



# **TweetLyticsAI User Guide: Understanding Arguments on Twitter Using Argument Mining and NLP**

Final Project Report

Author: Mohammad Albinhassan

Supervisor: Dr Odinaldo Rodrigues

Student ID: 1924581

April 6, 2022

# Contents

<b>A User Guide</b>	<b>1</b>
A.1 Instructions . . . . .	1

# Appendix A

## User Guide

### Contents

---

<b>A.1 Instructions . . . . .</b>	<b>1</b>
A.1.1 Running the Application . . . . .	1
A.1.2 Running the Tests . . . . .	5
A.1.3 Running the Data Preparation . . . . .	6
A.1.4 Running the Random Forest Classifier Training . . . . .	7
A.1.5 Running the Twitter RoBERTa Classifier Training . . . . .	7
A.1.6 Using the Application . . . . .	10

---

### A.1 Instructions

#### A.1.1 Running the Application

There are two different ways of running the application, the first being using docker, the other is downloading all the software tools manually and running series of commands. The docker approach is HIGHLY recommended.

#### Running the Application Using Docker

**NOTE:** Docker is a containerisation platform that is a virtualisation software. Its benefit is it will work EXACTLY the same on any device, regardless of the operating system. If you cannot run the application, it will be likely because you installed docker incorrectly or missed one of the steps. Also, make sure that you have localhost port 3000 and 80 free, as this application requires them; this might be a cause for not working.

To run the application using docker, follow these steps:

1. Install docker if you do not already have it, use the following link as a guide:

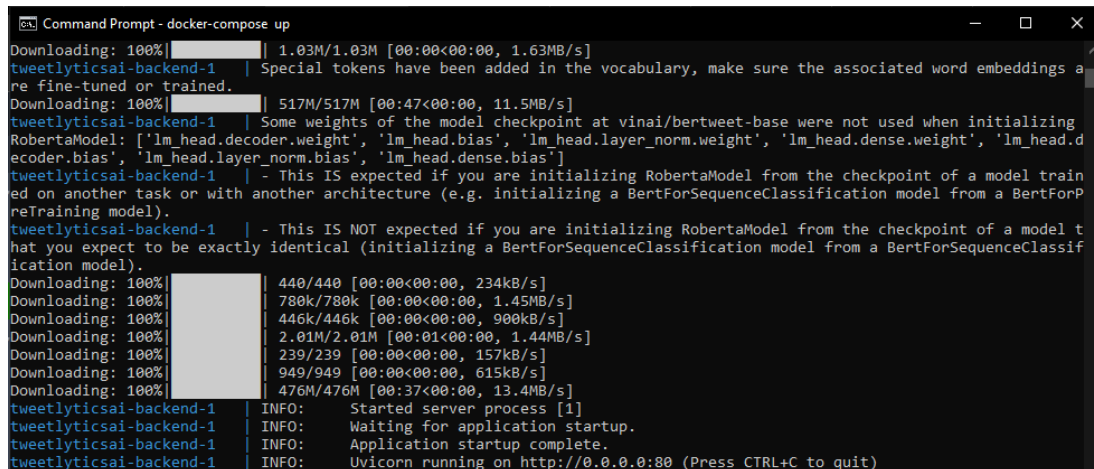
<https://docs.docker.com/get-docker/>

2. Navigate in your terminal to the ‘tweetlyticsai’ directory, which is the root folder in the submitted project. It should contain folders such as ‘ml’, ‘backend’, ‘frontend’, etc. and contain a `docker-compose.yml` file.

