Twitter Sentiment Analysis with Toxic analysis

J COMPONENT PROJECT REPORT

Submitted by

Guransh Dua (17BCE0029)
Charan Lalchand Soneji (17BCE2196)

in partial fulfillment for the award of the degree of

B. Tech

in

Computer Science and Engineering

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Vellore-632014, Tamil Nadu, India

School of Computer Science and Engineering

March, 2020

Contents

Title: Twitter Sentiment Analysis with Toxic analysis

(i) Abstract

Sentiment analysis is the computational study of people's opinions and sentiments expressed in written language. We can analyze the opinions of the mass in an effective manner using Sentiment Analysis. In this project, we propose to create a Sentiment Analysis model using Google's Natural Language processing API and Python. This model can help Government analysts understand the sentiments of the crowd on sensitive social and political issues and can prove to be extremely helpful. Google Natural Language API is a set of powerful pre-trained models of the Natural Language API let developers work with natural language understanding features including sentiment analysis, entity analysis, entity sentiment analysis, content classification, and syntax analysis. This project also identifies all the different toxic elements and helps filter them out in order to provide convenience of use for the user. Toxic elements refers to usage of racist, abusive language etc. It helps in providing a better user experience.

(ii) Introduction

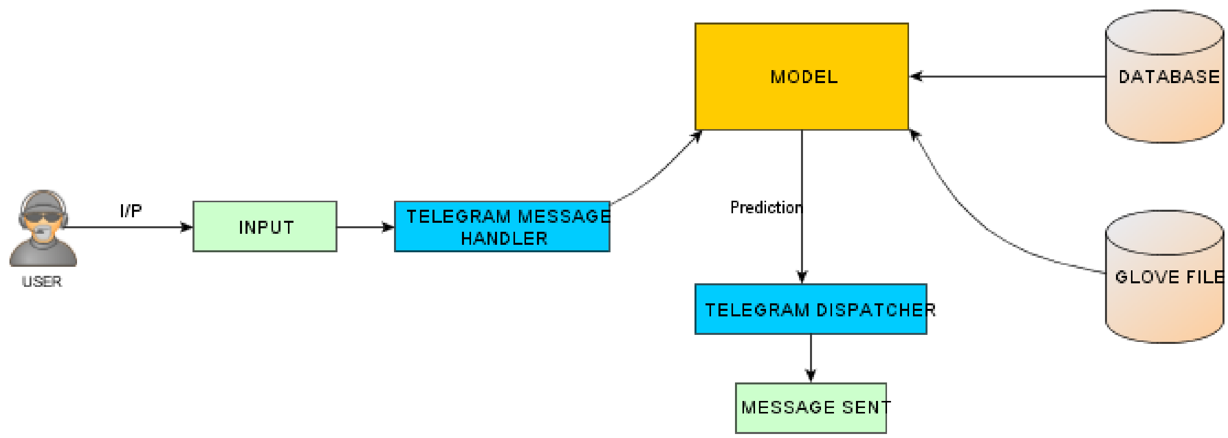
Sentiment Analysis is the process of 'computationally' determining whether a piece of writing is positive, negative or neutral. It's also known as opinion mining, deriving the opinion or attitude of a speaker. Sentiment Analysis is the automated process of analysing text data and sorting it into sentiments positive, negative or neutral. Performing Sentiment Analysis on data from Twitter using machine learning can help companies understand how people are talking about their brand.

With an excess of 321 million dynamic clients, sending a daily average of 500 million Tweets, Twitter enables organizations to contact an expansive crowd and associate with clients without intermediaries. On the drawback, it's harder for brands to rapidly identify negative substance, and on the off chance that it becomes a web sensation you may wind up with a unexpected PR emergency on your hands. This is one reason why social listening — checking discussion and criticism in online networking — has become an urgent procedure in internet based life showcasing. This is where Toxic Analysis comes in. Monitoring Twitter allows companies to understand their audience, keep on top of what's being said about their brand and their competitors, and discover new trends in the industry.

Online reputation is one of the most precious assets for brands. A bad review on social media can be costly to a company if it's not handled properly and swiftly. After all, 80% of consumers get advice on products from social media. Twitter sentiment analysis allows you to keep track of what's being said about your product or service on social media and can help you detect angry customers or negative mentions before they turn into a major crisis. At the same time, Twitter sentiment analysis can provide interesting insights to understand customer feedback.

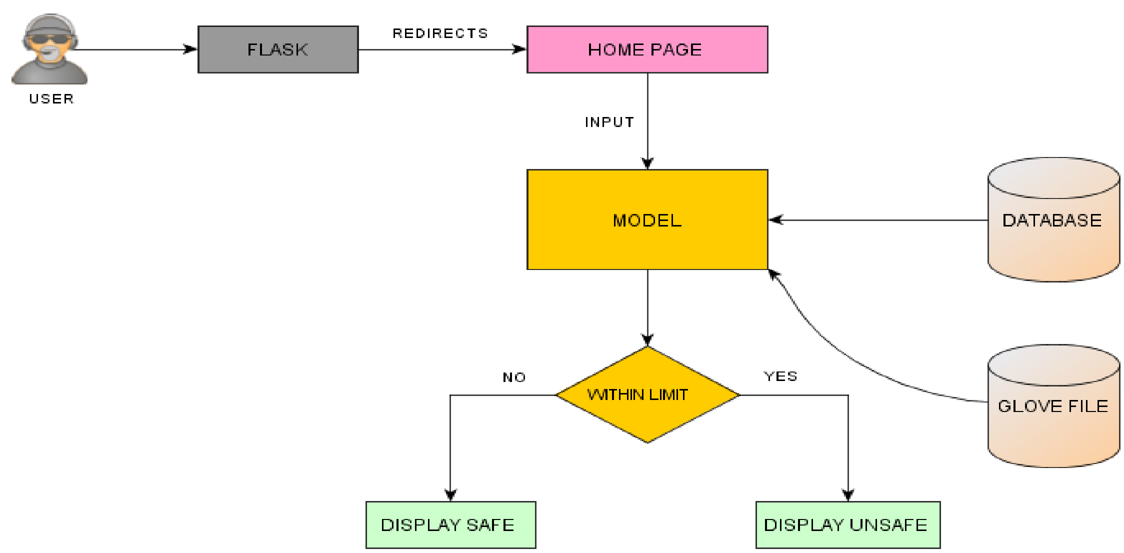
(iii) Architecture diagram

The high-level architecture diagram is mentioned below:



