TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Twitter Sentiment Classification Project

Machine Learning - A.A. 2024/25

Emanuele Valzano - Matricola 0341634

TensorFlow

Agenda

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

01
Introduction

02
Possible approaches

03
Steps involved

04
Dataset Analysis

05
Preprocessing and Vectorization

06
Naive Bayes as baseline

07
A Bidirectional LSTM based model

08
Conclusions

Introduction

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The goal of the project is to develop a Machine Learning model to classify Twitter messages as either positive or negative. We discuss possible approaches to solve the task and build a model most suited for the goal and to experiment with the overall architecture.

Mainly exploit Deep NNs, particularly RNNs.

Possible Approaches

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

- **Simple ML algorithms**, that take all the tweet sequence as the model's input and perform a binary classification (e.g. logistic regression).
  - Simple to build.
  - Not built to handle sequences.
  - Not too high accuracy.

- **Probabilstic Models** (Naive Bayes):
  - Simple and efficient, can be used for text classification.
  - Can use Bag of Word rapresentation (with its limits).
  - Not the best accuracy.

Possible Approaches (2)

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

More advnced models

- **RNNs** (e.g. LSTM based model):
  - Built to handle sequences and maintain a hidden state.
  - Bidirectional LSTM layers can capture context given both previous and successive tokens.
  - Harder to train and optimize.

- **Transformer-based language models** (e.g. BERT):
  - State-of-the-art to handle token sequences.
  - Too many parameters to train from scratch with the available resources.
  - Mostly fine-tuning of pre-trained models, would not allow us to experiment too much with the architecture.

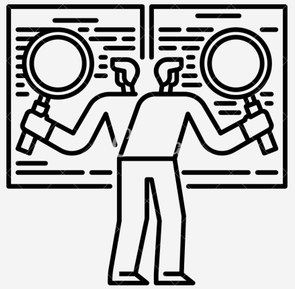
Possible Approaches (3)

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

chosen

It has been chosen to experiment with **RNNs**, particularly we built an LSTM based model.
It is the best comprimise given our goal to experiment on the architecture and to avoid utilizing mostly pre-trained models with already defined architectures.

Moreover, in order to have a solid banchmark for performance comparisons, it has been used Naive Bayes.



Steps Involved

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The steps involved in the project follow a modular apprach, by implementing reusable functions that follow different strategies. These modular components are than put together to build the model of interest.

Overall steps:

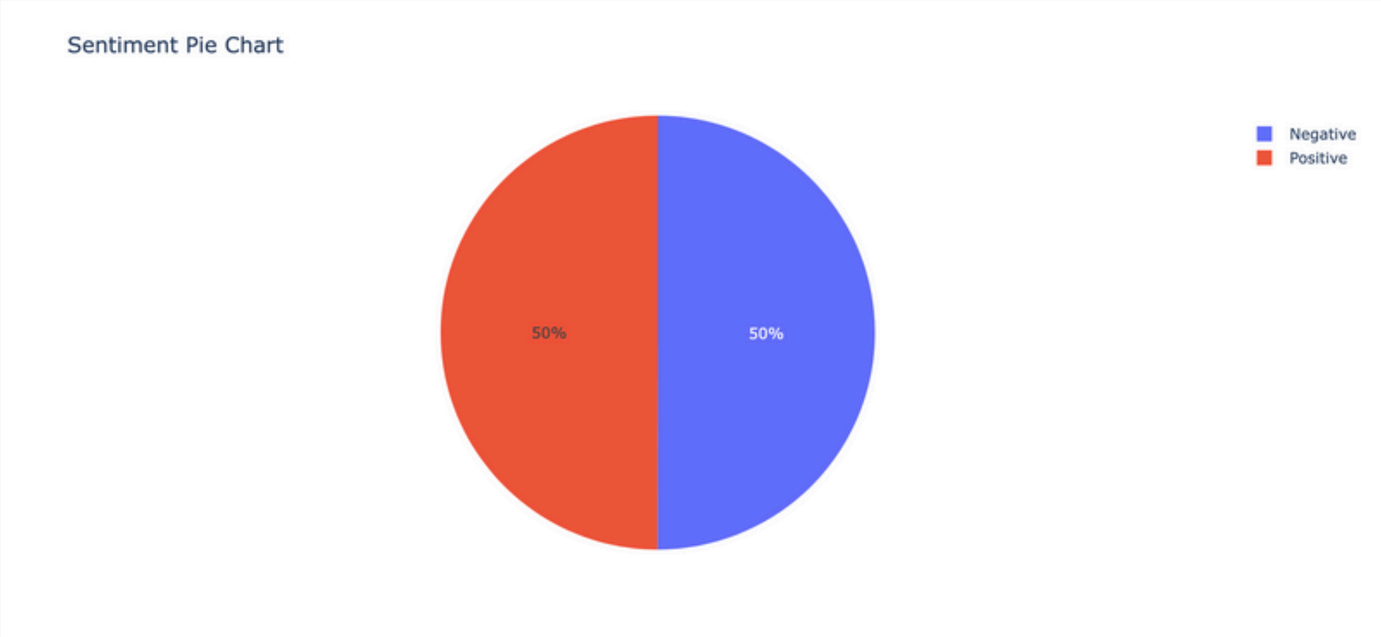
1. Load the dataset and perform a visual analysis.
2. Employ two different preprocessing strategies, along with its two embedding techniques.
3. Implement Naive Bayes as a Baseline.
4. Build and evaluate a bidirectional LSTM-based model with different configurations.
5. Informally test the built model on our own tweets.