3. From your terminal, execute the following command:

```
docker-compose up
```

4. The front end should start before the back end, which you can access from `http://localhost:3000/`. The back end will take a few minutes, given it needs to download and set up a few GB worth of deep learning models. There will be several progress bars that track the download progress of these deep learning models, which can be seen in Figure A.1. When the back end is done setting up and is ready, it should look something like Figure A.1 with a message “Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)”. To double-check it is running, open the following link from your browser `http://127.0.0.1/api/health`; you should get a JSON response that says “API is up and running!”. The terminal should contain logs of the application start-up, which you can monitor to see progress. There will be some front end and back end warnings, which you shouldn’t worry too much about. However, there should be no errors.



```
Command Prompt - docker-compose up
Downloading: 100%|██████████| 1.03M/1.03M [00:00<00:00, 1.63MB/s]
tweetlyticsai-backend-1 | Special tokens have been added in the vocabulary, make sure the associated word embeddings are fine-tuned or trained.
Downloading: 100%|██████████| 517M/517M [00:47<00:00, 11.5MB/s]
tweetlyticsai-backend-1 | Some weights of the model checkpoint at vinai/bertweet-base were not used when initializing RobertaModel: ['lm_head.decoder.weight', 'lm_head.bias', 'lm_head.layer_norm.weight', 'lm_head.dense.weight', 'lm_head.decoder.bias', 'lm_head.layer_norm.bias', 'lm_head.dense.bias']
tweetlyticsai-backend-1 | - This IS expected if you are initializing RobertaModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
tweetlyticsai-backend-1 | - This IS NOT expected if you are initializing RobertaModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).
Downloading: 100%|██████████| 440/440 [00:00<00:00, 234kB/s]
Downloading: 100%|██████████| 780k/780k [00:00<00:00, 1.45MB/s]
Downloading: 100%|██████████| 446k/446k [00:00<00:00, 900kB/s]
Downloading: 100%|██████████| 2.01M/2.01M [00:01<00:00, 1.44MB/s]
Downloading: 100%|██████████| 239/239 [00:00<00:00, 157kB/s]
Downloading: 100%|██████████| 949/949 [00:00<00:00, 615kB/s]
Downloading: 100%|██████████| 476M/476M [00:37<00:00, 13.4MB/s]
tweetlyticsai-backend-1 | INFO: Started server process [1]
tweetlyticsai-backend-1 | INFO: Waiting for application startup.
tweetlyticsai-backend-1 | INFO: Application startup complete.
tweetlyticsai-backend-1 | INFO: Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

Figure A.1: Terminal Displaying Output of Application Successfully Starting Using `docker-compose up`

## Running the Application Manually

Running the application manually requires a few more steps; however, it is needed if you want to run tests, data preparation tasks or train the Random Forest model. Again, make sure you have localhost port 3000 and 80 free.

To run the back end manually, follow these steps:

1. For the back end, you need Python version 3.7.11. It is **HIGHLY** recommended you use an Anaconda virtual environment to run the application and other tasks to prevent version dependency mismatch or unforeseen errors due to different devices. To set up an Anaconda virtual environment, follow the following steps:

- (a) Install Anaconda using the following guide:

```
https://docs.anaconda.com/anaconda/install/
```

- (b) Open the Anaconda terminal

- (c) Create an Anaconda environment using the following command in your Anaconda terminal

```
conda create -n tweetlyticsai python=3.7.11.
```

- (d) Activate the environment using the following command in your Anaconda terminal:

```
conda activate tweetlyticsai
```

- (e) Execute all back end commands from the Anaconda terminal from now on

2. Navigate in your terminal to the ‘tweetlyticsai’ directory, which is the root folder in the submitted project.

3. In your terminal, execute the following command to install all packages required:

```
pip install -r backend/requirements.txt.
```

If that doesn’t work try:

```
pip3 install -r backend/requirements.txt
```

4. Start the backend by running the command in your terminal:

```
uvicorn backend.api.app --host 0.0.0.0 --port 80
```

5. Similar to the docker steps, this should take a few minutes as it needs to download several GB worth of deep learning models. The terminal should have progress bars for these downloads. The terminal should print the following message when the back end is

ready “Uvicorn running on `http://0.0.0.0:80`”, and should look something like Figure A.2. To double-check it is running, open the following link from your browser `http://127.0.0.1/api/health`; you should get a JSON response that says “API is up and running!”.

A terminal window with a dark background and green text. It shows four lines of log output: 'INFO: Started server process [1376]', 'INFO: Waiting for application startup.', 'INFO: Application startup complete.', and 'INFO: Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)'.

```
INFO: Started server process [1376]
INFO: Waiting for application startup.
INFO: Application startup complete.
INFO: Uvicorn running on http://0.0.0.0:80 (Press CTRL+C to quit)
```

Figure A.2: Terminal Displaying Output of Back End Successfully Starting Using a Manual Set-up

To run the UI manually, follow these steps:

**NOTE:** Execute these commands in another standard terminal, not your Anaconda terminal.

1. Download nodejs version 16.14.2, which is the LTS version (Long Term Support), use the following link to do so: <https://nodejs.org/en/>
2. After installing nodejs, install yarn using the following command:

```
npm install --global yarn
```

To double-check it is downloaded run the command to check the version installed:

```
yarn --version
```

Alternatively, you can follow the guide:

<https://classic.yarnpkg.com/lang/en/docs/install/>

3. Navigate to the ‘tweetlyticsai/frontend’ directory, which should contain all the front end code and folders such as ‘src’.
4. Run the command to install all the needed packages:

```
yarn install
```

5. Run the command to start the front end:

```
yarn start
```

6. The front end should now start and can be accessed from the URL `http://localhost:3000/`.

## A.1.2 Running the Tests

To run the automated tests of this software system, you must have followed the “**Running the Application Manually**” steps in Section A.1.1. This subsection will detail how to run the automated tests for the back end and UI.

### Running the Back End Tests

To run the back end tests, follow the following steps:

1. If you downloaded Anaconda, execute these steps from the Anaconda terminal in the ‘tweetlyticsai’ environment, i.e., `conda activate tweetlyticsai`.
2. Navigate in your terminal to the ‘tweetlyticsai’ directory, which is the root folder in the submitted project.
3. Execute the following command to run the tests and collect code coverage:

```
python -m pytest --cov backend/tests --cov-branch --cov-report term-missing
```

4. The terminal should output the code coverage with 100% coverage as shown in Figure A.3.

```
===== test session starts =====
platform win32 -- Python 3.7.11, pytest-7.1.1, pluggy-1.0.0
rootdir: C:\Users\MohammadABH\Documents\Code\University\KCL\Year 3\PRJ\TweetLyticsAI
plugins: anyio-3.5.0, cov-3.0.0, mock-3.7.0
backend\tests\services\test_keyword_extraction_service.py      38      0      6      0 100%
backend\tests\services\test_relation_based_classifier_service.py 21      0      2      0 100%
backend\tests\services\test_sentiment_analysis_service.py     50      0      6      0 100%
backend\tests\services\test_tweet_tree_builder_service.py     72      0      6      0 100%
backend\tests\services\test_twitter_api_service.py            28      0      2      0 100%
backend\tests\services\utils\test_preprocessor.py             17      0      2      0 100%
-----
TOTAL                                                         289      0     30      0 100%
===== 38 passed in 46.75s =====
```

Figure A.3: Back end code coverage using `pytest` with 100% code coverage

### Running the User Interface Tests

To run the UI tests, follow the following steps. Execute all the commands in a standard terminal, not in an Anaconda terminal:

1. Navigate to the ‘tweetlyticsai/frontend’ directory, which should contain all the front end code.
2. Execute the following command to run the tests:

```
yarn test
```

3. Execute the following command to run the tests and collect code coverage:

```
yarn test --coverage .
```

The output should showcase high code coverage, as shown in Figure A.4.

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s
All files	88.63	100	88	87.8	
src	100	100	100	100	
App.tsx	100	100	100	100	
src/api	100	100	100	100	
FetchTweets.tsx	100	100	100	100	
src/components	86.36	100	76.92	85.71	
ColorModeSwitcher.tsx	100	100	100	100	
ErrorAlert.tsx	100	100	100	100	
HomeButton.tsx	100	100	100	100	
LoadingAnimation.tsx	100	100	100	100	
TweetTree.tsx	100	100	100	100	
TweetTreeVisualization.tsx	77.77	100	62.5	76	31,57-59,63-64
src/pages	84	100	100	83.33	
HomePage.tsx	100	100	100	100	
PageNotFound.tsx	100	100	100	100	
TweetAnalysisPage.tsx	80.95	100	100	80	17-19,24
src/tests	100	100	100	100	
test-utils.tsx	100	100	100	100	

Figure A.4: UI code coverage using React Testing Library with high code coverage

### A.1.3 Running the Data Preparation

This project involved preparing data by constructing and cleaning a large dataset from two datasets, namely, Debagreement and SRQ. The following steps will detail how to construct that dataset. It is important to note that if you want to run the Random Forest classifier steps in Section A.1.4 or fine-tune Twitter RoBERTa in Section A.1.5, you must follow these steps:

1. Make sure that you followed the “**Running the Application Manually**” steps in Section A.1.1.
2. If you installed the Anaconda terminal, use it here.
3. Navigate in your terminal to the ‘tweetlyticsai’ directory, which is the root folder in the submitted project.
4. Execute the command to generate the dataset:

