Electronics and Computer Science

Faculty of Physical Sciences and Engineering

University of Southampton

William Hirth
August 2022

Researching Sentiment Analysis using Natural Language Processing

Project supervisor: Professor Adam Prugel-Bennett
Second examiner: Dr. Istebreq Saeedi

A project report submitted for the award of
Bsc Computer Science

Abstract

Natural language processing is the field of research focused on computers using text or speech as input and providing some meaningful output (such as predicting the next word or summarising the text), often using machine learning. It has a rich history dating back to the 1950s with famous examples such as the Turing test, a test devised to test if a computer can think like a human, being commonly spoken of outside of computer science. With advances in the field such as GPT-3 computers can produce authentic sounding text, compare two passages, and calculate the similarity between them, or produce a written form of spoken text automatically. On the other hand, one area in which machine learning struggles is deriving the meaning behind a sentence but with the field ever changing and advances in sentiment analysis computers may be able to “understand” speech soon. This project aims to research these current techniques and produce a model able to, with high accuracy, take multiple pieces of text and classify them into two opposing groups, for example, positive and negative sentiments, or real and fake.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

I have not used any resources produced by anyone else.

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly-accessible repositories e.g. Github, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report, or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

I have not submitted any part of this work for another assessment.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report, and list the ethical approval reference number(s) in the box below.

My work did not involve human participants, their cells or data, or animals.

ECS Statement of Originality Template, updated August 2018, Alex Weddell aiofficer@ecs.soton.ac.uk

Contents

Table of Contents

1 Introduction

1.1 Problem

Sentiment analysis is using methods such as machine learning for natural language processing to determine the meaning or intent behind a certain phrase. Natural language processing, in turn, refers to the field of study in which computers are used to process language, such as written text, often using artificial intelligence. The problem that the research will be applied to is a competition on the website Kaggle.com which hosts competitions relating to machine learning and data science. This eliminates the need to find training and testing data for my model and will provide a concrete metric of how well my model is performing. Applying the research to a competition gives a real-world use case and allows the report to delve deeper into specific technologies, methods, and algorithms which are currently being used in the field.

1.2 Aims

The goal of this project is to research the topic of sentiment analysis and natural language processing in general then develop software which can take real world data, such as tweets provided by Kaggle, and classify them using a binary classifier into two classes (in this case real and fake). This software will be trained using a training set also provided by Kaggle and the metric used to evaluate the accuracy of the model will be the percentage of correct classifications it makes on an unseen test dataset from Kaggle. The goal with this Kaggle competition is to reach between 80 and 85% accuracy as this is where the top scores lie are achieving and will be challenging to achieve but still attainable. According to research "with Mechanical Turk, humans only agree 79% of the time[1]" when it comes to sentiment analysis which means that with 80% accuracy the model will have better predictive power than a human (although it is not stated what type of sentiment analysis the humans were required to do). The fields that are being researched in this project are rapidly changing and have a wide breadth of research which will make it impossible to research all of it, however, the aim of this project is to understand the fundamental concepts and apply them to the specific problem.

1.3 Scope

The project will focus on sentiment analysis and natural language processing; however, it will more specifically be researching binary classifiers as this is the problem which the research will be applied to. This project will explore the current techniques used for sentiment analysis

[1] (Ogneva, 2010)

and the methods used in other applications of artificial intelligence for natural language processing.

2 Research

2.1 Terms

- Loss (also known as cost) – The difference between the expected value and the predicted value (There are different loss metrics such as MAE, RMSE etc.)
- Model – The machine learning algorithm after the weights of the layers have been trained
- Activation function – Defines whether a neuron is “activated” or not, it takes the sum of the (weight*input)+bias and depending on the function (for example, ReLU or sigmoid) the neuron fires.
- Batch – The amount of data seen over one training iteration
- Epoch – A number of iterations where all the training data has been seen once by the model (1 Epoch = Datasize/Batchsize number of iterations)
- Overfitting – When a model’s predictive power has become too specific to the training data (the underlying trend has not been learnt)
- Underfitting – When model’s predictive power is not specific enough

Underfitting vs overfitting can be thought of in terms of functions, a model that is overfit will be a very high order polynomial where all the datapoints will directly fit on the line and underfitting would be linear or very low order and the line does not fit to many datapoints.

2.2 Natural language processing

Natural language processing (NLP) is the branch of computer science relating to allowing computers to process text or speech. This includes speech recognition, parts of speech tagging, grammar induction, name entity recognition, machine translation etc. Classical NLP started with hard coded rules, for example, regular expressions (regex) which take text as input and finds matches or makes replacements to the text. Programs using regex are still commonly used today, including in text pre-processing or in conjunction with other more complex algorithms and machine learning. Modern NLP on the other hand does not explicitly specify features but instead uses neural networks and artificial intelligence to infer rules or features of the text to model natural language. While the focus of this paper is on modern NLP that is not to say that it is inherently superior; there are issues with the opaque “black box” nature of these language models which only exacerbates the issues of models developing biases due to bad data and makes it challenging to debug or locate these issues.

2.3 Embedding

For a machine to understand text a representation of the data is required; computers deal with numbers so the text must be translated into numbers which could be achieved by simply using the ascii code of the individual characters. However, this would not be very efficient as not all combinations of characters are valid words and looking at individual words would be much more important than looking at individual characters. A lot of the model's effort would also be used trying to figure out what valid words are instead of how the words fit together which would not be a useful way to train a model. Therefore, the text is split up into words instead (which could be considered unique tokens), however, neural networks require the input to be in vector form. This leads onto a way to embed text into a computer readable format called one hot embedding. This approach to vectorising text is using a one hot vector, which is a vector filled with all 0s except the index which refers to the given word, for example if the indexes represent [“this”, “and”, “out”] then the one hot encoding of “out” would be [0,0,1]. This has many issues, for example, as the dictionary size grows then the size of the vectors grows too so every representation of a word will have an extra dimension. When encoding sentences there is also the issue of word order being lost e.g., “cat in house” and “house in cat” are the same and repeat words being lost e.g., “run run run” and “run” are the same. This can lead to information being lost which could be useful for classification and training.

A useful embedding for a word will mean that similar words in the vector space will be close to one another, one way to do this is with word2vec (or GloVe) which uses words that are often used in similar contexts and maps them onto a vector that has hundreds of dimensions. This is useful for training as it means that the model can see how close the predicted word was to the correct solution by checking how close the vector it predicted, and the true vector was. In this way, unlike with one hot encoding, the model can understand the relationships or similarities between words. If you were simply using the dictionary index of each word (e.g., assigning each word a unique id) then words close together in the dictionary would be closer to each other which is not ideal for training. The classic example of how these vectors encode relationships between words is using the vector for king subtracting the vector for man and adding the vector for woman and the closest vector in the feature space should be the vector for queen which means that the relationships between these words has been encoded in the vectors that represent them.

2.4 Data pre-processing

Pre-processing the data is vital to improve the accuracy of the model as machine learning algorithms are often thought of as garbage in garbage out, meaning that if the input data is bad, it is irrelevant how good the code because the output will also be meaningless. To prevent this data must be presented to the machine in a format it can understand and extract the most information from (without having anything unnecessary), this often requires steps such as: removing punctuation, removing URLs, removing stop words, noise removal, lower casing, tokenization, stemming, and lemmatization.

Noise removal and tokenisation are about identifying what the proper tokens are by breaking down the text and removing abnormal characters. Noise removal depends on what information you want the model to process in the text, for example, information such as numbers or special characters, however, in tweets the hashtag symbol could be important. Usually special characters such as tags, numbers, non-ascii characters, punctuation, and emojis could be considered noise. Tokenisation on the other hand is breaking down the text into smaller chunks such as words or sentences which can be processed. This is often done by just splitting the text on spaces or if you want the tokens to be sentences then a full stop, exclamation point, or question mark.

Removing information that is not necessary (such as punctuation, URLs, and case) will help the model not overfit to this information that may not be present in the testing set or does not relate to the objective of the model. Sometimes this information is helpful (the difference between “go away” and “GO AWAY!”) as it can help convey tone so a note can be made, for example tagging the second phrase with <all caps>. However, in the general case there is no difference between “go away” and “Go away.” other than one having proper grammar, so the casing and punctuation can be removed.

Stop words are common words that do not add any meaningful information to the text such as “I”, “was”, “am”, “this” etc. and can be removed without losing the keywords of the sentence. For example, “The cat was in the house” without stop words could become “cat house” which would save memory over a large corpus and mean that the computer is able to pick up on the key words in a sentence. This could be useful in contexts such as comparing similarity between two texts as text with a lot of words in common with another text is likely to be similar, however, the context/ relationships between the words are often lost. For example, in the above case the word “in” is lost which provides the key information about the relationship between the cat and the house. There are different lists of stop words which can be used in different contexts, and it is often a compromise between speed/ memory size and detail.

Information such as grammar or the form a word takes is often not useful to the computer for figuring out the keywords of the text and are thus superfluous. This is the intuition behind stemming and lemmatization which involve putting words into their base form. Stemming refers to taking the root “stem” of the word such as “run” for “running”, “runner”, and “run”, etc. as they all involve the same concept. Lemmatization on the other hand is like stemming in that it tries to find the base form of a word, however, it does not just use the characters themselves. It often has a lookup table which associates words, for example, “was” becomes “be” and “mice” becomes “mouse” and “Lemmatization is typically seen as much more informative than simple stemming[2]”.

All the processes explained above are not necessary and models can be trained on unprocessed text, however, this will take up more memory and may lead to a lower quality output. While text cleaning is still important, in sentiment analysis you often want as much

[2] (Beri, 2020)

information as possible and the relationships between words is key (not just the keywords) so some of the above steps such as stop word removal, stemming, and lemmatization are skipped while others such as noise removal and removing unnecessary information is still vital.

2.5 Sentiment analysis

Sentiment analysis (or opinion mining) is used to determine emotion, or subjective qualities from text using natural language processing and machine learning. Some useful implementations of sentiment analysis are separating spam emails from real ones or companies trying to understand how users feel about a particular feature. There are multiple approaches that can be taken and the two most popular will be discussed in this paper. The first is knowledge-based, which involves having a dictionary of words or attributes associated with each class of sentiment (real or fake, positive or negative, spam or not spam) and classifying the text based on what appears. This is a relatively simple approach but requires the conditions of each class to be hard coded and creating an exhaustive list can be time consuming and often impossible, but even after one is created it is possible that the text cannot even be classified if nothing from either (or something from both) list comes up. There is also no context behind these words, for example, "an article about cancer may talk about "unstoppable growth" which, in this context, is bad. But in the context of an article about start-up businesses, "unstoppable growth" is considered good.[3]" One way to factor in the context would be to use conjunctions such as "and" and "however" which relates the adjectives to the subject then use the subject to see how the adjective should be interpreted. For example, if there was a conjunction such as "deterioration of a patient <u>and</u> unstoppable growth[3]" the algorithm can interpret that as a bad thing as deterioration would have negative associations in the dictionary.

Classifying a challenging example:

"I usually love shopping, looking at phones is so fun, but this mall was too loud.

My brother enjoyed his milkshake. They're not really my thing."

Identifying who holds what opinion and what they are talking about is vital for sentiment analysis. In the first sentence "love" and "fun" are related to "shopping", however, the sentiment is not positive as the "but" negates the previous positive sentiment. If you were recommending activities for this person, you might still recommend shopping, however, overall, the sentiment for the mall is negative. In the second sentence the person who "loves" the milkshake is the brother, and this negation handling is more difficult for the computer to parse. If the program only looked at the individual words "love, fun, enjoyed, too loud" it might be assigned positive sentiment as a majority of the words are positive, however, the surrounding context is important.

