

# Sentiment Classification with TensorFlow Machine Learning

Theodore Perigo

Western Governors University

C964 - Computer Science Capstone

June 2, 2022

## Table of Contents

<b>A. Requirement A</b> .....	4
<b>A.1. Letter of Transmittal</b> .....	4
<b>A.2. Project Proposal</b> .....	5
<b>A.2.1 Problem Summary</b> .....	5
<b>A.2.2 Project Benefits</b> .....	5
<b>A.2.3 Project Description</b> .....	5
<b>A.2.4. Data Description</b> .....	6
<b>A.2.5. Objective and Hypothesis</b> .....	6
<b>A.2.6. Methodology</b> .....	6
<b>A.2.7. Funding Requirements</b> .....	7
<b>A.2.8. Stakeholder's Impact</b> .....	7
<b>A.2.9. Data Considerations and Precautions</b> .....	7
<b>A.2.10. Developer Expertise</b> .....	7
<b>B. Requirement B</b> .....	8
<b>B.1. Executive Summary</b> .....	8
<b>B.1.1. Problem Statement</b> .....	8
<b>B.1.2. Customer Summary</b> .....	8
<b>B.1.3. Existing System Analysis</b> .....	8
<b>B.1.4. Data</b> .....	9
<b>B.1.5. Project Methodology</b> .....	9
<b>B.1.6. Project Outcomes</b> .....	9
<b>B.1.7. Implementation Plan</b> .....	10
<b>B.1.8. Evaluation Plan</b> .....	11
<b>B.1.9. Resources and Costs</b> .....	11
<b>B.1.10. Timeline and Milestones</b> .....	12
<b>C. Required Development Attributes</b> .....	13
<b>C.1. Project files</b> .....	13
<b>C.2 Descriptive and Non-descriptive Method</b> .....	14
<b>C2.1. Descriptive Method:</b> .....	14
<b>C2.2. Non-descriptive Method:</b> .....	14
<b>C.3. Collected Datasets</b> .....	15
<b>C.4. Decision-support Functionality</b> .....	15

<b>C.5. Parsing and Cleaning Functionality .....</b>	<b>15</b>
<b>C.6. Data Exploration and Preparation Functionality .....</b>	<b>16</b>
<b>C.7. Data Exploration and Inspection Functionality .....</b>	<b>16</b>
<b>C.8. Interactive Queries.....</b>	<b>16</b>
<b>C.9. Machine-learning methods.....</b>	<b>17</b>
<b>C.10. Evaluation Functionality .....</b>	<b>17</b>
<b>C.11. Security Features.....</b>	<b>17</b>
<b>C.12. Tools to Monitor and Maintain.....</b>	<b>17</b>
<b>C.13. User-friendly dashboard with Visualizations .....</b>	<b>18</b>
<b>D. Post-implementation Documentation .....</b>	<b>19</b>
<b>D.1. Post-implementation Report .....</b>	<b>19</b>
<b>D.1.1 Project Purpose .....</b>	<b>19</b>
<b>D.1.2. Datasets .....</b>	<b>19</b>
<b>D.1.3. Data Product Code.....</b>	<b>20</b>
<b>D.1.5. Hypothesis Verification .....</b>	<b>23</b>
<b>D.1.6. Effective Visualizations and Reporting.....</b>	<b>23</b>
<b>D.1.7 Accuracy Analysis .....</b>	<b>24</b>
<b>D.1.8. Application Testing .....</b>	<b>24</b>
<b>D.1.9. Source Code and Executable Files.....</b>	<b>24</b>
<b>D.1.10 Quick-start Guide.....</b>	<b>25</b>
<b>E. References .....</b>	<b>28</b>

## A. Requirement A

---

### A.1. Letter of Transmittal

May 30<sup>th</sup>, 2022

Dr. Hal Emmerich  
Chief Technology Officer  
Alpha Analytics  
500 North Street  
San Francisco, CA, 90334

Dr. Emmerich,

The impact of social media and the side effect of ‘brand sentiment’ been a hot topic of discussion in recent years. Businesses and consumers are deeply concerned about sentiment towards themselves, their brands, products, etc., and are constantly changing their marketing to obtain more desired results. As an Analytics company, I feel that further understanding this emotional landscape is paramount to our company’s future success and can lead to a bounty of new business opportunities. To support this effort, I am proposing exploring this space by utilizing machine learning and developing a AI model that can help us classify sentiment for text posts on social media.

I hypothesize that the model, by itself through machine learning, will be able to predict whether a post’s emotional sentiment is positive, negative, or neutral in polarity with 80-90% human-like accuracy. Once properly trained, the AI model can be scaled to analyze millions of text posts in a fraction of the time it would take a human to simply just read a few posts, let alone properly identify its sentiment. This enables automatically aggregated feedback, allowing us to use the information for a variety of purposes for us and our clients.

To develop this model, I will create a small development team of 3 members. 1 software developer, 1 data analyst, and myself, a computer scientist/full-stack engineer with experience developing machine learning models dealing with text classification. As the bulk of the project is derived from free open-source utilities the cost of this project will mainly be in equipment costs. I am estimating that \$25,000 will need to be invested into additional equipment and R&D of our AI model during its development.

I believe this model will enable us to develop more intuitive and useful applications for our clients and should be seen as a great opportunity to grow our company’s operations by our shareholders.

Sincerely,

Theodore Perigo

## **A.2. Project Proposal**

### **A.2.1 Problem Summary**

The ability for users of a public social media platform to express their opinions creates an opportunity for data analysts to further understand sentiment towards particular subjects. Both customers and businesses alike are deeply concerned with the online “image” attached to their brands, products, and personalities.

Much of this feedback data is readily available online through public forums and social media networks. Searching for a particular topic or product on a social network such as Twitter can produce thousands of results, each with varying opinions and sentiments toward the queried subject.

Going through the vast amount of data would take much time and resources for a human. Going through each post one by one and classifying each based on their sentiment to understand the consensus on opinions on a topic would be a very tedious task.

### **A.2.2 Project Benefits**

Automating the task of reading a text post and classifying its sentiment polarity can save a lot of time and resources, especially if the task contains thousands of posts. To support this effort, I propose researching and developing a custom-built AI neural network model that will utilize deep machine learning to quickly automatically process social media posts and accurately rate text based on its sentiment polarity.