```
python ml/datapreprocessing/prepare_dataset.py
```

5. You should now have the file `dataset.csv` in the directory ‘tweetlyticsai/ml/datasets/’.



### A.1.4 Running the Random Forest Classifier Training

Part of this project involved training, hyperparameter tuning and evaluating a Random Forest classifier. To run these processes, follow the following steps:

**NOTE:** Running the following commands will take a prolonged amount of time, given a Random Forest classifier needs to be trained on thousands of data points.

1. Make sure that you followed the “**Running the Application Manually**” steps in Section A.1.1.
2. Make sure that you generated the `dataset.csv` file as detailed from the steps in Section A.1.3.
3. Navigate in your terminal to the ‘tweetlyticsai’ directory, which is the root folder in the submitted project.

4. Execute the following command to train and evaluate a Random Forest model with the optimal hyperparameters found from grid search:

```
python ml/training/random_forest.rbam.py best_grid
```

5. Execute the following command to train and evaluate a Random Forest model with the optimal hyperparameters found from random search:

```
python ml/training/random_forest.rbam.py best_random
```

6. Execute the following command to run grid search hyperparameter tuning and evaluation on the Random Forest model:

```
python ml/training/random_forest.rbam.py tune_grid
```

7. Execute the following command to run random search hyperparameter tuning and evaluation on the Random Forest model:

```
python ml/training/random_forest.rbam.py tune_random
```

### A.1.5 Running the Twitter RoBERTa Classifier Training

Running the Twitter RoBERTa hyperparameter tuning and training must be done on Google Colab as local machines are unlikely to have sufficient GPU. Hence, it is not recommended to test this file due to the large number of steps. Nevertheless, the steps are as follows:

1. Download the `bert_rbam.ipynb` file from the ‘tweetlyticsai/ml/training/’ directory and upload it to Google Drive, URL is as follows:  
  
`https://drive.google.com/drive/my-drive`.
2. Download the `dataset.csv` in the directory ‘tweetlyticsai/ml/datasets/’ and upload it to Google Drive. The training script expects the `dataset.csv` file to be located in the root of your Google Drive; if you place it in another directory, you must modify the `DATASET_PATH` variable in the `bert_rbam.ipynb` file.
3. Open the uploaded `bert_rbam.ipynb` file on Google Drive by double-clicking it. Google Colab should now open.
4. In the toolbar at the top of the page, there is a tab called “Runtime”, click it.
5. A list of options should open, click the “Change runtime type” option as shown in Figure A.5.

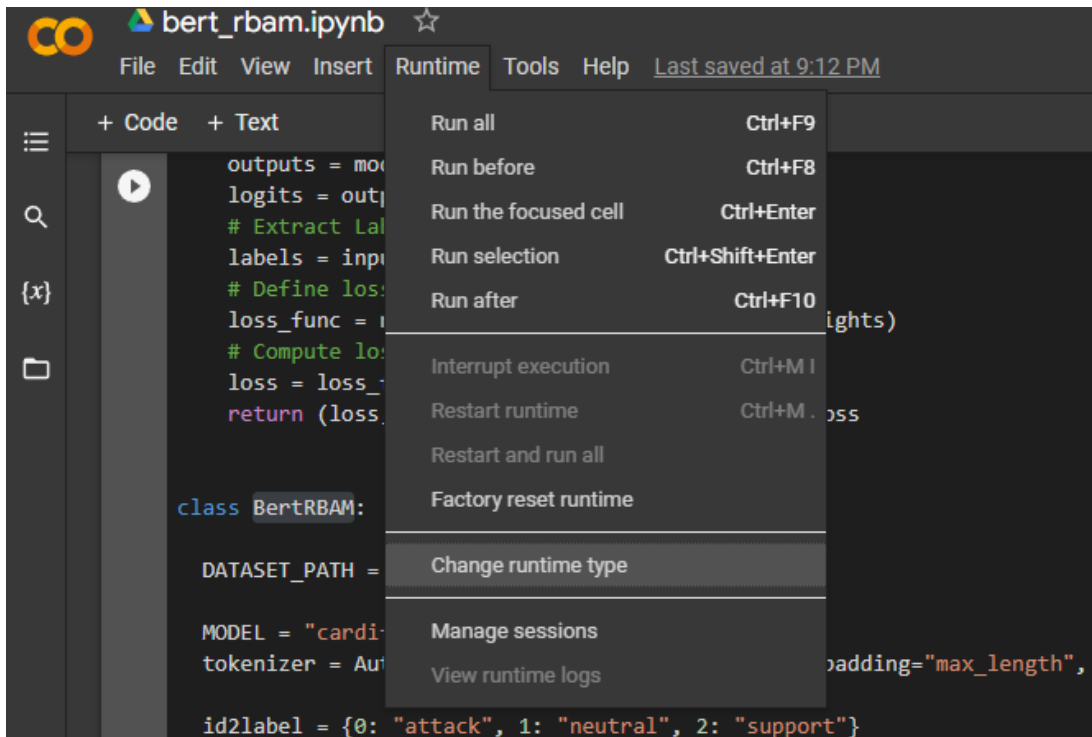


Figure A.5: Google Colab Change Runtime Option

6. Change the “Hardware accelerator” to “GPU” and click “Save”, as shown in Figure A.6.

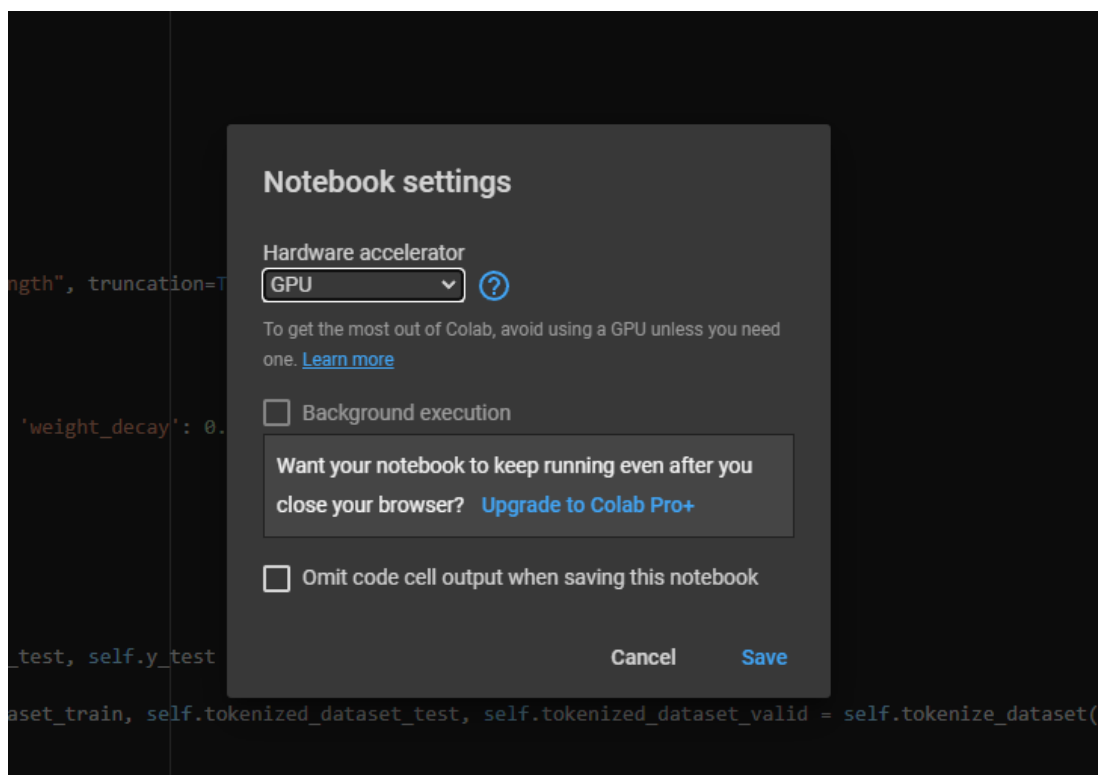


Figure A.6: Google Colab Change GPU Hardware accelerator

7. Once again, in the toolbar, open the tab called “Runtime” by clicking it.
8. Click “Run all” as shown in Figure A.7.

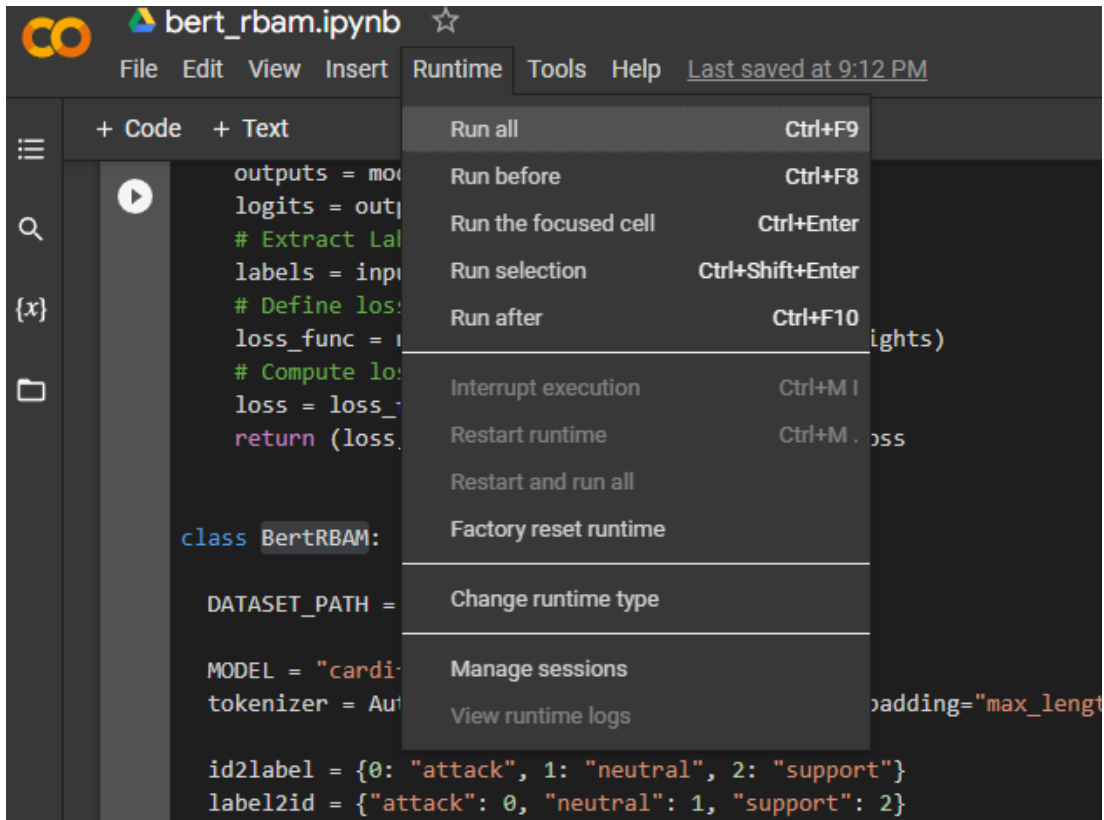