[3] (Basques, 2020)

The second, more modern, approach is the statistical machine learning approach which generates the co-occurrence between two words (how likely they are to appear near each other) by sampling a lot of training data then using this information to predict how likely a given sentiment is. Generally, features extraction is used which is when a model is trained to learn what features are associated with each classification using training data then extracting those features from unseen testing data to classify it. This is often used in conjunction with a dictionary that associates certain words with a sentiment, however, the model will learn what other words appear in similar contexts and what sentiments to associate these new words with. It is important to perform, the aforementioned, text cleaning (such as stop word removal) and provide good data so the model is able to learn useful co-occurrences, however, this method requires a lot of training data and time which may not always be possible. In addition, machine learning struggles with handling sarcasm, tone, and comparisons. These can be considered, and advancements are being made, but currently computers will find these types of challenges more difficult than humans.

2.6 Evaluation

The evaluation of a model is how well it performs, one naïve approach for a classification model is simply the accuracy which is given as the number of correct classifications divided by the total number of classifications. An example of why this could be flawed is given a classification problem where there are 90 Positives and 10 Negatives. A model that simply classifies everything as positive will have a 90% accuracy which does not seem like an accurate judge of how sophisticated this model is (as given a problem with 100 Negatives the model will get 0). The purpose of an evaluation metric is to get to get an estimate of how well the model will perform given any random data and although it is impossible to get a perfect judge the aim is to get as close as possible.

A more sophisticated approach would be the F_1 score which is calculated using the precision and recall. The precision is the ratio of True Positives (TP) to total Positive classifications (TP/(TP+FP)) and the recall is the ratio of True Positives to the total number of Positives (TP/(TP+FN)) where FP is False Positives and FN is False Negatives. The F1 score considers the ratio between the number of Positives and Negatives so if there are fewer Negatives, then misclassifying one impacts the score by more.

$$F1 = \frac{TP}{TP + \frac{1}{2}(FP + FN)} = 2\,\frac{precision * recall}{precision + recall}$$

Figure 1, F1 score formula

3 Applying the research to the problem

3.1 Problem

The task set by the competition is separating tweets “announcing a disaster” from tweets that are not about a real disaster. In essence a model must be created that can discern between real and fake disaster tweets using the information given, however, the aim of this model is to be a general binary classifier (the code will not refer to disaster tweets, so give different training data it could classify, for example, between spam or not spam). The dataset provided contains five columns for the training set (id, keyword, location, text, target) and the testing set had the same columns, however, the models aim was to predict the target correctly, so this column is of course not included. The input of the model should be two csv files, one for training and one for testing, and the output should be two columns, every id from the testing set and the predicted target for that id by the model. The model will be assessed and evaluated using only the accuracy of the predicted target values, in other words, the metric used to judge will be the number of correct predictions divided by the total number of entries as no other data is available from Kaggle. The most useful information for the model will be the text and its relationship between the target (which will either be 0 or 1) so this is what will be used for training and predicting.

3.2 Preparation

Before writing any software the environment that will be used has to be set up. The language used in this project was python as it has by far the most support for NLP with libraries such the natural language toolkit, which provides support for and packages which could be useful. Python is also the most widely used language for machine learning in general as it has libraries such as Pytorch and TensorFlow which streamline the process of training and testing the model. The latest version of python (which was 3.9.12) and the appropriate libraries were installed then the test and training datasets were downloaded from the Kaggle website.

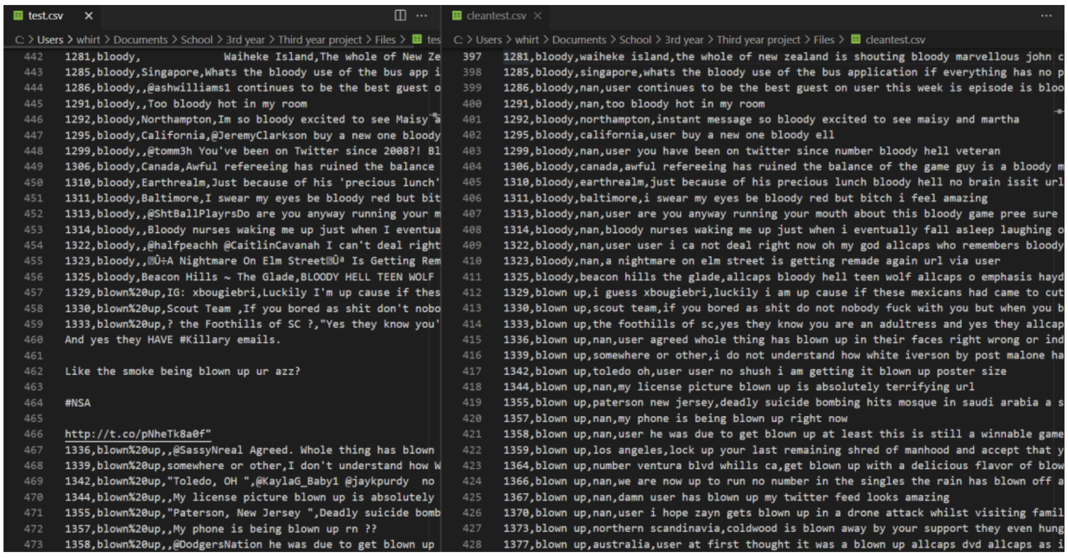
3.3 Cleaning datasets

The dataset provided to train and test the model contained phrases and terms that would not be useful for classifying the data. This included URLs which often linked to images that were attached to the tweets and tags such as @username where the username will not give any meaningful information to the model. The text also contained other irregularities that were necessary to remove, for example there were non-ascii characters, emojis, extra whitespace, and in the keywords column spaces were replaced with %20. As mentioned in the research it is beneficial to have as many words in their base form as possible so to clean the text contractions (such as can’t, won’t, and don’t), abbreviations and slang (such as ttyl, afk, and i.e.), and symbols and emoticons (such as £, >:(, and $) should be expanded or replaced and words should be put into lowercase.

The library used for text cleaning was ekphrasis[4] which is a “Collection of lightweight text tools” which use text from social media (including twitter) which makes it an ideal library for this situation. It includes features such as a social tokenizer which can recognise censored words, emoticons, different date formats (Feb 18th, 10/17/94, November 24th-2016 etc.), and currencies ($220M, £2B, €65.000 etc.), a spell corrector (e.g., to change korrect to correct) and a word segmenter. The segmenter can break down strings such as smallandinsignificant into their individual words (for the previous example small and insignificant). This is particularly useful for twitter as it can tag and then break down hashtags such as #johndoe into <hashtag> john doe </hashtag>. Other useful tags include allcaps, elongated (e.g., this suckssssss becomes this sucks <elongated>), and censored (f*** becomes f censored) which can all be useful for identifying if a tweet is about an authentic disaster as someone breaking news about a real situation might be less likely to swear or type in all caps.

Through a combination of regular expressions and implementing the preprocessor from the ekphrasis library, a program that takes the original train.csv and test.csv files and produces cleantrain.csv and cleantest.csv was developed. This program was able to fulfil all the above conditions, such as remove whitespace, and symbols thus making the text easier to embed and useful for the model which will be developed.

As shown in the figure below for ID: 1323 the non-ascii characters were removed and after the text had been cleaned “url” and “via user” have been used to tag where the original tweet contained @user and contained a URL.


test.csv
C: > Users > whirt > Documents > School > 3rd year > Third year project > Files > tes

```
1281,bloody,            Waiheke Island,The whole of New Ze
1285,bloody,Singapore,Whats the bloody use of the bus app i
1286,bloody,,@ashwilliams1 continues to be the best guest o
1291,bloody,,Too bloody hot in my room
1292,bloody,Northampton,Im so bloody excited to see Maisy a
1295,bloody,California,@JeremyClarkson buy a new one bloody
1299,bloody,,@tomm3h You've been on Twitter since 2008?! Bl
1306,bloody,Canada,Awful refereeing has ruined the balance
1310,bloody,Earthrealm,Just because of his 'precious lunch'
1311,bloody,Baltimore,I swear my eyes be bloody red but bit
1313,bloody,,@ShtBallPlayrsDo are you anyway running your m
1314,bloody,,Bloody nurses waking me up just when I eventua
1322,bloody,,@halfpeachh @CaitlinCavanah I can't deal right
1323,bloody,,�Û÷A Nightmare On Elm Street�Ûª Is Getting Rem
1325,bloody,Beacon Hills ~ The Glade,BLOODY HELL TEEN WOLF
1329,blown%20up,IG: xbougiebri,Luckily I'm up cause if thes
1330,blown%20up,Scout Team ,If you bored as shit don't nobo
1333,blown%20up,? the Foothills of SC ?,"Yes they know you'
And yes they HAVE #Killary emails.

Like the smoke being blown up ur azz?

#NSA

http://t.co/pNheTk8a0f"
1336,blown%20up,,@SassyNreal Agreed. Whole thing has blown
1339,blown%20up,somewhere or other,I don't understand how W
1342,blown%20up,"Toledo, OH ",@KaylaG_Baby1 @jaykpurdy  no
1344,blown%20up,,My license picture blown up is absolutely
1355,blown%20up,"Paterson, New Jersey ",Deadly suicide bomb
1357,blown%20up,,My phone is being blown up rn ??
1358,blown%20up,,@DodgersNation he was due to get blown up
```

cleantest.csv
C: > Users > whirt > Documents > School > 3rd year > Third year project > Files > cleantest.csv

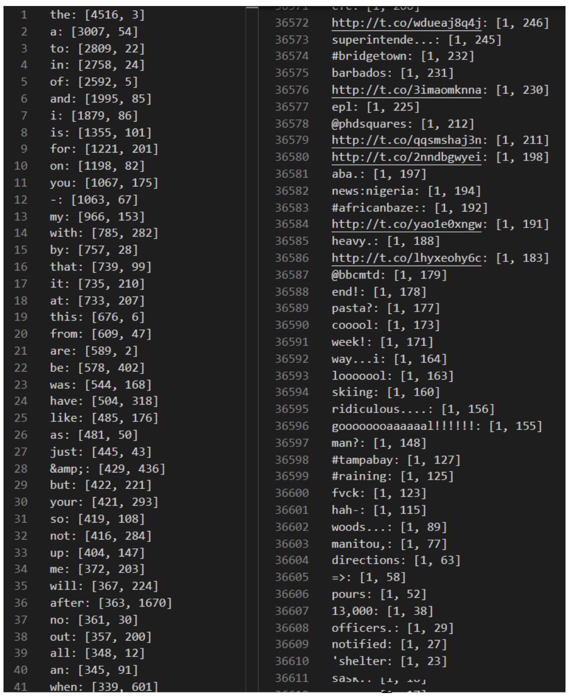
```
1281,bloody,waiheke island,the whole of new zealand is shouting bloody marvellous john c
1285,bloody,singapore,whats the bloody use of the bus application if everything has no p
1286,bloody,nan,user continues to be the best guest on user this week is episode is bloo
1291,bloody,nan,too bloody hot in my room
1292,bloody,northampton,instant message so bloody excited to see maisy and martha
1295,bloody,california,user buy a new one bloody ell
1299,bloody,nan,user you have been on twitter since number bloody hell veteran
1306,bloody,canada,awful refereeing has ruined the balance of the game guy is a bloody m
1310,bloody,earthrealm,just because of his precious lunch bloody hell no brain issit url
1311,bloody,baltimore,i swear my eyes be bloody red but bitch i feel amazing
1313,bloody,nan,user are you anyway running your mouth about this bloody game pree sure
1314,bloody,nan,bloody nurses waking me up just when i eventually fall asleep laughing o
1322,bloody,nan,user user i ca not deal right now oh my god allcaps who remembers bloody
1323,bloody,nan,a nightmare on elm street is getting remade again url via user
1325,bloody,beacon hills the glade,allcaps bloody hell teen wolf allcaps o emphasis hayd
1329,blown up,i guess xbougiebri,luckily i am up cause if these mexicans had came to cut
1330,blown up,scout team,if you bored as shit do not nobody fuck with you but when you b
1333,blown up,the foothills of sc,yes they know you are an adultress and yes they allcap
1336,blown up,nan,user agreed whole thing has blown up in their faces right wrong or ind
1339,blown up,somewhere or other,i do not understand how white iverson by post malone ha
1342,blown up,toledo oh,user user no shush i am getting it blown up poster size
1344,blown up,nan,my license picture blown up is absolutely terrifying url
1355,blown up,paterson new jersey,deadly suicide bombing hits mosque in saudi arabia a s
1357,blown up,nan,my phone is being blown up right now
1358,blown up,nan,user he was due to get blown up at least this is still a winnable game
1359,blown up,los angeles,lock up your last remaining shred of manhood and accept that y
1364,blown up,number ventura blvd whills ca,get blown up with a delicious flavor of blow
1366,blown up,nan,we are now up to run no number in the singles the rain has blown off a
1367,blown up,nan,damn user has blown up my twitter feed looks amazing
1370,blown up,nan,user i hope zayn gets blown up in a drone attack whilst visiting famil
1373,blown up,northern scandinavia,coldwood is blown away by your support they even hung
1377,blown up,australia,user at first thought it was a blown up allcaps dvd allcaps as i
```