The above diagram represents the routing of information in our given application. Whenever the user makes a request or gives an input in the telegram bot, it is directed to the Telegram Message Handler which then sends or interprets the value from the given Model. From the model, the output is sent to the Telegram Dispatcher which then links to the HTTP portal and displays the output. The blue dialogs represent the Telegram modules. The database represents the dataset which is being used to train the model and the glove file represents a dataset which is used to represent the relation of words in the vector form. The HTTP architecture of the web app is represented below:

The given architecture represents the way in which responses are redirected within the Web App



which has been created whereby FLASK has been used in order to use the python model created and embed it into the Web App. Based on the value predicted by the model, the tweet is classified as safe or unsafe.

(iv) Background study (Related papers and study)

Authors and year (Reference)	*Title (Study)*	*Concept/Theoretical mode/Framework*	*Methodology used/Implementation*	*Relevant Finding*	*Limitations/Future Research/Gaps Identified*
Betty van Aken, Julian Risch, Ralf Krestel, Alexander Loser	Challenges for Toxic Comment Classification	Toxic comment classification has become an active research field with many recently proposed approaches. To this end, they compare different deep learning and shallow approaches on a new, large comment dataset and propose an ensemble that outperforms all individual models. Further, they validate our findings on a second dataset.	Each field uses different definitions for their classification, still similar methods can often be applied to different tasks. In our work we focus on toxic comment detection and show that the same method can effectively be applied to a hate speech detection task.	We showed that the approaches make different errors and can be combined into an ensemble with improved F1-measure. The ensemble especially outperforms when there is high variance within the data and on classes with few examples. Some combinations such as shallow learners with deep neural networks are especially effective.	An important research topic for future work is investigating improved semantic embeddings, which can better distinguish different paradigmatic contexts. For future research, it is essential to know which challenges are already addressed by state-of-the-art classifiers and for which challenges current solutions are still error-prone.
Manav Kohli, Emily Kuehler, John Palowitch	Paying attention to toxic comment online	Deep Learning methods are already being used in order to detect abusive comments made in online forums. Detecting and classifying online abusive language is a non-trivial NLP challenge because online comments	In the given paper, a wide range of RNN models for the task of classifying abusive comments authored by users of Wikipedia. This is a challenging NLP because	Character level models offer a potential way to addressing out of vocabulary problems common in online comments, due to high frequency of slang and lack of credits. Using	A future avenue of research to pursue would be an architecture that attempts to infer word level features from character level input.

		are made in a wide variety of contexts.	the number of positive examples is low.	custom word embeddings saw a marginally low decrease in the word level GRU and LSTM accuracies.	
Kevin Khieu Neha Narwal	Detecting and Classifying Toxic Comments	This paper introduces various deep learning approaches applied to the task of classifiying toxicity in online comments. We study the impact of SVM, LSTM, CNN and MLP methods in combination with word and character-level embeddings, on identifying toxicity in text.	The project focusses on studying the effects of three different kins of neural network models (MLP, STM and CNN) at two levels of granularity- word level and character level.	For the multi-label classification task, we achieved a best performance of 0.927 label accuracy from our LSTM model. LSTM also performs best with regards to sentence accuracy. The toxic models were compensated with weights.	The overall accuracy of the project needs to be improved and the overall classification in toxicity needs to be increased and the scope needs to be widened.
Hafiz Hassaan Saeed, Khurram Shahzad Faisal Kamiran	Overlapping Toxic Sentiment Classification using Deep Neural Architectures	Propose Deep Neural Network (DNN) architectures to classify the overlapping sentiments with high accuracy. Moreover, we show that our proposed classification framework does not require any laborious text pre-processing and is capable of handling text pre-processing.	Our problem lies particularly in the domain of multi-label text classification. A multi-label classification problem is now being addressed. We are given a multi-label training dataset D which contains n pairs of documents and label vectors, where each document	This study draws a comparison among DNN models when there is an overlapping multi-label text classification problem. Based on the empirical results, considering overlapping text classification, we recommend not to spend too much time in data pre-	Recommend the use of focal loss to deal with imbalanced classes. Focal loss alleviated the skewed class problem, though not much significantly, but still it did not exacerbate the skewness problem. Use of multiple model performance metrics can prove decisive in the choosing and rating DNN models. In overlapping multi-label text

			corresponds to a single comment in the dataset.	processing. As a matter of fact, stop words, punctuation, etc. proved to be a vital constituent of data when it comes to the training of the models.	classification, it is observed that the per-label measurement of performance metrics corroborates the better understanding of model performance.
Shuichi Hashida, Keiichi Tamura, Tatsuhiro Sakai	Classifying Tweets using Convolutional Neural Networks with Multi-Channel Distributed Representation	This paper is focused on a state-of-art classification method for short text messages. With the increasing interest in social media, people are posting many short text messages, not only to communicate with other people. Short term memory model. The results showed that the classification performance of the deep learning models was superior to that of the naive Bayes classifier. Moreover, a CNN with multi-channel distributed representation can classify tweets better than a CNN	A topic analysis system based on density-based spatiotemporal Clustering. It shows an overview of the topic analysis system. The system has three main stages: tweet classification, spatiotemporal clustering, and visualization through a Web application. Users set a monitoring, topic such as heavy rain and snow. The tweet classifier classifies tweets according to whether or not their content is related to the monitoring topic. If the tweet classifier achieves a high performance	The proposed model is based on the Kim's model, which utilizes a CNN with distributed representation, a word embedding technique in which words are mapped to vectors in a multi-dimensional space. In Kim's model, text data are converted to a sequence of distributed representations. To enhance the capability of distributed representation, multi-channel distributed representation combines multiple matrices of distributed representation, which	A CNN with multi-channel distributed representation can classify tweets better than a CNN without multi-channel distributed representation. In our future work, we plan to enhance the representation of input data to improve the proposed model furthers.

			level, the effectiveness of the system is improved.	are constructed based on time delay.	

