UNIVERSITÉ PARIS-SACLAY

CentraleSupélec

Emotion Classification and Sentiment Analysis

Machine Learning Project

GitHub Code[1]

26th Jan 2024

Authors:
Mayorga Llano, Mariana[2]
Khan, Mir Wise[3]

[1] https://github.com/mirwisek/emotion-classification-twitter/
[2] mariana.mayorga-llano@student-cs.fr
[3] mirwise.khan@student-cs.fr

Contents

1 Introduction

Our project is focused on classification of preprocessed tweets into emotions i.e. joy, love, angry, sadness, surprise, and fear, using various deep learning models. Moreover, we also exploit the same models on a movies dataset for binary classification for sentiment analysis.

2 Dataset Preprocessing

We have used the two datasets. As ML models do not process raw text directly. Text inputs must first be transformed into a numerical format that the model understands. This involves breaking down the strings into smaller units known as tokens and then encoding these tokens as numerical vectors.

2.1 Tweets Dataset

Tweets dataset[4] came as highly preprocessed dataset with all kinds of special symbols removed. The distribution of 20,000 tweets is depicted in fig 1. Tweets dataset had class imbalance where representation of joy and sadness were comparatively high. We didn't address class imbalance issue considering the scope of the project.

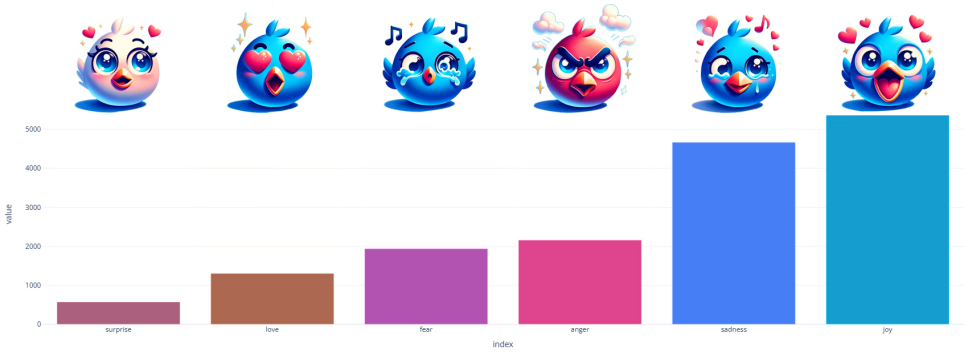

Figure 1: Tweets dataset class distribution

2.1.1 Data Distribution in Latent Space

We illustrate the text encodings from tokenizer into latent space in fig 2 to get an idea of how well the classes are separable.

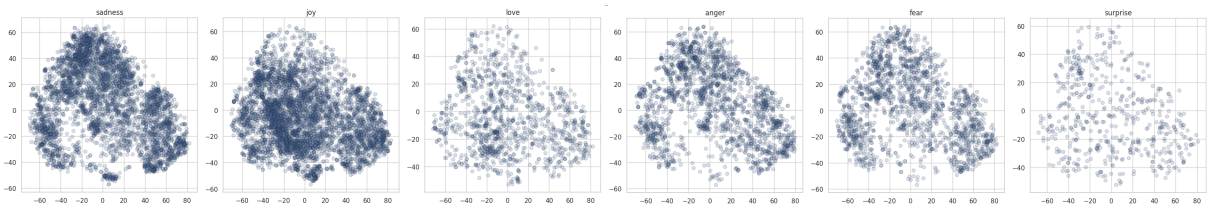

Figure 2: Tweets distribution in latent space

For positive emotions **(joy and love)** are well separated from the negative emotions and also share a similar space. While negative emotions **(sadness, anger, and fear)** occupy very similar

[4]https://huggingface.co/datasets/dair-ai/emotion

regions with slightly varying distributions. Finally, surprise is somewhat scattered all over the lower dimension space

While the separability of classes doesn't necessarily could be represented by tSNE or any other dimentionality reduction technique it just guides us to get some idea.

2.2 Movies Dataset

Movies dataset from IMDB[5] offered 50,000 movie reviews with equal classes as depicted in fig 3.

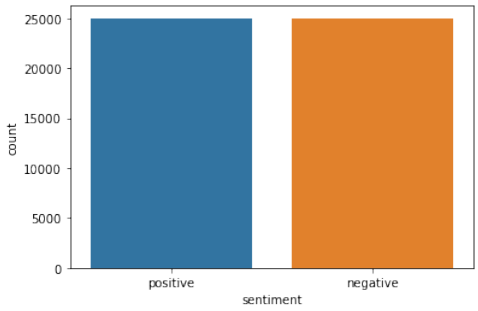

Figure 3: Movies dataset class distribution

2.2.1 Data Distribution in Latent Space

The representation would be different on PCA or UMAP. Therefore, fig 4 don't necessarily represent separability rather some intuition. Which can be verified after training a model.