Dataset Loading and Analysis

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The dataset has been loaded through the *text_dataset_from_directory* function and split into 70% for training, 15% for validation, and 15% for testing.

- BATCH_SIZE = 32

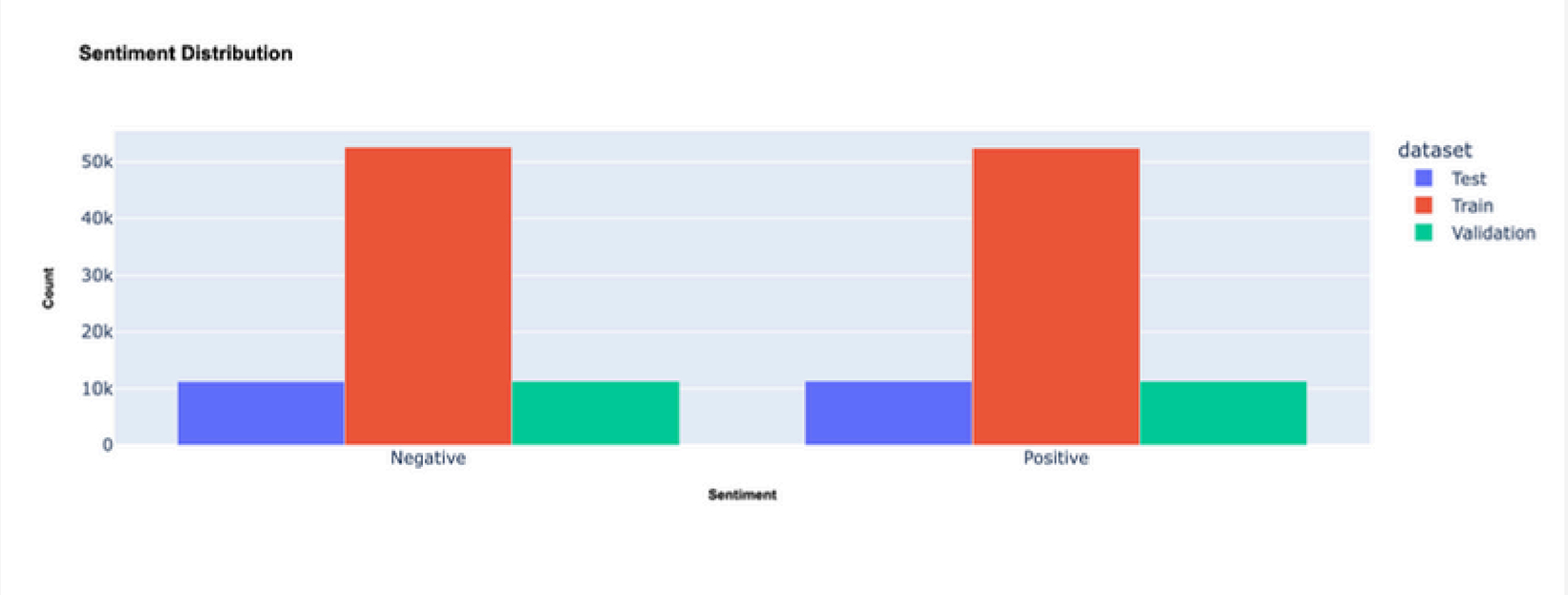


Dataset Loading and Analysis

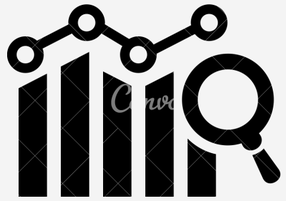
TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The dataset has