I believe that after its development, we could then purpose this AI technology for various future applications that need to analyze the emotional feedback based on the vast amount of publicly available text feedback available online.

Providing our clients with this automatically aggregated sentiment information will help them understand sentiment towards their brand and how online perception is formed. U this analyzed feedback toward products and brands can show potential flaws in marketing or development for clients. This can greatly increase our feedback analytics output, one of our company’s major exports.

### **A.2.3 Project Description**

This project consists of the research and development of an AI neural network model, and then the training of the model. The AI model will be developed using the Python programming language and the neural network will be based on TensorFlow, an end-to-end open-source machine learning platform.

Using TensorFlow, our AI model will be constructed to learn via LSTM (Long-Short-Term-Memory), a type of recurrent neural network. The AI model will be trained on a pre-classified dataset of Twitter posts. Once the AI model is trained, it will be able to take text as input, for example, another Twitter post, and then as output, predict the text's sentiment polarity as [Positive], [Negative], or [Neutral].

#### **A.2.4. Data Description**

The data we will be using to build, train, and test the AI model will be from a combination of sources containing extracted raw Twitter text and assigned sentiment. This collection includes multiple open-source dataset repositories: kaggle.com, github.com, and TensorFlow-datasets. In these open-source datasets, the community has already provided pre-classified data, provided directly from the Twitter API.

The data contains multiple attribute fields, including raw text posts, usernames, post IDs, and sentiment labels. We will be further extracting, processing, and cleaning the data to specifically suit our AI model's needs.

#### **A.2.5. Objective and Hypothesis**

The goal of This project is to automate the task of analyzing text and classifying its sentiment polarity by training a neural network AI model to learn how. This would reduce the required manpower and time and cost of resources that would need to be dedicated to read potentially tens of thousands of text posts. I hypothesize that the AI model will be able to predict the sentiment of a single text post instantaneously and accurately with at least 90% accuracy after initial training. The model will also be able to be improved over its life cycle, increasing in speed and accuracy, and can be scaled to analyze many thousands of posts at a time.

#### **A.2.6. Methodology**

We will be managing this project following the Agile development methodology, as I believe the Agile approach would best fit a project of this size. The development of the AI model will be broken down into several steps:

1. Data procurement from open-source dataset repositories
2. Merge data sources and pre-process.
3. Clean data and ready for use in the model
4. Split the cleaned and processed data into two groups: a training set to teach the model, and a testing set for validating the model's results.
5. Convert the text data to tokens for use in the model.
6. Construct and compile the LSTM model using the TensorFlow RNN platform.
7. Train the Model with the training dataset
8. Evaluate the Model
9. Document findings and produce visuals for data representation

10. Provide a simple Command Line Interface to demonstrate the AI.

Once this explorative venture is complete, we can then see into expanding the capabilities of the AI model and work to implement it into our future applications and products.

#### **A.2.7. Funding Requirements**

As the bulk of the project is derived from free open-source utilities the cost of this project will mainly be in equipment costs. Developer salaries are already configured into the company payroll so team members will just need to allocate their time to this project. Machine learning utilizes hefty CPU and GPU resources and requires powerful computing machines to train. In training the initial model, we will start with the minimum hardware requirements needed and upon completion and further approval, scale the operation from there. I am estimating that \$25,000 will need to be invested in the research and development of our AI model.

#### **A.2.8. Stakeholder's Impact**

With a modest initial investment, I believe that the manpower, time, and resources that will be saved after a successful implementation of the AI model into future applications will most definitely be a positive Return-on-Investment. I believe this model will enable us to develop more intuitive and useful applications for our clients so that they can improve their businesses. This will increase client interest, acquisition, and retention, and therefore should be seen as a great opportunity to grow our company's operations by all stakeholders.

#### **A.2.9. Data Considerations and Precautions**

Data from datasets we are procuring are all from free open-source repositories and have been legally acquired from their respective sources. Concerning potentially sensitive data, it should be noted we will be dropping personally identifiable information (e.g., usernames connected to specific tweets) from our dataset tables, even though the information is publicly available online via the openly provided Twitter API, and access and use of this data will not violate any privacy laws in the United States.

#### **A.2.10. Developer Expertise**

I will create a small development team of 3 members. 2 software developers and 1 data analyst. As one of the full-stack software developers on board, I have significant experience in developing machine learning models specializing in text classification.

## B. Requirement B

---

### B.1. Executive Summary

#### B.1.1. Problem Statement

Businesses are deeply concerned with the online “image” attached to their brands, products, and personalities. Much of this feedback data is readily available online through public forums and social media networks. For example, searching for a particular topic or product on a social network such as Twitter can produce thousands of results, each with varying opinions and sentiments toward the queried subject.

Going through the vast amount of data would take much time and resources for a human. Going through each post one by one and classifying each based on their sentiment to understand the consensus on opinions on a topic would be a very tedious task. Therefore, automating this process of analysis and classification may be beneficial to increasing the productivity and output of our analysis operations.

#### B.1.2. Customer Summary

This project is intended as an expansion of our internal operations for our data analysis division. Once developed, it can be released stand-alone or be integrated into our existing data analytic applications for use by clients and business partners. With the project being open source in nature, making aspects of the project available online would be in provide interesting data science to study and contribute to the further development of forks of the main project.

#### B.1.3. Existing System Analysis

Currently, we utilize several teams of junior data analysis to go over thousands of lines of consumer feedback found on various social media platforms. Our human analysts try their best to determine the sentiment of a line of text based on their perception of what they read. This is obviously very time consuming and has been reported to be a very tedious task by many junior analysts.

In the past, there have been some efforts to try to automate the task of classifying text by filtering certain keywords and phrases from a pre-formed dictionary of desired terms. However, this method is not very accurate and is limited to whatever keywords are in the dictionary and still requires human intervention to adjust the automation to fit specific types of data. We believe that by utilizing our AI neural network we can get similar results to a human analyst, and be able to automatically parse, analyze and classify thousands of lines of text in a fraction of the time it would take a human to.



#### **B.1.4. Data**

The data we will be using to build, train, and test the AI model will be from a combination of sources containing extracted raw Twitter text and assigned sentiment. This collection includes multiple open-source dataset repositories: kaggle.com, github.com, and TensorFlow-datasets. In these open-source datasets, the community has already provided pre-classified data, provided directly from the Twitter API.