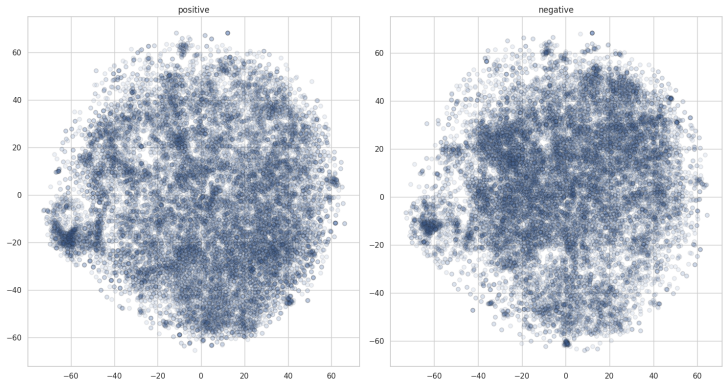

Figure 4: Movies distribution in latent space

Fig 5 illustrates the word count distribution for tweets and movies. Most of the tweets are around 25 words while the movie reviews are limited to 300 words on average.

3 Explored Models

3.1 CNN

For this section, we implement a binary classification (positive/negative) and a multiclass classification (sadness, anger, love, surprise, fear, joy) using 1D Convolutional Networks.

[5]https://datasets.imdbws.com/

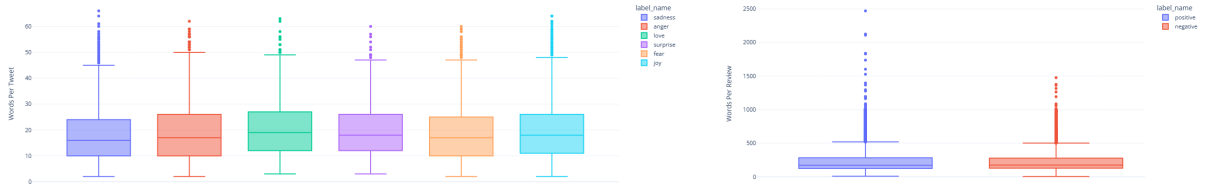

Figure 5: Words count per Tweets & Movie reviews

3.1.1 Input preparation

For CNN, it is necessary to clean and standardize the text data to ensure that the model learns from relevant features and generalizes well to unseen data. To achieve this goal, the following actions were taken.

HTML Tag Removal: Given that some of our data had HTML tags, we removed them to prevent noise in the sentiment analysis.

Text Normalization

- *Lowercasing*: All text was converted to lowercase to maintain consistency and avoid duplicity in word recognition due to case differences.
- *Removing Punctuations and Numbers*: Non-alphabetic characters were removed.
- *Single Character Removal*: Isolated characters, often resulting from punctuation removal, were eliminated as they hold minimal contextual importance.
- *Space Normalization*: Multiple spaces were reduced to single spaces to maintain text uniformity.

Stopword Removal: Stopwords, do not contribute much to the sentiment of the text, so they were removed to help reducing the size of the feature set and focus the analysis on words with higher potential for sentiment indication.

Tokenization and Padding The processed text was then tokenized, converting words into numerical tokens. This transformation is to allow the neural network models to process our data. Padding was applied to ensure that all input sequences had the same length, which is a requirement for our CNNs.

Word Embedding with GloVe To capture the semantic relationships between words, GloVe (Global Vectors for Word Representation) embeddings were used with 100 dimensions. These pre-trained embeddings map words into high-dimensional vectors, which allowed us to create the embedding layers at the beginning of the CNN models. Importantly, this layer was set as non-trainable to retain the semantic information embedded in the pre-trained vectors.

Dataset Preparation For binary classification, the sentiment labels were encoded as 0 and 1 representing negative and positive sentiments, respectively. In the case of multiclass classification, labels were converted to one-hot encoding. The datasets were split into training and testing sets, ensuring a representative distribution for model training and evaluation.

3.1.2 Models Training

We started with an initial simple architecture [16] of one pair of 1D Convolutional and GlobalMaxPooling layers. This was later adapted to have 2 and 4 pairs of 1D Convolutional and MaxPooling layers followed by a GlobalMaxPooling [17]. Additionally, we tested a model with 2 pais of Convolutional and MaxPooling layers followed by a flatteining layer without GlobalMaxPooling [18]. MaxPooling is used in all our models to reduce dimensionality and maintain the best performance among pooling options[SSA20]. All models start with the embedding layer and finish with the fully connected layer. The reasoning behind using these layers is described in the following points.

Embedding Layer The first layer of the model is a pre-trained embedding layer, utilizing GloVe embeddings to take advantage of the existing semantic relationships between words.