Figure 2, Before and after cleaning the text

[4] (Christos Baziotis, 2017)

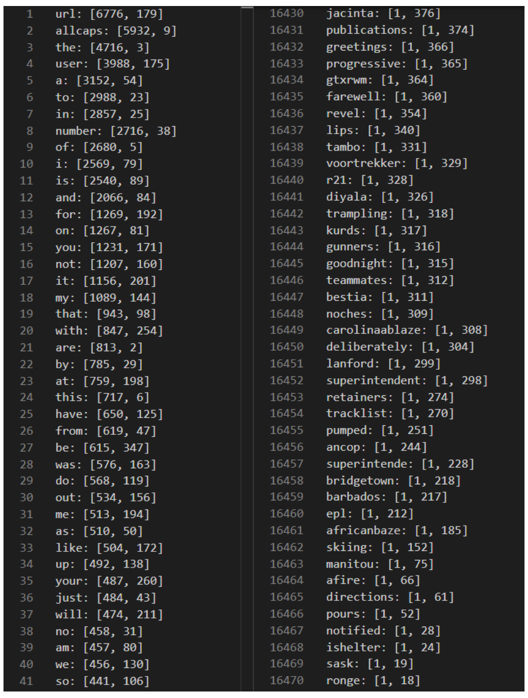
3.4 Creating the model

The first step was creating a list of words to see what the most and least common words were, the output of which is shown below. The process of coding the model was begun before cleaning the text, however, after cleaning the text the output of the same program can be seen on the right. The output takes the form “word: [frequency, unique_id]” with words like “the” and “a” predictably having the highest frequency and more obscure words such as trampling or tambo only appear once. After cleaning there are 16470 unique words, however, only 7731 appear more than once and 3113 appear more than five times.



```
1      the: [4516, 3]
2      a: [3007, 54]
3      to: [2809, 22]
4      in: [2758, 24]
5      of: [2592, 5]
6      and: [1995, 85]
7      i: [1879, 86]
8      is: [1355, 101]
9      for: [1221, 201]
10     on: [1198, 82]
11     you: [1067, 175]
12     -: [1063, 67]
13     my: [966, 153]
14     with: [785, 282]
15     by: [757, 28]
16     that: [739, 99]
17     it: [735, 210]
18     at: [733, 207]
19     this: [676, 6]
20     from: [609, 47]
21     are: [589, 2]
22     be: [578, 402]
23     was: [544, 168]
24     have: [504, 318]
25     like: [485, 176]
26     as: [481, 50]
27     just: [445, 43]
28     &amp;: [429, 436]
29     but: [422, 221]
30     your: [421, 293]
31     so: [419, 108]
32     not: [416, 284]
33     up: [404, 147]
34     me: [372, 203]
35     will: [367, 224]
36     after: [363, 1670]
37     no: [361, 30]
38     out: [357, 200]
39     all: [348, 12]
40     an: [345, 91]
41     when: [339, 601]
```

```
36572  http://t.co/wdueaj8q4j: [1, 246]
36573  superintende...: [1, 245]
36574  #bridgetown: [1, 232]
36575  barbados: [1, 231]
36576  http://t.co/3imaomknna: [1, 230]
36577  epl: [1, 225]
36578  @phdsquares: [1, 212]
36579  http://t.co/qqsmshaj3n: [1, 211]
36580  http://t.co/2nndbgwyei: [1, 198]
36581  aba.: [1, 197]
36582  news:nigeria: [1, 194]
36583  #africanbaze:: [1, 192]
36584  http://t.co/yao1e0xngw: [1, 191]
36585  heavy.: [1, 188]
36586  http://t.co/lhyxeohy6c: [1, 183]
36587  @bbcmtd: [1, 179]
36588  end!: [1, 178]
36589  pasta?: [1, 177]
36590  cooool: [1, 173]
36591  week!: [1, 171]
36592  way...i: [1, 164]
36593  looooool: [1, 163]
36594  skiing: [1, 160]
36595  ridiculous....: [1, 156]
36596  gooooooaaaaaal!!!!!!: [1, 155]
36597  man?: [1, 148]
36598  #tampabay: [1, 127]
36599  #raining: [1, 125]
36600  fvck: [1, 123]
36601  hah-: [1, 115]
36602  woods...: [1, 89]
36603  manitou,: [1, 77]
36604  directions: [1, 63]
36605  =>: [1, 58]
36606  pours: [1, 52]
36607  13,000: [1, 38]
36608  officers.: [1, 29]
36609  notified: [1, 27]
36610  'shelter: [1, 23]
36611  sask.: [illegible]
```



```
1      url: [6776, 179]
2      allcaps: [5932, 9]
3      the: [4716, 3]
4      user: [3988, 175]
5      a: [3152, 54]
6      to: [2988, 23]
7      in: [2857, 25]
8      number: [2716, 38]
9      of: [2680, 5]
10     i: [2569, 79]
11     is: [2540, 89]
12     and: [2066, 84]
13     for: [1269, 192]
14     on: [1267, 81]
15     you: [1231, 171]
16     not: [1207, 160]
17     it: [1156, 201]
18     my: [1089, 144]
19     that: [943, 98]
20     with: [847, 254]
21     are: [813, 2]
22     by: [785, 29]
23     at: [759, 198]
24     this: [717, 6]
25     have: [650, 125]
26     from: [619, 47]
27     be: [615, 347]
28     was: [576, 163]
29     do: [568, 119]
30     out: [534, 156]
31     me: [513, 194]
32     as: [510, 50]
33     like: [504, 172]
34     up: [492, 138]
35     your: [487, 260]
36     just: [484, 43]
37     will: [474, 211]
38     no: [458, 31]
39     am: [457, 80]
40     we: [456, 130]
41     so: [441, 106]
```

```
16430  jacinta: [1, 376]
16431  publications: [1, 374]
16432  greetings: [1, 366]
16433  progressive: [1, 365]
16434  gtxrwm: [1, 364]
16435  farewell: [1, 360]
16436  revel: [1, 354]
16437  lips: [1, 340]
16438  tambo: [1, 331]
16439  voortrekker: [1, 329]
16440  r21: [1, 328]
16441  diyala: [1, 326]
16442  trampling: [1, 318]
16443  kurds: [1, 317]
16444  gunners: [1, 316]
16445  goodnight: [1, 315]
16446  teammates: [1, 312]
16447  bestia: [1, 311]
16448  noches: [1, 309]
16449  carolinaablaze: [1, 308]
16450  deliberately: [1, 304]
16451  lanford: [1, 299]
16452  superintendent: [1, 298]
16453  retainers: [1, 274]
16454  tracklist: [1, 270]
16455  pumped: [1, 251]
16456  ancop: [1, 244]
16457  superintende: [1, 228]
16458  bridgetown: [1, 218]
16459  barbados: [1, 217]
16460  epl: [1, 212]
16461  africanbaze: [1, 185]
16462  skiing: [1, 152]
16463  manitou: [1, 75]
16464  afire: [1, 66]
16465  directions: [1, 61]
16466  pours: [1, 52]
16467  notified: [1, 28]
16468  ishelter: [1, 24]
16469  sask: [1, 19]
16470  ronge: [1, 18]
```

Figure 3, The most and least common words in the testing and training corpus

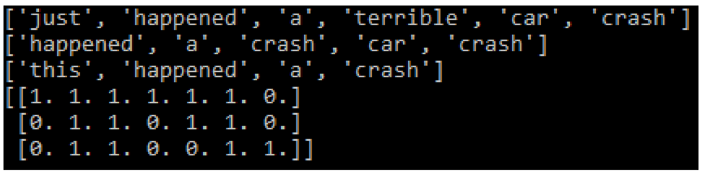
This process of giving each word a unique id and finding out how many words there are in the text was done to make turning the text from the tweets into a one hot encoded vector easier as each word has a unique id. Although one hot vectors have issues (as mentioned above) they are the fastest to implement and provides enough information for the first attempt at the model. First the text is turned into a sequence of numbers (representing the unique ids of the words in the text) then that is turned into a one hot encoded version of the text.

For example, given the table:

WORD	ID
left	1
cat	2
far	3
the	4

The phrase “the cat left” would have each word replaced by its ID giving: [4,2,1] then a vector of 0s with 1s at positions 4, 2, and 1 giving: [1,1,0,1]

Below is an example using the encoding from the model where the words represented by the position in the vector are: [‘just’, ‘happened’, ‘a’, ‘terrible’, ‘car’, ‘crash’, ‘this’], the text is shown at the top and its one hot encoding is shown below.



```
['just', 'happened', 'a', 'terrible', 'car', 'crash']
['happened', 'a', 'crash', 'car', 'crash']
['this', 'happened', 'a', 'crash']
[[1. 1. 1. 1. 1. 1. 0.]
 [0. 1. 1. 0. 1. 1. 0.]
 [0. 1. 1. 0. 0. 1. 1.]]
```

Figure 4, One hot encoding of three phrases

Adapted from an approach by Keldenich[5] the text can be encoded into a format that can be understood by the computer and the model can be created. The shape of the model can be seen in the figure below and has an input layer, two hidden layers followed by an output layer. The first layer, the input layer, takes the data as a list of size 16470 which is the number of unique words in the dataset then the data is passed through and shrunk down to 16 through a ReLU layer then passes through another ReLU layer before finally passing to the output layer which uses a sigmoid function to come up with the probability of which class the output belongs to. When predicting this is rounded so a prediction of 0.64 is given the target 1.