Figure A.7: Google Colab Run All Cells

9. You will soon be prompted with a page that asks for you to authorise the notebook to mount your notebook drive to Google Drive to access the `dataset.csv` file. Authorise the notebook by following the steps.
10. Training will run on the optimal hyperparameters. To change the hyperparameters, modify the `BertRBAM.hyperparameters` dictionary. The training will approximately take 2 hours, after which the output of the training will be printed to the console of the notebook and the output of running the evaluation set on the model.

### A.1.6 Using the Application

This subsection will detail how to use the application. While there are tooltips, instructions and help buttons everywhere on the UI, instructions will still be given. Follow the following steps to use the application:

1. Run the application either using docker or manually as detailed in Section A.1.1.

2. Open the UI by opening the following link in a browser: <http://localhost:3000/>.  
Figure A.8 showcases the page you should see.

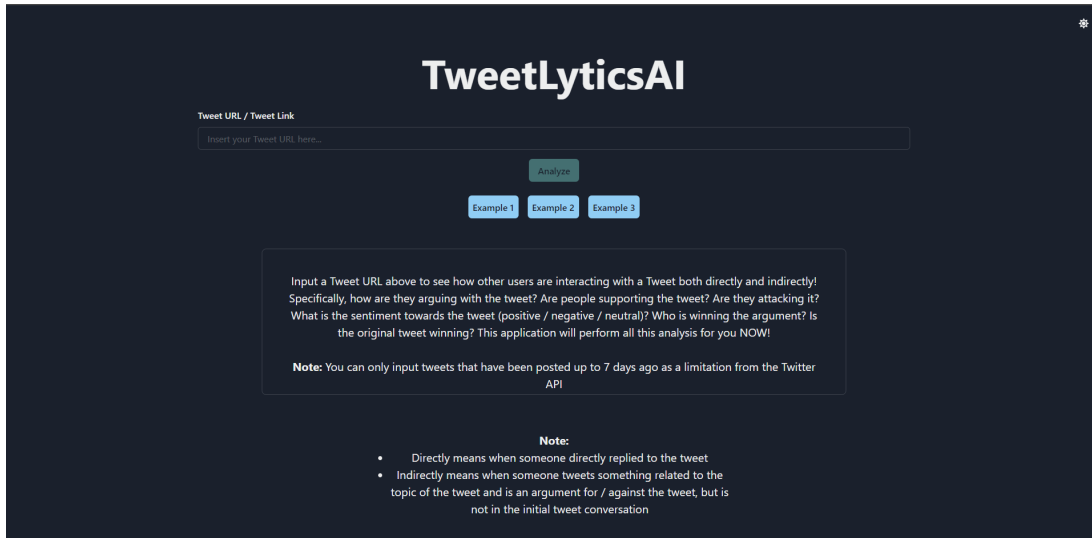


Figure A.8: TweetLyticsAI UI Home Page

3. There are several things you can do on this page:
- (a) Input a tweet URL into the input box that says “Insert your Tweet URL here...” for the system to analyse and click the “Analyse” button. It is highly recommended that you use one of the pre-loaded examples as detailed in the next bullet point. This is because running the system on a new tweet will require making many requests to the Twitter API, taking a prolonged time of a few minutes to finish