Convolutional Layer (1D) We used initially 128 filters and a kernel size of 5 for all models to compare their results, but later we tested the performance of different parametrizations in Section 3.1.4.

MaxPooling GlobalMaxPooling is used to get a more generalized summarization considering our inputs are short. Additionally, we experiment with the integration of regular MaxPooling in certain models to ascertain whether the extraction of more localized feature information enhances overall performance. This approach is intended to evaluate the efficacy of combining both local and global pooling strategies in the context of our specific dataset and model architecture.

Flattening Layer The Flattening layer allows transitioning from the convolutional and pooling layers to the dense layer within our CNN architecture. After applying several convolutional and pooling layers to extract and condense the most relevant features from the input data, the Flattening layer transforms the multidimensional output of these layers into a one-dimensional array. This process allows the model to transition smoothly from feature extraction to classification by feeding the flattened array into the dense layer(s) for the final classification decision. We can see in the results that this layer seems to be more effective in binary classification than in multiclass classification. More details about this analysis in Section 3.1.3.

Dense Layer This final layer uses a sigmoid activation function and a single unit output for binary classification and SoftMax activation with a unit per class in multiclass classification, each representing the probability of class membership. The sigmoid function is used in binary classification since it constrains the output to a range between 0 and 1 mirroring the probabilistic nature of binary classification and facilitating the interpretation of results as class probabilities, while softmax is used for multiclass to provide a single classification among the multiple classes.

Model Compilation For model compilation, the Adam optimizer is selected due to its efficiency and adaptability, particularly in handling the sparse gradients often encountered in NLP tasks. Binary cross entropy is chosen as the loss function for binary classifications, and categorical cross-entropy is used for multiclass classification models. Accuracy and loss is employed as metrics for further performance comparisons.

Training The training of the model was originally set for 6 epochs, but later we introduced early stopping to prevent overfitting. A portion of the training data, specifically 20%, is used as a validation set, allowing for periodic evaluation of the model's performance on unseen data. This

validation strategy is vital for ensuring the model's generalizability and robustness, providing insights into its performance in real-world scenarios.

3.1.3 Evaluation of Architectures

Model Description	Test Accuracy	Test Loss	# of Conv1D Layers	Epochs
Original Model	0.8666	0.3109	1	3
Double Convolution	0.8599	0.3258	2	2
4 Conv Layers	0.8576	0.3262	4	3
2 Conv + Flattening	0.8641	0.3244	2	3

Table 1: Performance Analysis of CNN Models for Binary Classification

The Original Model with only one Conv1D layer, achieved a high test accuracy for binary classifications, indicating efficient learning with minimal overfitting. Evidence suggests that introducing more convolutional layers in the Double Convolution model resulted in a slight decrease in accuracy and an increase in test loss, suggesting that added complexity does not necessarily enhance performance. Similarly, the 4 Conv Layers model for the binary classification, despite its complex architecture, showed a further decrease in accuracy (0.8576) and the highest test loss (0.3262), highlighting the diminishing returns and potential overfitting associated with excessive complexity. This decrease in performance along more complex architectures contradicts the results found in [SSA20]. This may be due to the nature of the input data, where the variable length of our input may affect its performance, therefore, for more similar results further data cleaning may be needed. This may also be a result of different parametrization settings, however, further investigation would be needed to verify this.

In the case of multiclass classification, the best model results were found in the double convolutional architecture without flattening. The challenge of classifying among more options may require additional convolutions to better understand the data since is not only a binary solution anymore.

Additionally, the Double Convolution + Flattening model, which differentiates from the Double convolution model by using a flattening layer instead of a GlobalMaxPooling before the Dense layer, nearly matches the Original Model's performance in the binary classification with an accuracy of 0.8641 and a test loss of 0.3244, suggesting that strategic layering and flattening can achieve a balance between complexity and performance. However, oppositely to the binary classification, the 2 Conv + Flattening model has a significant drop in accuracy and an increase in loss, highlighting that the introduction of a flattening layer might detract from the model's ability to effectively process and classify multiclass data.

In binary classification tasks, the decision boundary between the two classes may be more straightforward, allowing a flattened feature representation to effectively capture the necessary information for classification, while multiclass tasks involve distinguishing among more than two classes, which typically requires capturing more complex relationships in the data. Considering that GlobalMaxPooling reduces each feature map to its most prominent feature while flattening might lead to a loss of spatial hierarchies and could introduce noise or irrelevant information, we hypothesize that keeping a layer of GlobalMaxPooling instead of Flattening in Multiclass Classification helps prevent overfitting, which may be less likely for binary classification.

