

Covid-19 Impacts on Global Education

CSEo44o – ARTIFICIAL INTELLIGENCE

Teoman Berkay Ayaz - 1800004169

Muhammed Safa Uslu - 1700003646

İbrahim Ağcabay - 1900002653

Table of Contents

Introduction, Project Goals and Scope	2
Preferred Data Collection Method and Why?	3
Data Collection Process and Logic	4
Post Data Collection	5
An example communication regarding labels	10
Categorical Distributions of Our Dataset	11
Additional Information on Our dataset	11
Introduction to Our Neural Networks	12
All-classes classification NN	12
Hierarchical classification neural networks	12
Data Preprocessing.....	14
To Lowercase.....	14
Removal of Non-Alphabetic Characters.....	14
Tokenization of Tweets.....	14
Vocabulary Indexing	14
Padding.....	14
Shuffling	14
General Optimization Process	15
Adding Dropout layers to Our Network	16
Increasing & Decreasing the Neuron Count	16
Adding Regularizers	17
Learning Rate Adjustments	18
Adding and Removing Layers.....	19
Different Optimizers and Activation Functions.....	19
Issues Regarding Data Quality & Data Quantity	20
All-Classes Classification Neural Network.....	22
Hierarchical Classification Neural Networks	23
Relevance Binary Classifier	23
Sentiment Multiclass Classifier	25
Combined Model Performance	27
Information Extracted from the Data	28

Introduction, Project Goals and Scope

The following work was assigned to our group (G15) as a term project for the CSEo440 – Artificial Intelligence course at IKU. The group conducting the study consists of the following members:

- Teoman Berkay Ayaz - 1800004169
- Muhammed Safa Uslu - 1700003646
- İbrahim Ağcabay - 1900002653

The topic of the project, as defined in the project description is:

“During pandemic, conditions are changed in education just like all other areas. To adapt to this new environment, governments decided to continue education online many times. For two years and even now, sometimes education and even exams are taken online. But what is the effect of this on students? Was it better for their learning, or worse? These effects can also change from person to person, and different learning skills can respond in different ways. In this project, your aim is to inspect this topic in many aspects and parameters by using tweets.”

The project in accordance with the project description seeks to:

- Develop a (series of) Neural Network(s) to classify tweets based on the sentiment they may or may not contain alongside their relevance.
- Minimize underfit & overfit in our neural networks.
- Optimize our neural networks to the highest possible accuracy rate.
- Extract information regarding the impact of covid-19 on education.

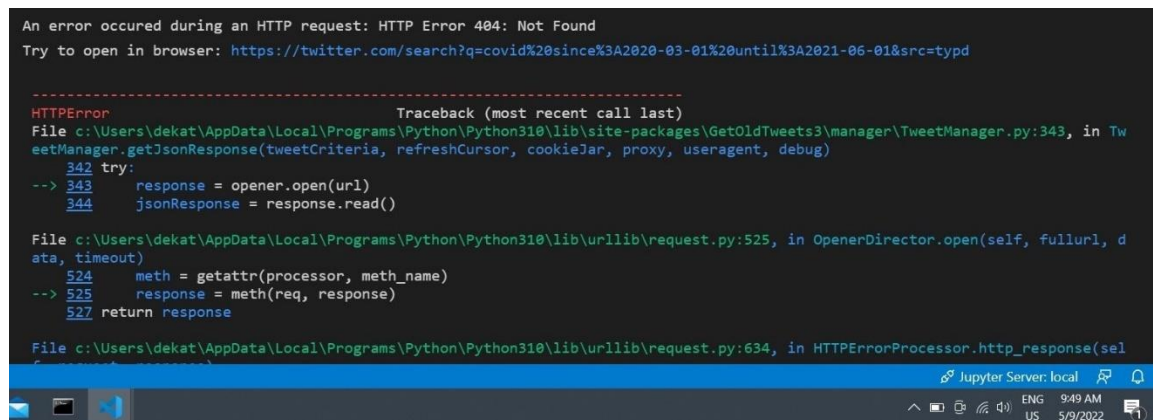
Preferred Data Collection Method and Why?

With regards to data collecting there were a couple of options we could have chosen. There are multiple API's and tools alongside the option to manually scrape data off Twitter by writing a web scraping script with python.

The web scraping approach was unfortunately not successful due to Twitter not allowing for web scraping in its user agreement alongside other issues regarding time constraints and a lack of experience in data collecting via web scraping.

Aside from the web scraping approach, the first method we tried was Twitter API since it is the "Official" method for collecting data from Twitter. This approach however had an issue, very detrimental to our project; mainly with limiting the specific period of time we can collect tweets from to the last week. Since we could only collect tweets from a week ago instead of our desired period which was March 2020 – March 2021, this method was found to be "Unsuitable" regarding the data collection process of our project.

The next option we attempted to use was GetOldTweets3 for python. GetOldTweets3 is a tool that is used to scrape data off Twitter and can go back to our desired period to collect tweets. The Problem however was that the tool was no longer working in 2022. As far as it is to our knowledge the problem seems to be that since twitter does not allow for web scraping and GOT3 simply collects data with web scraping. Hence, we could not use GOT3.



```
An error occurred during an HTTP request: HTTP Error 404: Not Found
Try to open in browser: https://twitter.com/search?q=covid%20since%3A2020-03-01%20until%3A2021-06-01&src=typd

-----
HTTPError                                Traceback (most recent call last)
File c:\Users\dekat\AppData\Local\Programs\Python\Python310\lib\site-packages\GetOldTweets3\manager\TweetManager.py:343, in TweetManager.getJsonResponse(tweetCriteria, refreshCursor, cookieJar, proxy, useragent, debug)
    342 try:
--> 343     response = opener.open(url)
    344     jsonResponse = response.read()

File c:\Users\dekat\AppData\Local\Programs\Python\Python310\lib\urllib\request.py:525, in OpenerDirector.open(self, fullurl, data, timeout)
    524     meth = getattr(processor, meth_name)
--> 525     response = meth(req, response)
    526     return response

File c:\Users\dekat\AppData\Local\Programs\Python\Python310\lib\urllib\request.py:634, in HTTPErrorProcessor.http_response(self, req, response, code, msg, headers, readdata)
    634     if code < 200 or code > 399:
        raise HTTPError(req.get_full_url(), code, msg, headers, readdata)
    635     return response
    636     raise HTTPError(req.get_full_url(), code, msg, headers, readdata)
```

With all the options above failing we resorted to use Twint as our tool for data collection. With that said however, Twint did come with its fair share of issues despite being a preferable choice compared to our other options at the time.

Data Collection Process and Logic

In the process of data collection, after deciding to use Twint, we needed to decide on how to collect the data. To collect the relevant tweets, we decided to use queries composed of a set of relevant keywords.

Keyword		Keyword
education	&&	covid-19
education	&&	coronavirus
education	&&	online
distance	&&	learning
student(s)	&&	covid-19
student(s)	&&	coronavirus

We used the “AND” operator in our queries to pair our keywords to make sure we only collect relevant tweets. Please note that the term “relevant” in this context stands for a tweet that contains the pair of keywords we are looking for.

The pairs consist of three generalized subjects; first one is online education, second one is education and covid-19, and the last one is student(s) and covid-19.

With these in mind we set our limit to 10.000 tweets per pair and started to collect as many tweets as we could.

After we had collected a number of tweets deemed sufficient regarding the subject at hand (which was a total of 14002 tweets), we noticed that; although, our keywords were all English, some of the tweets we collected we marked as other than English. In our case, that is any row that did not have the “en” value in its “*language*” column. After cleaning these samples we were left with a total of 13388 tweets.