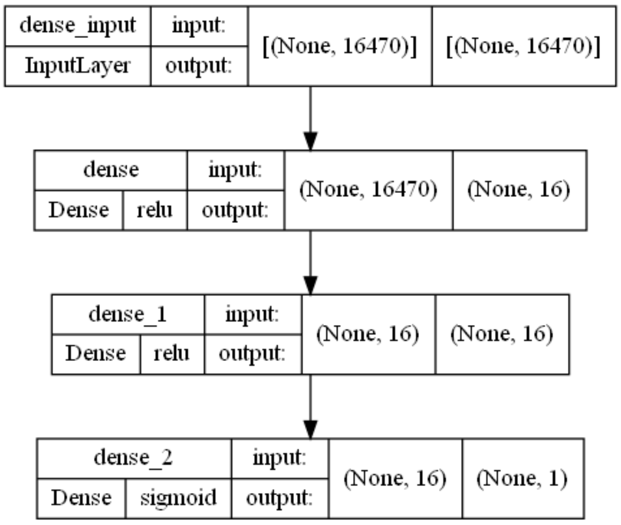


Figure 5, Shape of the model

As shown above all layers used are dense, also known as a fully connected layer, meaning each neuron receives input from all the neurons in the layer before and these are the “most commonly used layer in models[6]”. The units or shape of the layer is the input and output of the layer, for example, for the second layer it takes the input from the input layer and reduces the output to a list of size 16 to pass onto the next layer. The output of a given layer is calculated by “activation(dot(input, kernel) + bias)” where the bias (not used for this model) is a vector, the activation function is specified, and it is applied to the dot product of the kernel (the weights matrix) and the input tensor.

[5] (Keldenich, 2021)
[6] (Sharma, 2020)

The activation functions used are the rectified linear unit, or ReLU, function and the sigmoid function. The ReLU function will either output 0 if the input is negative or simply return the input - f(x) = max(0, x). Some benefits of the ReLU layer is that it is less susceptible to the vanishing gradient problem which effects function such a tanh and sigmoid as the value of x increases (as their gradients decrease as x increases) and it is computationally efficient as the function is so simple. The sigmoid function on the other hand transforms the data into the range between 0 and 1 using the equation below. This is used for the output layer as the value being between 0 and 1 also acts as the probability of the input belonging to each class. For example, if the sigmoid function returned 0.75 then there is a 75% chance that the input belongs to the class 1. The advantage of the sigmoid is that it also prevents the value from exploding since unlike the ReLU unit (where x can increase infinitely) the value is bound between 0 and 1.

$$sigmoid(x) = \frac{1}{1+e^{-x}}$$

Figure 6, Sigmoid function

Once the model has been trained and the weights in each layer have been tuned the testing data is prepared by using the same one hot encoding algorithm and fed into the model which predicts the class using the text from the tweet. This first attempt at the model gives an accuracy of 0.79497, however, with some changes to how the data was cleaned and the changing the number of epochs a final accuracy of 0.79895 was reached (all submissions to Kaggle can be seen in the appendix). The changes in accuracy and loss when the number of epochs changes is shown in the figure below. The accuracy on the Kaggle website is consistently higher than the validation accuracy reached on splits of the training set which is possibly due to having more training data (as none of it needs to be saved for validation) and it is also possible that the testing set is easier to classify than the training set.

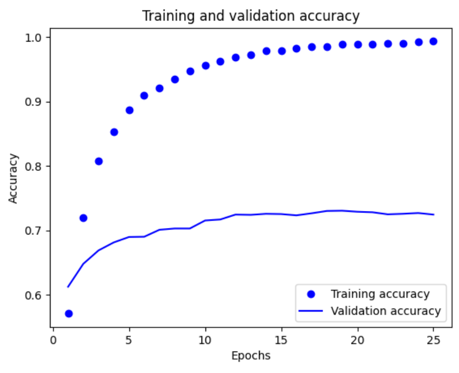


Figure 7, Accuracy of the Model

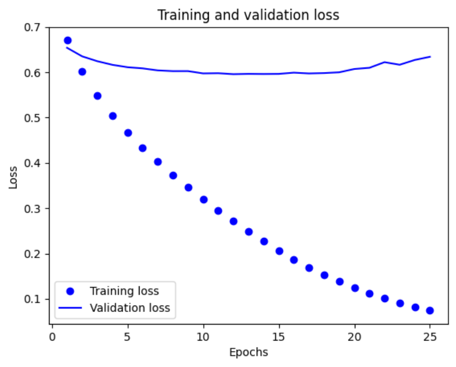


Figure 8, Loss of the Model

3.5 Tuning the model

Following the completion of the code for the model and submitting the output to the Kaggle competition, coding for tuning the model was begun. The library used was called Keras Tuner[7] which is used for tuning the hyperparameters in TensorFlow projects. As Keras was used for the model it was ideal to use Keras Tuner to find the optimal hyperparameters for the program, however, these parameters are not the only variables that impact the performance of the model. The optimizer and the loss function will also impact how the model updates so in total the variables that were tested are: the loss function, the optimizer, the number of epochs, the learning rate, and the shape of the first ReLU layer in the model.

The value for the learning rate was changed between 0.01, 0.001, and 0.0001, the value for the shape of the first layer was varied from 32 to 512 in increments of 32 and the model ran for a maximum of 50 epochs with the best value of each being selected. This was tested for two different optimizers and two different loss functions, namely Adam, and Root Mean Square Propagation as the optimizers and Binary Cross-Entropy, and Root Mean Square Error as the loss functions. The loss function is the measure of how well your model is doing and this is evaluated after every iteration with a higher loss meaning that your predicted values are more different to the target values.

This loss function is what your model is aiming to reduce so changing the loss function could have a large impact on the performance of the model. Mean Square Error is the most used loss function and is calculated using the equation below which is the sum of the squares of the difference between the target and predicted values divided by the number of predicted values. This is what the model was originally using for the first submissions as this was before the research into loss functions was conducted.

$$Mean\ Square\ Error = \frac{1}{n}\sum_{i=1}^{n}(Yi - \hat{Y}i)^2$$

Figure 9, MSE formula

The Binary Cross-Entropy (also known as log loss) on the other hand is the most used loss function for binary classification problems (which this problem falls under) and is calculated using the average error of the probability distribution for the prediction with the equation below. Both loss functions were tested; however, it was expected that the Binary Cross-Entropy would perform better as it is well suited to this specific task.

[7] https://www.tensorflow.org/tutorials/keras/keras_tuner

$$Log\ loss = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Figure 10, Binary Cross-entropy formula

To explain Binary Cross-Entropy an example of potential for this project can be seen below. Since, for this project, the classification is binary, and the output of the sigmoid function is a probability between 1 and 0 we can expect the output of the model will look like the table below.

ID	Actual Class (y_i)	Prediction $p(y_i)$
1	1	0.76
2	1	0.32
3	0	0.23
4	1	0.93
5	0	0.53

$$\begin{gathered}-\frac{1}{5}\cdot(\\ (1 \cdot log(0.76) + (1-1)\cdot log(0.24))+\\ (1 \cdot log(0.32) + (1-1)\cdot log(0.68))+\\ (0 \cdot log(0.23) + (1-0)\cdot log(0.77))+\\ (1 \cdot log(0.93) + (1-1)\cdot log(0.07))+\\ (0 \cdot log(0.53) + (1-0)\cdot log(0.47)))\end{gathered}$$

Figure 11, Example data and the calculation for its Binary Cross-Entropy

The equation on the right is how you would calculate the binary cross entropy of the above predictions. As you can see, when y_i is 1 then the righthand half of the equation cancels out and when it is 0 the lefthand side cancels out which means that the Binary Cross-Entropy equation can be cancelled down to -1/5(log(0.76)+log(0.32)+log(0.77)+log(0.93)+log(0.47)) and means the log loss of these predictions is about 0.217. The reason that the log function is used is so that when the prediction is further from the correct label the cost becomes higher (it grows exponentially as the log function has an asymptote at 0). This function breaks when the model guesses completely incorrectly as log(0) is undefined, however, this can be fixed by simply defining log(0) equal to the negative max for the datatype you are using to store your numbers. This information was based on Saxena's explanation of log loss.[8]

The optimizer is the algorithm to update the weights in each layer to minimize the loss function. They have parameters such as the learning rate, which is how much the weights are updated by for each iteration, and epsilon, which is a "small constant for numerical stability[9]" (so that the divisor in the update function does not approach 0 meaning the weights explode). The two optimizers researched were Adam and Root Mean Square Propagation (RSMprop) which are both first order differential algorithms that use gradient descent to tune the weights for each layer of the model. Gradient descent aims to change the weights of the model to minimize the loss through taking the differential of the line then updating the weight in the direction of the gradient. This can be visualised with the figure below (which also shows how

[8] (Saxena, 2021)
[9] https://keras.io/api/optimizers/adam/

the learning rate affects the change in weights) where the optimizer is changing the weight with each iteration.

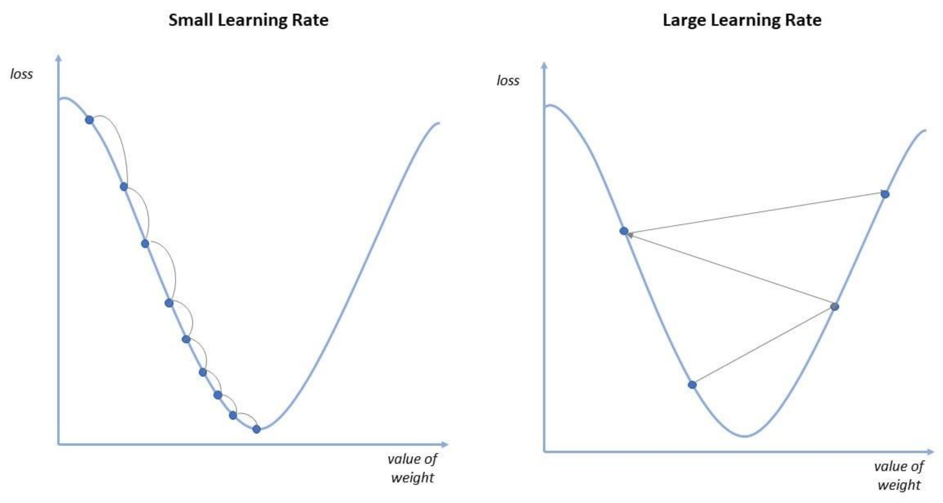


Figure 12, Model loss with different weights
Found at: https://1.cms.s81c.com/sites/default/files/2020-10-27/Learning%20Rate.jpg

RMSprop can be fundamentally explained with two steps: “1. Maintain a moving (discounted) average of the square of gradients. 2. Divide the gradient by the root of this average”[10]

Given parameters in different directions b and w

For each iteration t compute the derivatives dw, db on the current batch:

Calculate Exponentially Weighted Average of squares for dw and db:

$$\text{Weighted average of dw } (S_{t,dw}) = \beta \cdot S_{t-1,dw} + (1-\beta) \cdot dw^2$$

$$\text{Weighted average of db } (S_{t,db}) = \beta \cdot S_{t-1,db} + (1-\beta) \cdot db^2$$

Apply Update Functions:

$$w_t = w_{t-1} - \alpha \cdot \frac{dw}{\sqrt{S_{t,dw}} + \epsilon} \qquad b_t = b_{t-1} - \alpha \cdot \frac{db}{\sqrt{S_{t,db}} + \epsilon}$$

Where α is the learing rate and β is the ratio of current value to previous values

Figure 13, Equation for RMSprop

The issue with other gradient descent algorithms is that they can get stuck mostly moving in one specific direction (in the example above with two parameters b and w it might get stuck moving in the b direction and w could optimize slowly). RMSprop fixes this as you can increase the learning rate (α) which will increase the learning speed without the weights that