Model Description	Test Accuracy	Test Loss	# of Conv1D Layers	Epochs
Original Model	0.8243	0.4847	1	11
Double Convolution	0.8556	0.41083	2	7
2 Conv + Flattening	0.8274	0.4524	2	6

Table 2: Performance Analysis of CNN Models for Multiclass Classification

3.1.4 Hyperparameters Combinations

We tested multiple combinations of parametrization of filter values [64, 128, 256] and kernel sizes [3, 5, 7] as shown in Figure 6. An increase of filters will potentially capture a broader range of features, but we risk overfitting, because of this we tested with a higher and a lower filter than the one originally used, which was 128. Additionally, small kernels were chosen considering our inputs are short sentences, so we want to avoid the input being covered in a single window to exploit the use of the convolutions.

Parameters		Binary		Multiclass	
Filters	Kernel	Accuracy	Loss	Accuracy	Loss
64	3	0.8659	0.315642	0.794375	0.553796
64	5	0.8591	0.321995	0.775937	0.609681
64	7	0.8599	0.330753	0.761563	0.6498
128	3	0.8742	0.297925	0.808438	0.506729
128	5	0.8652	0.311775	0.775625	0.582586
128	7	0.8557	0.336927	0.789688	0.577662
256	**3**	**0.876**	**0.290941**	**0.843125**	**0.436101**
256	5	0.8656	0.309868	0.799688	0.525083
256	7	0.8678	0.317418	0.781875	0.567334

Figure 6: CNN Model Performance of Binary Classification with Varying Filters and Kernel Sizes

Increasing the number of filters seems to allow the model to learn a wider array of patterns from the text data, contributing positively to model performance. Meanwhile, considering a smaller window of words at a time allows for a more precise capture of relevant features. The best model configuration, in terms of accuracy, was found with 256 filters and a kernel size of 3. This setup not only achieved the highest accuracy but also the lowest loss, indicating a robust model that generalizes well to unseen data.

3.2 DistilBERT for Emotion Classification

DistilBERT is a distilled version of BERT, which is smaller, faster, cheaper, and lighter [Cor21]. DistilBERT are pretrained to predict masked words in a sentence of text. But in our case, we utilize it for classification. We use **"bert-base-uncased"** which comes with 512 tokens (context size) and since it is uncased it offers 30522 vocabulary size.

For the two transformer models we explored following two approaches namely feature extraction and fine tuning. The results are listed in table 3.

3.2.1 Input preparation - Tokenization

First the text is tokenised, represented using one-hot vectors called token encodings The size of the tokenizer vocabulary determines the dimension of the next encoding (usually 20-200k). Next these token encodings are converted to token embeddings (vectors living in a lower dimensional space).

3.2.2 Feature Extraction Approach

The input token embeddings are passed through the encoder block layers to yield a hidden state for each input token. While for the pretrained model, the objective is of language modeling, where each hidden state is fed to a layer that predicts the masked input tokens, we change it for our use case. Thus, for the the classification task, we replace the language modeling layer with a classification layer (logistic classifier).

3.2.3 Fine-Tuning Approach

In fine tuning, which comparatively takes longer training time than feature extraction, because unlike feature extraction where initial layers are frozen, all the weights are fine tuned.

3.2.4 Comparative Results

The results from table 3 show that Fine-Tuning approach with accuracy of 92.90% beats feature extraction approach with accuracy of 63.35%. Mainly, because the fine-tuned model updates all weights and is trained for task specific classification while in feature-extraction approach only one layer is focus on classification which does not seem to be enough.

To further clarify the results, fig 8 indicates that **love** is still often confused with **joy** (0.11), but much less than that indicated by fig 7 which is (0.46). Moreover, there are lots of other high values for confusion in fig 7 which is why the Fine-Tuning Approach 3.2.3 is better for DistilBERT.

3.3 BERTweet for Emotion Classification

BERTweet is a pre-trained language model for English Tweets that has a context size of 128 tokes, which is comparatively less than DistilBERT with 512 tokens. But it has more vocabulary size of 64000 than DistilBERT.

3.3.1 Feature Extraction Approach

We utilize the similar pipeline as DistilBERT for fair comparisons only replacing the type of model.

