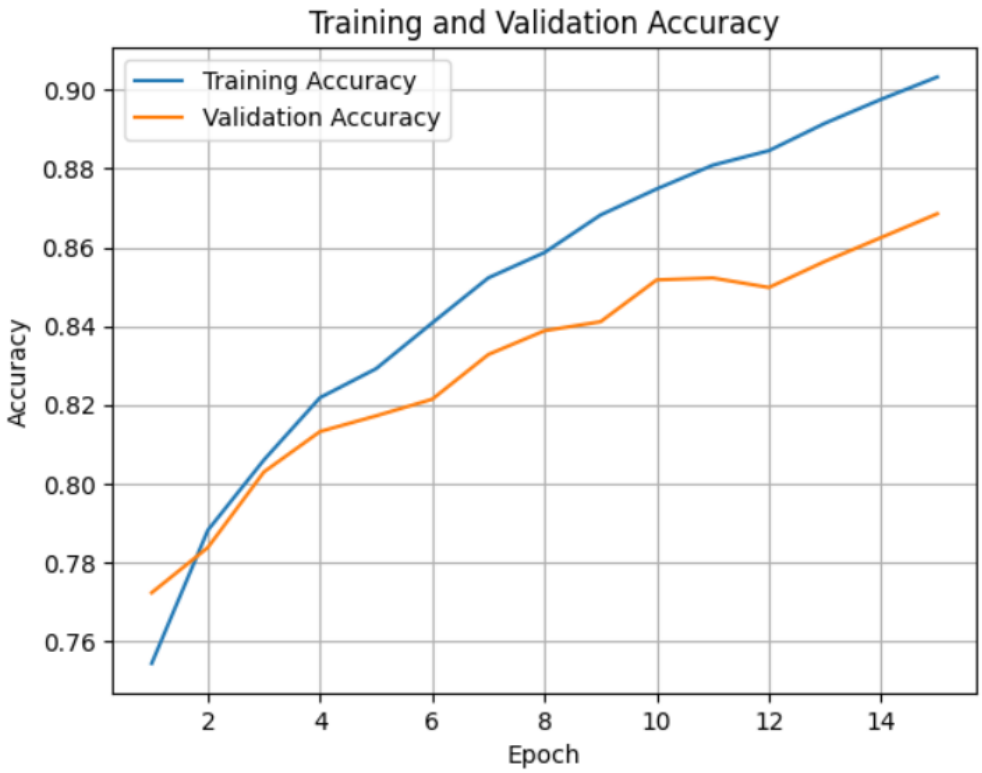
We have decided to start with a simple forward network thar will either have 64, 128,256 neurons. Learning rate is either 0.01, 0.001,0.0001 and epochs either 5,10,15. The activation function will be relu and optimizer Adam. Also 64 neurons will be present before the last single layer that will have the function sigmoid because our label is 0 or 1 so binary classification.This specific architecture was decided randomly, and it was decided if it will effective it will be kept. As a result, grid searchwere implemented to see what are the best hyperparameters 3^3 =27 different combinations where tested to see what hyperparameters are the best. As a result, 256 neurons will be implemented, 0.001 learning rate and 15 epochs. The accuracy for the validation dataset was 0.87 and the test accuracy was 0.92. The results are satisfying for a simple neural network.

Classification Report:

	precision	recall	f1-score	support
0	0.87	0.88	0.88	4382
1	0.87	0.85	0.86	3900
accuracy			0.87	8282
macro avg	0.87	0.87	0.87	8282
weighted avg	0.87	0.87	0.87	8282



To evaluate the performance of the binary classification model, a confusion matrix was generated on thevalidation (train)set. The matrix showed the following results: 3874 true negatives, 3319 true positives, 508 false positives, and 581 false negatives. This indicates that the model correctly identified the majority of both classes, with an overall accuracy of approximately 87%. The precision for the positive class (class 1) was approximately 87%, meaning that when the model predicted a positive class, it was correct 87% of the time. The recall for the positive class was around 85.0%, indicating that it successfully detected 85% of all actual positive cases. The F1 score, which balances precision and recall, was approximately 86%, suggesting a strong and balanced performance. While the model shows high performance, the presence of 581 false negatives suggests that further improvements could be made to reduce missed positive cases, which might be important depending on the application domain.