Post Data Collection

When collecting tweets, Twint collect a total of thirty-six features about those tweets including the tweet itself. We did not need twenty-four of them, so we decided to drop the said twenty-four. The list of features we kept and did not are, as the following:

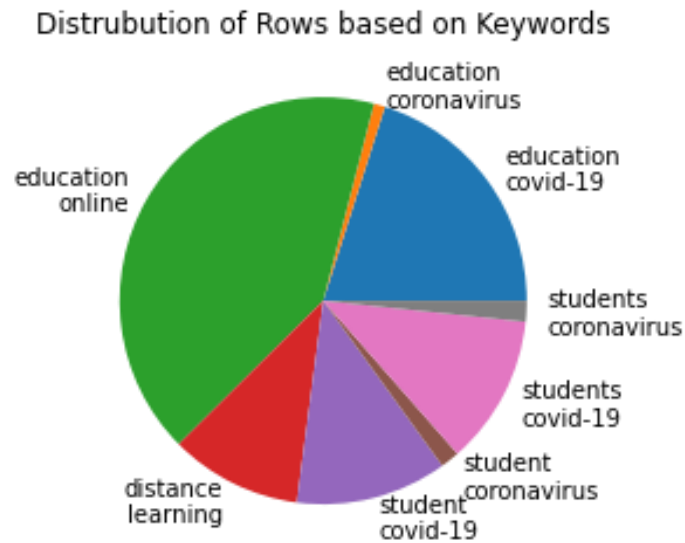
Feature	Removed	Feature	Removed
id	0	place	1
conversation_id	1	tweet	0
created_at	0	language	0
date	0	mentions	1
time	0	urls	1
timezone	0	photos	1
user_id	0	replies_count	1
username	0	retweets_count	1
name	0	likes_count	1
hashtags	1	source	1
cashtags	1	user_rt_id	1
link	0	user_rt	1
retweet	0	retweet_id	1
quote_url	1	reply_to	1
video	1	retweet_date	1
thumbnail	1	translate	1
near	1	trans_src	1
geo	1	trans_dest	1

The features we were interested in were mostly about the language of the said tweet, whether it was a retweet, link to that tweet for potential future use, time information on that tweet etc. .

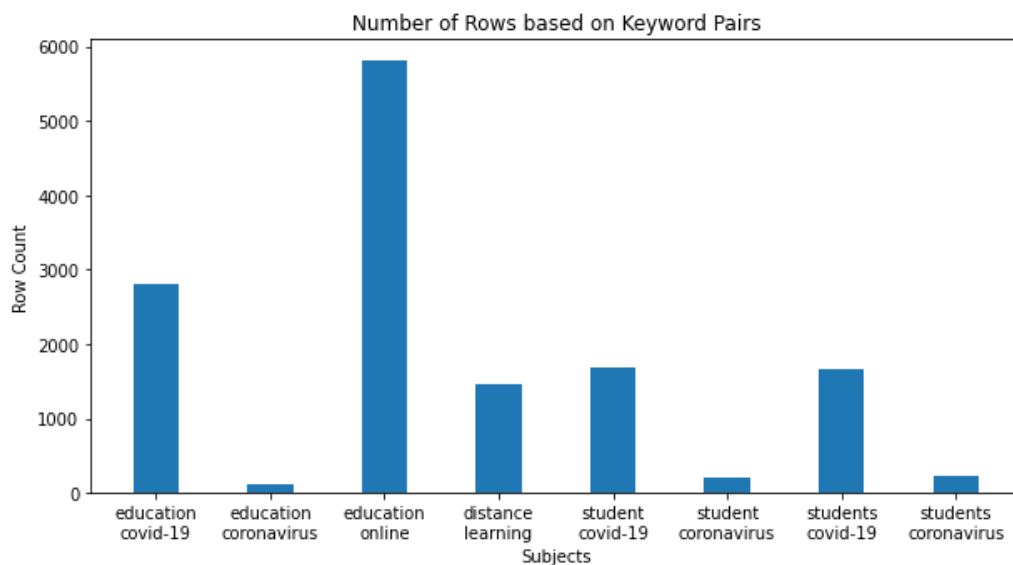
Aside from this matter we also decided to use “ISO-8859-1” encoding to load our dataset as a pandas dataframe.

Keyword Based Distributions of Our Dataset

The following graph is a pie chart displaying the distributions of rows in our dataset, belonging to their respective keyword pair. From the graph we can see that most of our data belongs to “online&&education” and “distance&&learning” keyword pairs. We inspect these two pairs in the same subject since they, for the most part are synonymous with each other.



The second graph on this subject is a bar plot of same keyword pairs, displaying the approximate number of rows belonging to each keyword pair.



Process of Thought Regarding Our Dataset Classes

To conduct our study, we first needed to add labels to our dataset. We first needed to decide how many labels we wanted to have and what these labels will indicate. Since we were doing a sentiment analysis project, we decided to have three classes regarding the sentiments; alongside a class of tweets that do not contain a sentiment. The classes in our dataset are as the following:

Sentiment	Index
Negative	0
Neutral	1
Positive	2
Irrelevant	3

Please note that initially, when we were labeling our tweets, we hadn't decided to have a *“Neutral”* class; hence, initially the classes were:

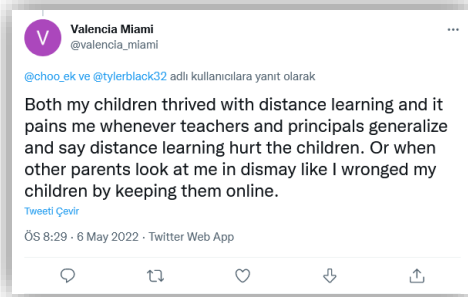
Sentiment	Index
Negative	0
Positive	1
Irrelevant	2

During the labeling process, when we had added the label *“Neutral”* to a tweet, we used the index 3 during this process. Later down the road, we used Pandas to fix the issue and switched over to the set of indexes mentioned earlier.

Labeling Process of Our Dataset

After deciding on the labels, we needed to decide on a standard on what label to give to each respective tweets in our dataset.

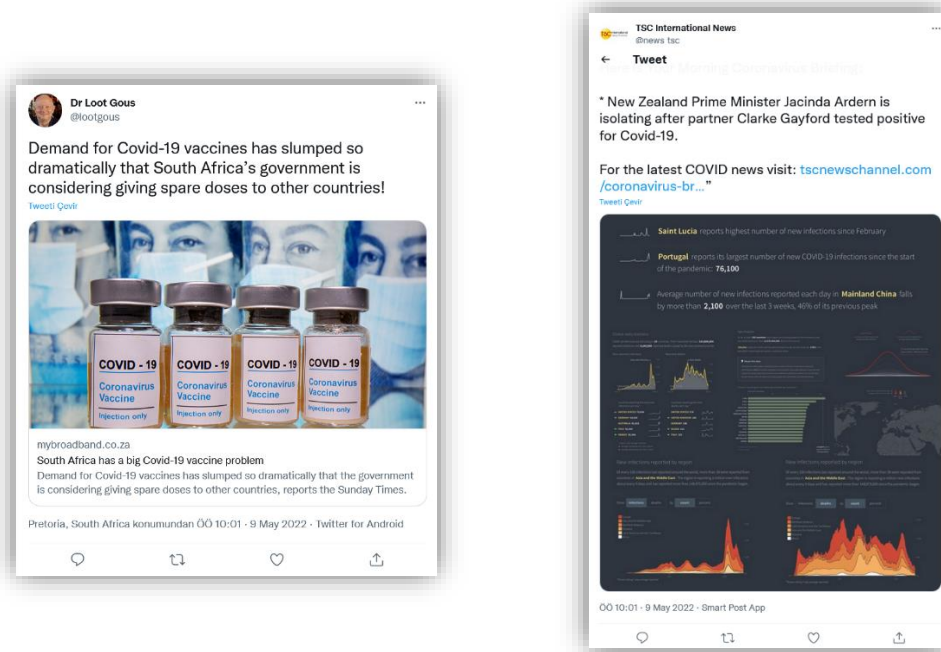
As a standard for positive sentiments, we decided to label anything positive regarding online education or any overall positive sentiment as a positive (2) as in the examples below:



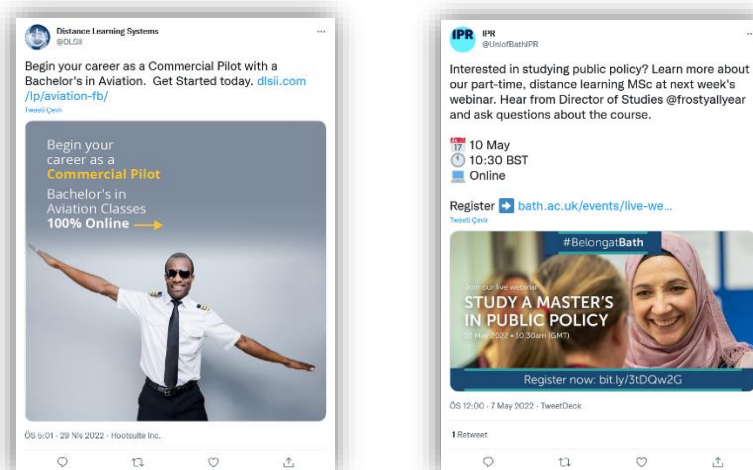
As a standard regarding negative sentiments, we decided to label any tweet bashing distance education or any overall negative sentiment as in the examples below:



Regarding neutral sentiments, we labeled any statistical, factual or any non-emotional relevant tweet as a neutral sentiment as in the examples below:



Lastly, on the matter of irrelevant tweets, anything from ads, news, bot tweets picked up tweet which is irrelevant to our study was labeled as irrelevant as in the following examples:



After setting a standard on the definition of each class we manually added labels to 5752 rows of tweets; of which only 2223 were usable. Due to the time constraints, we were unable to label all 13388 tweets and fully.

Regarding the labeling process we had certain issues with certain tweets. Mainly due to the ambiguous language a significant part of the tweets contained which was quite concerning regarding the quality of data. Even though members of our group were always in communication we often had conversations like the following, where we could not agree on the class of a given tweet.

An example communication regarding labels

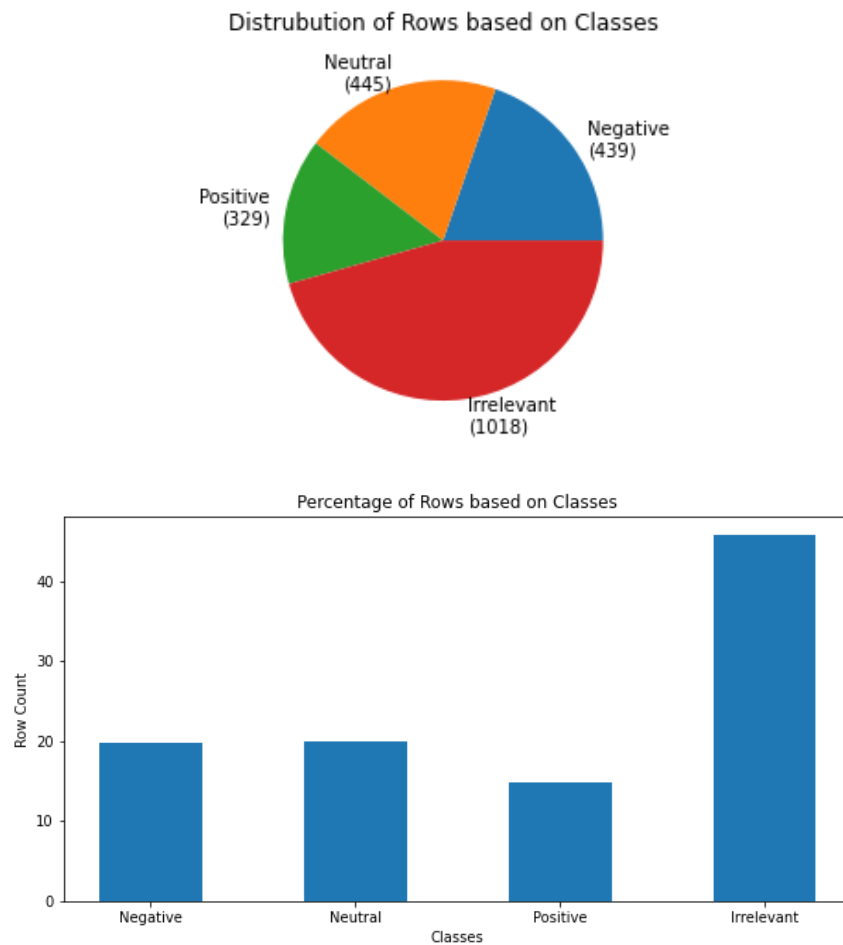
Taken from WhatsApp, 14/05/2022, paraphrased with fixed grammar, translated between Teoman Berkay Ayaz (1800004169) and İbrahim Ağcabay (1900002653)

- 1900002653: "Education has given me the confidence to fulfill my dreams," says ...
- 1900002653: Is this supposed to be a 0 or a 2?
- 1800004169: I think 1.
- 1800004169: It says "Education has given me the confidence to fulfill my dreams"
- 1900002653: Are you sure?
- 1900002653: It says that "We should help them catch up... what they've missed"
- 1800004169: Not really.
- 1800004169: Regardless, it is going to be a false classification as far as I can tell.
- 1900002653: Yeah, I think I agree.



Categorical Distributions of Our Dataset

A rather large chunk of our dataset, an approximate 45% of it is made up of irrelevant samples which are mostly composed of advertisements. The second largest class is the neutral class which makes up an approximate 21% of our dataset. The remaining two classes, negative and positive each respectively make up approximately 19% and 15%.



Additional Information on Our dataset

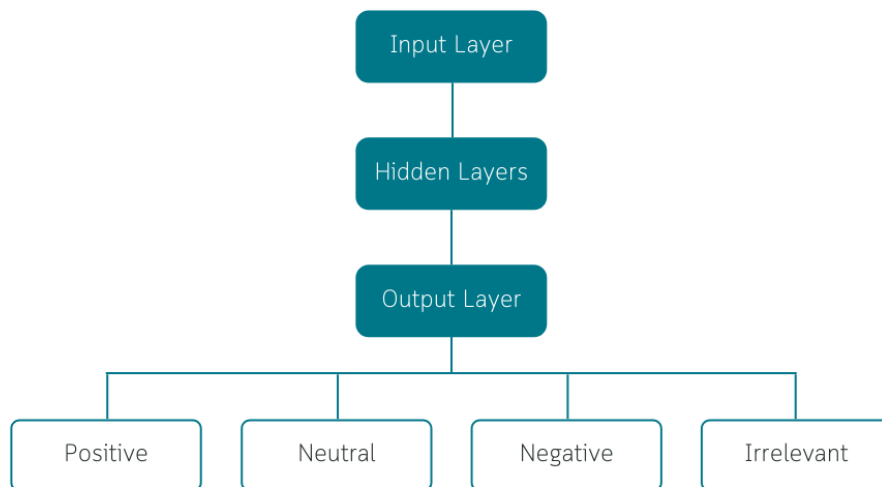
Our dataset contains a total of 9053 unique tokens with a maximum length of 37 tokens in a single tweet. Dataset consists of 2223 rows of total data and contains 12 features as mentioned earlier in the report.

Introduction to Our Neural Networks

For this project we decided to use two main approaches regarding how we wanted construct our neural networks.