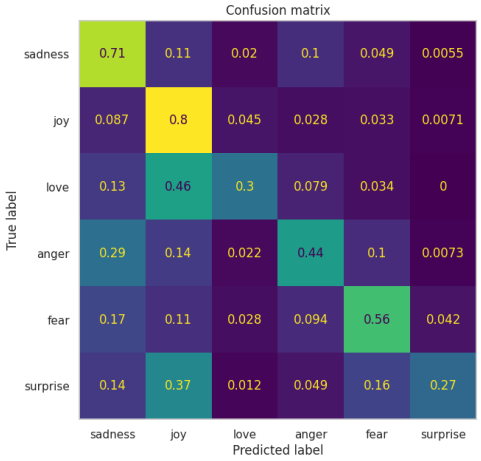

Figure 7: Feature extraction

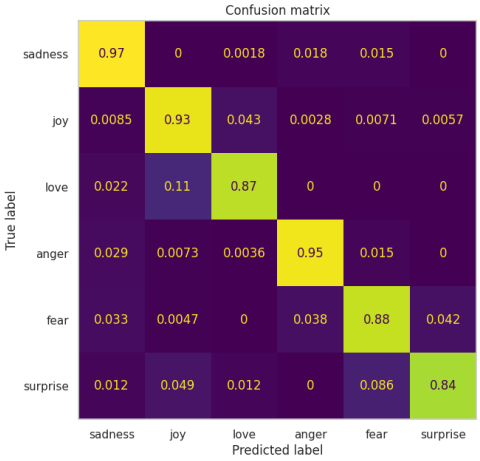

Figure 8: Fine tuning

3.3.2 Fine-Tuning Approach

Again the similar procedure is followed as for DistilBERT but with different model.

3.3.3 Comparative Results

The results from table 3 show that Fine-Tuning approach with accuracy of 93.35% beats feature extraction approach with accuracy of 60.95%.

Moreover, it's verified by the higher F1-score of fine-tuning approach with 0.9292 compared to that of feature extraction which is 0.544. Also, it is noticeable from confusion matrix in fig 10 that **surprise** is confused a lot with **fear** i.e. 0.09, despite the model's high accuracy. The reason is that because **surprise** is represented less in data distribution due to class imbalance therefore the effect of this confusion is minor on accuracy of the model.

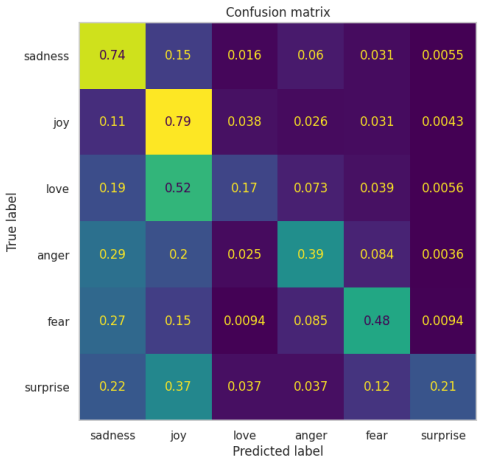

Figure 9: Feature extraction

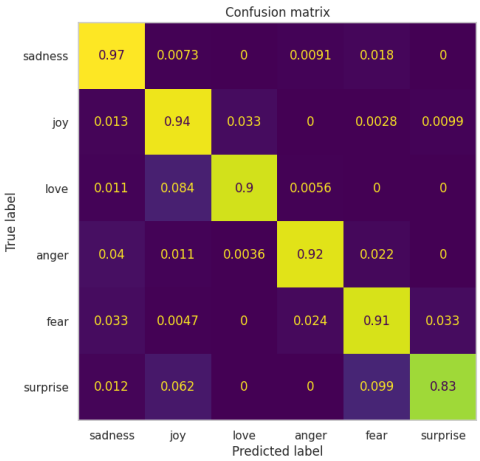

Figure 10: Fine tuning

[6]FE: Feature Extraction, FT: Fine Tuning

Model	Approach[6]	Test Accuracy %	F1 Score	Learning Rate	Epochs/Iterations
DistilBERT (Emotion Classification)	FE	63.35	0.5440	-	2000
	FT	90.70	0.9064	0.00002	1
	FT	92.85	0.9294		2
	FT	**92.90**	0.9292		3
BERTweet (Emotion Classification)	FE	60.95	0.4980	-	2000
	FT	90.15	0.8980	0.00002	1
	FT	92.10	0.9213		2
	FT	**93.35**	0.9336		3
BERTweet (Sentiment Analysis)	FE	84.88	0.8500	-	2000
	FT	89.76	0.8975	0.00002	1
	FT	**90.66**	0.9066		2
	FT	90.42	0.9042		3

Table 3: Transformer model results on Emotion Classification and Sentiment Analysis

3.4 BERTweet for Sentiment Analysis

BERTweet performed well on tweets therefore we also tested the same model on sentiment analysis usecase for movies review classification.

3.4.1 Feature Extraction Approach

The feature extraction approach for binary classification (sentiment analysis) offers good accuracy of 84.88% in table 3 but its superseded by CNN's best accuracy of 87.6% in table 6.

3.4.2 Fine-Tuning Approach

As illustrated in fig 11 the model's accuracy in the second epoch was 90.66% which degraded to 90.42% in the third epoch because it started to overfit, as observed in table 3. The same can be stated about the relatively high F1-score of second epoch compared to third epoch.