been loaded through the *text_dataset_from_directory* function and split into 70% for training, 15% for validation, and 15% for testing.

- BATCH_SIZE = 32



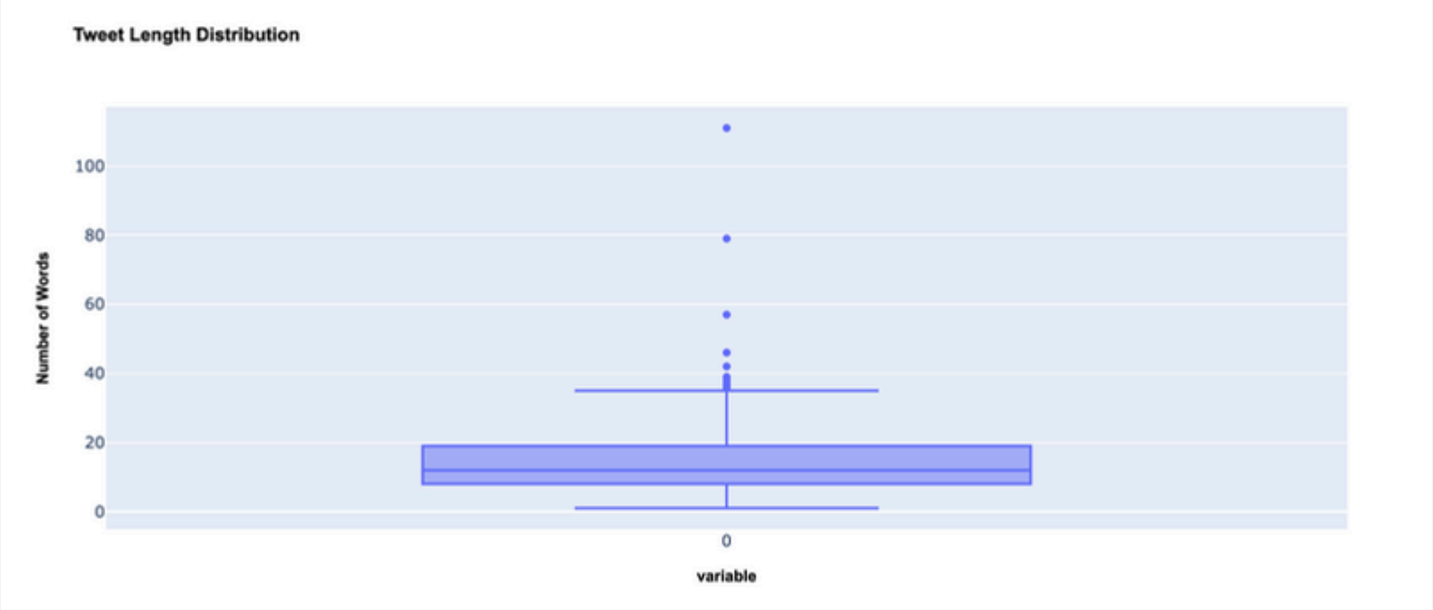
Dataset Loading and Analysis



TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The dataset has been loaded through the *text_dataset_from_directory* function and split into 70% for training, 15% for validation, and 15% for testing.