ALL-CLASSES CLASSIFICATION NN

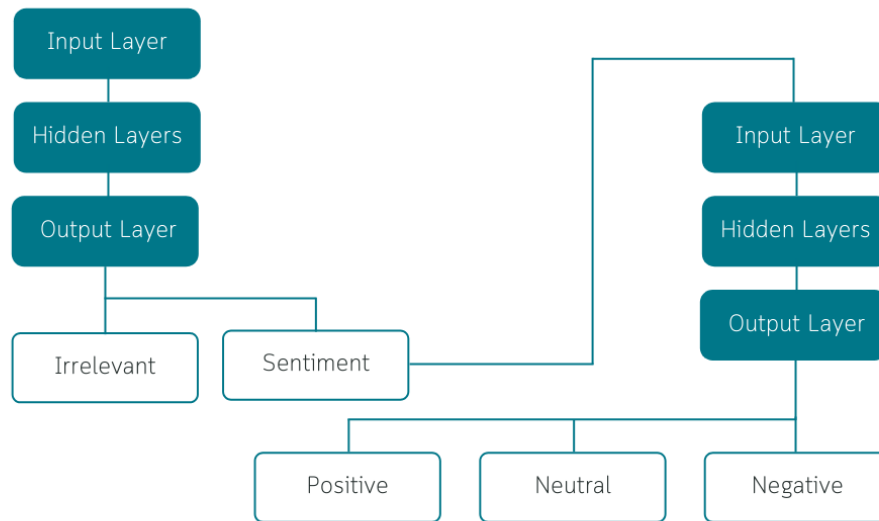
The first and primal (for the lack of a better term) of our neural networks is the “All Classes Classification” model. The model as shown in the flowchart below takes the input and simply classifies it between one of our four classes.



HIERARCHIAL CLASSIFICATION NEURAL NETWORKS

The second neural network is a set of two distinct neural networks. These networks work together to do the same classification in two separate steps, aiming to increase the accuracy of the final prediction.

The prior neural network in the hierarchy is the “Relevance Binary Classifier” responsible with classifying whether a given sample is irrelevant or does it contain a sentiment whether it be negative, positive or neutral to the subject.



After the initial classification, if the sample is classified as relevant; it is transferred over to the latter neural network in the hierarchy. After the transfer, the multiclass classification network classifies it as one of the three possible classes: positive, negative, and neutral.

One thing that should be noted with this approach is that, in the event that the first classification is false, the second one has no chance of being correct since irrelevant class does not contain any sentiment within it or whatsoever.

Data Preprocessing

To Lowercase

The initial step we took in our preprocessing was to convert all letters to lowercase for later purposes. We used the built in “.to_lowercase” function in python to achieve this. By the end of this step, all tweets in our dataset are converted into all lowercase like in the following example:

[Link to the Tweet](#)

- “Distance Learning wild asf frfr.” => “distance learning wild asf frfr.”

Removal of Non-Alphabetic Characters

To remove non-alphabetic characters in our dataset, we used “stopwords” from NLTK’s corpus. In this step we remove anything that is not found between lowercase a to z in English alphabet.

- “distance learning wild asf frfr.” => “distance learning wild asf frfr”

Tokenization of Tweets

To split our tweets into tokens, we used the “word_tokenizer” from NLTK. In this step we convert every word found in a tweet into a token in an array.

- “distance learning wild asf frfr” => [“distance”, “learning”, “wild”, “asf”, “frfr”]

Vocabulary Indexing

After we split our tweets into tokens, we store the index that is ordered based on how common a word is into an array which becomes a row in our training set

- [“distance”, “learning”, “wild”, “asf”, “frfr”] => [3, 2, 2801, 4497, 4498]

Padding

To pass our data to our model, we need each row to be the same length. To achieve this, we add padding to every set of indexes using “pad_sequences” from keras, preprocessing, sequence.

- [3, 2, 2801, 4497, 4498] => [3, 2, 2801, 4497, 4498, 0, 0, 0, 0, ..., 0]

Shuffling

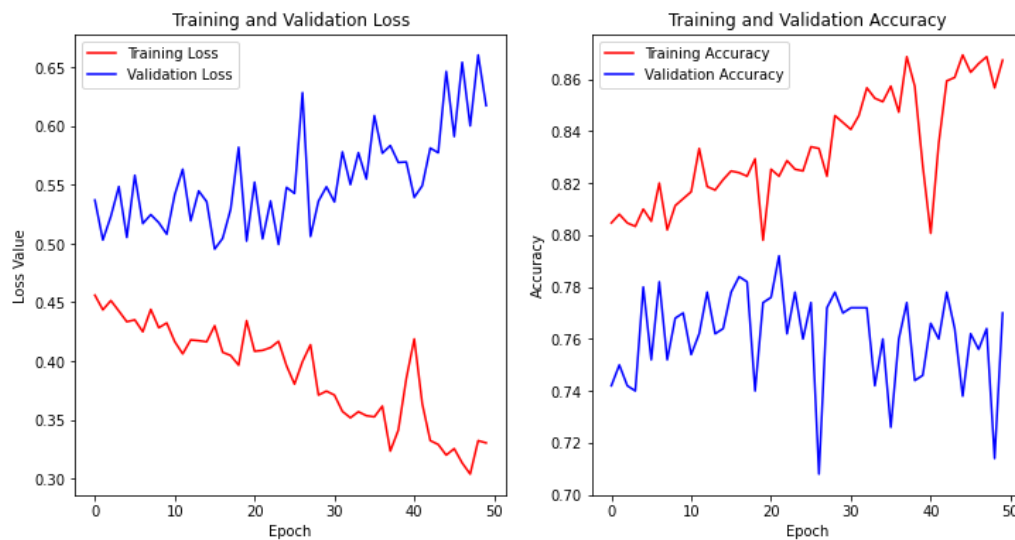
One additional step we took with our data is that we shuffled it, so samples are distributed more evenly throughout the dataset.

General Optimization Process

All our models are meant to be used in Natural Language Processing. Their architectures are rather similar as well. Hence, the optimization process in each model was also rather similar aside from some key distinctions like using tanh activation instead of relu, output layer units etc. which will be told about later in each respective sub-section of each model.

We began with a simple neural network consisting of an embedding layer as its input layer, an LSTM layer with 64 units, a dense layer with 10 units and relu activation, and lastly, an output layer with softmax or sigmoid activation depending on the task alongside 2-4 units depending on the task as well.

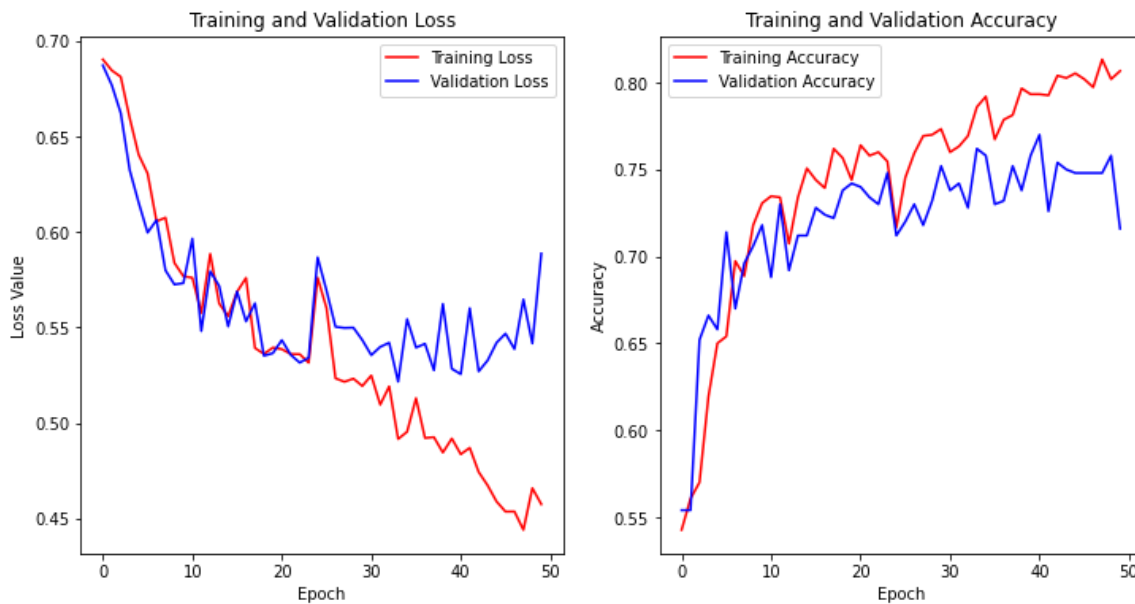
This architecture on average caused quite intense cases of overfitting. The following graph, taken from our Relevance Binary Classification Neural Network is a fitting example of this.