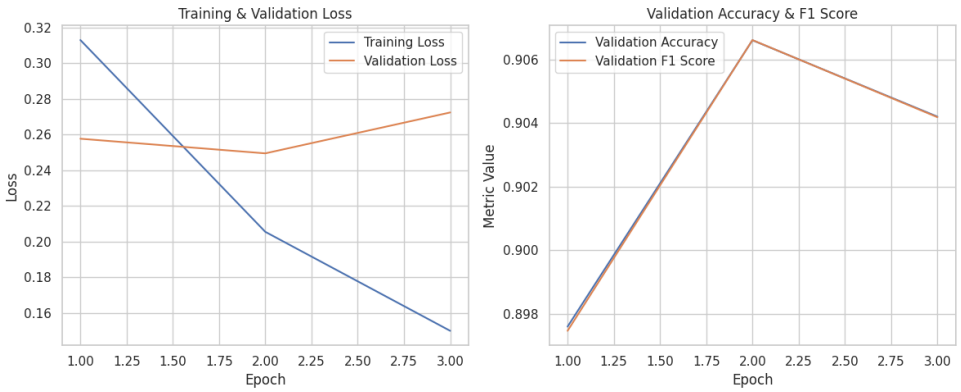

Figure 11: Training loss for BERTweet on sentiment analysis

3.4.3 Comparative Results

Once again even though the BERTweet model is specialized for tweets it performs well on movie review classification as well beating the best accuracy of CNN.

Moreover, it can be seen from the confusion matrices in 12 that in feature extraction approach 15% of labels are misclassified while in case of fine tuning the **positive** labels are misclassified as **negative** 8.6% almost half of 15%. Similarly, **negative** sentiments are labeled as **positive** 11% times which again is lower than 15% from feature extraction approach. Thus, it explains why fine tuning approach performs well in comparison to feature extraction approach.

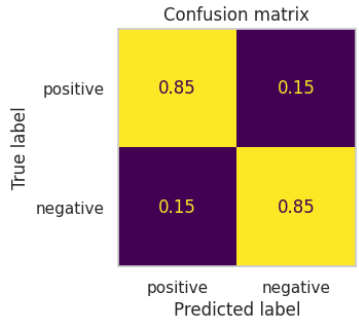

Figure 12: Feature extraction

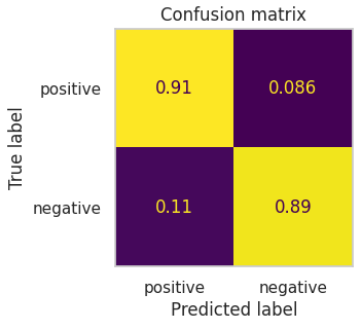

Figure 13: Fine tuning

3.5 Results - Comparison between CNN and Transformer Models

Whether it is emotion classification or sentiment analysis, transformers performed much better than CNN with variety of architectures.

Among the transformer models we observed that fine-tuning approach showed better results than feature extraction approach for both transformer models and in each usecase.

From the results in table 3 and the confusion matrices it is evident clearly that Fine-tuning approach of BERTweet performs better than DistilBERT. And the accuracy is much better than CNN's best model in table 6 which is 84.31% with 256 filters and 3 kernel. Moreover, the accuracies are justified by the relative F1-scores which are as much better as the accuracies of relevant models. Because the transformer models are more complex and since they are pretrained models, they have huge amount of parameters. In addition, the attention mechanism gives them more power for more concrete and robust classification.

4 Experimental Models Crossing

We were curious to try the model trained on sentiment analysis and observe how well it perform on dataset of tweets. And decided to test it on our best performing model i.e. BERTweet (Sentiment Analysis) which was trained on movies dataset.

4.1 Tweets data preparation

Since the tweets dataset was categorized into 6 (emotions) labels. The target variable of the tweets dataset was refactored to convert (Joy, Love, Surprise) labels to positive sentiment and (Fear, Sadness, Anger) as negative sentiments to match the sentiment analysis model (BERTweet).

4.2 Results

While in all previous cases of transformers fine-tuning approach was performing better but in this case feature extraction with accuracy of 63.1% and F1-score of 0.673 which performs better

than fine tuning approach that has accuracy of 54.1% with F1-score of 0.664.

In addition to F1-score, the correlation matrices indicate that model in fine-tuning approach has overfitted too much and thus miss classifying negative sentiments as positive (0.73) in fig 15 thus lower F1-score. Although, feature extraction also has this problem with negative labels classified as positive (0.46) in fig 14 but comparatively less and hence relatively higher F1-score.