[10] https://keras.io/api/optimizers/rmsprop/

are closer to the minima diverging because the update function is divided by the moving average of the squares. The first term of the update function is β (which is between 0 and 1) multiplied by the previous value of S_{dw} or S_{db} which controls how much the previous value of the average effects the average (lower β means the average is spread across more previous values and the previous value has less weight). Finally, as mentioned before, ϵ is in the divisor of the update functions to prevent the average being too close to 0 when the weighted average is small (it is usually a very small number). The reason that α can be increased without the correct weights diverging can be understood because if the weights are close to the minima in the b direction, then S_{db} will be large so you are dividing by a large number so b updates less while S_{dw} will be small so you are updating w by a larger number until both converge on the minima.[11]

By contrast Adaptive Moment Estimation (Adam) can be summarized as “a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.”[12] It is one of the more modern approaches to optimization when compared to standard gradient descent and is often considered to be one of the most powerful as it combines RMSprop with momentum and adding a correction term. It is for this reason it is one of the most widely used algorithms which is why it is important for this project to understand how it works.

$$\text{Given parameters in different directions b and w}$$
$$\text{Set variables for momentum (V) and Exponentially Weighted Average of squares (S) to 0}$$
$$V_{dw} = 0, S_{dw} = 0, V_{db} = 0, V_{db} = 0$$
$$\text{For each iteration t compute the derivatives dw, db on the current batch:}$$
$$V_{t,dw} = \beta_1 \cdot V_{t-1,dw} + (1-\beta_1)dw \qquad S_{t,dw} = \beta_2 \cdot S_{t-1,dw} + (1-\beta_2)dw$$
$$V_{t,db} = \beta_1 \cdot V_{t-1,db} + (1-\beta_1)db \qquad S_{t,db} = \beta_2 \cdot S_{t-1,db} + (1-\beta_2)db$$

$$V_{t,dw}^{corrected} = \frac{V_{t-1,dw}}{1-\beta_1^t} \qquad S_{t,dw}^{corrected} = \frac{S_{t-1,dw}}{1-\beta_2^t}$$
$$V_{t,db}^{corrected} = \frac{V_{t-1,db}}{1-\beta_1^t} \qquad S_{t,db}^{corrected} = \frac{S_{t-1,db}}{1-\beta_2^t}$$

$$\text{Apply Update Functions:}$$
$$w_t = w_{t-1} - \alpha \cdot \frac{V_{t,dw}^{corrected}}{\sqrt{S_{t,dw}^{corrected}} + \epsilon} \qquad b_t = b_{t-1} - \alpha \cdot \frac{V_{t,db}^{corrected}}{\sqrt{S_{t,db}^{corrected}} + \epsilon}$$

Figure 14, Equation for Adam

As you can see the weighted averages and update functions from RMSprop are still used, however, on top of that we also have the momentum which replaces the differential used to update in RMSprop and adds a new parameter β_2 which is used to store the average of past

[11] (DeepLearningAI, 2017)
[12] https://keras.io/api/optimizers/adam/

gradients (like how the average squares are stored). After the momentum and average weight are calculated, they are corrected to account for bias (denoted as $V^{corrected}$ and $S^{corrected}$ respectively) and these corrected values are used in the update function to calculate the new parameters for w and b. Naturally in the previous examples for RMSprop and Adam only two parameters on separate planes have been used for simplicity, however, in reality the model would be much more complex. When tuning the recommended values of ϵ (10^{-7}), β_1 (0.9), and β_2 (0.999) were used as tuning them does not have as much impact on the performance of the model, however, different values for α were tested.

After researching different loss functions and optimizers the code for tuning the model was written first by defining a hypermodel which is the model building function, then the objective (how the model should be evaluated – validation accuracy in this case) should be chosen along with the maximum number of epochs (50 was used in this case). The hypermodel must be changed for each optimizer and loss function which means that the tuning must be run 4 times as these cannot be changed in during the tuning. Once these have been defined then the models can be created and the optimal hyperparameters can be searched for using hyperband which is a variation of random search. It randomly samples several hyperparameters and runs them for a given number of iterations while calculating the validation accuracy. The lowest performing half are then discarded along with any that appears to be performing worse than it has been for the past five iterations (this is calculated using an early stop algorithm). This is repeated until only one model is left remaining which is run for the remaining number of epochs (if the max is 50 and it has only run for 30 epochs then it will continue to run even if it is regressing). This model is then saved, and the optimal number of epochs is recorded. One drawback of this method is that if there are many possible variations and some good configurations take a while to converge then they may be discarded early and conversely if there are fewer, then some bad configurations could be kept for longer than necessary as there are so few models.

The outcome of these trails can be seen in the appendix which also includes graphs for each of the best parameters against their accuracy and loss to find the best epoch for the best parameters. These graphs also highlight how the model performs in training vs the validation set which is produced using a split of the original training data. The validation set is an approximation for how the model will perform with unseen data as it is not used for training. As expected, the models reach a peak performance quite quickly then the accuracy and loss stop improving in the validation set as the number of epochs keep increasing which is due to overfitting. The optimal parameters for the model from this testing appear to be using an Adam optimizer with a learning rate of 0.001, the Binary Cross-Entropy loss function, an output shape of 96 from the first layer, and 5 epochs. This produced the highest accuracy at just over 75% on the validation set which means that the model correctly labels over 3 out of every 4 tweets. As expected, the Binary Cross-Entropy and Adam were superior to the MSE and RMSprop in this case, however, the difference was not as pronounced as expected.

Over the past 2 sections this project has explored from the encoding to the shape of the model and the activation function, followed by the loss functions and the optimizers. This deep understanding of how the model works and the testing and tuning of the model meant the final submission, after tuning all the hyper parameters, managed to achieve an accuracy of 0.80049. Top scoring submissions on Kaggle (that are not cheated to have 100% accuracy) do not above 85%. Thus, the final accuracy rating is an acceptable result and reaches the goals that were set at the beginning of the report.

4 Project management

4.1 Hazard analysis

Risk event	Probability	Impact Rating	Risk (P*I)	Prevention
Missing deadlines.	5	5	25	Keep motivation high and work on the project consistently.
Losing data on computer.	2	5	10	Keep backups of data on GitHub or external storage.
Getting sick.	4	1	4	Take time to recover and adjust schedule accordingly.
Underestimating project size.	3	3	9	Break down the project into smaller sections and work on them over time.
Underestimating project difficulty.	3	4	12	Move on from parts of the project that are too difficult or ask for help.

4.2 Tools

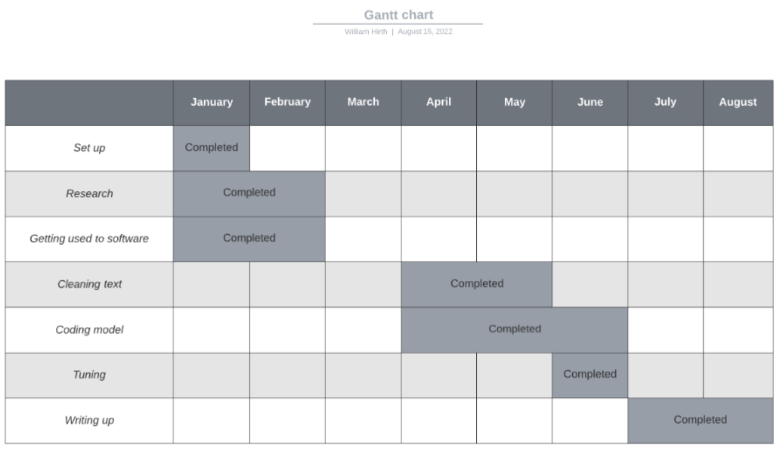
Tool	Purpose	Evaluation
VSCode	IDE	While it does not have as much functionality as a more powerful IDE its lightweight nature and customisability make it useful. Setting up a file and being able to quickly open the program and make changes was important for this project.
GitHub	Version control and backup	It was used to store my project files and changes on a remote repository and although it was not used very much it helped with the risk of work being lost or rolling back any changes that did not work. (A screenshot can be seen in the appendix)
Word	Word Processing	While it is a “what you see is what you get” word processor it is easy to use and effective. There are fewer customisability options than LaTeX and inputting formulae is more difficult, however, as I was working on a time limit it was efficient as I was more familiar with it.
Lucid	Making Charts	Used for making the gantt chart, it was easy and intuitive to use and the result was sufficient.

latex.codecogs.com	Writing equations	This online LaTeX compiler was used to write and screenshot equations that could not be produced in word. While LaTeX is useful for writing equations the overall user experience on word was easier and with easy-to-use compilers online equations were still able to be produced.

Overall, not many tools were used, however, this is not necessarily a negative. Work was mainly conducted from VSCode and the command prompt which meant that there was more control and understanding over what tools were being used and which files were necessary. One tool that should have been used is one for time management; setting tasks and reaching certain deadlines was difficult which meant that, as a result, the project suffered.

4.3 Time management

For this project time management was the biggest weakness; failure to breakdown the tasks and plan accordingly meant that the project was not completed in time and thus could not be submitted by the deadline. A combination of struggling to stay motivated and the project being left until late into the year which meant that once it begun there was not enough time left to extend the project beyond its original goals. Below is a gantt chart of how time was spent after January as work on the project did not begin until after the new year which was ultimately the reason the project was delayed.



Gantt chart

William Hirth | August 15, 2022

	January	February	March	April	May	June	July	August
Set up	Completed							
Research	Completed							
Getting used to software	Completed							
Cleaning text				Completed				
Coding model				Completed				
Tuning						Completed		
Writing up							Completed	

Figure 15, Gantt chart of time spent

5 Conclusion

5.1 Further work

Unfortunately, as time was not allocated properly for this project there were possibilities that were not explore but may have yielded higher accuracy. For example, only one type of embedding for the text was used which was one hot encoding while others such as GloVe was not tested. There was also only one approach in the application section which performed well, however, it could have been useful to research other libraries and approaches. The model also fails to consider features other than the text of the tweet, such as the keyword or location, which could have been useful when predicting. Finally, as the only metric Kaggle provided was accuracy no other performance metric could be used, for example, as mentioned in the research section accuracy is not always the best judge of a model.

5.2 Final conclusion

The aim of this project was to apply research conducted into natural language processing and sentiment analysis to a Kaggle competition and achieve a high accuracy. To this end this project has satisfied its goals as a model was created that was able to take the take the input data and produce an output that scored a 0.80049. Research was also conducted into sentiment analysis and the wider field of natural language processing while remaining within the scope and providing a solution to the problem outlined at the beginning of this paper.

Word count: 7282

6 Bibliography

Basques, R., 2020. *What is Sentiment Analysis and how is it used?.* [Online]
Available at: https://towardsdatascience.com/what-is-sentiment-analysis-and-how-is-it-used-217074887277
[Accessed 23 04 2022].

Beri, A., 2020. *Stemming vs Lemmatization.* [Online]
Available at: https://towardsdatascience.com/stemming-vs-lemmatization-2daddabcb221
[Accessed 29 02 2022].

Brownlee, J., 2019. *A Gentle Introduction to the Rectified Linear Unit (ReLU).* [Online]
Available at: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
[Accessed 11 05 2022].

Christos Baziotis, N. P. C. D., 2017. *DataStories at SemEval-2017 Task 4: Deep LSTM with Attention for Message-level and Topic-based Sentiment Analysis,* Vancouver, Canada: Association for Computational Linguistics.

Deepanshi, 2021. *Text Preprocessing in NLP with Python codes.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/06/text-preprocessing-in-nlp-with-python-codes/
[Accessed 12 02 2022].