As it can be seen in the graph, the sheer amount of overfit makes the neural network unsuitable for use in any application. To combat the overfitting issue, we used certain approaches which can reduce overfitting in a neural network.

Adding Dropout layers to Our Network

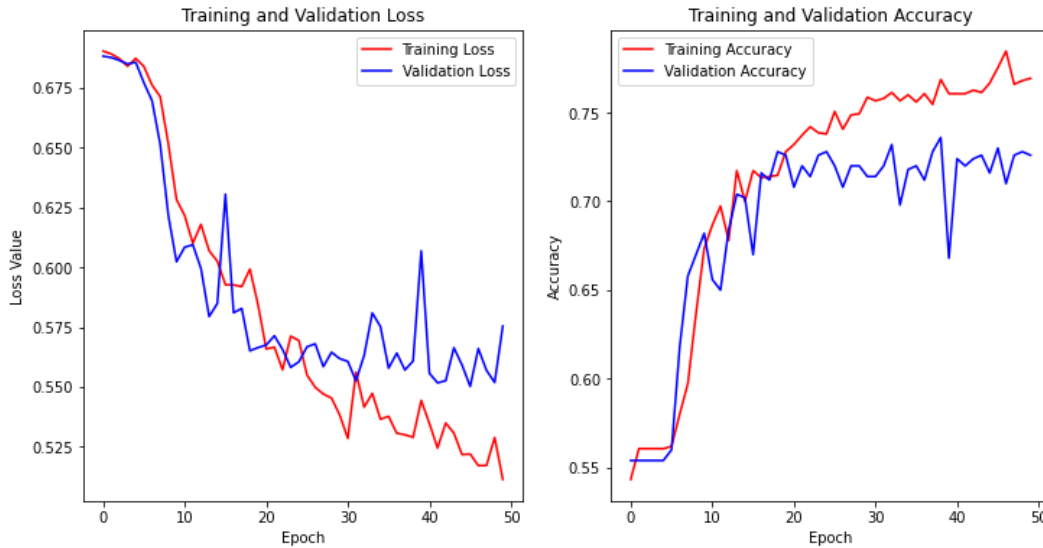
Adding dropout layers is one the simplest and most reliable ways to combat overfitting. In this approach we needed to find the optimal dropout rate and how many dropout layers we needed in the network. For most of the optimization process adding more than a single dropout has caused exacerbated underfitting issues so we mostly added a single dropout layer with a dropout rate of 0.4-0.6. Although this approach helped immensely with the overfitting issue, it was not the entirety of the solution we were looking for



Increasing & Decreasing the Neuron Count

Increasing the neuron (unit) count unnecessarily can cause an exacerbated level of overfitting. During the optimization process we increased and decreased the neuron counts to find the optimal amount in each respective neural network.

The following graph is the same network as the previous with the exception of 16 units in the LSTM layer and the 8 units in the dense layer.



When we look at the graph, we can see that the differences in training and validation accuracies and losses are quite notably smaller. Hence, we can say that this approach did help us with the overfitting issue.

Adding Regularizers

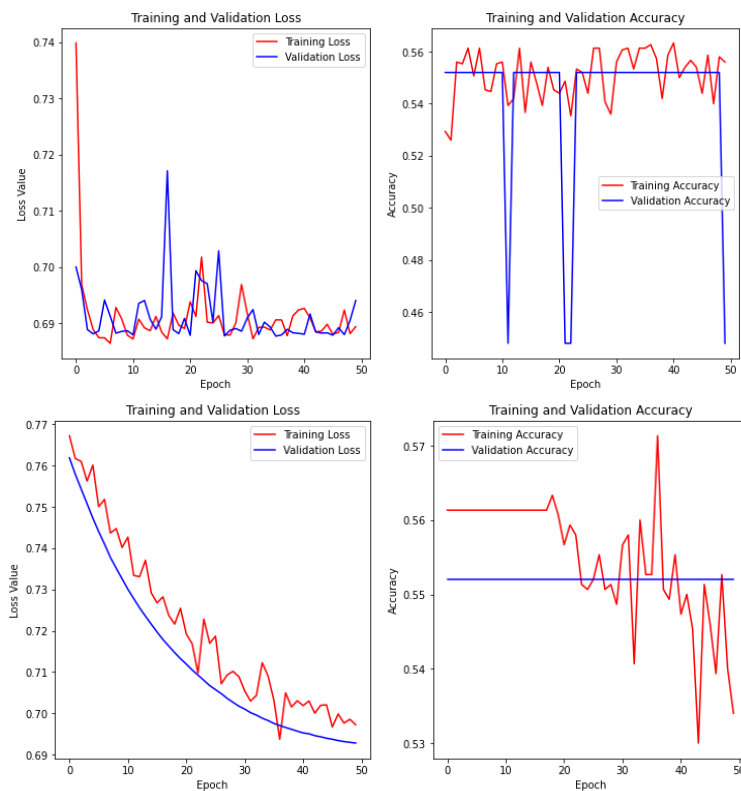
To calibrate our models further we used l_2 regularizers as our bias and kernel regularizers. This approach was mainly used to smooth the peaks in our graphs and stabilize our models. We typically used a rate of 0.001-0.00001



When we compare current graph with the previous graph, we can see that even though, the same level of overfitting is still there, the peaks have been considerably smoothed out, indicating that the current model is much more stable.

Learning Rate Adjustments

Adjusting the learning rate is one of the other methods to stabilize our models. When we adjust the learning rate however finding the right rate is essential since too high or too low of a learning rate (0.1 and 0.0001 in the following example) will cause our model to flatline as shown in the following graphs.

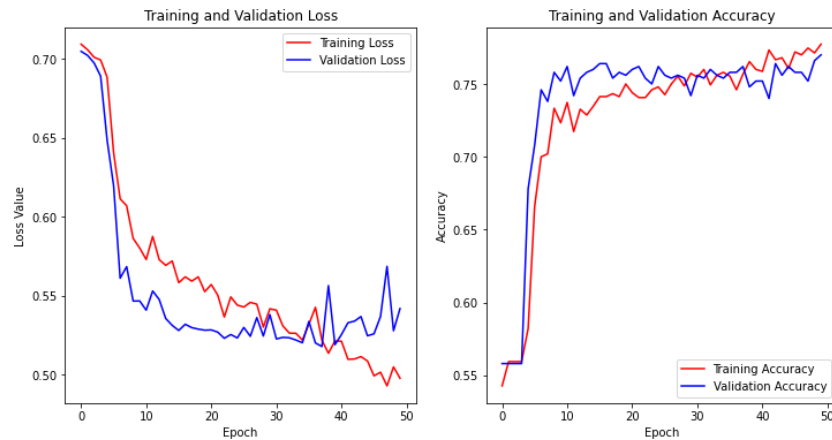


Throughout the optimization process, we found the default learning rate of Adam optimizer which is 0.001 to be a good learning rate for our models.

Adding and Removing Layers

Although the overfit at that point was minimized, our model could still perform better with a more suitable architecture. So far, we only worked on changes on the initial architecture. Hence, we tried adding more layers to see how the model would perform.