the computation due to rate limiting and cooldowns. Moreover, the API key can only handle 2 million tweets a month, so be cautious not to go over that limit. Additionally, you can only input tweets that have been posted up to 7 days ago as a limitation from the Twitter API.  
  
You must input a valid tweet URL that follows this structure (note, this is not a real tweet): <https://twitter.com/SomeUser/status/123456789>.
  - (b) Click one of the example buttons, namely “Example 1”, “Example 2” or “Example 3” that are under the input box that says “Insert your Tweet URL here...”. It is highly recommended that you use one of these examples instead of a new tweet.
  - (c) Toggle dark mode and light mode by clicking the moon/sun icon in the top right of the screen.

- (d) Read the instructions on the page
- Once you input a tweet URL or click one of the examples, a loading page will be shown. Please wait patiently. If you used one of the examples, loading will take a very short amount of time. However, if you use a new tweet URL, it can take up to a few minutes, depending on the Twitter API.
  - Once the loading has finished and the computation has been performed, you will be presented with a page similar to that in Figure A.9. There are several tasks/features you can do on this page:

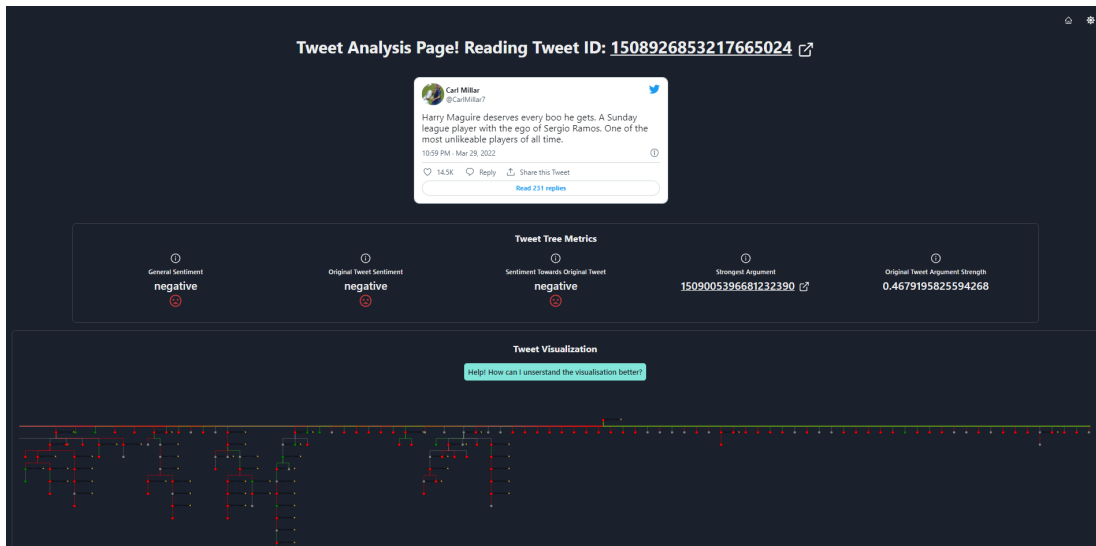


Figure A.9: TweetLyticsAI UI Tweet Tree Visualisation Page

- View the original tweet widget at the top of the page, or open the tweet using the URL hyperlink.