(v) Methodology

We start by collecting real time examples of tweets and then combining them with the dataset retrieved from Kaggle. The data was then preprocessed using NLTK module within python and tokenized. We remove stop words and punctuations from the dataset and optionally perform stemming and lemmatization. The model is then trained and created using the dataset which is retrieved and made. The glove dataset and the trained dataset are used to train the model. The glove dataset is used because of the embedding matrix that we are using in our model trainer. This is further explained below.

We then made a Flask based web app which can link our python trained module to the web app while taking the input from our telegram bot. The telegram bot is linked to the python web app using the HTTPS key obtained and the updater and massager modules inside the Python Telegram library. The key activities on the web app basically show the result output obtained from the Telegram bot which is then routed or accessed using the HTTPS token key and diaplyed with a pleasing UI.

(vi) Proposed model

In this section, we shall explain the different modules of this toxic analyzer. The given project is split up into several shorter modules which are explained briefly below:

1. Creation of model

This module discusses the creation of the model which is mainly involved in predicting the toxicity in the tweets that are being inputted to the user or being scrapped of twitter using tweepy. Some of the main python modules/libraries being used are numpy, pandas, tensorflow, nltk and keras.

a) Identifying dataset: This is one of the most important steps towards creation of a model. For the creation of our model, we used a toxic dataset from Kaggle which consists of a number of toxic statements made by users and a column predicting the class (i.e. intensity of toxicity or class of toxicity such as racism, sexism, etc.) and combined it with a few more results that we felt would help optimize the dataset.

b) Reading and pre-processing of data including tokenizing of data: While optimizing the dataset or accessing raw data, the data needs to be cleaned and then optimized for efficient usage to prevent anomalies or incorrect values to be printed during output. Hence, the entire Training dataset was scanned and all the missing values were manually replaced by us. A lot of the sentences were added by the members of the project in order to give an idea of real time tweets. Some of the real time values that we have added into our dataset are mentioned below.After this, we used the inbuilt NLTK module in python in order to read the data from the given CSV file/Dataset and create tokens of all the sentences which would further be provided to the embedding matrix.

c) Creation of model.py using embedding matrix: For the creation of our model, we have made use of an embedding matrix since it is a linear mapping from the original space to a real-valued space where entities can have meaningful relationships.

d) Saving the model to a model.h5 file

This is the last phase of creation whereby we use the TensorFlow backend in order to create the model whereby the models and its weights are returned to a file with an extension of .h5.

On compiling all of the function as mentioned above and running them with reference to specific functions respectively in Google collab, the output returns the trained model.

2. Linking with Telegram Bot

Telegram can be used to create a number of bots which may be made responsive by linking them to models as we shall be doing in this given module. The model.h5 file is linked to our bot in the following steps.

a) Creation of Bot in BotFather: For the creation of a bot, we use a predefined bot named Botfather in Telegram whereby we can create our own custom bots and rename them based on our requirement. The following snippet shows the creation of our bot:

The bot created in Telegram has been given:

- Name of bot: ToxicAnalysisBot
- HTTPS token of bot: 1076597208:AAH8whDdXKhF9xgSMj1P50EzZ-3JATLuaLs

b) Linking of bot with Python: This step helps us in accessing the model thought our bot created in Telegram. For the given process, “Python-telegram bot” library is made use of and the updater.dispatcher is used to dispatch our update to the bot.

A few snippets of the code which are linked to the connection of the Telegram bot are mentioned below:

In the given code, the Updater function has the HTTP access key of the given bot and Messagehandler is used to handle incoming messages. It also uses a callback whenever it gets a message. This callback is handled by the UDF named my_func. My_func receives the message sent by the user. It then checks if the message is a text message or something else and predict various parameters like toxicity.

3. Creating a front end and developing a UI for the result

The UI for this project is made using HTML, CSS and Bootstrap Framework. A simple and responsive UI is made so that the user, at no point feels lost in the UI. We have integrated this UI into a Jinja template. Jinja is a modern and designer-friendly templating language for Python, modelled after Django's templates. It is fast, widely used and secure with the optional sandboxed template execution environment. The reason for using Jinja here is that it provides a great amount of flexibility with respect to making changes in the User Interface during runtime. Here are some of the code snippets of the Front-End design.

In the screenshots above, we can see the main components of both the screens. For Home Page, the code for the form where the user is required to submit their text that needs to be processed is displayed. Whereas in the Results page, the buttons and the formatting that is displayed as the result to the user is shown.