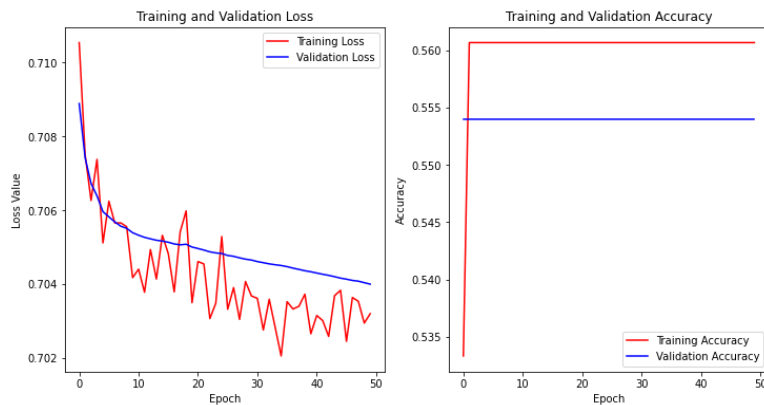
In the following example we added a flatten layer after our LSTM layer and a second dense layer with same regularizers and 8 units.



When we compare this graph to the previous graphs, we can tell that this new architecture fits our model much than the previous architectures we tried.

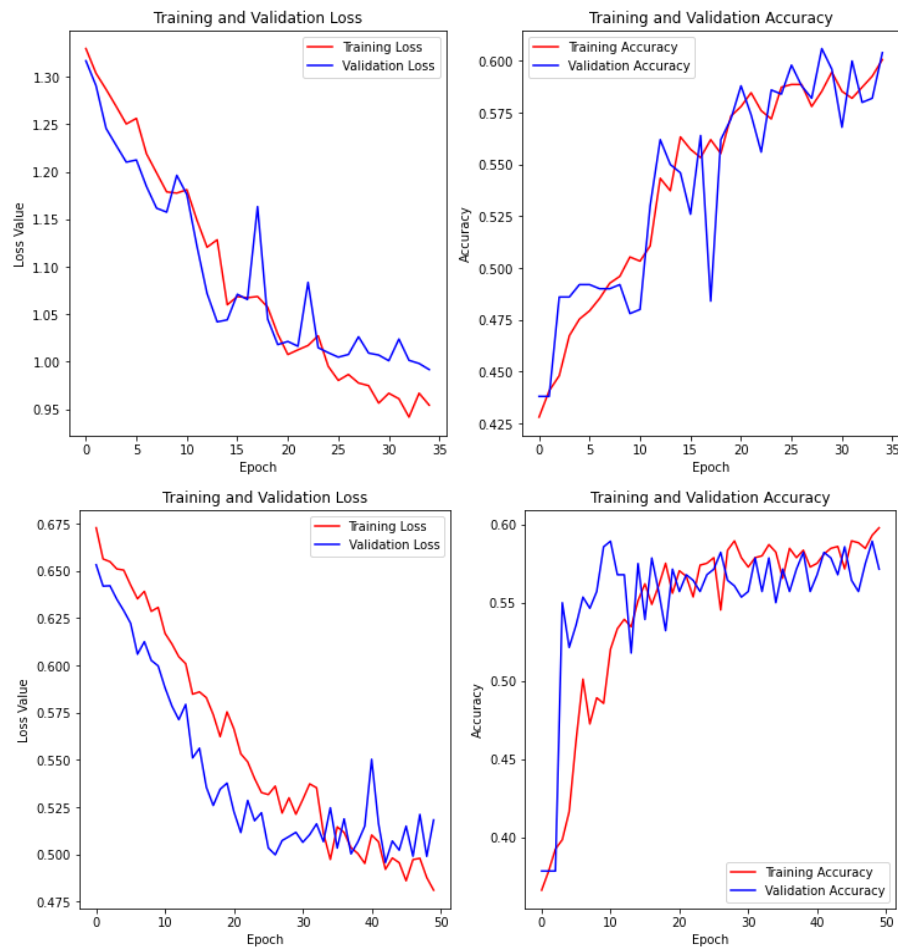
Different Optimizers and Activation Functions

Throughout our study we tried numerous activation functions in our layers (most notably tanh) and optimizers for our models. Generally, “Adam” was the best choice we could use regarding optimizers considering alternatives like “SGD” which caused the model to flatline as in the following graph.



Issues Regarding Data Quality & Data Quantity

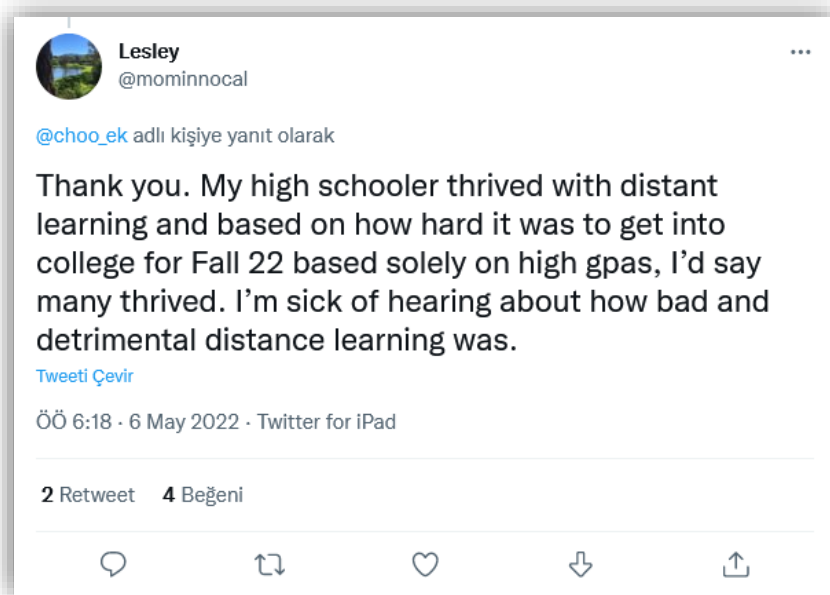
When we were optimizing our models, we noticed that some of were stuck at around 55% to 60% accuracy even though there was minimal to none overfit as in the following graphs.



Regardless of the approach we took we could never get our accuracies above 60% which was quite significantly below the desired 70%-80% range for the project, with the exception of our Relevance Binary Classification model which reaches roughly ($\pm 2\%$) 75% accuracy.

When we inspect the tweet samples in our dataset, we can see that the level of distinction between the classes are quite low aside from the “Irrelevant – 3” class and the other classes. Our irrelevant class samples contain quite distinct phrases like “Learn More” or “Register Today” quite commonly; hence, they can be distinguished quite easily unlike our sentiment classes.

Our sentiment classes sometimes tend to be rather ambiguous or not use clear and concise language, which can even make it difficult for a human distinguish them like the sample below. When we look at the sample tweet below; even though, the sentiment regarding distance learning is positive there is still distinguishable negative language within it.



Aside from the lack of a distinct language between classes, the amount of data is also a rather big issue since there is a direct correlation with model accuracy and amount of data to train with. Since we wanted to create our models earlier to have more time left for optimization, we could only work with 2000 rows of data which was also one of the more impactful factors regarding accuracy.

All-Classes Classification Neural Network

Purpose

All-Classes Classification Neural Network as the name suggests, is the neural network that classifies between all four possible classes (negative, neutral, positive, irrelevant).