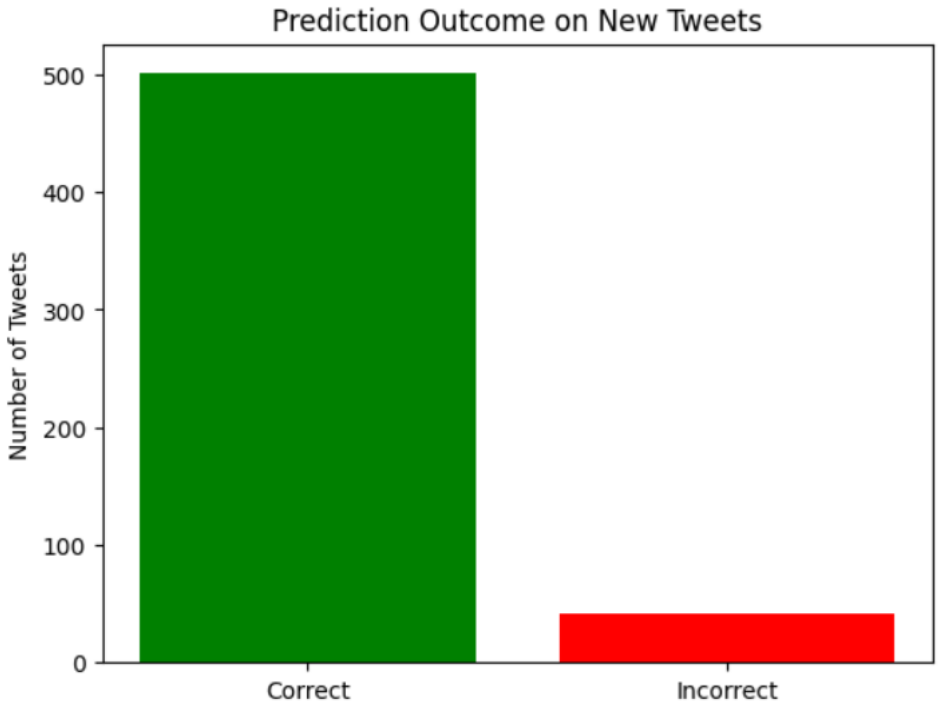


The Accuracy Line Plot shows both training and validation accuracy improving over time. The gap between training and validation accuracy is small, which is a good sign. Validation accuracy saturates, meaning further training wouldn't drastically improve generalization. So, as a conclusion, the model performs reliably on unseen data.

In this Loss Line Plot, training loss decreases steadily. Validation loss decreases but flattens out near the end. The lines don't diverge dramatically, so model isn't severely overfitting. As a conclusion, we have healthy training behavior.

Test dataset results

	precision	recall	f1-score	support
Negative	0.91	0.94	0.92	266
Positive	0.94	0.91	0.92	277
accuracy			0.92	543
macro avg	0.92	0.92	0.92	543
weighted avg	0.92	0.92	0.92	543



The bar chart further confirms this, showing that the vast majority of predictions were correct, with only a small number of incorrect classifications. These results demonstrate that the neural network generalizes well to new, unseen tweets and is highly reliable for binary sentiment classification tasks.

Also, we have 0.92 (or ~92%) test accuracy, which is strong because it's close to validation accuracy and that's proves that our model generalizes well. For many binary classification tasks, above 85% is considered solid, depending on the complexity and balance of the data. The consistent gap between train/val/test accuracy indicates stable training, not overfitting.

CNN

The first architecture was a CNN with 128 filters and kernel size 5 so it can spot 5grams.Here activation functions wererelu , it also contain 64 neurons and the output again goes thought 1 neuron that has function sigmoid because of the binary classification. The optimizer is Adam and the learning rate 0.001 and 15 epochs. Again the architecture is random just to see the accuracy which is satisfying 0.89.