The home page welcomes the user with a text input form. The user simply has to enter the text he wishes to analyses and click check. Post that, the user is redirected to the result page. Which looks like this one if all the parameters are below a satisfactory level:

(vii) Results and Discussion

A few snippets of the dataset used in the project are mentioned below:

	A	B	C	D	E	F	G	H
1	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
2	0000997932d777bf	Explanation	0	0	0	0	0	0
3	000103f0d9cfb60f	D'aww! He matches this background colour I'm seemingly stuck with. Thanks. (talk) 21:51, January 11, 2016 (UTC)	0	0	0	0	0	0
4	000113f07ec002fd	Hey man, I'm really not trying to edit war. It's just that this guy is constantly removing relevant information and talking to me	0	0	0	0	0	0
5	0001b41b1c6bb37e	"	0	0	0	0	0	0
6	0001d958c54c6e35	You, sir, are my hero. Any chance you remember what page that's on?	0	0	0	0	0	0
7	00025465d4725e87	"	0	0	0	0	0	0
8	0002bcb3da6cb337	COCKSUCKER BEFORE YOU PISS AROUND ON MY WORK	1	1	1	0	1	0
9	00031b1e95af7921	Your vandalism to the Matt Shirvington article has been reverted. Please don't do it again, or you will be banned.	0	0	0	0	0	0
10	00037261f536c51d	Sorry if the word 'nonsense' was offensive to you. Anyway, I'm not intending to write anything in the article(wow they woul	0	0	0	0	0	0
11	00040093b2687caa	alignment on this subject and which are contrary to those of DuLithgow	0	0	0	0	0	0
12	0005300084f90edc	"	0	0	0	0	0	0
13	00054a5e18b50dd4	bbq	0	0	0	0	0	0
14	0005c987bdfc9d4b	Hey... what is it..	1	0	0	0	0	0
15	0006f16e4e9f292e	Before you start throwing accusations and warnings at me, lets review the edit itself-making ad hominem attacks isn't going	0	0	0	0	0	0
16	00070ef96486d6f9	Oh, and the girl above started her arguments with me. She stuck her nose where it doesn't belong. I believe the argument w	0	0	0	0	0	0
17	00078f8ce7eb276d	"	0	0	0	0	0	0
18	0007e25b2121310b	Bye!	1	0	0	0	0	0
19	000897889268bc93	REDIRECT Talk:Voydan Pop Georgiev- Chernodrinski	0	0	0	0	0	0
20	0009801bd85e5806	The Mitsurugi point made no sense - why not argue to include Hindi on Ryo Sakazaki's page to include more information?	0	0	0	0	0	0
21	0009eaea3325de8c	Don't mean to bother you	0	0	0	0	0	0
22	000b08c464718505	"	0	0	0	0	0	0
23	000bfd0867774845	"	0	0	0	0	0	0
24	000c0dfd995809fa	"	0	0	0	0	0	0
25	000c6a3f0cd3ba8e	"	0	0	0	0	0	0
26	000cfee90f50d471	"	0	0	0	0	0	0

	A	B	C	D	E	F	G	H
103	003fa0c68deca750	Check the following websites:	0	0	0	0	0	0
104	0040017ef6277334	i can't believe no one has already put up this page Dilbert's Desktop Games so I did	0	0	0	0	0	0
105	004176f28a17bf45	"	0	0	0	0	0	0
106	0044cf18cc2655b3	What page shoudld there be for important characters that DON'T reoccur?	0	0	0	0	0	0
107	00472b8e2d38d1ea	A pair of jew-hating weiner nazi schmucks.	1	0	1	0	1	1
108	00480b6e1f19601b	I tend to think that when the list is longer than the rest of the article, there's a problem. Either the history and characteristic	0	0	0	0	0	0
109	0048de0c9422f64f	"	0	0	0	0	0	0
110	0048e4ed8a0af433	I'm not vandalizing	0	0	0	0	0	0
111	004a23742282fee4	Welcome to Wikipedia ! [bla] Discover Ekopedia, the practical encyclopedia about alternative life techniques.	0	0	0	0	0	0
112	004a789c03eda830	Including some appropriate mention of the Solomon article is not without some level of support .	0	0	0	0	0	0
113	004af8a71399a4dc	"	0	0	0	0	0	0
114	004b073d5b456b15	"	0	0	0	0	0	0
115	004b103182fb1eab	Thanks, Josette. I enjoyed meeting you, too. I was shocked by the decision, which does not begin to reflect consensus. Does	0	0	0	0	0	0
116	004b975fabbbffa9	Paleontologists agree that organic remains must be buried quickly so they can be preserved long enough to be come	0	0	0	0	0	0
117	004b97c80705a548	Also I think Vegetable Basket needs it's own Wikipedia page.	0	0	0	0	0	0
118	004d07d94cb92e35	Bigfoot Reference	0	0	0	0	0	0
119	004d912a00f79c89	Also see this if you cant trust Murkoth Ramunni	0	0	0	0	0	0
120	004de318396bbf8b	"	0	0	0	0	0	0
121	004e59705689a5ed	"	0	0	0	0	0	0
122	004f5608984d99f1	"	0	0	0	0	0	0
123	004f6dbe69f3545d	Conformity as healthy	0	0	0	0	0	0
124	004f981460421bdf	Hi, I am new to wikipedia. Read up on pandemics today. I am impressed by the quality of information, to my knowledge not	0	0	0	0	0	0
125	004fb9f655230909	Should say something about his views as an educationalist and socialist political commentator.	0	0	0	0	0	0
126	004fd4fb5c47c29f	"==Sandbox==	0	0	0	0	0	0
127	0050cb3bc226f94e	heh, it's a bit of a copy of Wikipedia:WikiProject Professional wrestling, but I thought it look a bit tidy and better that way.	0	0	0	0	0	0
128	00510c3d06745849	Ahh, Hello Witzeman	0	0	0	0	0	0
129	00521847b93eba3b	"	0	0	0	0	0	0
130	0052a7e684beeb1a	". (On Dec 14, 2006, a NIST scientist said ""...the collapse of the towers were not of any magnitude that was seismically signif	0	0	0	0	0	0
131	005306a4c109dab3	The statement drawn from Watchtower literature, honestly, not clandescently, not with hypocrisy, is drawn from	0	0	0	0	0	0
132	00537730daf8c5f1	Sorry about that. I had checked, but had only come up with Dulas Bay. Somehow I had missed the Dulas disambiguation pag	0	0	0	0	0	0
133	0053978373606ba4	TCM	0	0	0	0	0	0
134	0053bab79133c0fc	Well, it still needs expansion in areas, citations in others, more images (was going to get a pic of the Stanley Cup banner toni	0	0	0	0	0	0
135	00548d029d7f4783	A redirect somewhere couldn't hurt, though I still don't think any o the candidates are ideal. But then if he passes the notabi	0	0	0	0	0	0
136	00548f16a392d8ed	Meivazhi	0	0	0	0	0	0
137	0055208c660f7894	Though this is certainly a small article, but it is not so inconsequential as to warrant subsuming it within the Flame Trees	0	0	0	0	0	0
138	00562e78e0b34102	Yeah, let's merge the content. (Not sure if Devil's Canyon is the same type of codename.)	0	0	0	0	0	0
139	0057b7710cb5ebb2	"	0	0	0	0	0	0