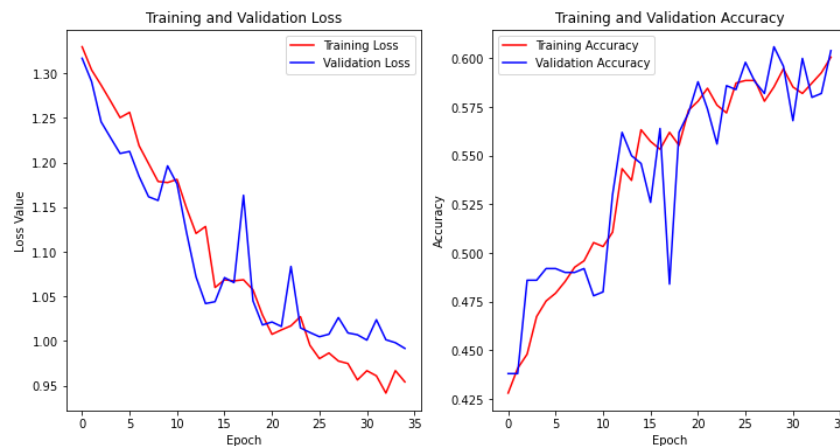
Architecture

Its architecture consists of five total layers starting with; an embedding input layer with “trainable” parameter set to false, an LSTM layer with 64 units, a dense layer with 16 units, “relu” activation and 0.1 as an l2 bias regularizer, a dropout layer with a dropout rate of 0.5, and lastly, an output layer with “softmax” activation since it is a multiclass classification model.



Performance

This model yields an accuracy of about ($\pm 2\%$) 60%. The training of this model continues until epoch 50 since overfit was rather minimal and dismissible.

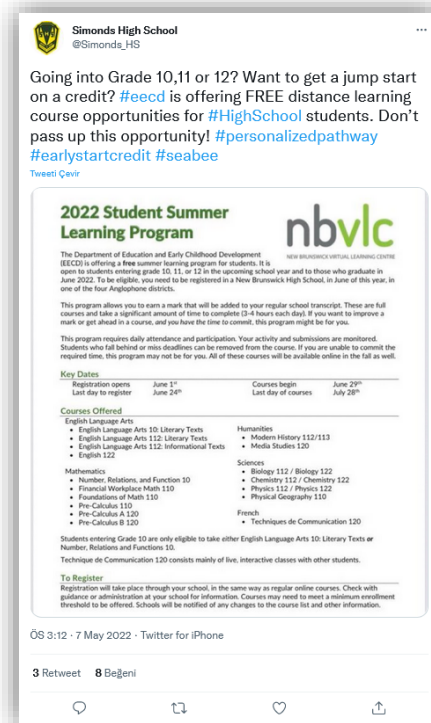
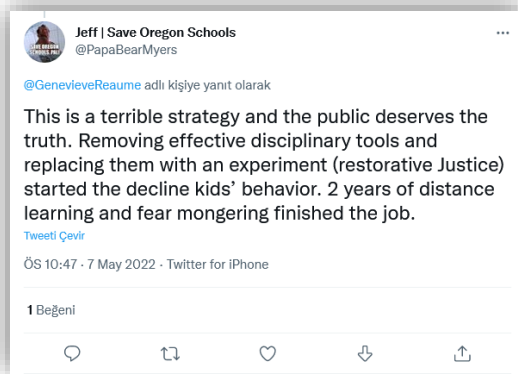


Hierarchical Classification Neural Networks

Relevance Binary Classifier

Purpose

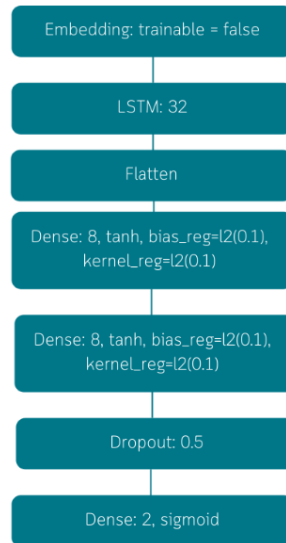
The first model in our hierarchy is the Relevance Binary Classifier. It classifies whether a tweet contains a sentiment whether it be negative, neutral, or positive. Or, if it simply is irrelevant to the subject such as an advertisement tweet such as examples below.



Although the model as a standalone model isn't worth a lot with regards to analyzing data, we still believe it has potential in being used to tell apart things like advertisements, irrelevant bot tweets etc. from actual tweets that represent a sentiment.

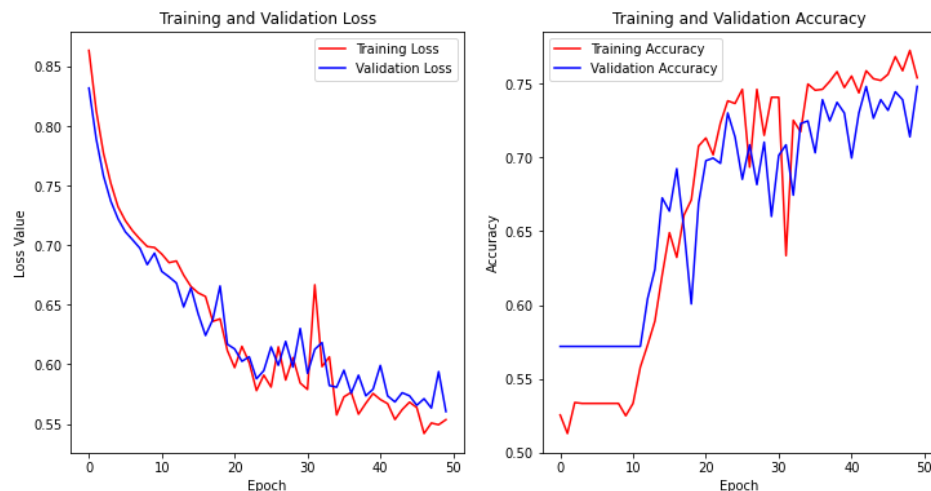
Architecture

This model consists of a total of seven layers starting with; an embedding layer with “trainable” parameter set to false as its input layer, an LSTM layer with 32 units, a flatten that comes after the LSTM layer, two back to back dense layers with 8 units, tanh activation function, l2 regularizers with 0.001 regularization rate as kernel and bias regularizers, a dropout layer with a dropout rate of 0.5 , and an output layer with sigmoid activation since it is a binary classification model.

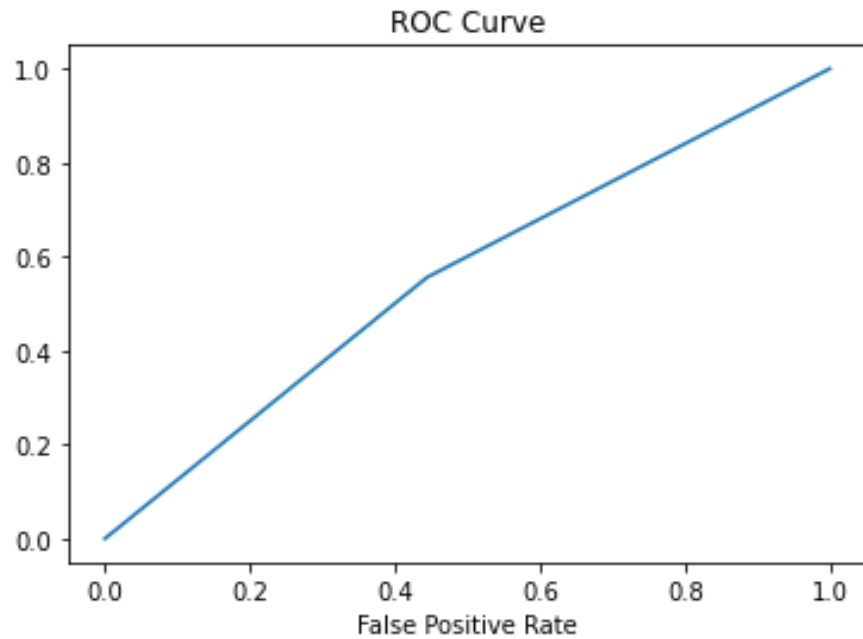


Performance

This model yields around ($\pm 2\%$) 75% accuracy. The training model continues until epoch 50 since even though, there is some noticeable overfit the overfit is quite minimal and dismissible.



One other detail regarding the model is that it seems to flatline until around epoch 10. We interpreted that as there simply not being enough data for the training of our model, which is something we experienced in the past.



