NLP ASSIGNMENT-2.

Prathap Karunakaran (M20MA007)

Aarthiee UK(M20MA001)

3) Using the same data as your Assignment 1, Develop an

a) Bi-LSTM based sentiment analysis model using (a) TF-IDF embeddings,

(b) word2vec embeddings (c) glove embeddings.

b) BERT based sentiment analysis model.

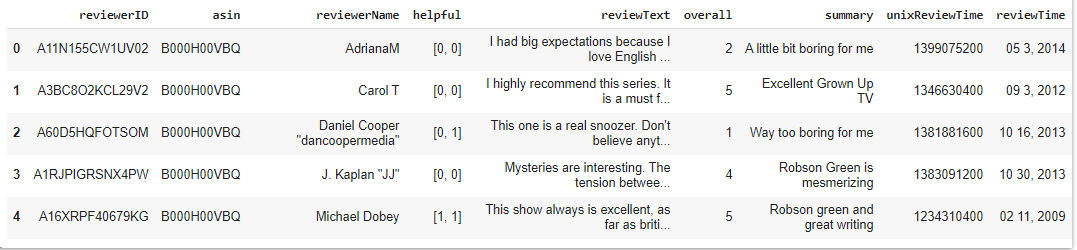
c) Compare the performance on the test set among these models. (Bi-LSTM

(with tf-idf, word2vec, glove), BERT)

d) Traditional ML Models developed in assignment 1.

**Dataset – Amazon Product Review Dataset.**

Displaying the First 5 rows in the dataset.



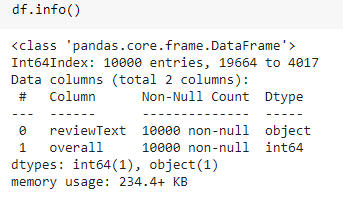
Total rows in the dataset – 37126.



We have chosen 10,000 rows in the Amazon Product Reviews dataset.



We are considering two columns in the dataset: ‘reviewText’ and ‘overall’.

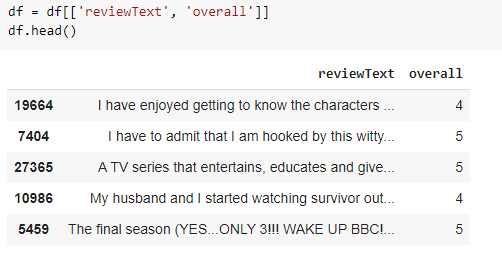


**CONDITION CHECK:**

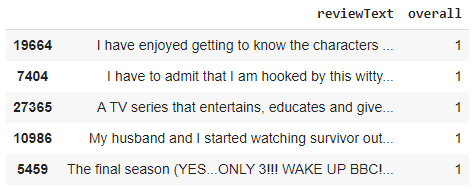
If the ‘overall’ column in the dataset is greater than 2 , ‘overall’ value becomes 1.

Otherwise the ‘overall’ column becomes 0.

Before checking the ‘overall’ condition with ‘reviewText’:



After the ‘overall’ condition check,



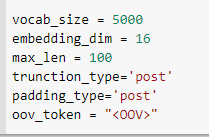
**Bi-LSTM based sentiment analysis model using Glove embeddings:**

**GloVe: Global Vectors for Word Representation.**

Glove is based on matrix factorization techniques on the word-context matrix. It first constructs a large matrix of (words x context) co-occurrence information, i.e. for each “word” (the rows), you count how frequently we see this word in some “context” (the columns) in a large corpus.  The number of “contexts” is of course large, since it is essentially combinatorial in size. So then we factorize this matrix to yield a lower-dimensional (word x features) matrix, where each row now yields a vector representation for the corresponding word. In general, this is done by minimizing a “reconstruction loss”. This loss tries to find the lower-dimensional representations which can explain most of the variance in the high-dimensional data.

**Train-Test-Split : 80:20 split**

**Tokenizing the words:**

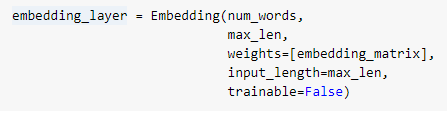


We have used pre-trained word embeddings to create own embedding layer.