The data contains multiple attribute fields, including raw text posts, usernames, post IDs, and sentiment labels. We will be further extracting, processing, and cleaning the data to specifically suit our AI model's needs. Our dataset will consist of an estimated 50,000 data points.

#### **B.1.5. Project Methodology**

We will be managing this project following the Agile development methodology, as I believe the Agile approach would best fit a project of this size. Once this explorative venture is complete, we can then see into expanding the capabilities of the AI model and work to implement it into our future applications and products.

#### **B.1.6. Project Outcomes**

This project if successful will significantly reduce the cost and time needed for our human analysts to process and classify text based on their sentiment polarity. The model can then be later upgraded and retrofitted to expand its capabilities to work with other applications. The initial model should have an estimated 80-90% sentiment prediction accuracy based on the length of text per post.

### B.1.7. Implementation Plan

1. Data procurement from open-source dataset repositories.  
Repositories include:
  - kaggle.com [<https://www.kaggle.com/datasets>]
  - github.com API [<https://github.com/datasets>]
  - TensorFlow-dataset packages  
[<https://www.tensorflow.org/datasets/overview>]
  - Twitter API [<https://developer.twitter.com/en/docs/twitter-api>]
2. Merge data sources and pre-process.
3. Clean data and ready for use in the model
  - a. Utilize Python and Regular Expression (regex library) scripts to parse data files and clean text for processing by the model.
    - i. Drop unneeded columns and rows from datasets (e.g., usernames, textIDs)
    - ii. Clean text by removing unnecessary alphanumeric characters and symbols, weblinks, retweets, mentions, etc.
    - iii. Convert sentiment label strings to the float datatype for processing by the model. [“positive”] = 1.0, [“negative”] = 0.0.
4. Split the cleaned and processed data into two groups: a training set to teach the model, and a testing set for validating the model’s results. This function can be adjusted by changing the ratio of the split. Initial settings will have a [0.30] (70% training data / 30% testing) data split.
5. Convert the text data into tokenized via an imported Tokenizer library, then convert the tokens into sequences for use in the model.
6. Construct and compile the AI model.
  - a. The model will be constructed using the TensorFlow recurrent neural network (RNN) platform retrofitted for text classification.
  - b. The neural network will be trained via the Long-short-term-memory (LSTM) type.
  - c. The model will then be compiled using [Binary cross-entropy] as the loss function and use [Adam] optimization with an initial learning rate hyperparameter set to [ $1 \times 10^{-4}$ ].
7. Train the model with the training dataset.
  - a. Initially, we will use [10] epoch iterations for training and [30] validation steps, validating loss and accuracy in each step.
8. Evaluate the Model
  - a. Additional Python functions will return estimated loss and accuracy values
  - b. Plot visually using the matplotlib Python library.
9. Document findings and produce visuals for data representation
10. Provide a simple Python Jupyter Notebook Interface to interact with and evaluate the model.

### B.1.8. Evaluation Plan

After training, the model will be evaluated using the allocated test data split from the cleaned data files. Additional python functions will also show the estimated loss and accuracy values.

A prediction function will be constructed utilizing the trained model and we can then run this function to view the results of the model's training. The prediction function will take a sample input string provided by a user in the form of a Twitter post. The text can be passed to the function as an argument and will be run through the model.

The function will analyze the text and return its predicted sentiment value in the form of a float datatype between [0.0 ~ 1.0]. The sentiment polarity will be evaluated based on pre-assigned criteria values:

- Positive sentiment will have a value of  $[x \geq 0.60]$ .
- Neutral sentiment will have a value of  $[x \geq 0.40; x < 0.60]$ .
- Negative sentiment will have a value of  $[x < 0.40]$ .

An additional helper function will read the sentiment value and return a text string containing the appropriate sentiment polarity label. This can be viewed and verified by human users to further evaluate the validity of the model. We believe the first iteration of this model should predict sentiment values with [80-90%] accuracy. We do also expect differing opinions of whether the model's results are accurate based on what some human users' own opinions would rate a text's sentiment.

### B.1.9. Resources and Costs

As the bulk of the project is derived from free open-source utilities the cost of this project will mainly be in equipment costs. Developer salaries are already configured into the company payroll so team members will just need to allocate their time to this project.

Machine learning utilizes hefty CPU and GPU resources and requires powerful computing machines to train. The TensorFlow machine learning platform also requires a tensor-core equipped Nvidia spec GPU for all machine learning applications.

Currently, our target specs for the model development machines are:

- CPU: Intel Core i7-8700k
- GPU: Nvidia RTX 3080
- RAM: 32 GB
- Storage: 1 TB
- OS: Windows 11

I am estimating that \$25,000 will need to be invested into additional equipment and R&D of our AI model during its development. In training the initial model, we will start with the minimum hardware requirements needed and upon completion and further approval, scale the operation from there.

### B.1.10. Timeline and Milestones

We are estimating the research and development of the model to take approximately 1 month with a Summer 2022 start. We will be managing this project following the Agile development methodology, as I believe the Agile approach would best fit a project of this size.

The development of the model will be broken down into several milestones:

Milestone	Task	Days	Start	End
1	Research and Planning	3	11-Jul-2022	14-Jul-2022
2	Data Procurement	2	12-Jul-2022	14-Jul-2022
3	Data Pre-processing Module	2	18-Jul-2022	20-Jul-2022
4	Data Conversion Module	2	20-Jul-2022	22-Jul-2022
5	Model Design	3	25-Jul-2022	28-Jul-2022
6	Model Development	9	27-Jul-2022	5-Aug-2022
7	Evaluation	4	8-Aug-2022	12-Aug-2022
8	Document Findings	3	9-Aug-2022	12-Aug-2022

## C. Required Development Attributes

---

See source code for full details.

For ease of evaluation, I will list a minimum of one example for each required attribute. Each attribute may have more than one method or implementation that can be found in the full source code.