Next 2 convolutional layers will be implemented each with 128 filters but the one will have kernel size 5 an the other 3 so they can capture 3grams and 5 grams. Again the accuracy is high 0.89.

Last but not least 3 convolutional layers will be implemented with 128 filters but different kernel size 3,4,5 so now it can capture 3grams, 4grams, 5 grams and to be able to learn fully ,from the text that it receives, the sentiment. The accuracy is 0.90.

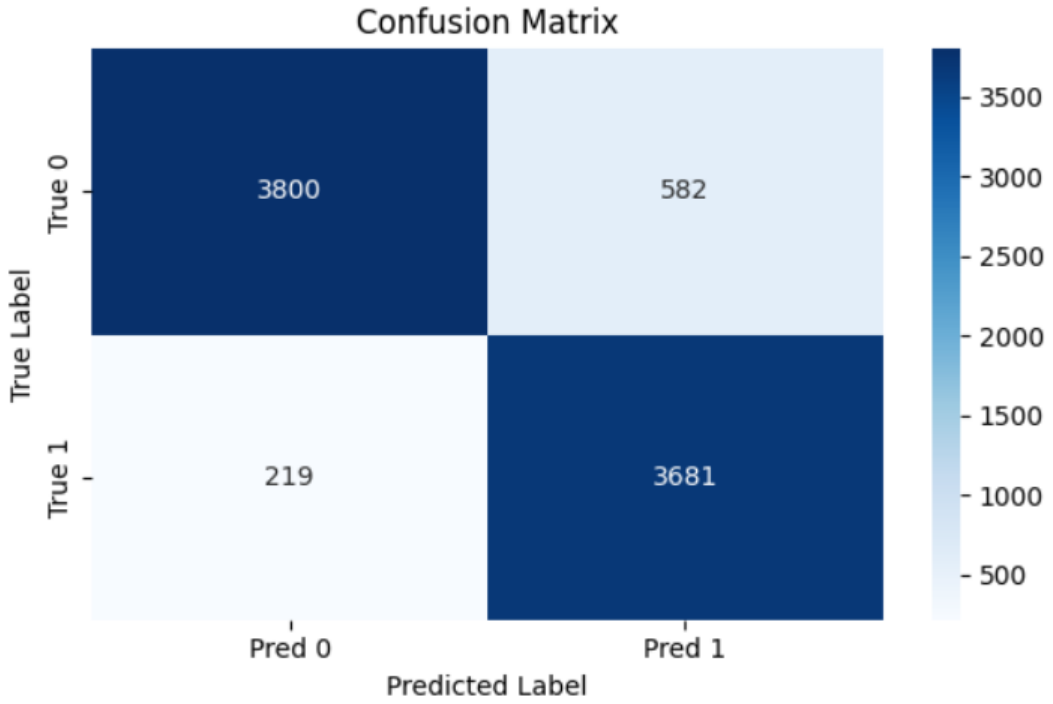
For this architecture (with 3 con layers) random search will be implemented. We decided not to use grid search because of the amount of time it will take to implement the search. Only 5 searches were tried to find the best hyperparameters and even with a small amount like that it took approximately 30 minutes to show the results. Because the hyperparameters it found did not show accuracy better than 0.90. The previous hyperparameters mentioned will be used.

The diagram is for the CNN 3 conv layers that had the best result that we wanted to visualize.