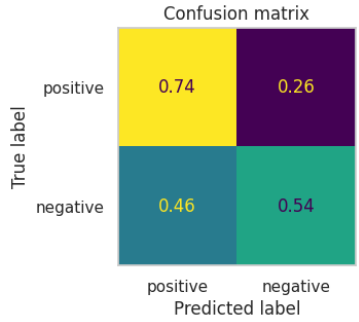

Figure 14: Feature extraction

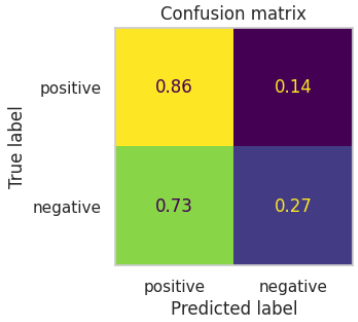

Figure 15: Fine tuning

Since the model was trained on movies review dataset which comparatively had high word count per movie review compared to tweets which were on average 25 words. Therefore, we can say that the BERTweet was trained more specifically for classifying movies review and when we intended to infer for tweets it couldn't generalize well. Thus, if we would also add the tweets data to training data of movies review, it would perform well to generalize. Another option would be to regularize the models.

Nevertheless, the purpose of this exploration was experimental and not solely the purpose of the project. Our take and learning from this exploration was that depending on the use case, a model can be trained for either a specific use case or generic use case from the start and training it for one purpose and utilizing the same model for another purpose doesn't offer good quality results.

References

[Cor21] Diogo Cortiz. "Exploring Transformers in Emotion Recognition: a comparison of BERT, DistillBERT, RoBERTa, XLNet and ELECTRA". In: *ArXiv* abs/2104.02041 (2021). URL: https://api.semanticscholar.org/CorpusID:233025474.

[SSA20] Nurul Ashikin Samat, Mohd Salleh, and Haseeb Ali. *The Comparison of Pooling Functions in Convolutional Neural Network for Sentiment Analysis Task*. Jan. 2020. DOI: 10.1007/978-3-030-36056-6_20.