The given snippet is used to get the data and extract in a given format

```
def get_data(train, test, max_features = 20000, maxlen = 50):
    list_sentences_train = train["comment_text"].fillna("_na_").values
    list_sentences_test = test["comment_text"].fillna("_na_").values
    list_classes = ["toxic", "severe_toxic", "obscene", "threat", "insult", "identity_hate"]
    y = train[list_classes].values

    tokenizer = Tokenizer(num_words=max_features)
    tokenizer.fit_on_texts(list(list_sentences_train))
    list_tokenized_train = tokenizer.texts_to_sequences(list_sentences_train)
    list_tokenized_test = tokenizer.texts_to_sequences(list_sentences_test)

    X_t = pad_sequences(list_tokenized_train, maxlen=maxlen)
    X_te = pad_sequences(list_tokenized_test, maxlen=maxlen)

    return X_t,X_te,y,tokenizer
```

NOTE: For the given project, we made use of Google Collab in order to run our code because we encountered several errors during the execution due to insufficient graphics on our personal laptops.

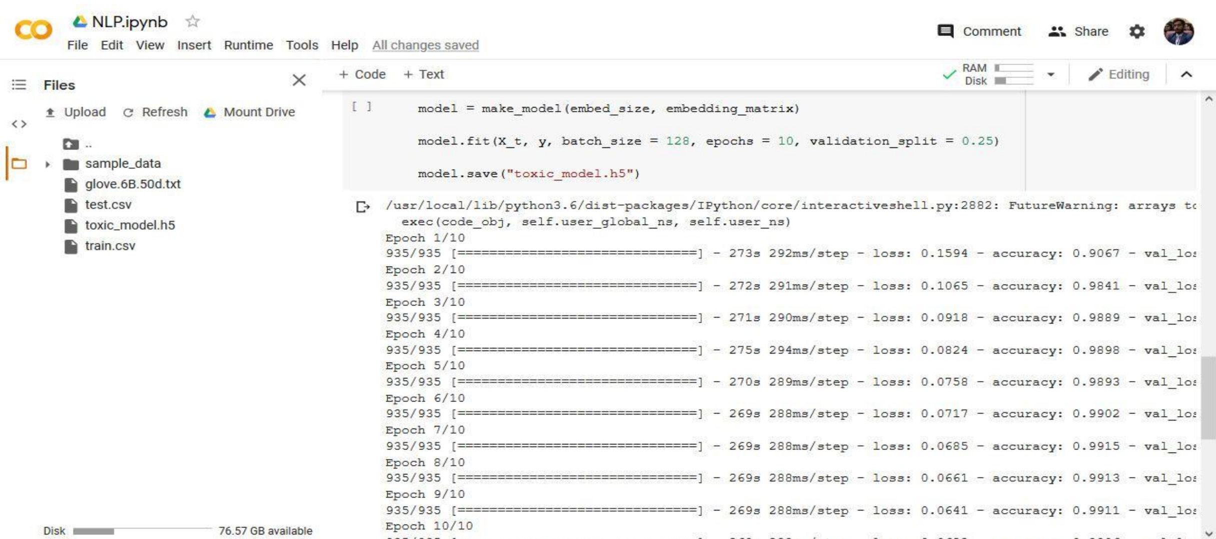
The code snippets for the formation of the embedding matrix is mentioned below:

```
def get_embedding_matrix(EMBEDDING_FILE, embed_size, tokenizer, max_features = 20000):
    embeddings_index = dict(get_coefs(*o.strip().split()) for o in open(EMBEDDING_FILE,'rb'))
    all_embs = np.stack(embeddings_index.values())
    emb_mean,emb_std = all_embs.mean(), all_embs.std()

    word_index = tokenizer.word_index
    nb_words = min(max_features, len(word_index))
    embedding_matrix = np.random.normal(emb_mean, emb_std, (nb_words, embed_size))

    for word, i in word_index.items():
        if i >= max_features:
            continue
        embedding_vector = embeddings_index.get(word)

    if embedding_vector is not None:
        embedding_matrix[i] = embedding_vector

    return embedding_matrix
```

The snippet of the model being trained is shown in the code below:

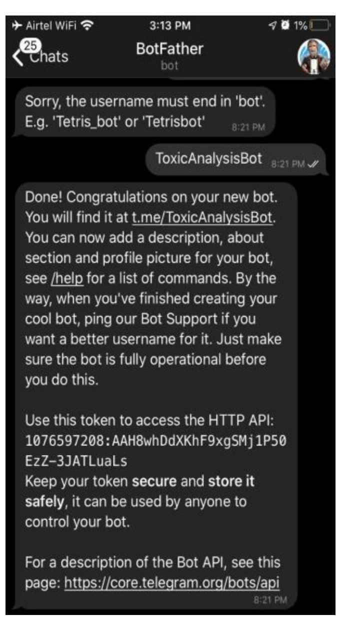
```
def make_model(embed_size, embedding_matrix, max_features = 20000, maxlen = 50):
    inp = Input(shape=(maxlen,))
    x = Embedding(max_features, embed_size, weights=[embedding_matrix])(inp)
    x = SpatialDropout1D(0.2)(x)
    x = Bidirectional(GRU(128, return_sequences = True, activation = "tanh",
                      recurrent_activation = "sigmoid", use_bias = "True", reset_after = "True", unroll = "False"))(x)
    # For CuDNN implementation with tf2
    x = Dropout(0.2)(x)
    x = Conv1D(64, kernel_size = 3, padding = "valid", kernel_initializer = "glorot_uniform")(x)
    avg_pool = GlobalAveragePooling1D()(x)
    max_pool = GlobalMaxPooling1D()(x)
    x = concatenate([avg_pool, max_pool])
    preds = Dense(6, activation="sigmoid")(x)

    model = Model(inp, preds)
    model.compile(loss='binary_crossentropy',optimizer=Adam(lr=1e-4),metrics=['accuracy'])

    return model
```

The snippet of the output is mentioned below:



The respective accuracies of the model at each step are shown above.