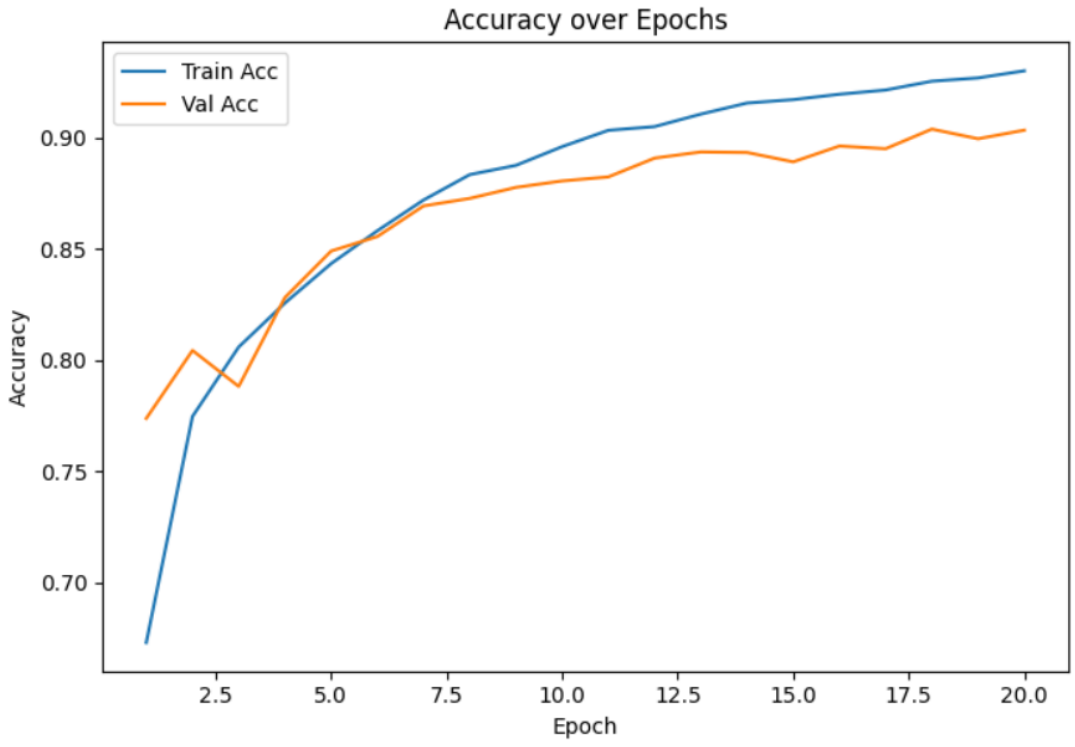
Classification Report:

	precision	recall	f1-score	support
0	0.95	0.87	0.90	4382
1	0.86	0.94	0.90	3900
accuracy			0.90	8282
macro avg	0.90	0.91	0.90	8282
weighted avg	0.91	0.90	0.90	8282

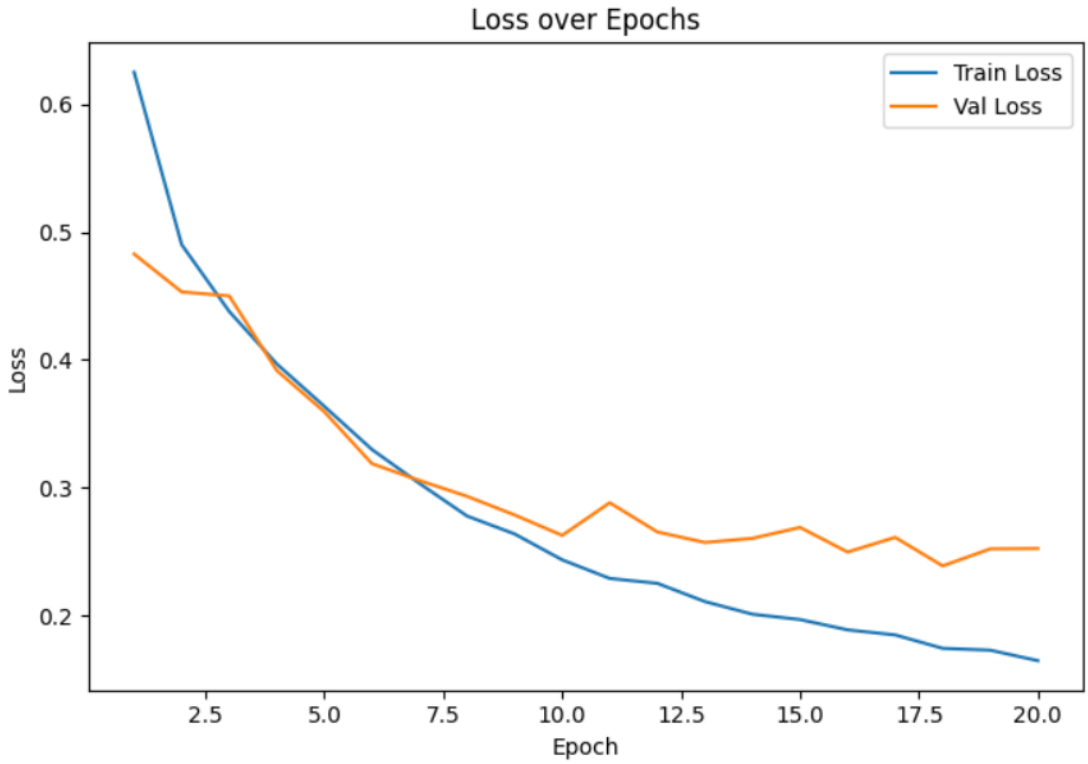
The classification report for the CNN model indicates strong performance across both classes, with precision of 0.95 for class 0 and 0.86 for class 1. Recall is higher for class 1 at 0.94, and both classes have an F1-score of 0.90. Both classes have an F1-score of 0.90, and the macro and weighted averages are also consistently around 0.90. The overall accuracy is 90%, confirming the model's robust and balanced predictive capability.