Appendix A Architectures

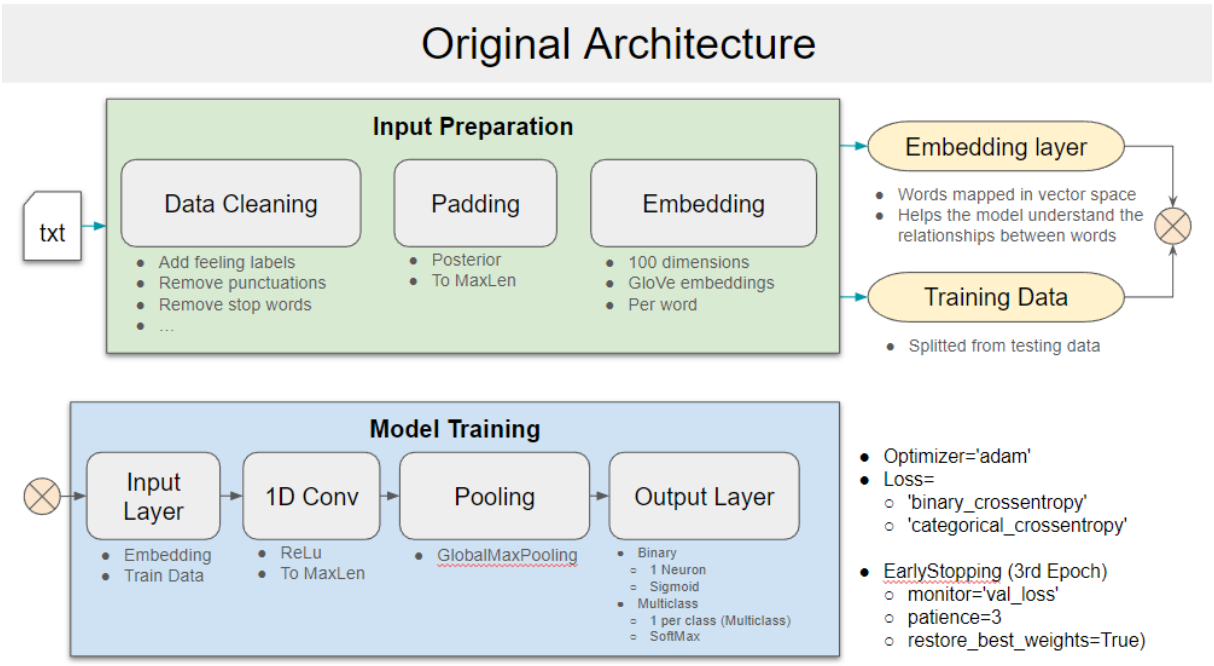

Figure 16: Initial CNN Architecture

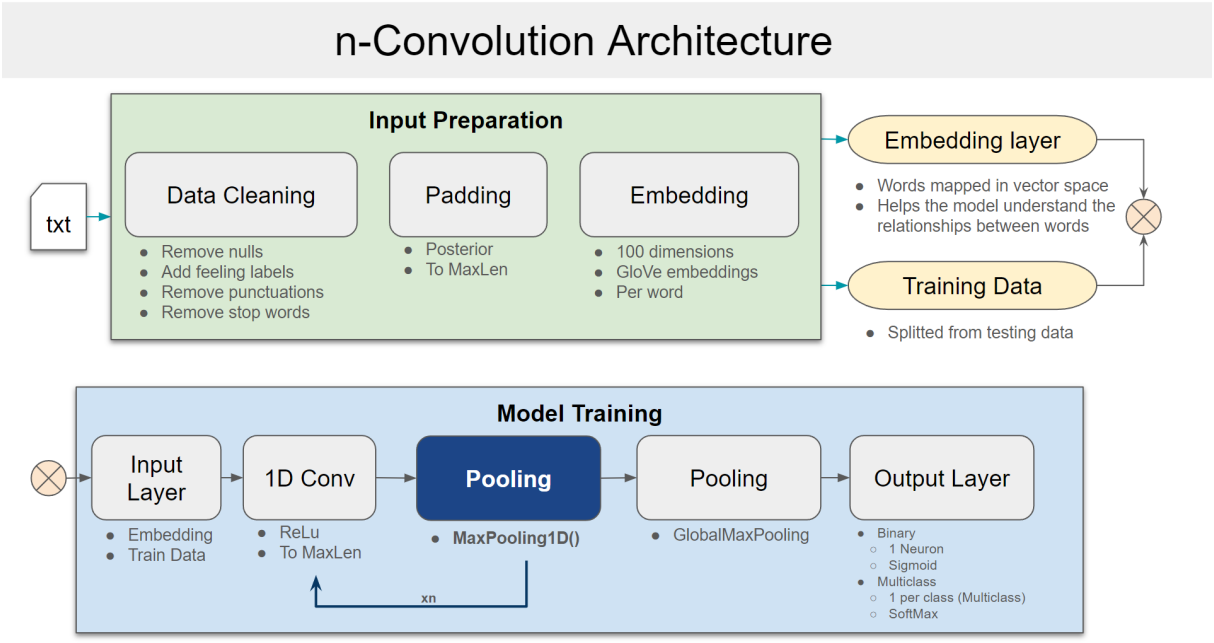

Figure 17: Adapted CNN Architectures with multiple Convolutions

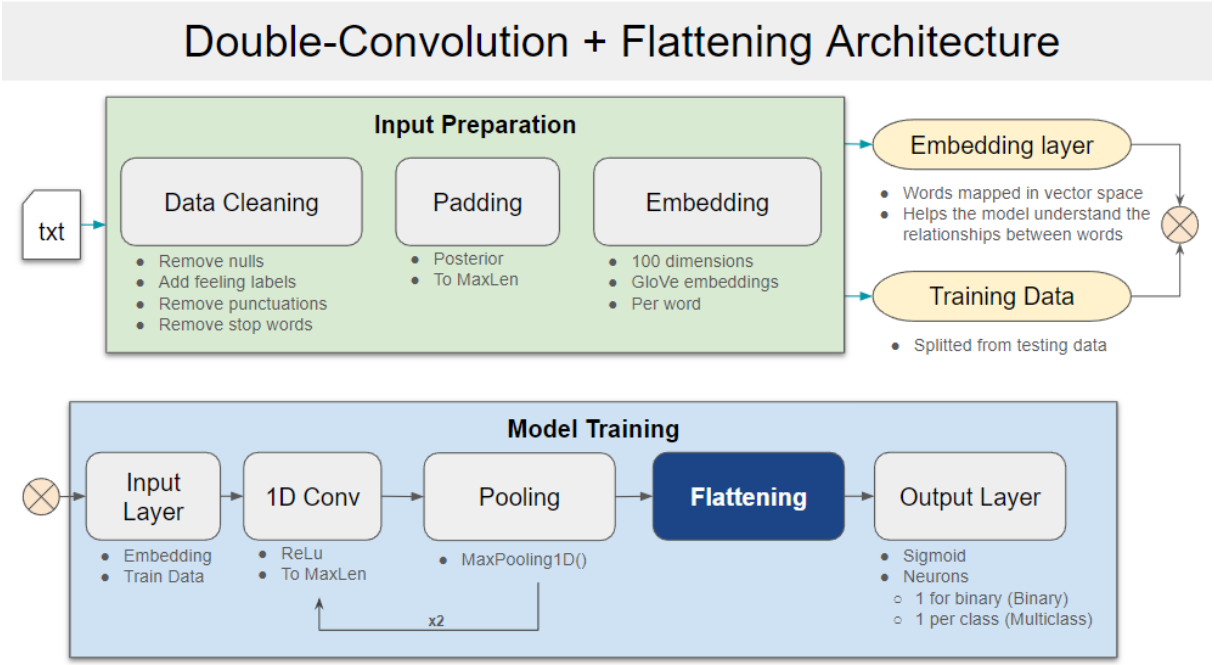

Figure 18: Adapted CNN Architectures with multiple Convolutions and using Flattening without GlobalMaxPooling