It contains 40000 word vectors.



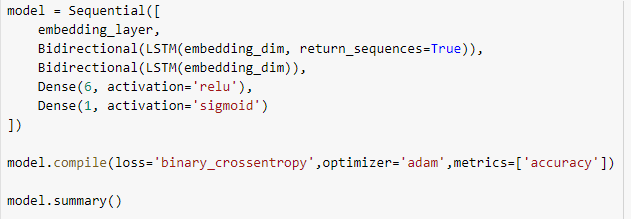
**Embedding Layer:**



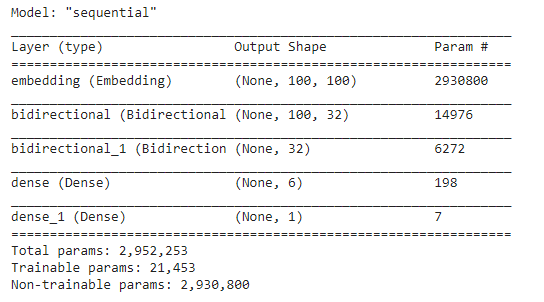
**Build the Model:**

Building the model using embedding layer and Bidirectional LSTM for Glove embeddings.

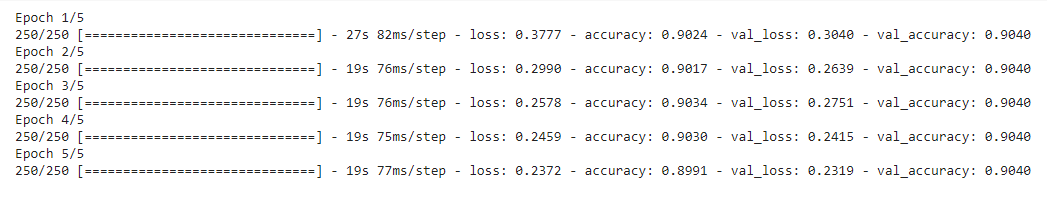
Using the optimizer as Adam Optimizer and loss as Binary cross-entropy loss.