### C.1. Project files

~\Sentiment Analysis

  \data

    └─ cTweets.csv

  \installation

    └─ Miniconda3-latest-Windows-x86\_64.exe

    └─ VC\_redist.x64.exe

  └─ READ\_ONLY\_sentiment-analysis.html

  └─ sentiment-analysis.ipynb

  └─ TPerigo\_Capstone\_Task2\_MASTER-DOCUMENT.pdf

## C.2 Descriptive and Non-descriptive Method

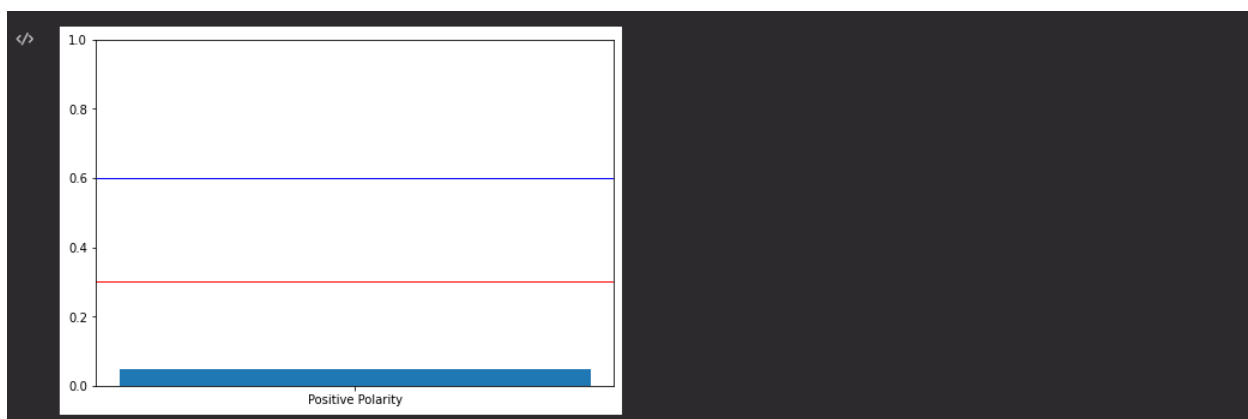
### C2.1. Descriptive Method:

This bar graph shows the distribution of the 3 different sentiment labels attached to each separate line of text in the data file.



### C2.2. Non-descriptive Method:

This bar graph shows the sentiment value of a particular line of text. We can predict the sentiment by referencing the threshold lines on the graph. If the bar is underneath the red threshold line, the sentiment is negative. If the bar reaches above the blue threshold line, the sentiment is positive.



### C.3. Collected Datasets

The datasets used in this project are found in the `.../data/cTweet.csv` file in the project directory

### C.4. Decision-support Functionality

The project itself is used to assist in the classification of text for analysis purposes. The results output from the model can be used by users to aid in analysis operations.

### C.5. Parsing and Cleaning Functionality

The project contains functions to read the provided data file and parse the information to a local variable for use.

```
3 df = pd.read_csv(LOCAL_DATA_FILE, names=["textID","text","sentiment"])
```

The project contains a function that will modify/format the data for use in training and testing the model.

```
1 def clean_text(text):
2     text = re.sub("[^a-zA-Z]", " ", str(text)) # Replace all non-letters with spaces
3     text = re.sub(r'@[A-Za-z0-9]+', '', text) # Remove @mentions
4     text = re.sub(r'@[A-Za-z0-9]+', '', text) # Remove @mentions
5     text = re.sub(r'@[A-Za-z]+', '', text) # Remove @mentions
6     text = re.sub(r'@[-]+', '', text) # Remove @mentions
7     text = re.sub(r'#', '', text) # Remove '#' sign
8     text = re.sub(r'RT[\s]+', '', text) # Remove RT
9     text = re.sub(r'https?\/\S+', '', text) # Remove the hyper Link
10    text = re.sub(r'&[a-z;]+', '', text) # Remove '&gt;'
11    text = re.sub(r'\s+', " ", (text)) # Remove excess whitespace
12    return text
```

[153] ✓ 0.6s

Python

```
1 df = df.drop(columns='textID')
2 df = df[df.sentiment != "neutral"]
3
4 df.text = df.text.apply(clean_text)
5
6 df = df.replace("negative", 0.0)
7 df = df.replace("positive", 1.0)
8
9 # Shuffle the data
10 df = df.sample(frac=1).reset_index(drop=True)
11
12 df.head(5)
```

[154] ✓ 0.5s

Python

## C.6. Data Exploration and Preparation Functionality

Throughout various points in the project, there is code that outputs the prepared raw data.

```
5 df.head(5)
```

[3] ✓ 0.1s Python

	textID	text	sentiment
0	tpryan	@stellargirl I loooooooooooooo my Kindle2. ...	positive
1	vcu451	Reading my kindle2... Love it... Lee childs i...	positive
2	mandanicole	how can you not love Obama? he makes jokes abo...	positive
3	kylesellers	@Karoli I firmly believe that Obama/Pelosi hav...	negative
4	theviewfans	House Correspondents dinner was last night who...	positive

## C.7. Data Exploration and Inspection Functionality

Throughout various points in the project, there is code that outputs the processed data for inspection.

```
10
11 df.head(5)
```

Python

	text	sentiment
0	Sure its easier to login every day and make p...	1.0
1	BankyHype or she could have just realized in ...	0.0
2	iamshur http cadence latchmered org new blog	1.0
3	Heading to quot hope in the city quot benefit ...	1.0
4	Just finished reading Chuck Palahniuk s Pygmy ...	0.0

## C.8. Interactive Queries

The section at the end of the project allows for a user to input a custom text string into the Jupyter Notebook to test the predictive capabilities of the model. A new result is returned showing some data related to the prediction task.