- BATCH_SIZE = 32

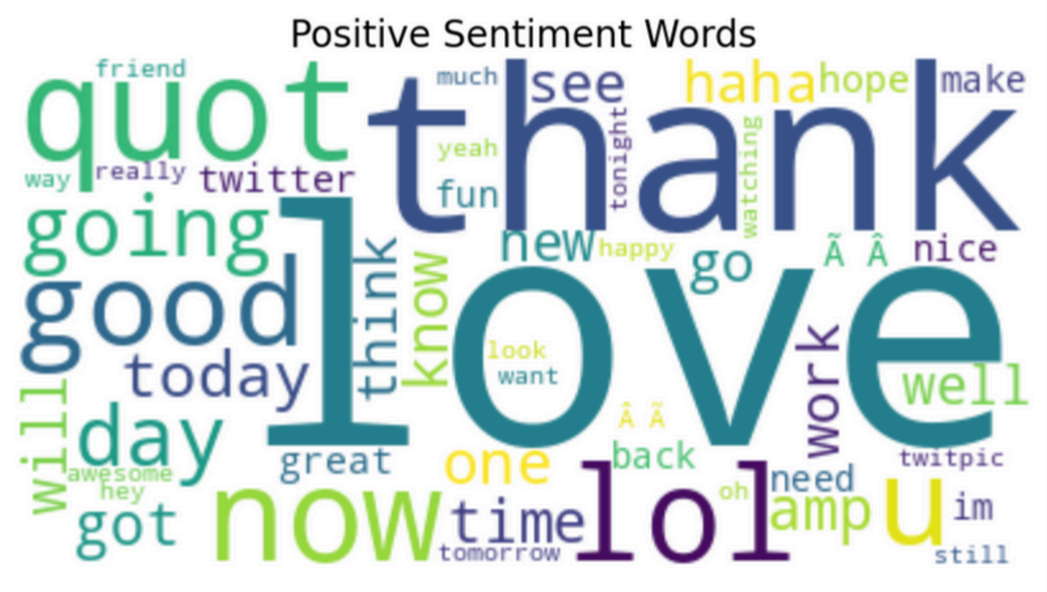


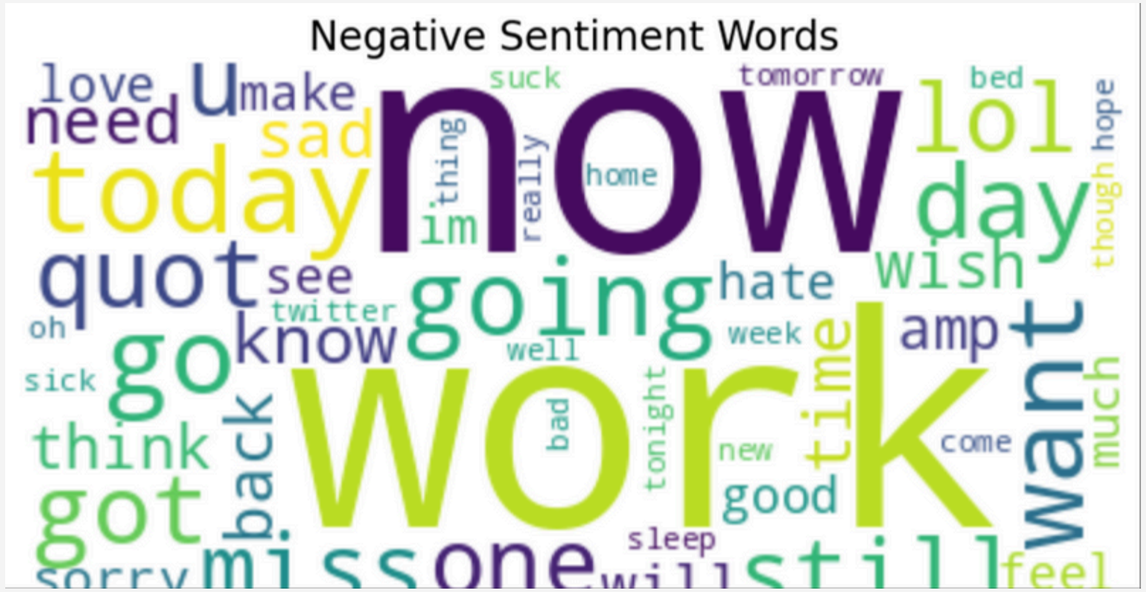
Dataset Loading and Analysis

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

The dataset has been loaded through the *text_dataset_from_directory* function and split into 70% for training, 15% for validation, and 15% for testing.