**SUMMARY OF THE MODEL:**

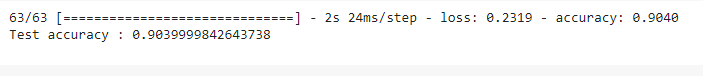


**Training the model for 5 epochs.**



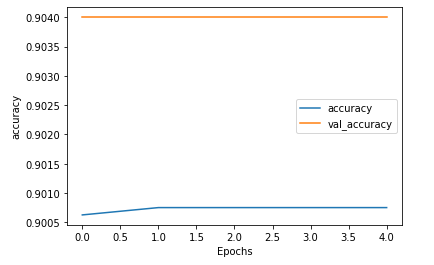
**Predictions on test set:**

Accuracy on Testset for Glove Model : 0.90399

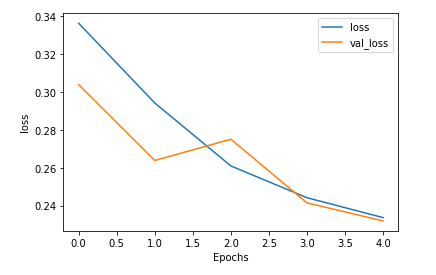


**Plotting the Graph:**

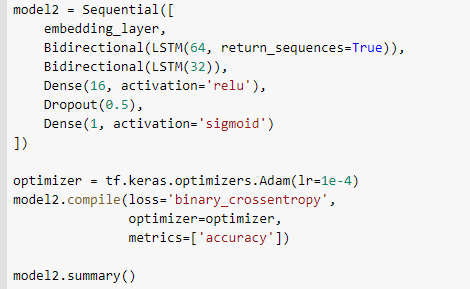
**Accuracy vs. Epochs.**



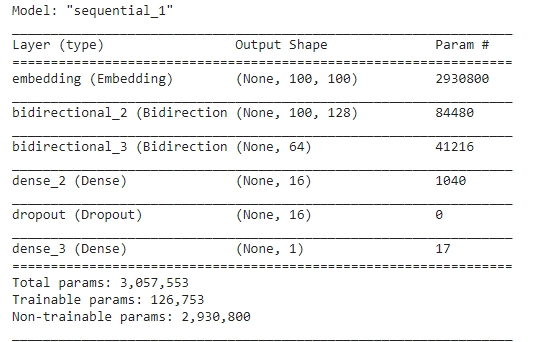
**Loss vs. Epochs.**



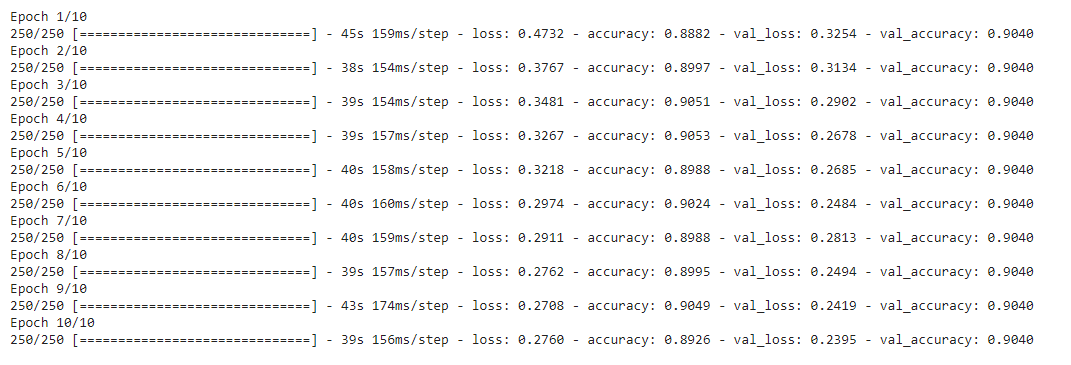
Adding the dropout layer for decrease overfitting and we increase number of epochs to 10.



Summary of the Model is:



We are running for **10 Epochs.**



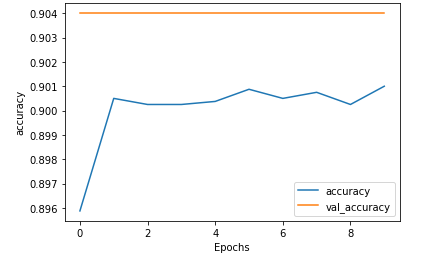
**Predictions on a test set:**

Test Accuracy is 0.9039

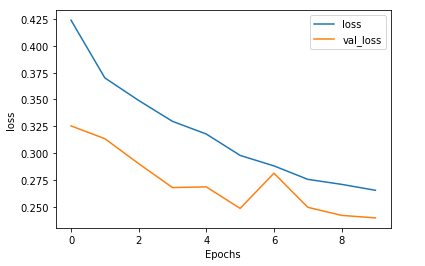


**Plotting the Graph:**

**Accuracy vs. Epochs.**



**Loss vs. Epochs.**



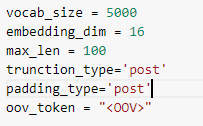
**Bi-LSTM based sentiment analysis model using Word-to-vector embeddings:**

Word2Vec is a feed forward neural network based model to find word embeddings. There are two models that are commonly used to train these embeddings: The skip-gram and the CBOW model.

We are using pre-trained models in Gensim for word-to-vector embedding.

**Train-Test-Split : 80:20 split**

**Tokenizing the words:**



We have used pre-trained word embeddings to create own embedding layer.

It contains 47375 word vectors.