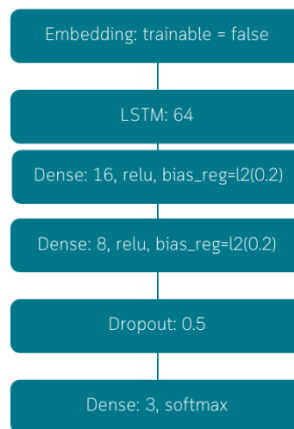
Sentiment Multiclass Classifier

Purpose

The second model in the hierarchy is the Sentiment Multiclass Classifier. It classifies the sample it gets from the Relevance Binary Classifier, and classifies it as one of the three possible classes (negative, neutral, positive).

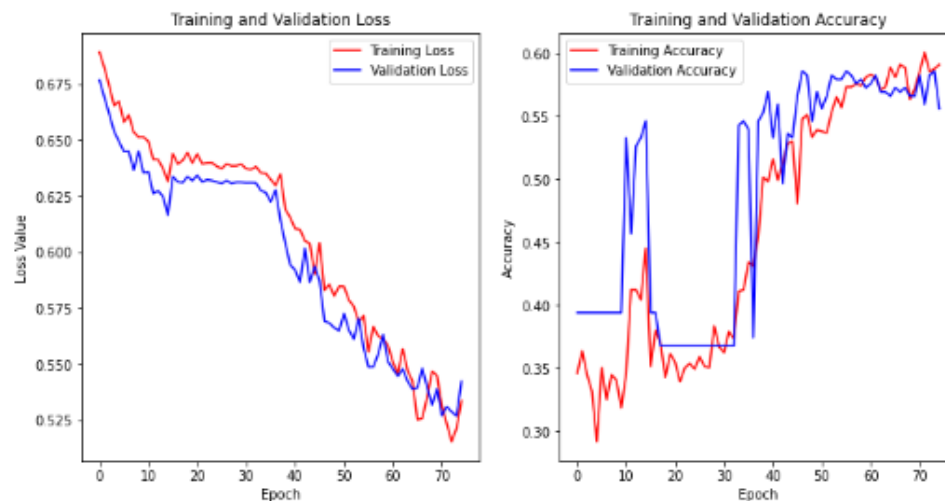
Architecture

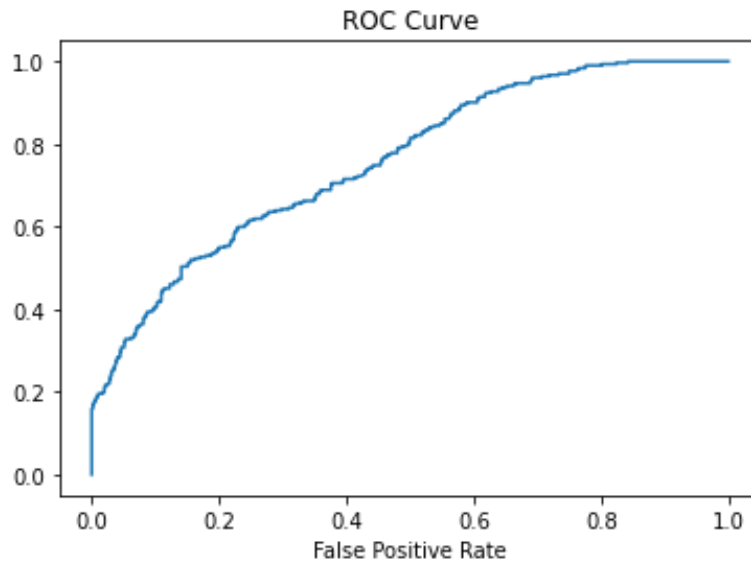
Its architecture is composed of a total of 6 layers. The initial layer is the embedding input layer with trainable set to false followed by an LSTM layer with 64 units. The next layer is a dense layer with 16 units, “relu” activation and an l2 regularizer with a regularization factor of 0.2 as its bias regularizer. Next, we have another dense layer with 8 units and, “relu” activation and an l2 regularizer with a regularization factor of 0.2 as its bias regularizer. Next, we have a dropout layer with a dropout rate of 0.55 followed by the output layer with “softmax” activation since it is a multiclass classification model.



Performance

The model has roughly ($\pm 3\%$) 59% accuracy with minimal to none overfit. With that said, the model is rather unstable since its peaks are much sharper compared to our other models. The model continues to train until epoch 75 since the overfit is minimal to none.





One other note regarding model performance, is that needs to be done is that the model flatlines between epoch 10 and then continues without flatlining until around epoch 20 only to end flatlining at around epoch 35.

The model seems to stabilize around epoch 50 hence unlike other models this one, keeps training until epoch 75.

Combined Model Performance

Our two model that work back-to-back originally aimed to increase the accuracy. The idea can be described as; instead of being mediocre at multiple things, being great at one thing will bring more success. The idea here was that instead of having a single model do all the distinction between classes, having two models that work on different classes would increase the accuracy.

Our tests show us that although Relevance Binary Classifier yields a much higher accuracy than the All-Classes Classifier which has about the same accuracy as Sentiment Multiclass Classifier, combined probability shows us that the combined accuracy is:

$$- \quad 0.75 * 0.59 = 0.4425 \Rightarrow 44.25\%$$

When we compare the 44.25% accuracy with 60% accuracy of the All-Classes Classifier, we can see that, in our case instead of using the models that work back-to-back, using the standalone All-Classes Classifier will yield a better analysis.

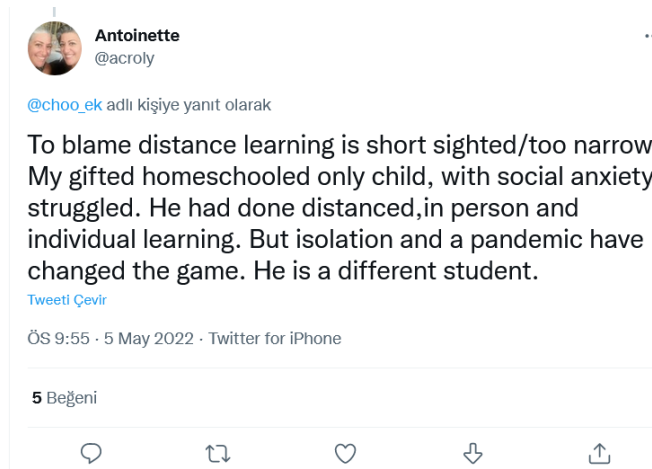
Information Extracted from the Data

When we look at the ratio of tweets who are happy about the situation of education under covid conditions (positive sentiments – Class 2) and the tweets that are negative regarding the situation (negative sentiments – Class 0) which is roughly (rounded down to two decimals) 0.75, we can see that significantly higher amount of people are unsatisfied with the situation of education in covid conditions.

When we thoroughly inspect the tweets, we can see that most of the negative tweets have certain subjects in common mostly regarding inequity, poverty and that some students are falling behind compared to their peers.



With that said however, despite being about 25% lower in numbers, some people are satisfied or happy about the fact that education can now be online. One of the more common things that the positive tweets was that students with social anxiety performed better in distance learning than they did with face-to-face learning.



According to the information on the tweets it seems that, distance learning gave students with certain disabilities and/or high levels of social anxiety a better chance at learning and having higher levels of academic success.

One other notable find was that about 45% of the tweets were in the irrelevant class (Class 3), which mostly comprise of advertisements. When we inspect the said tweets, we can see that a significant amount educational institutions such as universities, academies, business schools etc. are offering programs that would normally would have been a 4-year undergraduate program or a 2-year masters program pre-covid.

This shows us that the education industry has adapted to this new era of online learning and there are lots of more thing that can be learned online ranging from flying a commercial aircraft to developing a successful neural network model to analyze tweets off of twitter.

