TOR VERGATA
UNIVERSITÀ DEGLI STUDI DI ROMA

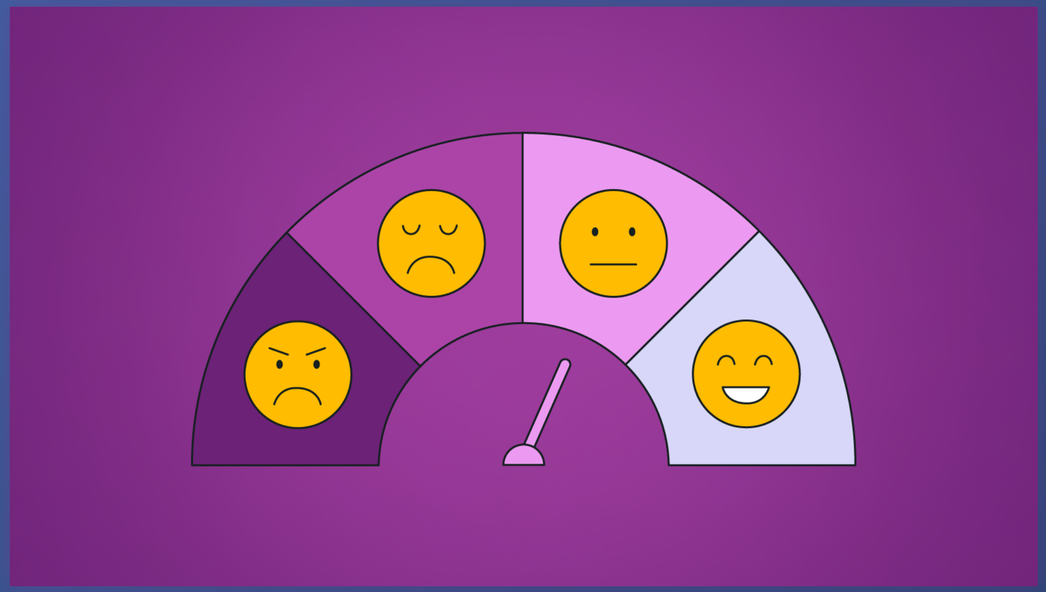
TWITTER SENTIMENT ANALYSIS

Machine Learning Project A.Y. 24/25

Matteo Avallone - 0357457

INTRODUCTION

General overview



Sentiment analysis is the task of processing the given textual information to analyze the emotions in it. In this particular task, a dataset comprising of almost 150k tweets is analyzed in order to acquire information regarding a positive or a negative feedback from each of its elements. The problem is therefore posed as a binary classification task.

2 models will be deployed: a much simpler one and a more complex one. Each of the two models will be further tuned for specific configurations involving a set of hyperparameters to understand how they behave under specific conditions.
Lastly, conclusions will be drawn up, highlighting which of the two models is more suitable for specific scenarios.

INTRODUCTION
Libraries

string

os

math is used to split the 30% of the dataset into validation and test set
numpy provides efficient numerical computational capabilities
rich is a Python library to format text and make it more stylish.
tensorflow is the core deep learning framework powering the sentiment analysis model.
string mainly for text preprocessing
regex offers powerful text processing and pattern matching functionalities.
matplotlib provides the means to visualize training history.
os facilitates interaction with the operating system for file and directory operations.
scikitlearn to evaluate the model's accuracy, precision, recall and f1 score metrics.