- View the general aggregated metrics from the tweet tree from the section “Tweet Tree Metrics”. This has “General Sentiment”, “Original Tweet Sentiment”, “Sentiment Towards Original Tweet”, “Strongest Argument” and “Original Tweet Argument Strength”. There are tooltips over each metric that you can hover over and will tell you more information about them.
- View the tweet tree visualisation (if your screen is small, you might have to scroll down). For help on how to understand the visualisation, click the “Help! How can I understand the visualisation better?” button.

Essentially, the analysis is modelled as a tree structure, where nodes represent tweets and edges represent replies or relevant and argumentatively related tweets. Red nodes are tweets with negative sentiment, green nodes have positive sentiment and grey nodes have a neutral sentiment. Similarly, red edges denote an argumentative attack relation, green edges are support relations, and grey edges are neutral relations.

- (d) The tweet tree visualisation is scrollable and draggable. This helps for navigation.
- (e) Expand nodes. This can be done by clicking the nodes/circles/tweets. This will open an expanded view that shows several key insights about the tweet, as shown in Figure A.10. It includes information such as the text, sentiment, argumentative type, argument strength, public metrics, the tweet widget, etc.

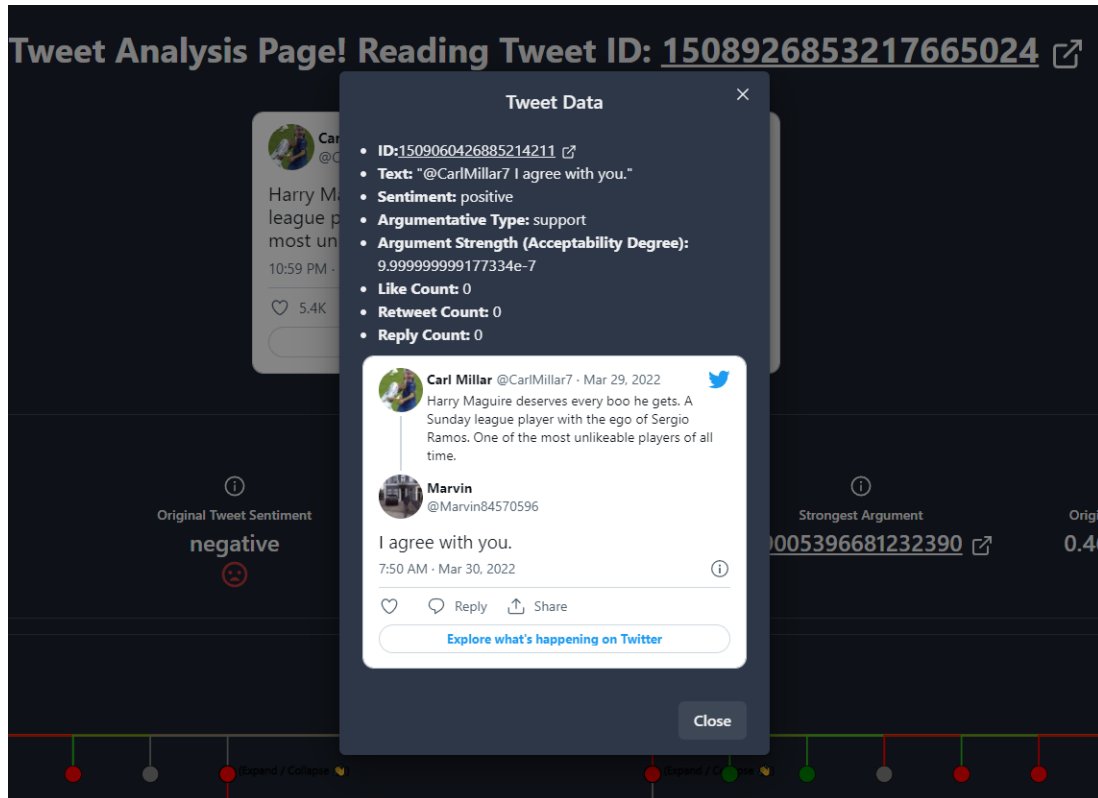


Figure A.10: TweetLyticsAI UI Tweet Node Expanded Modal

- (f) Expand/Collapse child nodes. You can expand and collapse the child nodes for nodes with children to make navigation easier. This can be done by clicking the "Expand / Collapse" button next to nodes with children.