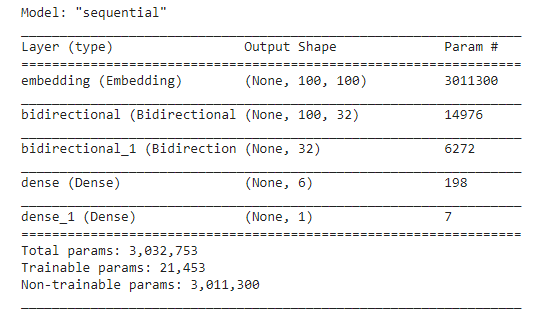


**Build the Model:**

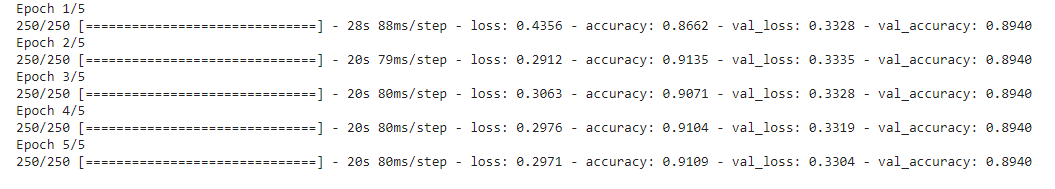
Building the model using embedding layer and Bidirectional LSTM for word-to-vec embeddings.

Using the optimizer as Adam Optimizer and loss as Binary cross-entropy loss.

**SUMMARY OF THE MODEL:**



**Training the model for 5 epochs.**



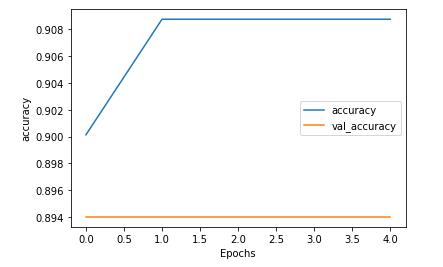
**Predictions on test set:**

Test accuracy : 0.893

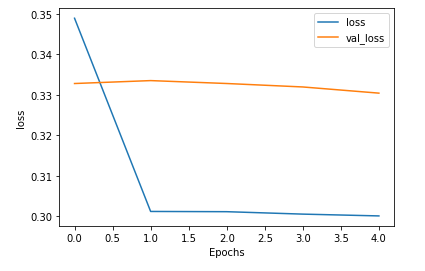


**Plotting the Graph:**

**Accuracy vs. Epochs.**

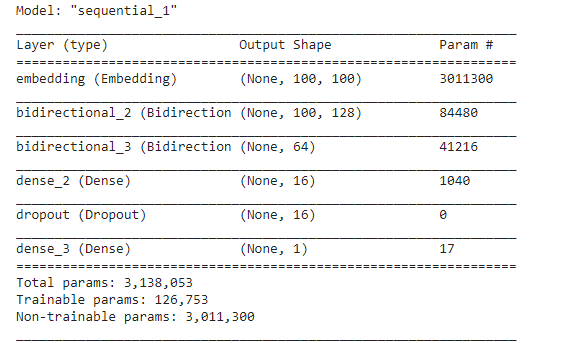


**Loss vs. Epochs.**

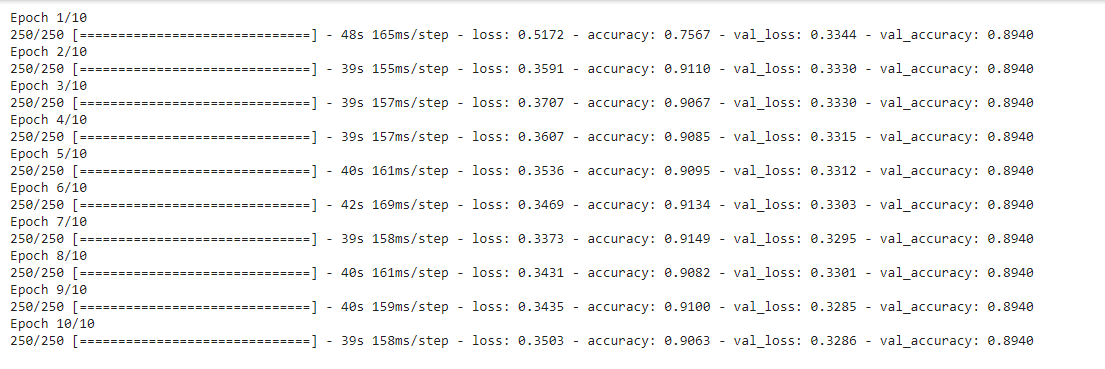


Adding the dropout layer for decrease overfitting and we increase number of epochs to 10.