KEY STAGES

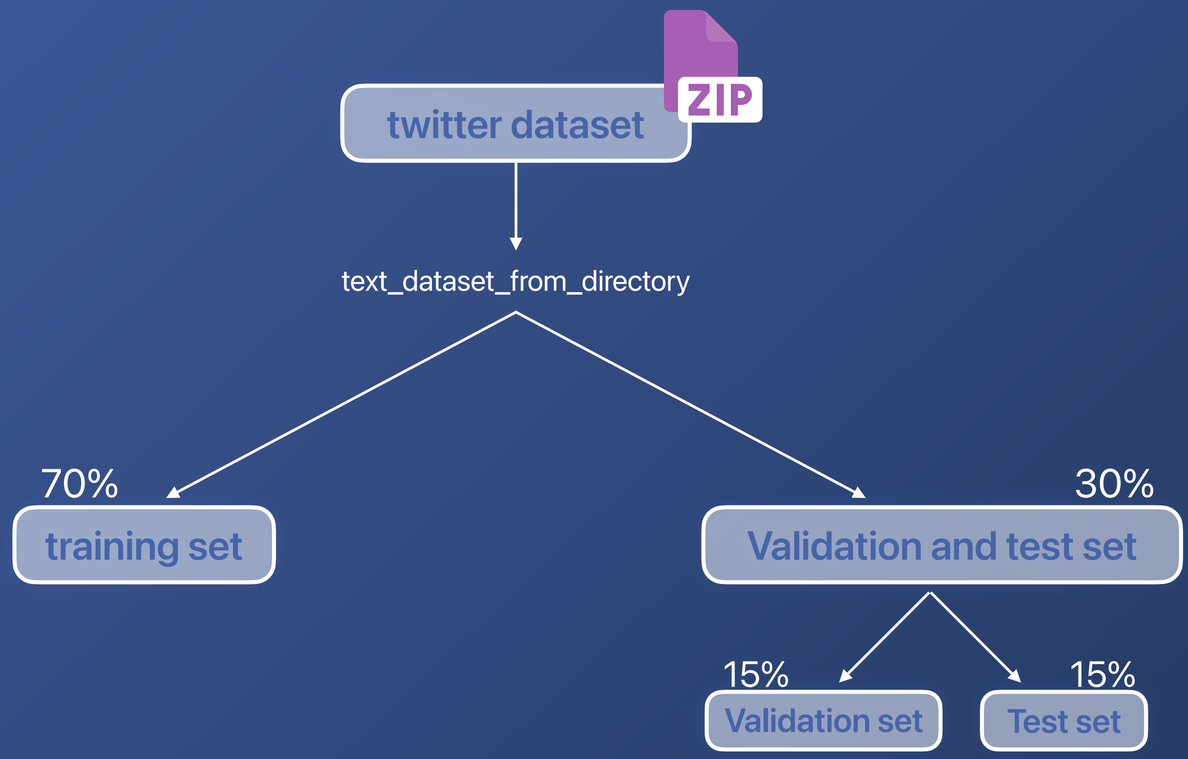
Project architecture

As we delve into the project's architecture, the following key stages can be summarized:

1 ***Data loading***: load the dataset and use a percentage ratio to split it accordingly.
2 ***Data preprocessing***: meticulously preprocess the data, encompassing different kinds of text cleaning.
3 ***Building the model***: construct a simple LSTM-based model as a first approach.
4 ***Training the model:*** train it by exploring its different configurations to get a general idea of how it behaves.
5 ***More complex model***: delve into a more complex approach and its different configurations, this time based on BiLSTM.
6 ***Visualization and Evaluation***: visualize the training history to gain insights into the model's learning process and evaluate its performance on the unseen test data to assess its generalization ability.
7 ***Further testing***: create some tweets on our own and check whether the final model is able to correctly predict their classification.