DeepLearningAI, 2017. *Adam Optimization Algorithm (C2W2L08).* [Online]
Available at: https://www.youtube.com/watch?v=JXQT_vxqwIs
[Accessed 23 02 2022].

DeepLearningAI, 2017. *RMSProp (C2W2L07).* [Online]
Available at: https://www.youtube.com/watch?v=_e-LFe_igno
[Accessed 21 02 2022].

Glorot, X. a. B. A. a. B. Y., 2011. *Deep Sparse Rectifier Neural Networks,* s.l.: PMLR.

Keldenich, T., 2021. *A simple and efficient model for Binary Classification in NLP.* [Online]
Available at: https://inside-machinelearning.com/en/a-simple-and-efficient-model-for-binary-classification-in-nlp/
[Accessed 24 01 2022].

Ogneva, M., 2010. *How Companies Can Use Sentiment Analysis to Improve Their Business.* [Online]
Available at: https://mashable.com/archive/sentiment-analysis
[Accessed 23 05 2022].

O'Malley, T. a. B. E. a. L. J. a. C. F. a. J. H. a. I. L. a. o., 2019. *KerasTuner,* s.l.: KerasTuner.

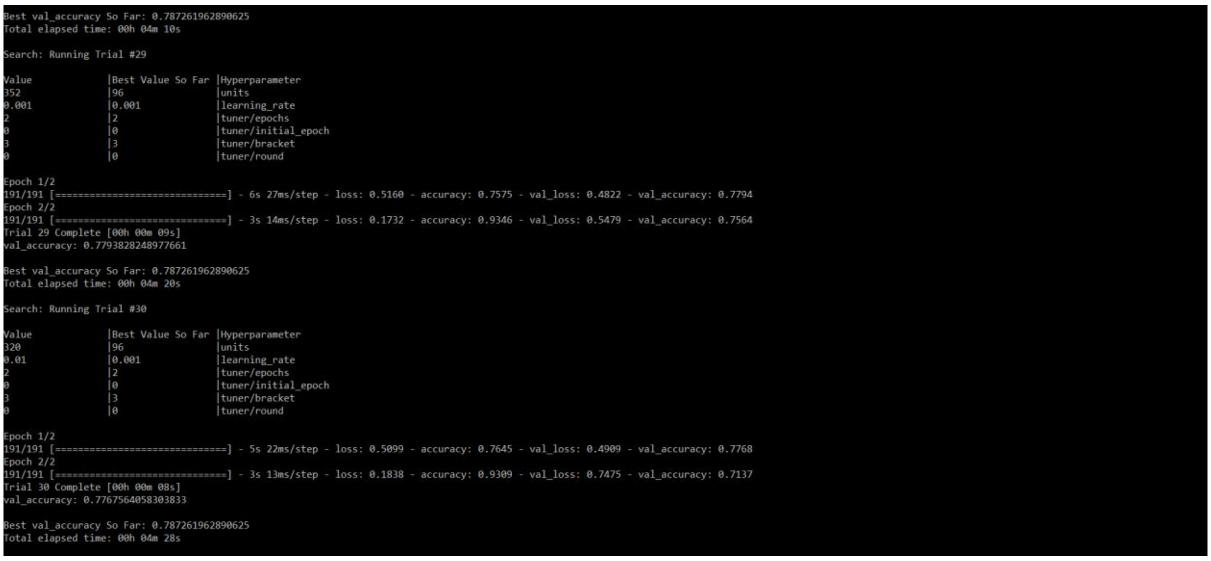
Saxena, S., 2021. *Binary Cross Entropy/Log Loss for Binary Classification.* [Online]
Available at: https://www.analyticsvidhya.com/blog/2021/03/binary-cross-entropy-log-loss-for-binary-classification/
[Accessed 14 02 2022].

Sharma, P., 2020. *Keras Dense Layer Explained for Beginners.* [Online]
Available at: https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/
[Accessed 11 06 2022].

7 Appendix

7.1 Graphs of the tuning data

Adam Binary Cross-Entropy



```
Best val_accuracy So Far: 0.787261962890625
Total elapsed time: 00h 04m 10s

Search: Running Trial #29

Value             |Best Value So Far |Hyperparameter
352               |96                |units
0.001             |0.001             |learning_rate
2                 |2                 |tuner/epochs
0                 |0                 |tuner/initial_epoch
3                 |3                 |tuner/bracket
0                 |0                 |tuner/round

Epoch 1/2
191/191 [==============================] - 6s 27ms/step - loss: 0.5160 - accuracy: 0.7575 - val_loss: 0.4822 - val_accuracy: 0.7794
Epoch 2/2
191/191 [==============================] - 3s 14ms/step - loss: 0.1732 - accuracy: 0.9346 - val_loss: 0.5479 - val_accuracy: 0.7564
Trial 29 Complete [00h 00m 09s]
val_accuracy: 0.7793828248977661

Best val_accuracy So Far: 0.787261962890625
Total elapsed time: 00h 04m 20s

Search: Running Trial #30

Value             |Best Value So Far |Hyperparameter
320               |96                |units
0.01              |0.001             |learning_rate
2                 |2                 |tuner/epochs
0                 |0                 |tuner/initial_epoch
3                 |3                 |tuner/bracket
0                 |0                 |tuner/round

Epoch 1/2
191/191 [==============================] - 5s 22ms/step - loss: 0.5099 - accuracy: 0.7645 - val_loss: 0.4909 - val_accuracy: 0.7768
Epoch 2/2
191/191 [==============================] - 3s 13ms/step - loss: 0.1838 - accuracy: 0.9309 - val_loss: 0.7475 - val_accuracy: 0.7137
Trial 30 Complete [00h 00m 08s]
val_accuracy: 0.7767564058303833

Best val_accuracy So Far: 0.787261962890625
Total elapsed time: 00h 04m 28s
```

Figure 16,Tuning to find the best parameters

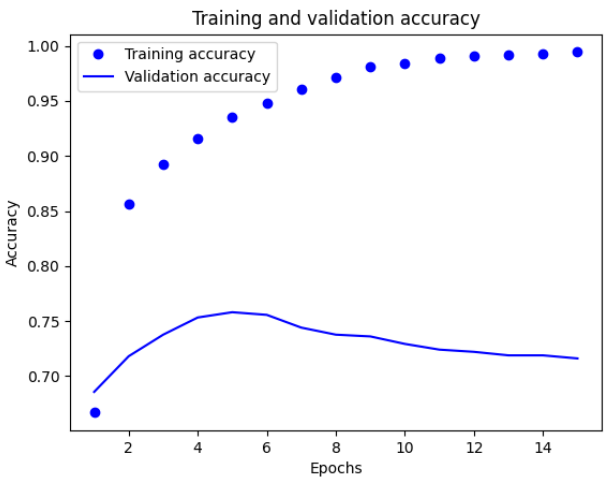


Figure 17, Accuracy of the best parameters

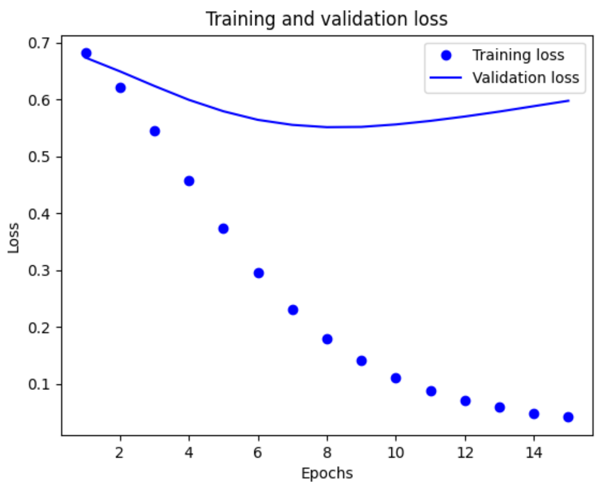
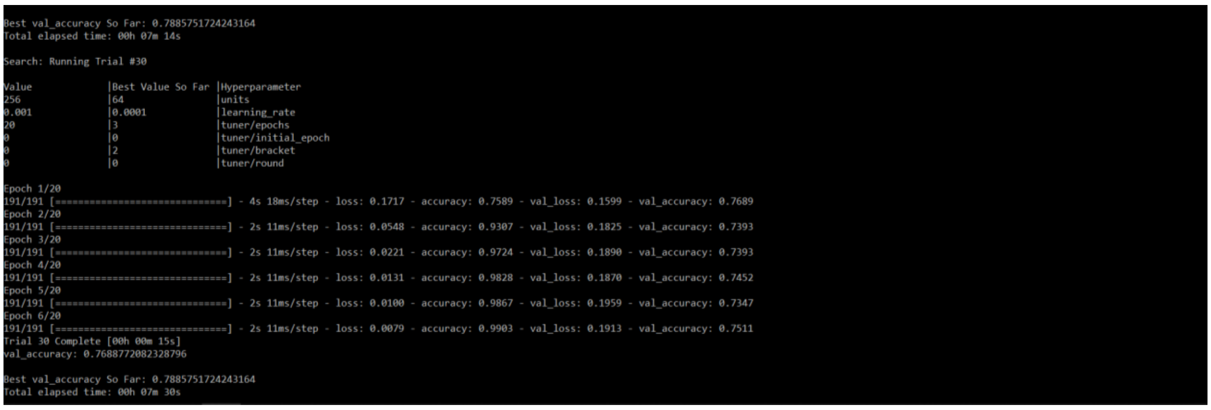


Figure 18, Loss of the best parameters

Adam Mean Square Error



```
Best val_accuracy So Far: 0.7885751724243164
Total elapsed time: 00h 07m 14s

Search: Running Trial #30

Value             |Best Value So Far |Hyperparameter
256               |64                |units
0.001             |0.0001            |learning_rate
20                |3                 |tuner/epochs
0                 |0                 |tuner/initial_epoch
0                 |2                 |tuner/bracket
0                 |0                 |tuner/round

Epoch 1/20
191/191 [==============================] - 4s 18ms/step - loss: 0.1717 - accuracy: 0.7589 - val_loss: 0.1599 - val_accuracy: 0.7689
Epoch 2/20
191/191 [==============================] - 2s 11ms/step - loss: 0.0548 - accuracy: 0.9307 - val_loss: 0.1825 - val_accuracy: 0.7393
Epoch 3/20
191/191 [==============================] - 2s 11ms/step - loss: 0.0221 - accuracy: 0.9724 - val_loss: 0.1890 - val_accuracy: 0.7393
Epoch 4/20
191/191 [==============================] - 2s 11ms/step - loss: 0.0131 - accuracy: 0.9828 - val_loss: 0.1870 - val_accuracy: 0.7452
Epoch 5/20
191/191 [==============================] - 2s 11ms/step - loss: 0.0100 - accuracy: 0.9867 - val_loss: 0.1959 - val_accuracy: 0.7347
Epoch 6/20
191/191 [==============================] - 2s 11ms/step - loss: 0.0079 - accuracy: 0.9903 - val_loss: 0.1913 - val_accuracy: 0.7511
Trial 30 Complete [00h 00m 15s]
val_accuracy: 0.7688772082328796

Best val_accuracy So Far: 0.7885751724243164
Total elapsed time: 00h 07m 30s
```