The confusion matrix shows a strong balance between the two classes, with 3800 true negatives, 3681 true positives, and relatively low false predictions (582 false positives and 219 false negatives).This reflects strong overall classification performance with balanced precision and recall, and demonstrates the model's ability to effectively distinguish between classes.



This plot shows the CNN model's training accuracy steadily increasing and reaching about 93%, while validation accuracy plateaus near 90%. The consistent performance and minimal gap between the two curves indicate a well-generalized model.



The loss plot shows that training and validation loss decrease steadily across epochs, indicating effective learning and minimal overfitting. Validation loss stabilizes after epoch 10, suggesting that the model maintains strong generalization performance beyond initial training.

Test dataset results

Classification Report:

	precision	recall	f1-score	support
0	0.98	0.96	0.97	266
1	0.96	0.98	0.97	277
accuracy			0.97	543
macro avg	0.97	0.97	0.97	543
weighted avg	0.97	0.97	0.97	543

After the implementation of this architecture to the test dataset the accuracy was 0.97. Which is expected because first of all due to the 3 convolutional

layers the model is able to find complex patterns. Second the test dataset is pretty small 543 instances but still it is unseen data for the neural network that can still predict the label with high accuracy.

RNN

RNN is designed to sequentially read the data and thanks to their inbuild memory they can remember and learn from the past. Also, they are sensitive to the order for the word since they also learn from that. Aa result they can capture the context of the sentences and words.

We tried the simple RNN with 128 neurons. The activation's function was decided to be tahn because it makes all the inputs to be in the range from [-1,1]. Since the RNN has this hidden state where it depends on the previous hidden state (because that's how it learns) it makes lot of computations and multiplications and if this computation become to be the model might not be effective and have problems learning. With smaller values created from tahn the network is more efficient.For the dense part of the network, we use relu. As a result, the tanh leans the sequence and the relu lakes the decision based on that sequence. Optimizer was Adam and learning rate 0.001. Again some hyperparameters were used randomly just to see how the network will work. As a result, the validation accuracy was 0.52 which is not satisfying at all.

Next, we tried bidirectional RNN to see if the accuracy can be improved. This technique allows the RNN to read the words backwards to be able to understand the sequence even better. The number of neurons was again 128 the activation function than, optimizer Adam, learning rate 0.001. As a result, the validation accuracy was 0.82 which is an amazing improvement. The RNN was able to learn the learn the sequence even better and show this amazing result.