DATA LOADING

Splitting the data

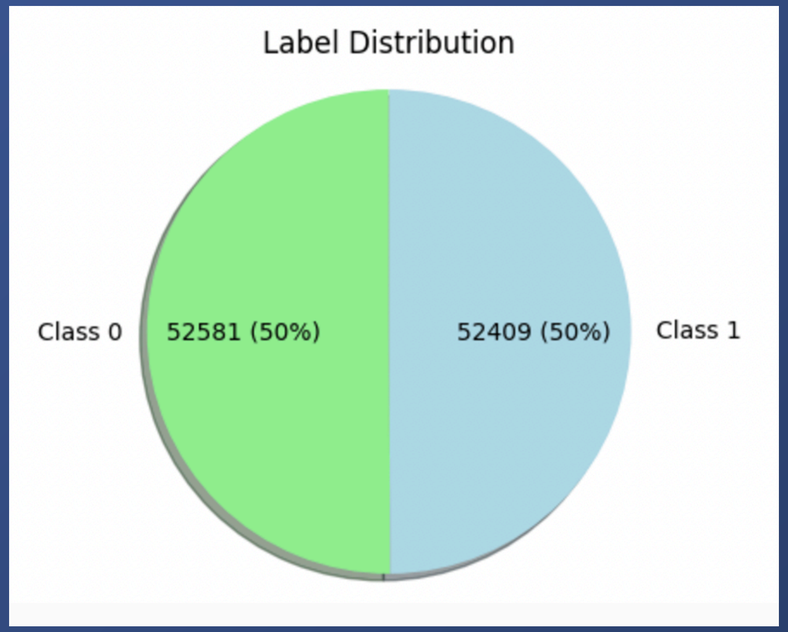


The data is extracted from the zip file and further split into training, validation and test set. 70% of the dataset will be used for the training phase, the remaining 30% is split in half and assigned to the other two sets.

text_dataset_from_directory is used to extract the dataset and split it into 70% of training set and 30% of validation_and_test set.

DATA LOADING

Label distribution



The dataset is pretty balanced, accounting for a nearly perfect 50% split of positive and negative tweets.

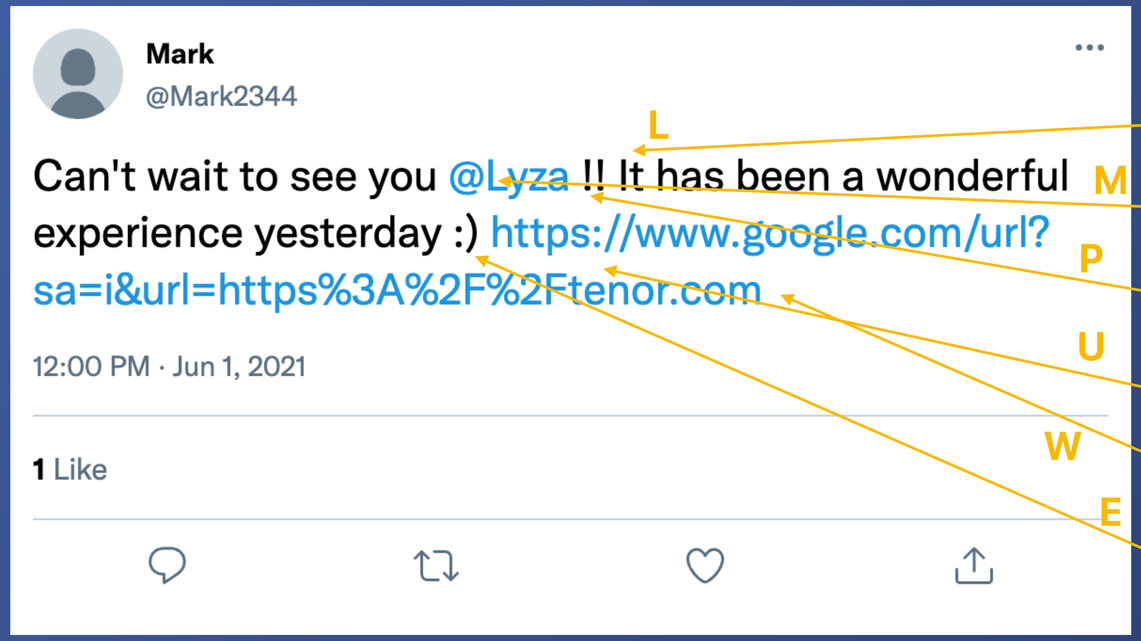
DATA PREPROCESSING

Slang handling

The dataset is cleaned and properly preprocessed in order to prepare it for the classification task.
One of the main areas of this process is the handling of slang words: the most common abbreviations and jargon have been handled by substituting them with the corresponding true meanings or alternate words.
For this sake, a dictionary data structure has been employed, with keys being the regular expressions that simplify the slang words lookup process and the values being the correct form of the expressions.