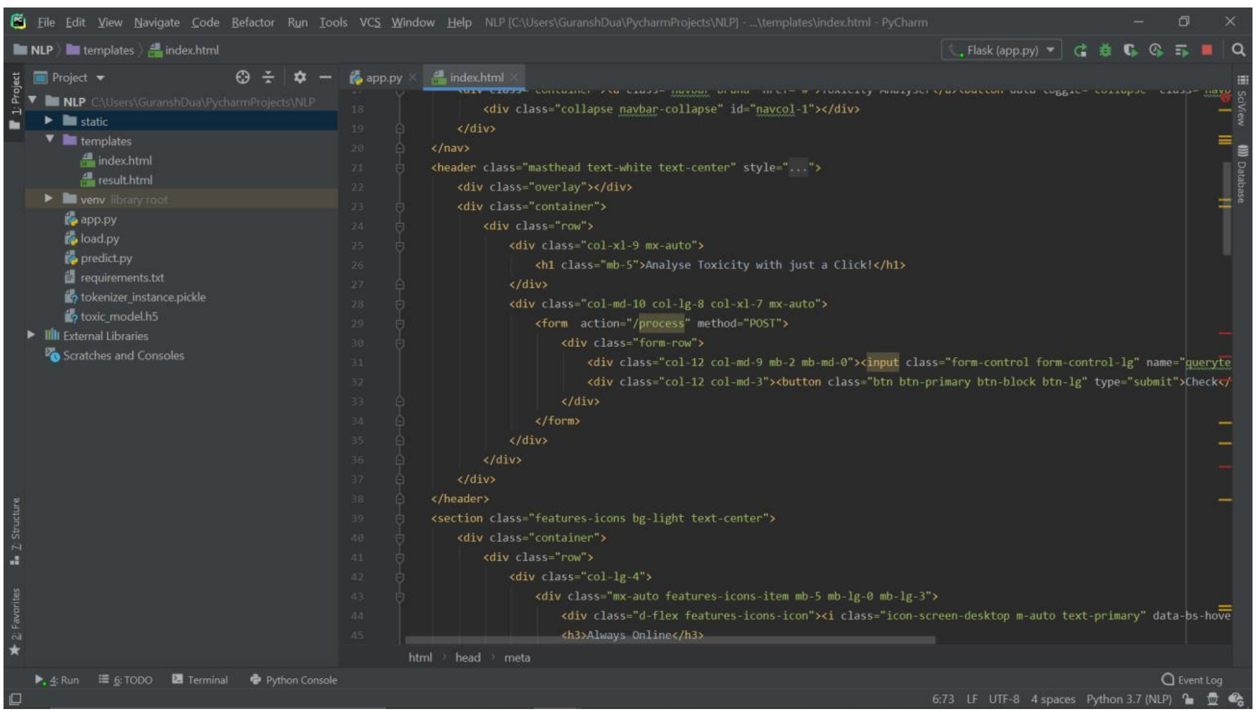
Creation of the bot in BOTFATHER is represented below:



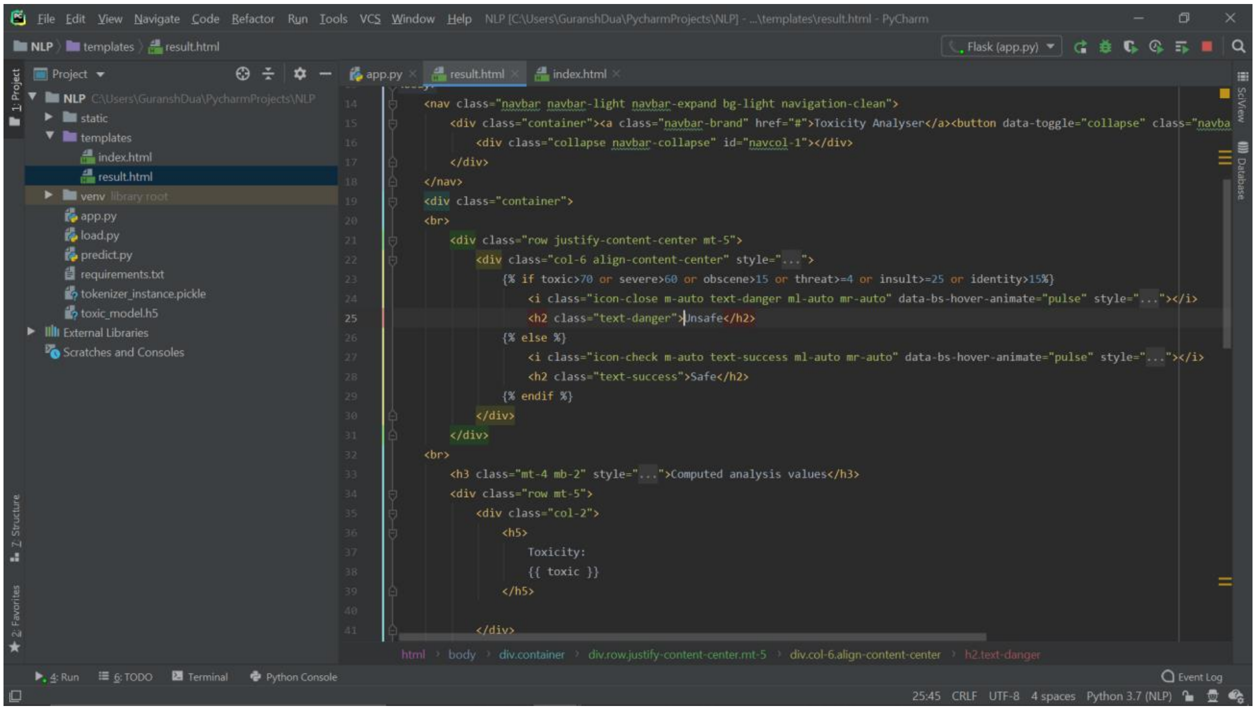
The bot is then linked with python using the following code:

```
updater = Updater('1076597208:AAH8whDdXKhF9xgSMj1P50EzZ-3JATLuaLs')
    dp = updater.dispatcher
    dp.add_handler(MessageHandler(Filters.all, my_func))
    updater.start_polling()
    updater.idle()

def my_func(bot, update):
    if not update.effective_message.text:
            update.effective_message.reply_text(text = "Cannot handle given format, getting aware now")
    else:
        msg = update.effective_message.text
        update.effective_message.reply_text(text = prediction(msg))
```