Summary of the Model is:



We are running for **10 Epochs.**



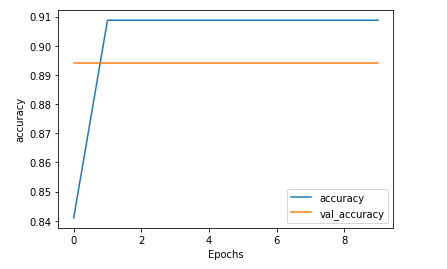
**Predictions on a test set:**

Accuracy is 0.8939

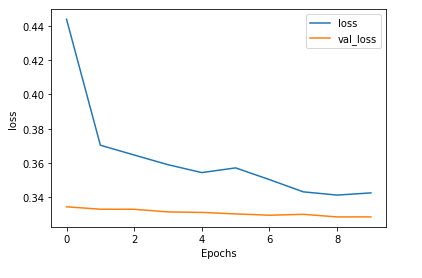


**Plotting the Graph:**

**Accuracy vs. Epochs.**



**Loss vs. Epochs.**



**Bi-LSTM based sentiment analysis model using TF-IDF embeddings:**

**Term Frequency** — Inverse Document Frequency (**TF**-**IDF**) is another more common tool in NLP for converting a list of text documents to a matrix representation.**TF**-**IDF** are sparse **vectors** where the number of non-zero values in the vector is equal to the number of unique **words** in the document.

MODEL SUMMARY:

Model summary for TF-IDF Embedding.

Model: "sequential\_18"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

bidirectional\_36 (Bidirectio (None, 100, 32) 2304

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

bidirectional\_37 (Bidirectio (None, 32) 6272

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_38 (Dense) (None, 6) 198

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_39 (Dense) (None, 1) 7

=================================================================

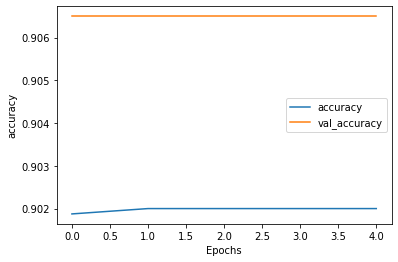
Total params: 8,781

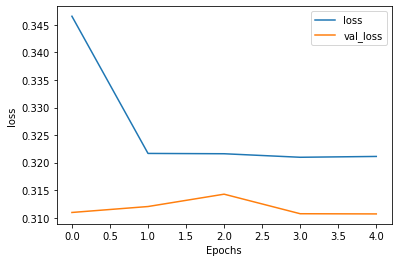
Trainable params: 8,781

Non-trainable params: 0

**Plotting the Graph:**

**Accuracy vs. Epochs.**





**b. BERT BASED SENTIMENT ANALYSIS MODEL.**

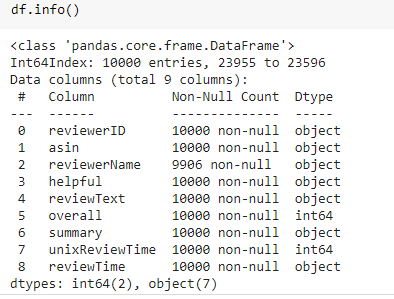
BERT stands for Bidirectional Representation for Transformers, was proposed by researchers at Google AI language in 2018. Although the main aim of that was to improve the understanding of the meaning of queries related to Google Search, BERT becomes one of the most important and complete architecture for various natural language tasks having generated state-of-the-art results on Sentence pair classification task, question-answer task, etc.

BERT is based on the Transformer architecture. Here I have used the BERT with Pytorch-Transformers library. PyTorch-Transformers known as Pytorch-pretrained-bert.