↓ Insert custom text below ↓

- Insert your text into the sample\_text\_here field between the quotes [' ']
- Then run the code block below
- Do not use quotes "" or apostrophes ' inside the custom text

```
1 #####
2
3 sample_text_here = ['That movie was terrible!']
4
5 #example_text_here1 = ['What a wonderful day. Im so happy.']
6 #example_text_here2 = ['That movie was terrible! I want my money back.']
7
8 #####
```



## C.9. Machine-learning methods

In the project, a Recurrent Neural Network (RNN) is created using the TensorFlow machine learning platform. Code construct, compile, train, and evaluate the model are present in the source code.

```

1 model = tf.keras.Sequential()
2 model.add(tf.keras.Input(shape=(T, )))
3 model.add(tf.keras.layers.Embedding(vocab+1, 20))
4 model.add(tf.keras.layers.LSTM(15, return_sequences=True))
5 model.add(tf.keras.layers.GlobalMaxPooling1D())
6 model.add(tf.keras.layers.Dense(32, activation='relu'))
7 model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

```

[160] ✓ 0.2s Python

### Compile the Model

```

1 model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer=tf.keras.optimizers.Adam(1e-4), metrics=['accuracy'])

```

[161] ✓ 0.4s Python

## C.10. Evaluation Functionality

The project contains code that will evaluate the accuracy of the model. The project also produces visuals to show the accuracy and loss functions during the training of the model. The user can also manually enter a query into the notebook to further evaluate the model.

```

1 eval_loss, eval_acc = model.evaluate(pad_train, y_train)
2
3 print('Loss:', eval_loss)
4 print('Accuracy:', eval_acc)

```

[16] ✓ 3.6s Python

```

... 887/887 [=====] - 3s 3ms/step - loss: 0.1292 - accuracy: 0.9572
Loss: 0.12920980155467987
Accuracy: 0.9571574330329895

```

## C.11. Security Features

The project is self-contained within the Python Jupyter Notebook. The project only accesses local files as the data and required module libraries are prepared in advance. No other security features are necessary at this time.

## C.12. Tools to Monitor and Maintain

The project is created in a Python Jupyter Notebook. The project is run in a top-down methodology and each code block is executed in sequential order and can be monitored during runtime. Testing and debugging took place during the development of each code block. The Jupyter notebook also contains a history of save-states in case a backup is needed. No other long-term maintenance is required at this time.

### C.13. User-friendly dashboard with Visualizations

The entire project is represented in a Python Jupyter Notebook. This is a notebook-type dashboard that is used for web-based interactive collaboration. The notebook-like structure makes viewing and executing python code and visualizations very user-friendly like you are reading down through a whitepaper or journal.

Visual information is output via 1) bar graphs (see above), 2) line graphs (see above), and 3) real-time text-line outputs (during the model training).

```

Train the Model

1 history = model.fit(pad_train, y_train, validation_data=(pad_test, y_test), epochs=10, validation_steps=30)
[15] ✓ 1m 12.3s Python

... Epoch 1/10
887/887 [=====] - 11s 9ms/step - loss: 0.6759 - accuracy: 0.6297 - val_loss: 0.6175 - val_accuracy:
0.6962
Epoch 2/10
887/887 [=====] - 7s 8ms/step - loss: 0.5253 - accuracy: 0.7698 - val_loss: 0.5024 - val_accuracy:
0.7750

```

## D. Post-implementation Documentation

---

### D.1. Post-implementation Report

#### D.1.1 Project Purpose

This project is an exercise in learning how to develop neural networks and machine learning applications to help with the automation of certain tasks involving specific datasets. Utilizing Python and various API's/libraries, Jupyter Notebooks, and the TensorFlow machine learning platform I was able to successfully create and train a model that can parse, analyze, and predict the emotional sentiment of a string of text with 80-90% accuracy. The model can then be later retrofitted to perform this analysis on a grand scale, analyzing thousands of related texts in a fraction of a second compared to a human user. This is very favorable as it allows users to do so without human intervention, saving much time and resources. Providing automatically aggregated sentiment information can help users understand sentiment towards topics, for example, specific phrases, words, hashtags, subjects, and brands.

#### D.1.2. Datasets

Data was procured from several different dataset repositories:

- kaggle.com [<https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset>]
- github.com API  
[[https://github.com/tensorflow/datasets/tree/master/tensorflow\\_datasets/text/](https://github.com/tensorflow/datasets/tree/master/tensorflow_datasets/text/)]
- TensorFlow-dataset packages  
[<https://www.tensorflow.org/datasets/catalog/sentiment140>]

The data was then manually combined and placed into a CSV file, [cTweets.csv] which is placed into storage for the model to access locally.

The data is based on text posts that were extracted from twitter.com using the Twitter API by the dataset repository's contributors and covers a variety of topics. Each line of data contains the raw text of a Twitter post with accompanying metadata such as usernames and textID's.

	textID	text	sentiment
0	tpryan	@stellargirl I loooooooooovvvveee my Kindle2. ...	positive
1	vcu451	Reading my kindle2... Love it... Lee childs i...	positive
2	mandanicole	how can you not love Obama? he makes jokes abo...	positive
3	kylesellers	@Karoli I firmly believe that Obama/Pelosi hav...	negative
4	theviewfans	House Correspondents dinner was last night who...	positive

Most importantly, each datapoint also contains a label stating the text's sentiment polarity as [positive] or [negative], which is why these datasets were selected for this project.

- [Positive] polarity sentiment describes sentences that convey a positive emotion (happy, excited, etc.).
- [Negative] polarity sentiment describes sentences that convey a negative emotion (angry, sad, frustrated, etc.).

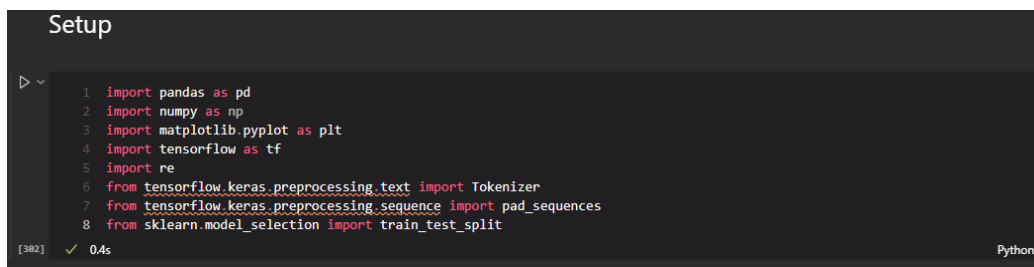
These labels were added both manually by the open-source community and with some assistance from text processors that filtered text including 'emoji-like characters. For example, text with a [😊] 'smiley-face' emoji was usually labeled as positive.

This dataset is then processed and cleaned for use in the model. The dataset originally includes [47,130] lines of data. Then that dataset is split into two groups: a training set and a testing set. The training set is used in the training of the model's neural network machine learning. And the testing set is then used afterward to validate the accuracy of the model.