Figure 19, Tuning to find the best parameters

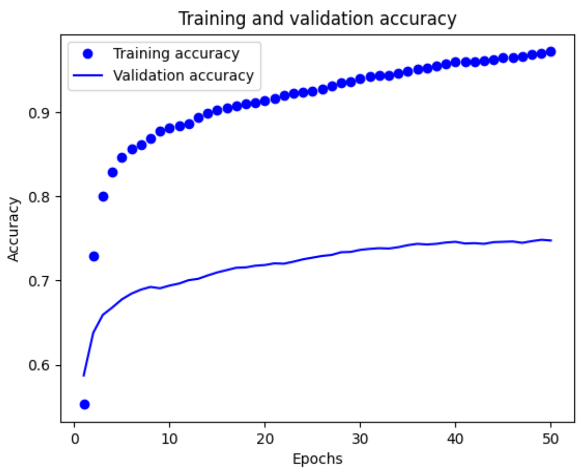


Figure 20, Accuracy of the best parameters

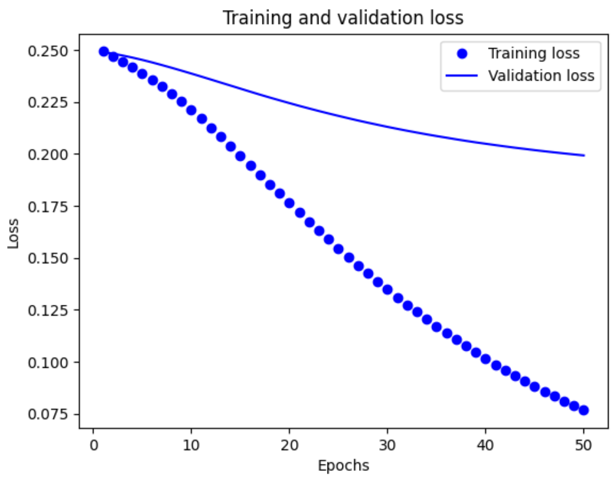
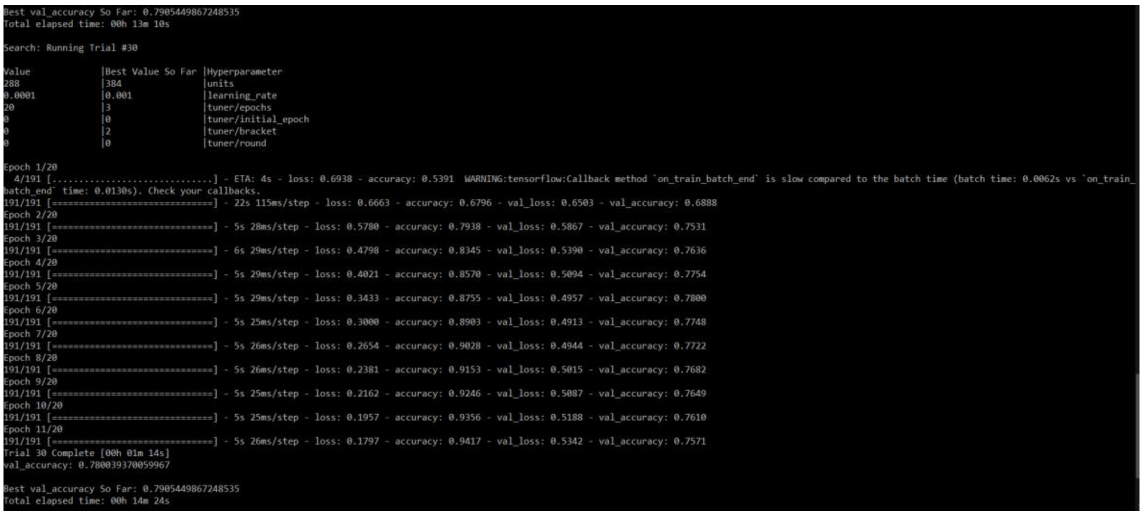


Figure 21, Loss of the best parameters

Root Mean Square Propagation Binary Cross-Entropy



```
Best val_accuracy So Far: 0.7905449867248535
Total elapsed time: 00h 13m 10s

Search: Running Trial #30

Value             |Best Value So Far |Hyperparameter
288               |384               |units
0.0001            |0.001             |learning_rate
20                |3                 |tuner/epochs
0                 |0                 |tuner/initial_epoch
0                 |2                 |tuner/bracket
0                 |0                 |tuner/round

Epoch 1/20
  4/191 [..............................] - ETA: 4s - loss: 0.6938 - accuracy: 0.5391  WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0062s vs `on_train_
batch_end` time: 0.0130s). Check your callbacks.
191/191 [==============================] - 22s 115ms/step - loss: 0.6663 - accuracy: 0.6796 - val_loss: 0.6503 - val_accuracy: 0.6888
Epoch 2/20
191/191 [==============================] - 5s 28ms/step - loss: 0.5780 - accuracy: 0.7938 - val_loss: 0.5867 - val_accuracy: 0.7531
Epoch 3/20
191/191 [==============================] - 6s 29ms/step - loss: 0.4798 - accuracy: 0.8345 - val_loss: 0.5390 - val_accuracy: 0.7636
Epoch 4/20
191/191 [==============================] - 5s 29ms/step - loss: 0.4021 - accuracy: 0.8570 - val_loss: 0.5094 - val_accuracy: 0.7754
Epoch 5/20
191/191 [==============================] - 5s 29ms/step - loss: 0.3433 - accuracy: 0.8755 - val_loss: 0.4957 - val_accuracy: 0.7800
Epoch 6/20
191/191 [==============================] - 5s 25ms/step - loss: 0.3000 - accuracy: 0.8903 - val_loss: 0.4913 - val_accuracy: 0.7748
Epoch 7/20
191/191 [==============================] - 5s 26ms/step - loss: 0.2654 - accuracy: 0.9028 - val_loss: 0.4944 - val_accuracy: 0.7722
Epoch 8/20
191/191 [==============================] - 5s 26ms/step - loss: 0.2381 - accuracy: 0.9153 - val_loss: 0.5015 - val_accuracy: 0.7682
Epoch 9/20
191/191 [==============================] - 5s 25ms/step - loss: 0.2162 - accuracy: 0.9246 - val_loss: 0.5087 - val_accuracy: 0.7649
Epoch 10/20
191/191 [==============================] - 5s 25ms/step - loss: 0.1957 - accuracy: 0.9356 - val_loss: 0.5188 - val_accuracy: 0.7610
Epoch 11/20
191/191 [==============================] - 5s 26ms/step - loss: 0.1797 - accuracy: 0.9417 - val_loss: 0.5342 - val_accuracy: 0.7571
Trial 30 Complete [00h 01m 14s]
val_accuracy: 0.780039370059967

Best val_accuracy So Far: 0.7905449867248535
Total elapsed time: 00h 14m 24s
```

Figure 22, Tuning to find the best parameters

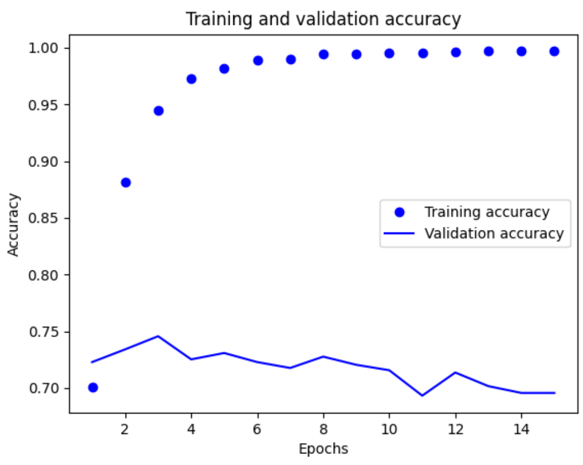


Figure 23, Accuracy of the best parameters

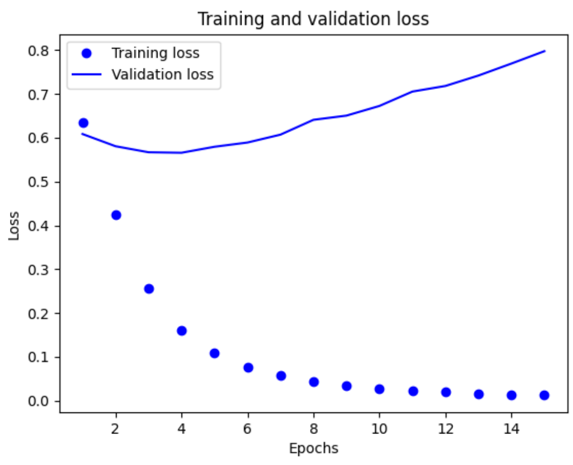
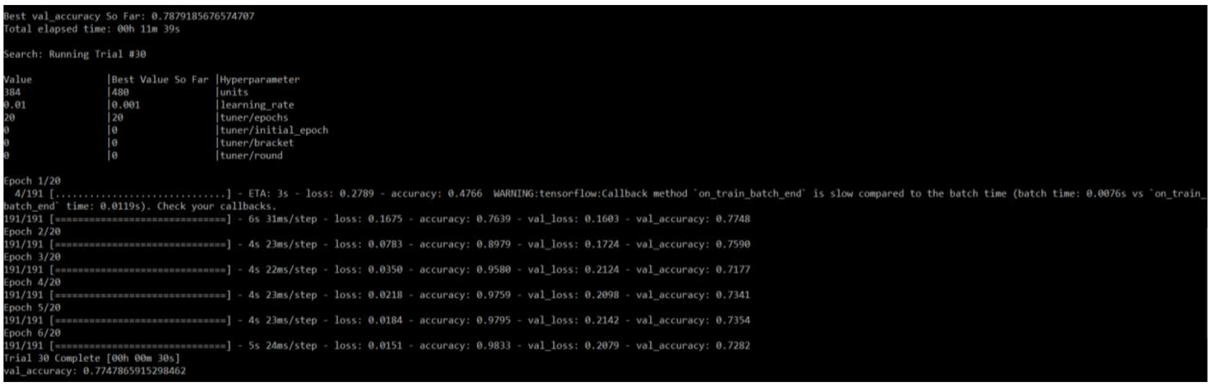


Figure 24, Loss of the best parameters

Root Mean Square Propagation Mean Square Error



```
Best val_accuracy So Far: 0.7879185676574707
Total elapsed time: 00h 11m 39s

Search: Running Trial #30

Value             |Best Value So Far |Hyperparameter
384               |480               |units
0.01              |0.001             |learning_rate
20                |20                |tuner/epochs
0                 |0                 |tuner/initial_epoch
0                 |0                 |tuner/bracket
0                 |0                 |tuner/round

Epoch 1/20
  4/191 [..............................] - ETA: 3s - loss: 0.2789 - accuracy: 0.4766  WARNING:tensorflow:Callback method `on_train_batch_end` is slow compared to the batch time (batch time: 0.0076s vs `on_train_
batch_end` time: 0.0119s). Check your callbacks.
191/191 [==============================] - 6s 31ms/step - loss: 0.1675 - accuracy: 0.7639 - val_loss: 0.1603 - val_accuracy: 0.7748
Epoch 2/20
191/191 [==============================] - 4s 23ms/step - loss: 0.0783 - accuracy: 0.8979 - val_loss: 0.1724 - val_accuracy: 0.7590
Epoch 3/20
191/191 [==============================] - 4s 22ms/step - loss: 0.0350 - accuracy: 0.9580 - val_loss: 0.2124 - val_accuracy: 0.7177
Epoch 4/20
191/191 [==============================] - 4s 23ms/step - loss: 0.0218 - accuracy: 0.9759 - val_loss: 0.2098 - val_accuracy: 0.7341
Epoch 5/20
191/191 [==============================] - 4s 23ms/step - loss: 0.0184 - accuracy: 0.9795 - val_loss: 0.2142 - val_accuracy: 0.7354
Epoch 6/20
191/191 [==============================] - 5s 24ms/step - loss: 0.0151 - accuracy: 0.9833 - val_loss: 0.2079 - val_accuracy: 0.7282
Trial 30 Complete [00h 00m 30s]
val_accuracy: 0.7747865915298462
```