Next, we tried the bidirectional LSTM. LSTM is an acronym long short-term memory meaning that it has an even better and bigger memory to remember a bigger sequence of words and has an even bigger ability to learn from the text the sentiment. Again, we used a bidirectional technique which means that it has the ability to read the sentences backwards. The neurons again are 128, activation tahn, optimizer Adam learning rate 0.001.Of course, the training time was the biggest of all the architecture we tried. The results are significant the accuracy was 0.92 because the neural network was able to learn the data and understand the sentiment.

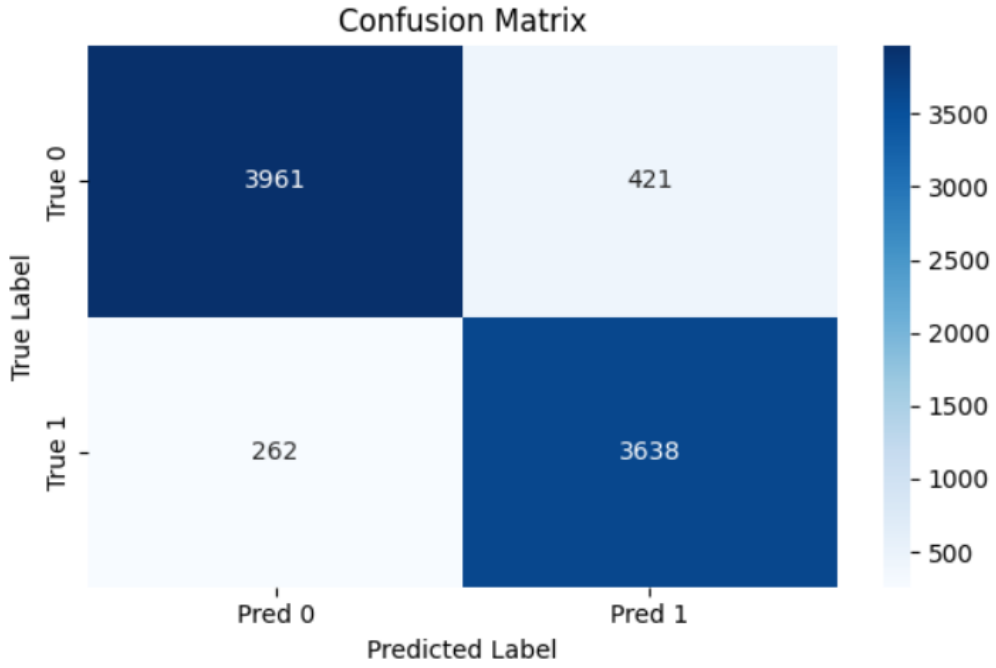
We decided to tune the hyperparameters for this architecture(bidirectional LSTM) we only used 5 random searches and still the time to complete them was approximately 40 minutes. Since during these 5 searches hyperparameters that provide accuracy bigger than 0.92 were not detected, we decided to keep the hyperparameters that where defined in the beginning.

The diagrams are for the LSTM because it showed the best results that we want to visualize.

Classification Report:

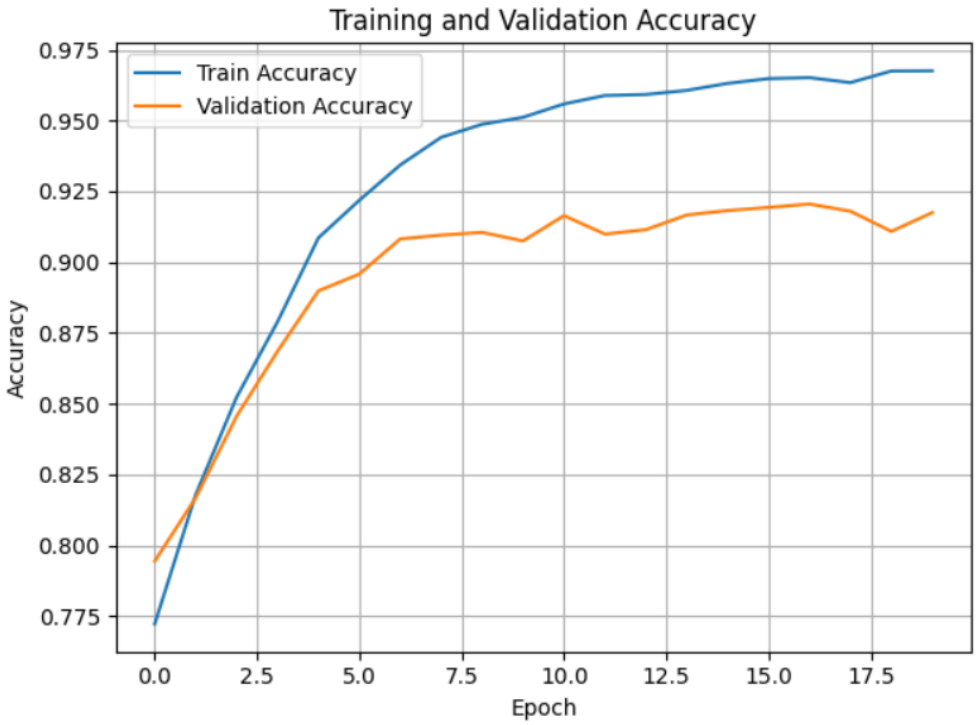
	precision	recall	f1-score	support
0	0.94	0.90	0.92	4382
1	0.90	0.93	0.91	3900
accuracy			0.92	8282
macro avg	0.92	0.92	0.92	8282
weighted avg	0.92	0.92	0.92	8282

The classification report further supports the model's effectiveness. Both classes achieve high precision (0.94 for class 0 and 0.90 for class 1) and recall (0.90 and 0.93, respectively). The overall accuracy is 92%, and both macro and weighted averages confirm balanced performance across the dataset. This reinforce the model's strong sentiment classification capabilities.



The confusion matrix reveals strong classification performance for both classes. The model correctly predicts 3961 of the negative samples and 3638 of the positive ones, while only misclassifying a small number (421 false positives and 262 false negatives). This balanced performance shows that the model is not biased toward either class and can distinguish between them

effectively, which is essential in sentiment analysis where both positive and negative classes carry equal importance.



Accuracy improves consistently over the training period, with training accuracy reaching around 97% and validation accuracy stabilizing near 92%. This strong performance shows that the model has learned meaningful features from the input sequences. The relatively small and consistent gap between training and validation accuracy suggests that the model generalizes well to new data, and the use of bidirectional LSTM has clearly helped it understand the context and sentiment in the sequences more effectively.

The training loss decreases steadily throughout the epochs, showing that the model is effectively learning the patterns in the data. The validation loss follows a similar trend until around epoch 10, after which it begins to rise slightly. This may suggest mild overfitting, as the model continues to improve on training data but starts to lose generalization performance on unseen data. However, the validation loss remains relatively low and stable overall, indicating that the model is still performing well and not significantly overfitting.

Test dataset results

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.98      0.98       266
           1       0.98      0.98      0.98       277