### D.1.3. Data Product Code

The project is based on a Python Jupyter Notebook. This allows easy modification to separate code modules since the code blocks are essentially separated into multiple steps. The steps are run in a top-down methodology. In this section, I will go over the source code and explain how it is constructed and performs.

First, we begin with the setup step. Here all necessary modules and libraries are loaded for use in the program.



```


Setup

1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import tensorflow as tf
5 import re
6 from tensorflow.keras.preprocessing.text import Tokenizer
7 from tensorflow.keras.preprocessing.sequence import pad_sequences
8 from sklearn.model_selection import train_test_split

[382] ✓ 0.4s Python

```

Now we load the data from the data file that is stored in a local folder in the project directory [.../data/cTweets.csv]. We use the read\_csv() python function to parse the file and create a dataframe with accompanying title names.



```

1 LOCAL_DATA_FILE = "data\cTweets.csv"
2
3 df = pd.read_csv(LOCAL_DATA_FILE, names=["textID", "text", "sentiment"])
4
5 df.head(5)

[323] ✓ 0.1s Python

```

This raw data is pre-processed and cleaned before use in training the model. We do this by running the text through a `clean_text()` function that modifies the text using regular expressions (regex) to the standard needed for the model. This includes removing unwanted symbols and formatting so that the model can easily read and interpret the text.

At this stage, we will also drop any unneeded columns from the data table. It is here we will also shuffle the data to make sure that the data is randomized for better training results as the grouping of the raw data may lead to inaccuracies.

```

1 def clean_text(text):
2     text = re.sub("[^a-zA-Z]", " ", str(text)) # Replace all non-letters with spaces
3     text = re.sub(r'@[A-Za-z0-9]+', '', text) # Remove @mentions
4     text = re.sub(r'@[A-Za-z0-9]+', '', text) # Remove @mentions
5     text = re.sub(r'@[A-Za-z]+', '', text) # Remove @mentions
6     text = re.sub(r'@[-]+', '', text) # Remove @mentions
7     text = re.sub(r'#', '', text) # Remove '#' sign
8     text = re.sub(r'RT[\s]+', '', text) # Remove RT
9     text = re.sub(r'https?\:\/\/\S+', '', text) # Remove the hyper link
10    text = re.sub(r'&[a-z;]+', '', text) # Remove '&gt;';
11    text = re.sub("\s\s+", " ", (text)) # Remove excess whitespace
12    return text

[153] ✓ 0.6s Python

1 df = df.drop(columns='textID')
2 df = df[df.sentiment != "neutral"]
3
4 df.text = df.text.apply(clean_text)
5
6 df = df.replace("negative", 0.0)
7 df = df.replace("positive", 1.0)
8
9 # Shuffle the data
10 df = df.sample(frac=1).reset_index(drop=True)
11
12 df.head(5)

[154] ✓ 0.5s Python

```

Now that the text is ready for use, we then split the data into two parts. One part for training and one part for testing. In this implementation, we use a 70/30 split.

```

1 x_train, x_test, y_train, y_test = train_test_split(df['text'].values, df['sentiment'].values, test_size=0.30)

[155] ✓ 0.4s Python

```

The data now must be converted into a sequence of tokens for the model to learn from the words. We use the `Tokenizer()` function to 'vectorize' text into a sequence of integers for processing by the model. This takes a text string and breaks it down word for word. Then that word is converted to a numeric representation of our vocabulary. Within the neural network, each vocab has a different weight that the model will learn to adjust based on its sentiment. We are also padding the sequences so that each sequence will be of equal length for training the model.

```

1 VOCAB_SIZE = 2000000
2 tokenizer = Tokenizer(num_words=VOCAB_SIZE)
3 tokenizer.fit_on_texts(x_train)
4 wordIndex = tokenizer.word_index
5 vocab = len(wordIndex)
6
7 print('Vocabulary size: ', vocab)

```

[156] ✓ 0.4s Python

... Vocabulary size: 31760

Convert the data into sequences

```

1 train_seq = tokenizer.texts_to_sequences(x_train)
2 test_seq = tokenizer.texts_to_sequences(x_test)
3
4 print('Training sequence: ', train_seq[0])
5 print('Testing sequence: ', test_seq[0])

```

[157] ✓ 0.4s Python

... Training sequence: [11383, 1, 94, 18, 6, 878, 15, 85, 83, 26, 10, 3, 2533]  
Testing sequence: [1454, 217, 1912, 3967, 25, 42]

Add padding to both sequences

```

1 pad_train = pad_sequences(train_seq)
2 T = pad_train.shape[1]
3 print('Training sequence length: ', T)

```

[158] ✓ 0.1s Python

... Training sequence length: 48

At this stage, all the text has been pre-processed and is ready to go. Now we will construct the neural network model using the TensorFlow Sequential() model type. The neural network will be trained using Long-short-term-memory (LSTM), a type of recurrent neural network (RNN). These types of networks are good for text-based classification due to their ability to capture long-term dependencies between word sequences. The model was constructed with the aid of a guide on the TensorFlow API documentation site and most of the hyperparameters values are set to default.

```

1 model = tf.keras.Sequential()
2 model.add(tf.keras.Input(shape=(T, )))
3 model.add(tf.keras.layers.Embedding(vocab+1, 20))
4 model.add(tf.keras.layers.LSTM(15, return_sequences=True))
5 model.add(tf.keras.layers.GlobalMaxPooling1D())
6 model.add(tf.keras.layers.Dense(32, activation='relu'))
7 model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

```

[160] ✓ 0.2s Python

Compile the Model

```

1 model.compile(loss=tf.keras.losses.BinaryCrossentropy(), optimizer=tf.keras.optimizers.Adam(1e-4), metrics=['accuracy'])

```

[161] ✓ 0.4s Python

Finally, the model is trained using the allocated set of training data that was split earlier. The model will train for 10 epochs (iterations), validating itself along the way for data loss and

accuracy. Here is where the model will utilize deep learning to learn how to associate sequence inputs with specific sentiments. Each layer of learning will then be recursively expanded upon during the next iteration.

```
1 history = model.fit(pad_train, y_train, validation_data=(pad_test, y_test), epochs=10, validation_steps=30)
```

[162] ✓ 1m 14.6s Python

### D.1.5. Hypothesis Verification

The hypothesis was to simply see if we could develop and train a neural network that can process, analyze, and predict the emotional sentiment of a given text string. I hypothesized that the machine would be able to do so with [80-90%] accuracy.