Figure 25, Tuning to find the best parameters

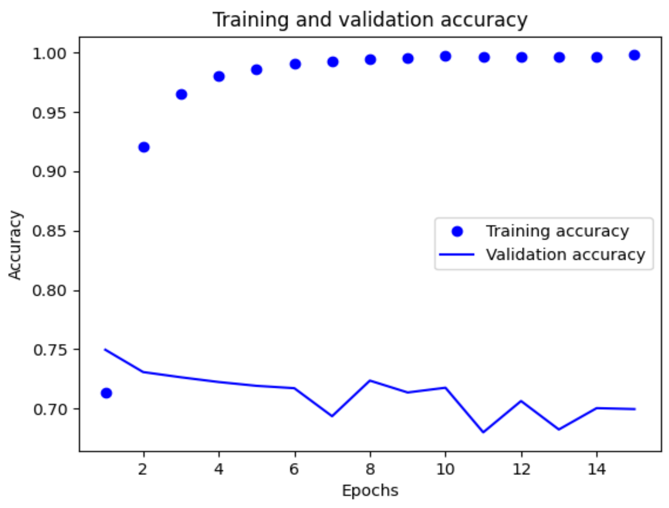


Figure 26, Accuracy of the best parameters

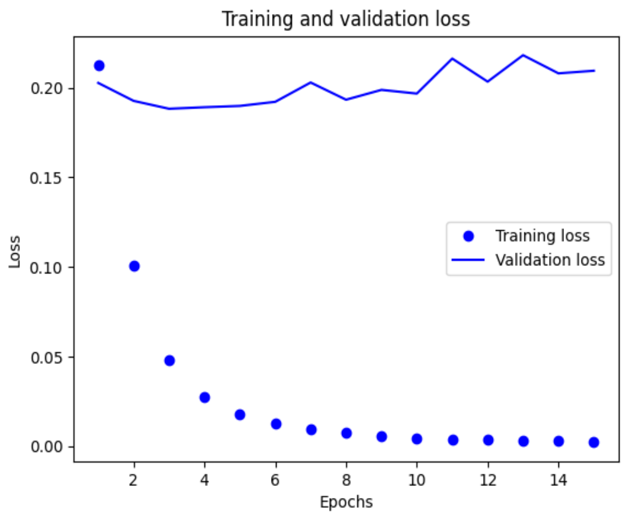
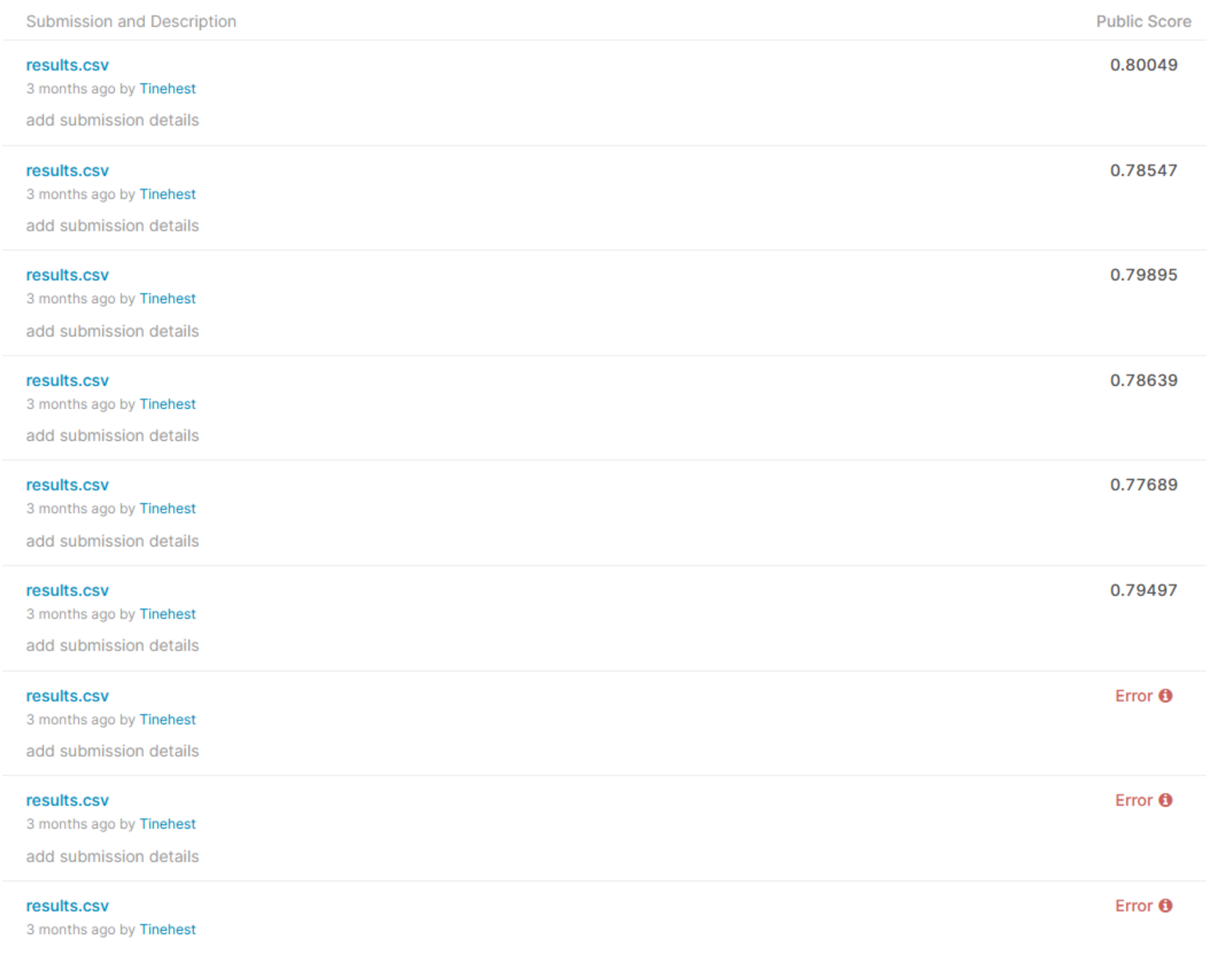


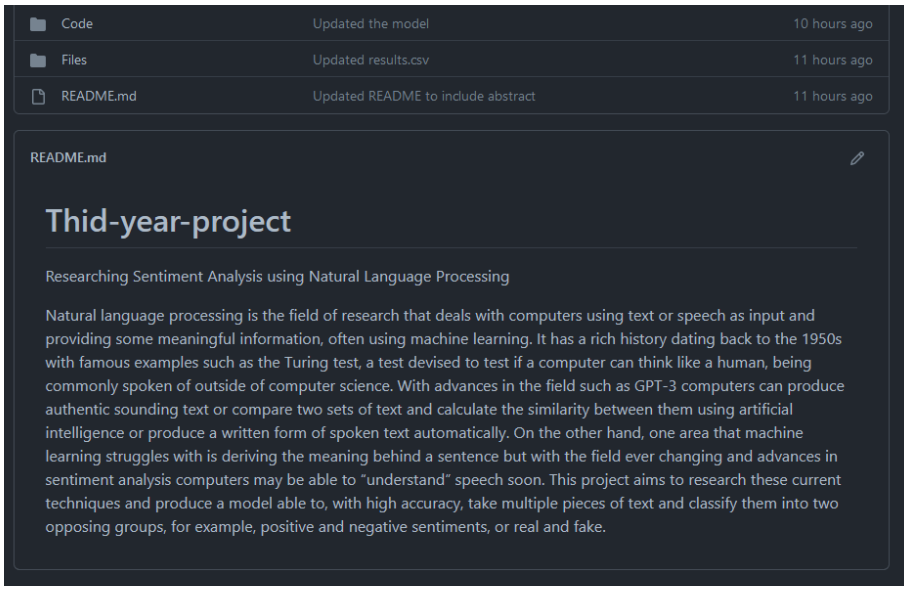
Figure 27, Loss of the best parameters

7.2 Screenshots



Submission and Description	Public Score
results.csv 3 months ago by Tinehest add submission details	0.80049
results.csv 3 months ago by Tinehest add submission details	0.78547
results.csv 3 months ago by Tinehest add submission details	0.79895
results.csv 3 months ago by Tinehest add submission details	0.78639
results.csv 3 months ago by Tinehest add submission details	0.77689
results.csv 3 months ago by Tinehest add submission details	0.79497
results.csv 3 months ago by Tinehest add submission details	Error
results.csv 3 months ago by Tinehest add submission details	Error
results.csv 3 months ago by Tinehest	Error

Figure 28, Submission made to the Kaggle competition



Code	Updated the model	10 hours ago
Files	Updated results.csv	11 hours ago
README.md	Updated README to include abstract	11 hours ago

README.md

Thid-year-project

Researching Sentiment Analysis using Natural Language Processing

Natural language processing is the field of research that deals with computers using text or speech as input and providing some meaningful information, often using machine learning. It has a rich history dating back to the 1950s with famous examples such as the Turing test, a test devised to test if a computer can think like a human, being commonly spoken of outside of computer science. With advances in the field such as GPT-3 computers can produce authentic sounding text or compare two sets of text and calculate the similarity between them using artificial intelligence or produce a written form of spoken text automatically. On the other hand, one area that machine learning struggles with is deriving the meaning behind a sentence but with the field ever changing and advances in sentiment analysis computers may be able to "understand" speech soon. This project aims to research these current techniques and produce a model able to, with high accuracy, take multiple pieces of text and classify them into two opposing groups, for example, positive and negative sentiments, or real and fake.

Figure 29, GitHub repository

7.3 Contents of archive

Files:

The files containing the data given to or returned by the model in csv format.

cleantest.csv – test.csv after being processed by Text Cleaning.py.

cleantrain.csv – train.csv after being processed by Text Cleaning.py.

results.csv – The output of Project.py containing the id from test.csv and the predicted target.

test.csv – The testing set from the Kaggle website containing the id, keyword, location, and text which will be used to predict the target.

train.csv - The training set from the Kaggle website containing the id, keyword, location, text, and target.

unique_words.txt – A list of the most frequent to least frequent words in the testing and training set combined.

Code:

The python code written to generate results.csv.

“Project - 0.79497.py” – The original python file which achieved an accuracy of 0.79497 before tuning, see section 3.4 for more information.

“slangdict.pickle” – The file generated by Text Cleaning.py which contains the slang dictionay which is used to replace abbreviations and slang in test.csv and train.csv.

“Text Cleaning.py” – The code which generates cleantest.csv and cleantrain.csv, see section 3.3 for more information.

“Tuned Project - 0.80049.py” – The final code which generated the results.csv which achieved 0.80049 accuracy on the competition, see section 3.5 for more information.

“Tuning.py” – The python file containing the code that found the optimal parameters for the model, see section 3.5 for more information.

8 Project Brief

Student name: William Hirth, wjth1g19

Supervisor name: Adam Prugel-Bennett

Project title: Researching current techniques of using deep learning for natural language processing

Description:

My aim is to research natural language processing using deep learning while applying it to a Kaggle competition with the task of discerning between authentic disaster tweets and regular tweets. The goal is to develop an understanding of current deep learning techniques and construct a machine learning model which can solve the problem of identifying which tweets are about real disasters and which are not.

https://www.kaggle.com/c/nlp-getting-started/overview