DATA PREPROCESSING

Main areas of interest



The main steps involved in the preprocessing phase are the following:

- Lowercasing
- Removing mentions
- Removing punctuations
- Removing URLs
- Removing trailing/leading/duplicated whitespaces
- Substituting emoticons with more expressive words conveying their actual meaning

DATA PREPROCESSING

Preprocessing pipeline

L M P U W E

DEPLOYED MODELS

General overview

Simple Model

- Vectorization layer
- Embedding layer
- LSTM layer

or

- BiLSTM layer
- Dense layer

+ Adam optimizer

More complex Model

- Vectorization layer
- Embedding layer
- BiLSTM layer
- LayerNorm
- Dense layer
- LayerNorm
- Dense layer

+ RMSProp optimizer

Callbacks = [early stopping, PrintBestValAccuracy]
Common to both the models are: Binary cross entropy as loss function and accuracy as metric.
Common hyper parameters: Vocabulary size, Number of units in the (Bi)LSTM layer, input sequence length.
The choice between BiLSTM and LSTM in the simple model is governed by the bidirectional parameter defined inside the model's function.
Both models use a vectorization layer to convert words into tokens and an embedding layer to convert the vectorized output of textual data into dense, continuous vectors.

DEPLOYED MODELS

Simple model - Configurations

	vocab size	input sequence length	(Bi)LSTM units	Bidirectional
Configuration 1:	5000	50	64	No
Configuration 2:	5000	50	64	Yes
Configuration 3:	5000	25	64	No
Configuration 4:	10000	50	64	Yes
Configuration 5:	20000	50	64	Yes
Configuration 6:	10000	50	128	Yes

Max Val Accuracy obtained

~79%

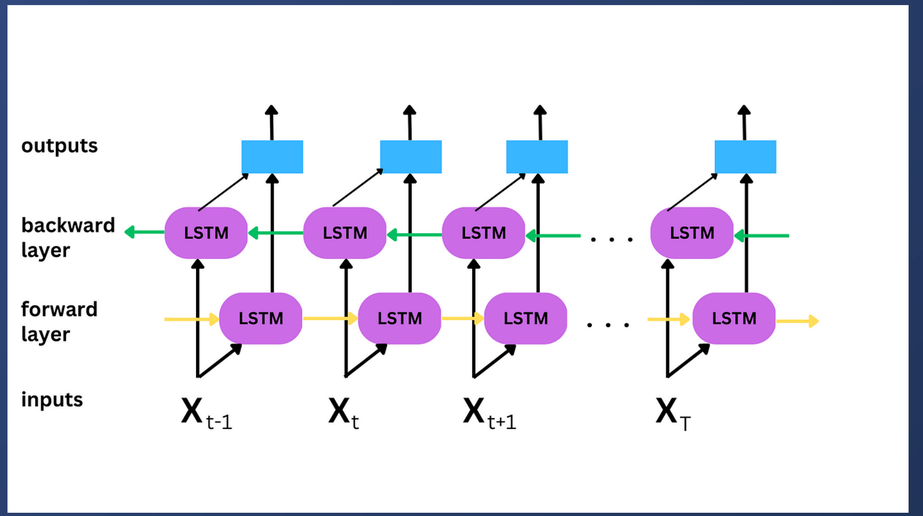
Almost all the configurations tend to underfit the training data, meaning that the model might not be complex enough to capture data patterns/relationships between features and labels.