    accuracy                           0.98       543
   macro avg       0.98      0.98      0.98       543
weighted avg       0.98      0.98      0.98       543
```

The test accuracy was again significant 0.97 proving that the model was able to learn the sentiment. Again, it must be mentioned that the test dataset was small but still it is completely unseen data to the networkand there is absolutely zero chance of data leakage.

Combination of tweets and movies

We tried the zero-shot technique where we used our 3 conv layer CNN on the test data of the second dataset movies reviews. The second dataset is this https://www.kaggle.com/datasets/atulanandjha/imdb-50k-movie-reviews-test-your-bert/data. Our goal was to see if the network can generalize well on unseen data but also from a different domain. The CNN had accuracy 0.63.Which is quite expecting because the dataset contains text from another domain the language is different the expressions also are different. People use different language in a tweet which is short and small and different in a movie review where in most cases they use arguments to justify their opinion. They also use different tone in twitter probably because it is more non formal but, in a movies reviews they may use more formal language. Nevertheless, our goal is that our network can generalize in many different domains to understand the sentiment from different text.So the low accuracy is expected but still approximately 60% is good for unseen different domain data.

Now we will add each time a percentage of the dataset movies to dataset twitters and each time we will train the CNN to see if the accuracy can be improved and test the CNN on the test dataset of the movies. The results are the following

percentage of movies using in twitter	Combine twitter and part of movies accuracy
0% ZERO SHOT	0.63
10%	0.68
20%	0.69
30%	0.73
100%	0.75

As we can observe each time we add a greater percentage of the dataset the accuracy is improving. Which is logical because the CNN sees each time more and more data of the dataset movies and learns better the sentiment. So when we use the whole train dataset of the movies we can see that the accuracy is 0.75 whichis satisfying, the CNN was able to learn the proper weights to predict the sentiment in the domain movies.

Transfer Learning

We experimented also with transfer learning. We took our saved model CNN that was trained on the dataset twitter and learned the weights of that sentiment prediction and used it on the different the percentage of the movies dataset to retrain the Dense part of it. The results are the following

percentage	transfer model twitter to movies (3 conv frozen)	2 conv frozen	1 conv frozen
0%	-	-	-
10%	0.63	0.63	0.64
20%	0.61	0.63	0.64
30%	0.6	0.66	0.66
100%	0.6	0.72	0.75

In the beginning we kept the tree layers frozen and retrained only the Dense part (64 neurons). Sadly each time we kept adding percentage the performance was only getting worse we can assume that due to the frozen convolution layers the CNN was unable to learn the sequence and the 3grams , 4gramns , 5 grams that are crucial in text.The more movie datawe use, the more the domain gap between Twitter and movies becomes a problem.

We tested what will happen if only 2 con layers were frozen and the results improved. The CNN started to learn the patterns from the movies dataset with at least 1 conv layer and when all the movie dataset was used it reached accuracy 0.72 which is an amazing improvement compared to the 0.60 with all the convolutional layers frozen.

Finally, we kept only 1 conv layer frozen. It must be mentions that in the Jupiter notebook we keep only the 1 conv frozen technique because it showed the best results the rest where experiments that we conducted and each time we kept the accuracies to present here in the report. As a result, this technique had amazing results , when all the dataset was used it reaches accuracy 0.75 which is the same when the CNN was trained on both datasets combined. Because of the unfrozen layers the CNN was able to learn 4grams and 5grams and a better sequence and weights.

Only use B

Last but not least we tried to train the CNN on the dataset b (each time used different percentage) to see what will happen and if this specific architecture is appropriate for the movies dataset. The results are the following

percentage	Only use movies
0%	-
10%	0.6
20%	0.59
30%	0.65
100%	0.78

We can see the gradual improvement of accuracy in the test dataset as we add more data. When used all the train dataset it reached accuracy in thes 0.78 wich so far is the best accuracy out of all the techniques(mix dataset,

transfer learning) which is expected since we train each time the CNN and it perfectly adjusted to the new word it encounters and learns new weights. Although we can say that different architectures can be used for the movies dataset. An accuracy 0.78 is satisfying but when used different architectures such as more convolutional layers, or different among of filters etc. an better accuracy for sure can be achieved.

Conclusion

We can conclude that the movies and twitter are two different datasets that have very different languages and tones that are used. An assumption can be made that the movie's dataset uses a more formal tone. Thats why during the transfer learning only 1 con layer must be kept frozen to achieve good accuracy. Still when the CNN was trained on the dataset movies it proved that more tailored models or architecture could potentially achieve even higher performance. Overall, these findings illustrate the trade-offs between generalization and domain-specific learning and emphasize the value of controlled fine-tuning in cross-domain sentiment analysis.

Postcript

You can find the code from the google colab in this link https://colab.research.google.com/drive/1R2ncOvjFRIF998EGfCFzC_-yxtjnvf3R?usp=sharing