- BATCH_SIZE = 32





Preprocessing and Vectorization

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Two different approaches have been tested to handle the preprocessing and tokenization of tweets, alongside two different embedding strategies.

TextVectorizationLayer and Trainable Embedding:
A simple preprocessing strategy is employed. After the vectorization layer there is a trainable embedding layer for converting tokens into real vectors. Additionally, to regularize the model within the embedding layer we add a Dropout layer at the end.

- Tokenization integrated within the architecture.
- Harder to train, especially if the dataset is not ideal.

Tokenizer and pre-trained GloVe Embedding
Keras Tokenizer to perform a more fine grained preprocessing strategy, to prepare tokens for the pre-trained GloVe Embedding.

- Training more efficient due to the freezing of the embedding layer.
- GloVe has been trained on 2B tweets.
- Tokenization and preprocessing not integrated within the architecture.

Naive Bayes as Baseline

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Both validation and test set have been concatenated in order to test the model using a single test set (30% of the data). Remind that this model just acts as a beseline comparison.

CountVectorizer is used to handle the bag-of-words representation. It constructs a matrix where each entry represents the count of occurrences of vocabulary words in a tweet. Each row corresponds to a single tweet.

- max_features = 5000

Dataset Preprocessing using the fine-grained preprocessing function already developed for GloVe embedding.

- Tried both to keep and remove stop words.
- "Surprisingly" keeping stop words leads to a 1.5% better accuracy.

Evaluation Results for Naive Bayes

Metric	Result
Accuracy	0.7714
Precision	0.7736
Recall	0.7685
F1-Score	0.7711

A Bidirectional LSTM based model

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

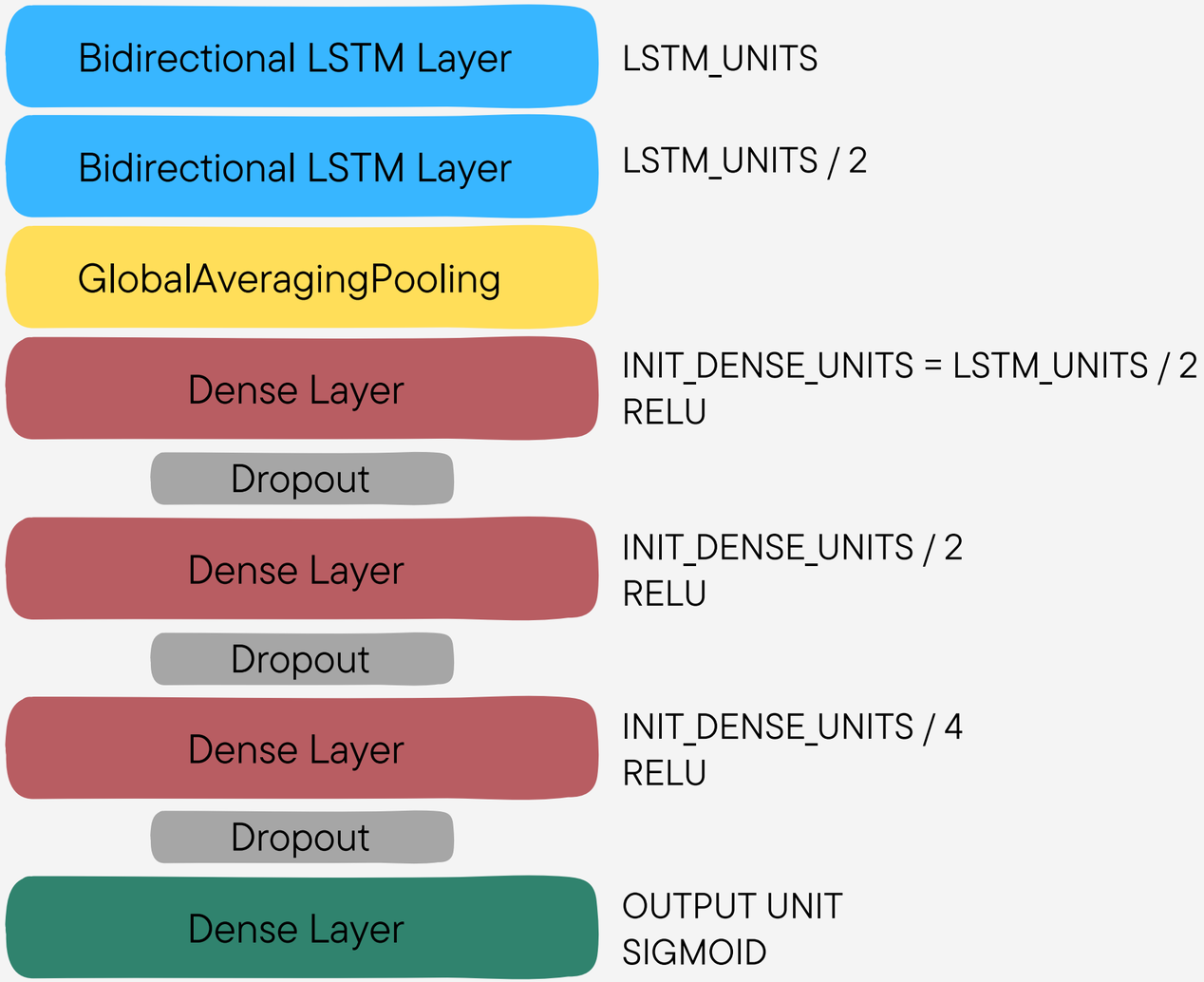
The two different configurations exploiting different Embedding layers are extended in order to implement a Bidirectional LSTM based model.