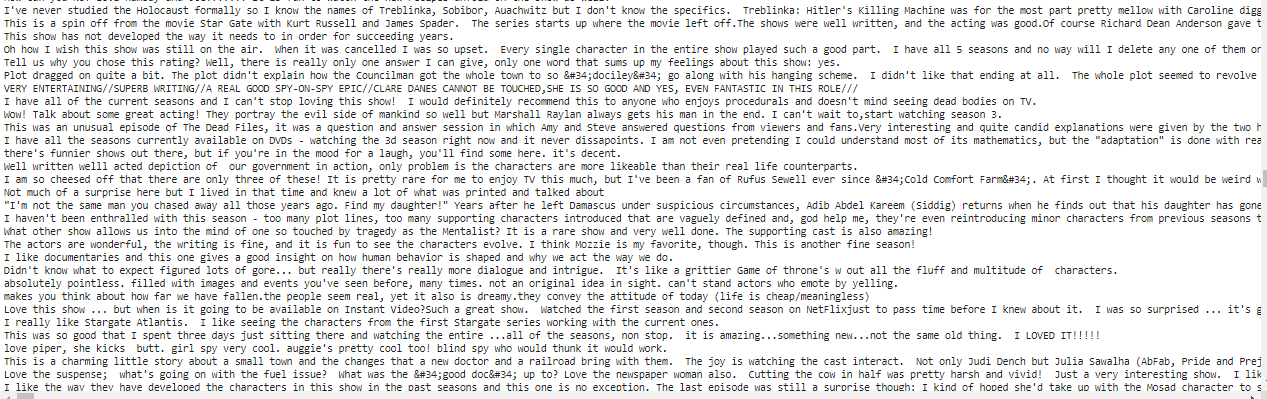
**Using Amazon Product Reviews Dataset.**

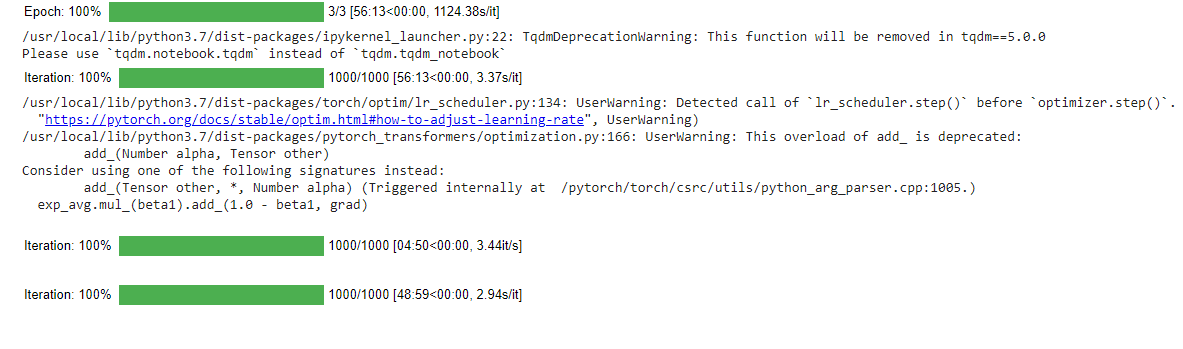
I have used the Amazon product review Dataset which contains only 10,000 rows for BERT Sentiment Analysis.

Amazon Product Review Dataset Information is:

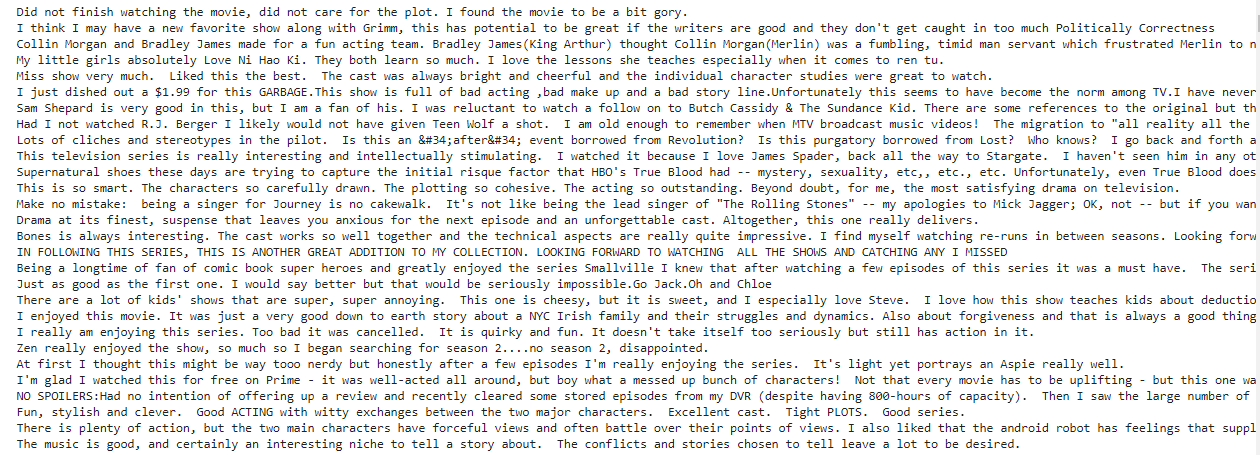


**Training the model:**

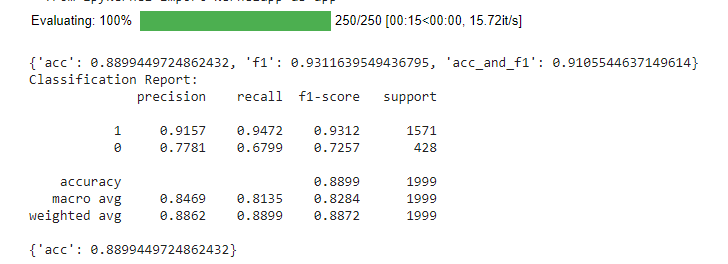




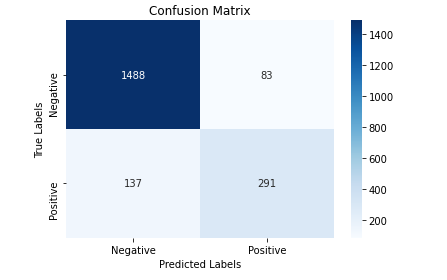
**Evaluating the Model:**



**Classification Report:**



**Confusion Matrix:**



**Accuracy of the Bert model : 0.8899**

**C. COMPARING THE PERFORMANCE OF (BI-LSTM WITH (WORD2VEC, GLOVE, TF-IDF)), BERT FOR SENTIMENT ANALYSIS.**

Here are the Sentiment Analysis Models and their accuracies.

Bi-LSTM With Glove Embeddings performs well with Accuracy 0.9039.

|  |  |
| --- | --- |
| **SENTIMENT ANALYSIS MODELS.** | **ACCURACY.** |
| BI-LSTM WITH GLOVE. | **0.9039** |
| BI-LSTM WITH WORD2VEC. | **0.8939** |
| BI-LSTM WITH TF-IDF. | **0.9065** |
| BERT | **0.8899** |

**D. TRADITIONAL ML MODELS IN ASSIGNMENT-1.**

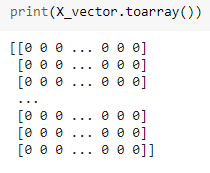
1. Naïve -Bayes.

2. Decision Tree.

3. Logistic Regression.

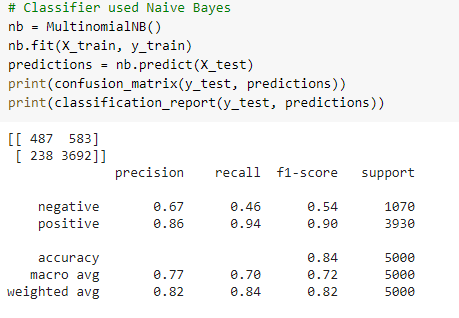
Using the **Amazon Product Review Dataset**, we have implemented the Traditional ML Models.