DEPLOYED MODELS

More complex model - Configurations

This model makes use of a fixed BiLSTM layer to better capture long-range dependecies, reading input both forward and backward.
Two LayerNorms are added to help stabilizing and accelerating training by normalizing the inputs to each layer; they normalize each feature within each sample (rather than considering batches of samples) .
The fully connected dense layer provides non-linearity and allows the model to learn higher-level representations.
A different optimization algorithm is used. RMSProp generally works well with models prone to vanishing or exploding gradients and may have less computational overhead with respect to Adam due to the absence of momentum.

	vocab size	input sequence length	(Bi)LSTM units
Configuration 1:	10000	50	64
Configuration 2:	20000	50	64
Configuration 3:	10000	50	128
Configuration 4:	10000	35	64
Configuration 5:	15000	25	64



Max Val Accuracy obtained

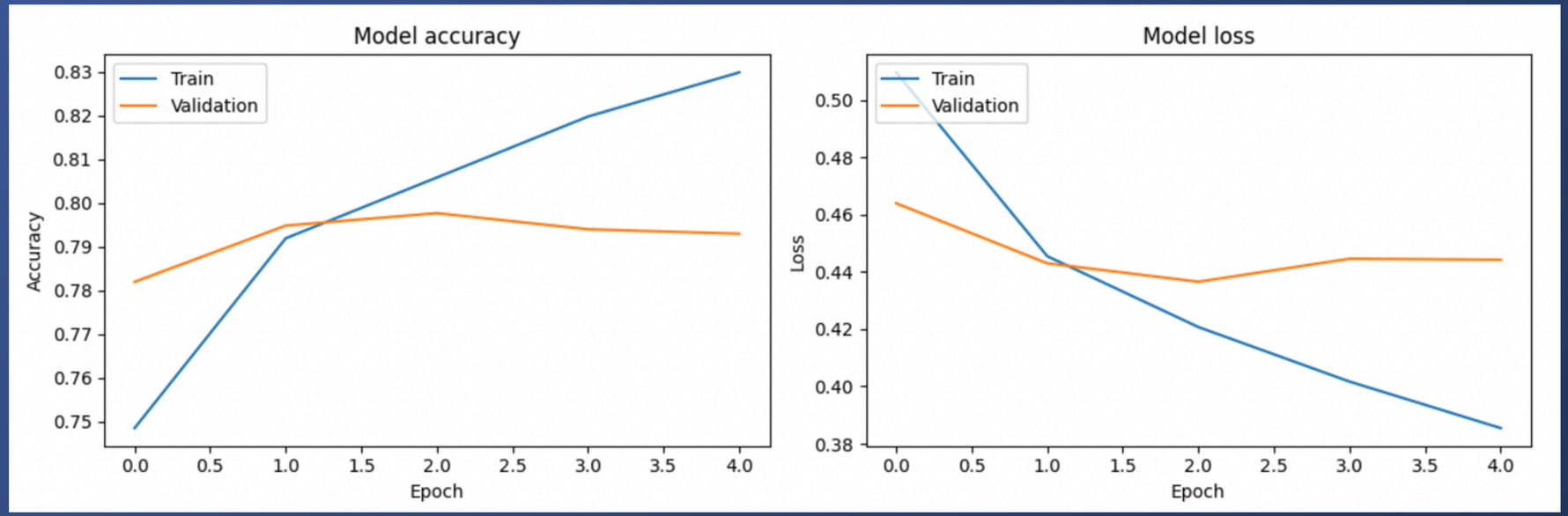
80%

This model appears to be more stable and is earning a slightly greater validation accuracy. It thus might suit better for more complex tasks.

DEPLOYED MODELS

More complex model - Visualization

Metric	Result
Accuracy	0.7972
Precision	0.7870
Recall	0.8170
F1-Score	0.8017



The best performing model is the first configuration; after evaluating it on the test data we get an almost 80% of accuracy.

CONCLUSION

Final considerations

The BiLSTM model appears to be more suitable for complex tasks and generally behaves better in terms of validation accuracy. It is also less prone to underfitting and thus better for large-scale datasets.
Despite being complex in architecture, it tends to have less parameters so it is more cost-effective in terms of computational cost and memory usage.