- It has been developed a function that extends an already initialized model, assuming to have the first input layers. It is modular in its design, allowing any other model with its own input layers to reuse the deeper architectural components.

Simple Idea

The idea behind the architecture is pretty simple: focus on capturing detailed information in the earlier layers of the network and then gradually reduce the complexity as we go deeper.

Model Architecture (does not include "input" layers)



The following values are tuned using KerasTuner:

- **LSTM_UNITS**
- **DROPOUT_LSTM**
- **DROPOUT_DENSE**

Optimizer: Adam
Loss: Binary Cross Entropy
Metrics: Accuracy

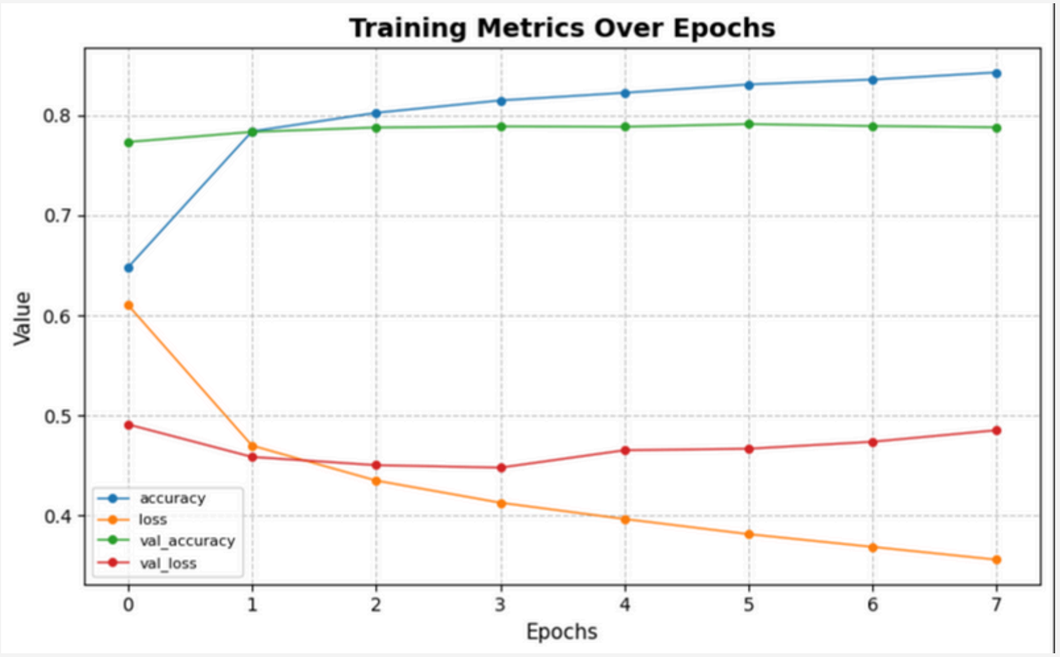
SentimentAnalysisTuner:
A wrapper around the *RandomSearch* tuner to ease even more the training process alongise hyperparameter tuning.