After completing the model, I was able to confirm the hypothesis as the model was able to correctly predict the sentiment of text to a max accuracy level of [95%].

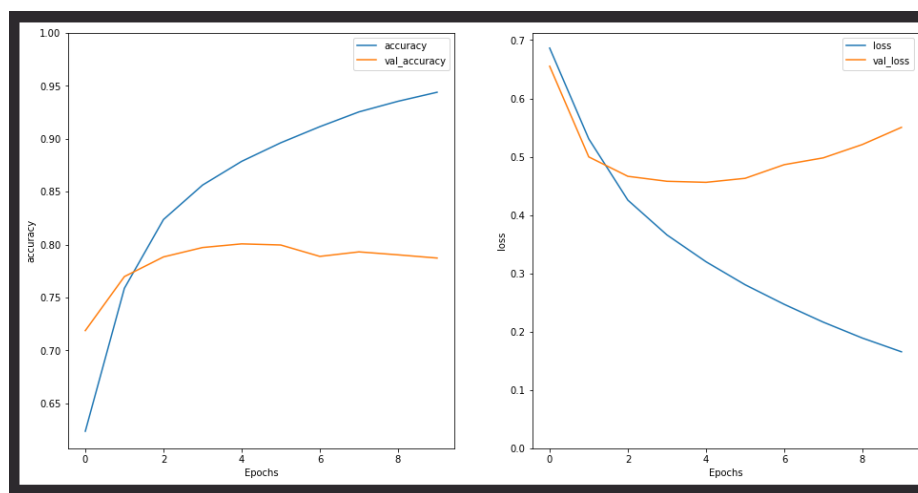
### D.1.6. Effective Visualizations and Reporting

In this project, we use several types of visuals to help describe the data and our results. Within the Jupyter Notebook, Python plotting libraries are used to output these visuals. And the modular layout of the Jupyter notebook itself lends to ease of understanding, as each code block is properly titled and described in the markup code.

First, we have a bar graph showing the distribution of sentiment labels out training and testing datasets. This gives us a greater understanding of the training and testing data sets and can help locate any bias in the data before its use in the model.



Second, we have line graphs showing the model's loss and accuracy changes over each of its training epochs, visualizing that with each iteration, the model is becoming more accurate in its ability to predict sentiment.



### D.1.7 Accuracy Analysis

Model accuracy was validated using a Python function included in the TensorFlow API, `[.evaluate()]` that scores the model based on the test dataset split. The model was able to attain a max accuracy score of `[0.9571]`. This shows that our model is well capable of performing the task of predicting sentiment. The emotion of online text is often misconstrued so certain inaccuracies are forgivable, even for human users. Accuracy can be further improved with model tuning and additional datasets.

### D.1.8. Application Testing

As the project is primarily maintained in a Python Jupyter Notebook, the application has been tested during development at runtime. Each code block module is run in order from a top-down methodology. For a subsequent block to run the prior block must have successfully compiled and run. Therefore, during development, each code module has been constructed and tested to run, accepting all inputs and producing the expected without error. Any changes to the modular pipeline must re-run the notebook to ensure that the application runs as expected.

### D.1.9. Source Code and Executable Files

All the required document source code and necessary data files have been included in this project package. See section C.1 for an overview of the package files.

Some files and packages may need to be installed manually to run the application. Please see the section D.1.10 [Quick-start Guide] for more details.



### D.1.10 Quick-start Guide

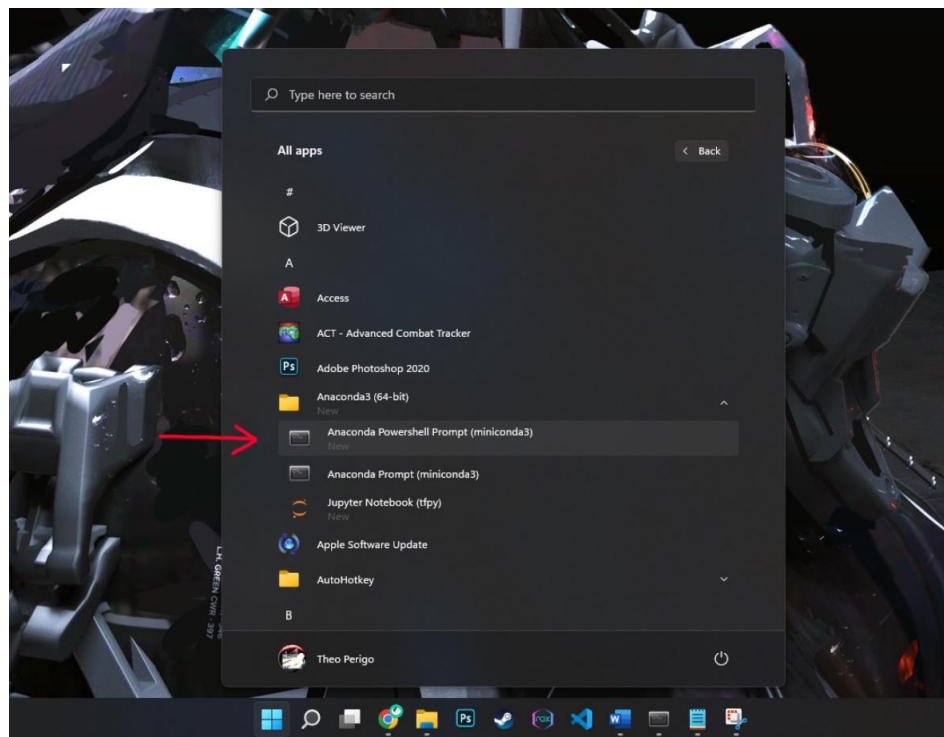
This will install all project dependencies needed to run the project.