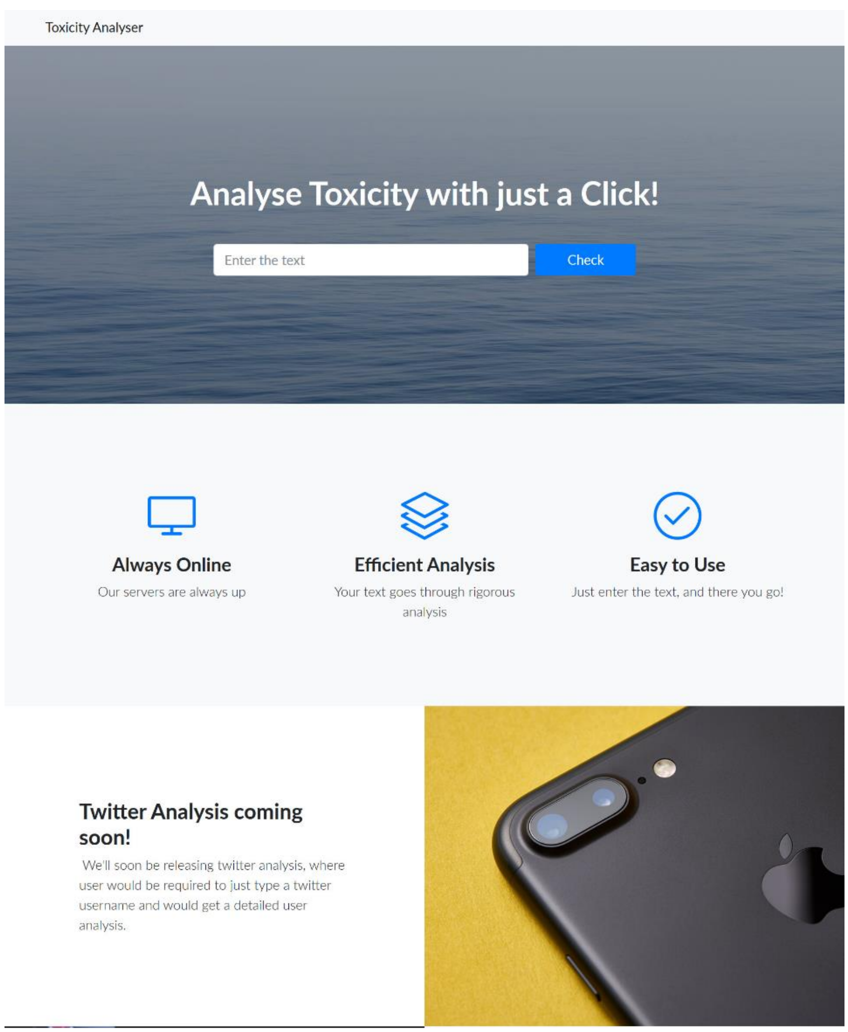
The respective code for the UI is shown below:



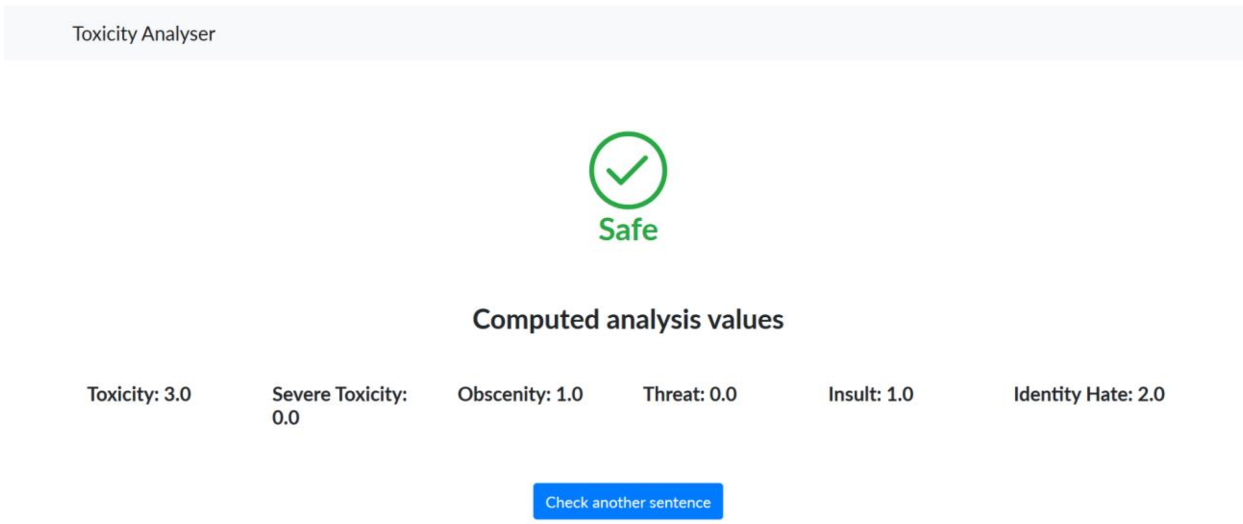
Home Screen



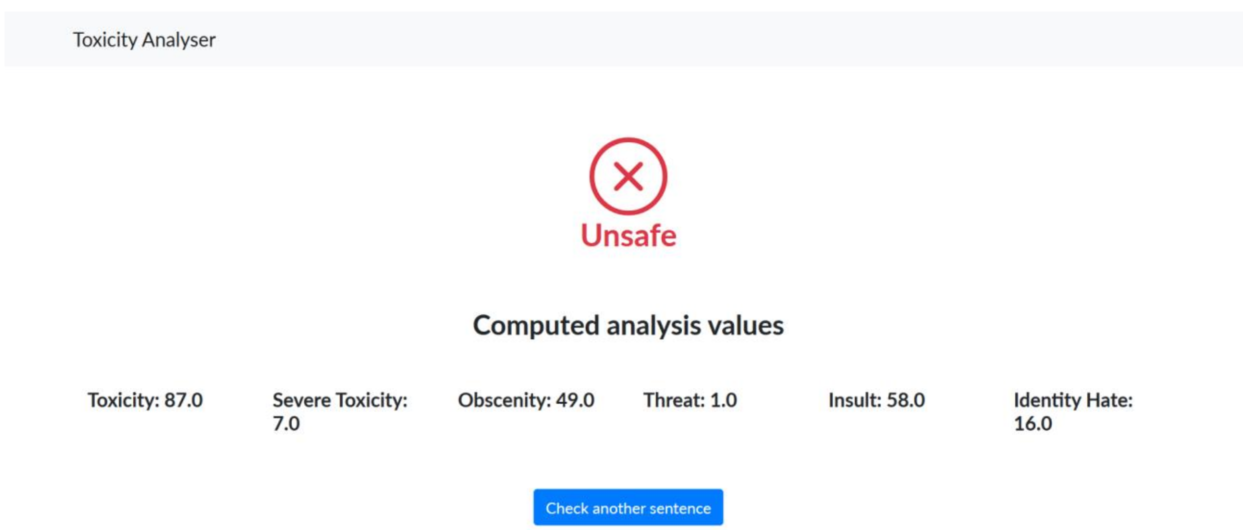
Results Page



Home Page

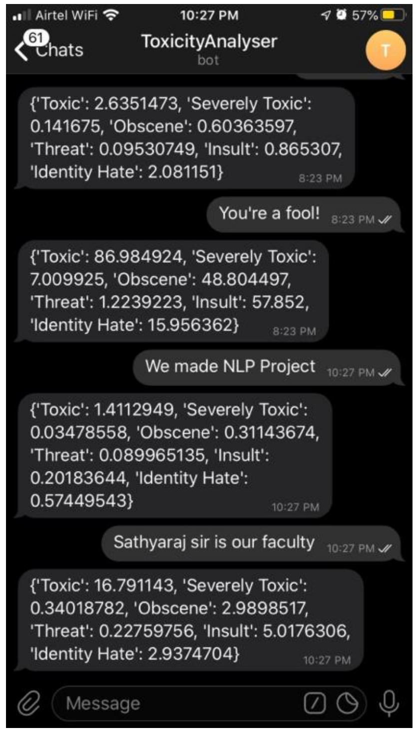


When the data is within the limit, this page is printed.



When the data is notwithin the limit, this page is printed.

Sample Output



(viii) Conclusion

It may be seen that the above model was trained and an overall accuracy of 97% was obtained for all the tweets obtained. We have also classified the data based on whether it is toxic, severely toxic, sexist, racist, obscene etc. and we have successfully linked the telegram bot with this model and linked it to a front end web app which can be used by users in order to link their Twitter and filter out the toxic comments or analyze the toxicity of statements or can also be used to check the toxicity of a tweet being made which shall give the result in the bo as well as a front end web-app.

(ix) References

[1] Betty van Aken, Julian Risch, Ralf Krestel, Alexander Löser, 2018 Sep. Challenges for Toxic Comment Classification: An In-Depth Error Analysis. In Sep 2018 Cornell University Journal.

[2] Manav Kohli, Emily Kuehler, John Palowitch, 2019 Feb. Paying attention to toxic comment online. In Feb 2019 Stanford Journal.

[3] Kevin Khieu, Neha Narwal. Detecting and Classifying Toxic Comments. In Stanford Journal.

[4] Saeed, Hafiz Hassaan & Shahzad, Khurram & Kamiran, Faisal. (2018). Overlapping Toxic Sentiment Classification Using Deep Neural Architectures. 1361-1366. 10.1109/ICDMW.2018.00193.

[5] Hashida, Shuichi & Tamura, Keiichi & Sakai, Tatsuhiro. (2018). Classifying Sightseeing Tweets Using Convolutional Neural Networks with Multi-channel Distributed Representation. 178-183. 10.1109/SMC.2018.00041.

[6] Fahim Mohammad (2018). Is preprocessing of text really worth your time for toxic commentclassification? In Int'l Conf. Artificial Intelligence ICAI'18 |447

[7] Saurabh Srivastava, Prerna Khurana, Vartika Tewar (2018) Identifying Aggression and Toxicity in Comments using Capsule Network. Proceedings of the First Workshop on Trolling, Aggression and Cyberbullying, pages 98–105Santa Fe, USA, August 25, 2018