Bidirectional LSTM with Trainable Embedding

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

During hyperparameter search it has been used early stopping, using a patiance of 2.
The best resulting model is than trained from scratch using a patience of 4.

Hyperparameter	Value
vocab_size	10000
dropout_emb	0.5
lstm_units	224
dropout_lstm	0.5
dropout_dense	0.3



Metric	Result
Accuracy	0.7895
Precision	0.7985
Recall	0.7767
F1-Score	0.7874

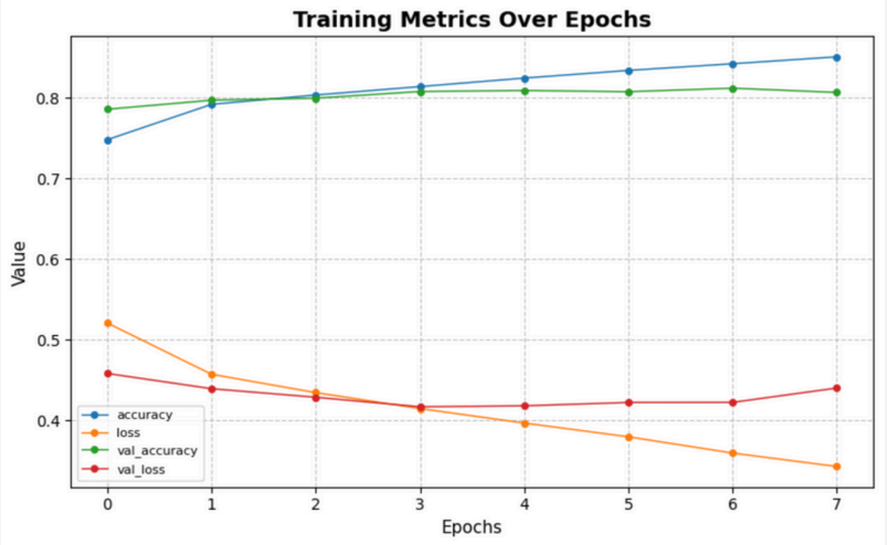
Bidirectional LSTM with GloVe Embedding

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Used the optimal vocabulary size found in the previous tuning search, since the vectorization logic is not embedded within the architecture, so we cannot leverage KerasTuner properly.

During hyperparameter search it has been used early stopping, using a patiance of 2.
The best resulting model is than trained from scratch using a patience of 4.

Hyperparameter	Value
lstm_units	256
dropout_lstm	0.3
dropout_dense	0.5



Metric	Result
Accuracy	0.8058
Precision	0.8065
Recall	0.8067
F1-Score	0.8066

Conclusions

TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

Best Model: Custom Tokenizer + Pre-trained GloVe Embedding.

- GloVe embeddings trained on extensive Twitter data, resulting in a better performance.
- More efficient during training since the embedding layer is freezed.

Accuracy Improvement: Achieved 1.63% increase compared to the version with trainable embeddings and a **3.44%** compared to the baseline. Could be beneficial for large scale data.

Wheter to use such model or a more cheap one (e.g. Naive bayes) depends on the application requirements.

Small Scale & Latency Critical:

- Prefer simpler, cost-efficient models (e.g., Naive Bayes).
- Prioritizes resource efficiency (lower power consumptions).

Large Scale & High Accuracy Needs:

- Opt for Bidirectional LSTM (remind that state-of-the art are Tranformers).
- Higher computational overhead, both for training and inference.