**\*\*This installation requires a 64-bit Windows 10/11 machine to run.**

**\*\*This installation requires a tensor-core equipped Nvidia GPU to run.**

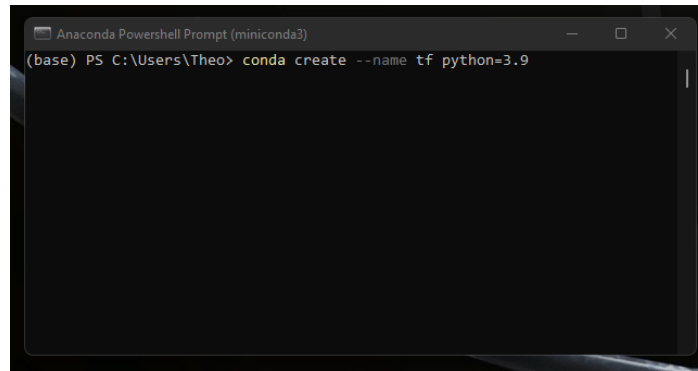
If you are having trouble running the project, I have included an HTML representation of the project in the project directory [READ\_ONLY\_sentiment-analysis.html] that you can view in your browser. However, this is read-only and not interactive (cannot be edited).

1. Install prerequisites located in the ../Installation folder (follow on-screen instructions)
  - a. Microsoft Visual C++ Redistributable [VC\_redist.x64.exe]
  - b. Miniconda 3 [Miniconda3-latest-Windows-x86\_64.exe]
2. Install the latest Nvidia GPU driver for your machine:  
<https://www.nvidia.com/Download/index.aspx>
3. After installing Miniconda, go to your installed programs, and in the Anaconda3 folder run [Anaconda Powershell Prompt (miniconda 3)]



4. Create a new conda development environment named [TF] using the following command inside the PowerShell. Type [y] when prompted to proceed.

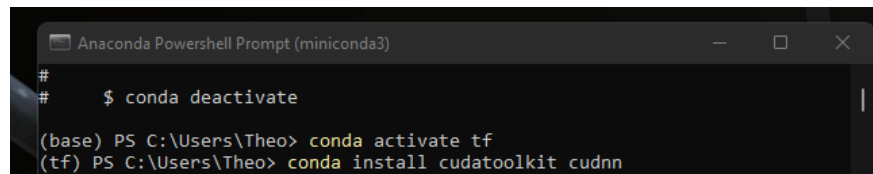
```
conda create --name tf python=3.9
```



Once complete you can activate the environment typing in this command.

```
conda activate tf
```

The (base) name should change to (tf) to show you are now in the tf environment.



5. Inside the tf environment install the CUDA and cuDNN. Type [y] when prompted to proceed.

```
conda install cudatoolkit cudnn
```

6. Inside the tf environment, install pip using the following code:

```
pip install --upgrade pip
```

7. Inside the tf environment, install Tensorflow and sklearn with the following code:

```
pip install tensorflow sklearn
```

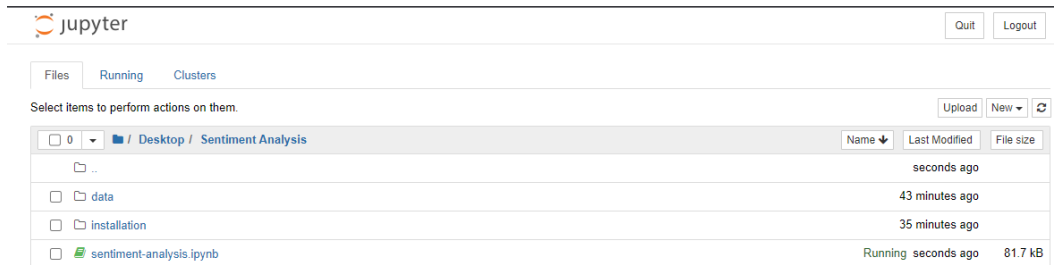
8. Inside the tf environment, install Jupyter Notebook, Pandas, and Matplotlib with the following code:

```
conda install notebook pandas matplotlib
```

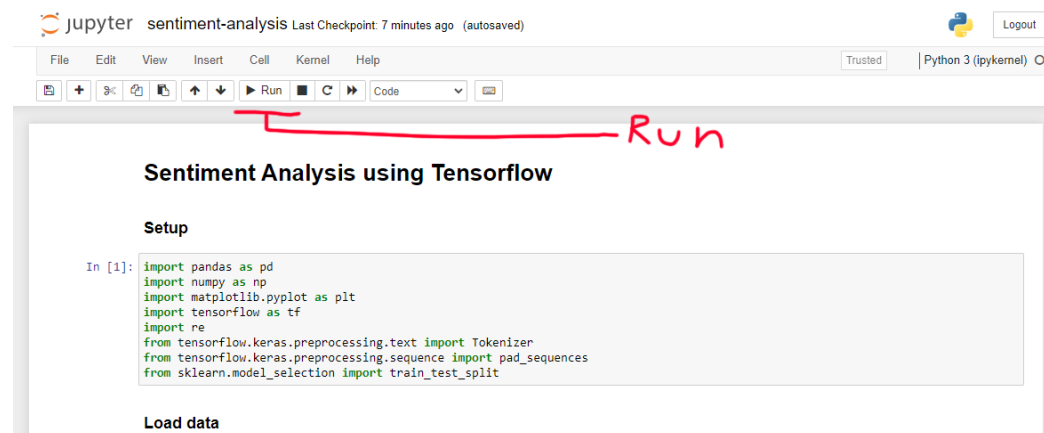
9. Now you can launch Jupyter notebook in your browser by typing in the following code:

```
jupyter notebook
```

10. Inside the Jupyter Notebook, navigate to the project folder wherever you have downloaded it to. Inside the [Sentiment Analysis] folder, click and run [sentiment-analysis.ipynb] to launch the project.



11. The project should load into your Jupyter Notebook window. Now you can run each block of code by clicking the run icon in the icon bar at the top. Or to run the entire project inside the [Cell] menu click [Run All].



12. If you installed the project correctly you should be able to run all blocks of code in order. Training the model may take some time depending on the specs of your machine. (On an Intel i7 8700k / Nvidia RTX 3080 it takes about 60-100 seconds).
13. If you wish to enter your own text to evaluate the model in code block 22, type your text inside the [ ' ' ] quotes and then click the run button to run that code block.
14. To exit, close your browser window, and inside the PowerShell press ctrl-c to end the notebook session. Then exit the PowerShell.
15. To uninstall, uninstall Miniconda 3 from your machine and delete its installed folders.

## E. References

---

No outside sources were cited in the creation of this document.