**NAÏVE- BAYES PIPELINE:**

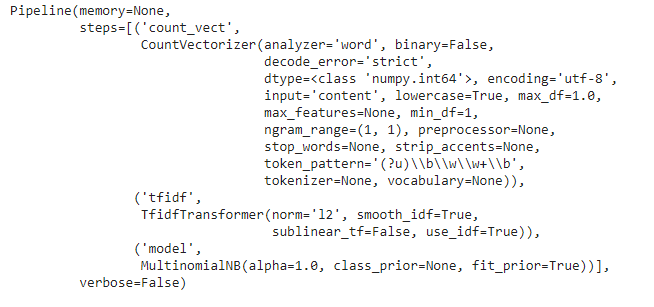


**Train-Test Split : 80:20**

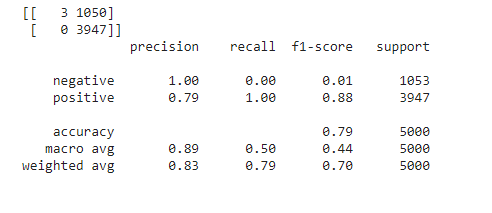
**Classification Report:**



**Creating the Pipeline:**



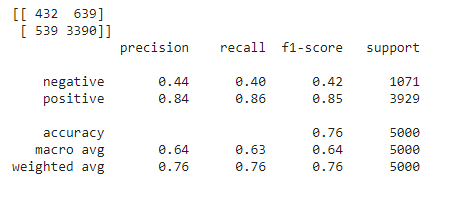
**Classification Report:**

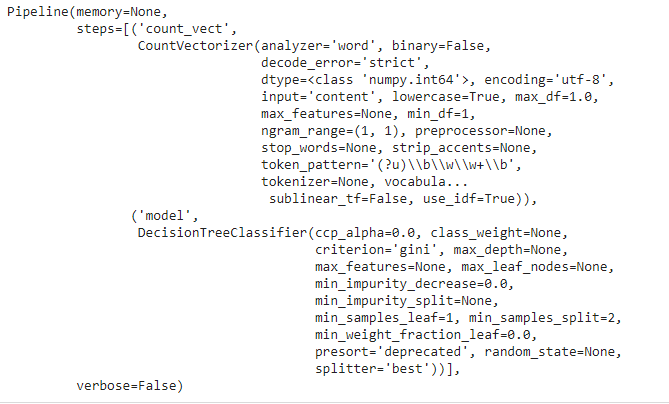


**DECISION TREE PIPELINE:**

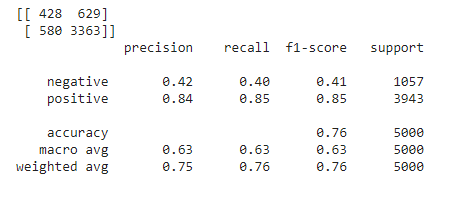
**Train-Test split : 80:20**

**Classification Report:**



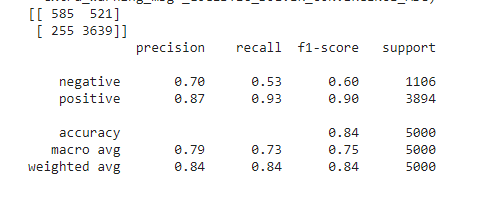


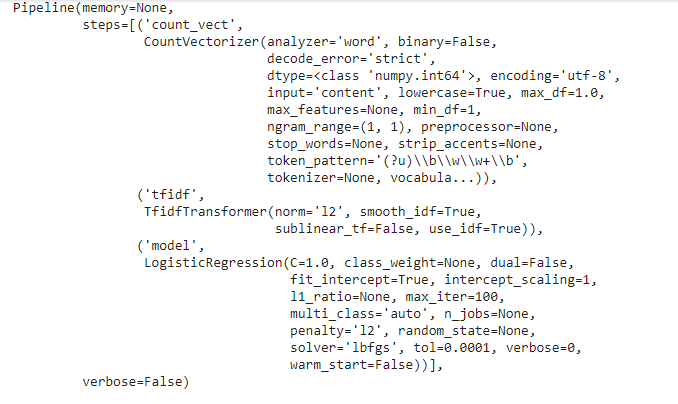
**Classification Report:**



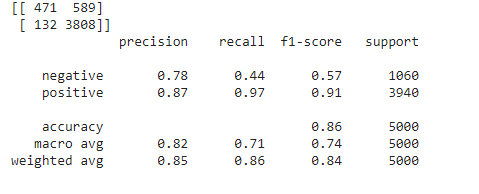
**LOGISTIC REGRESSION PIPELINE:**

**Classification Report:**





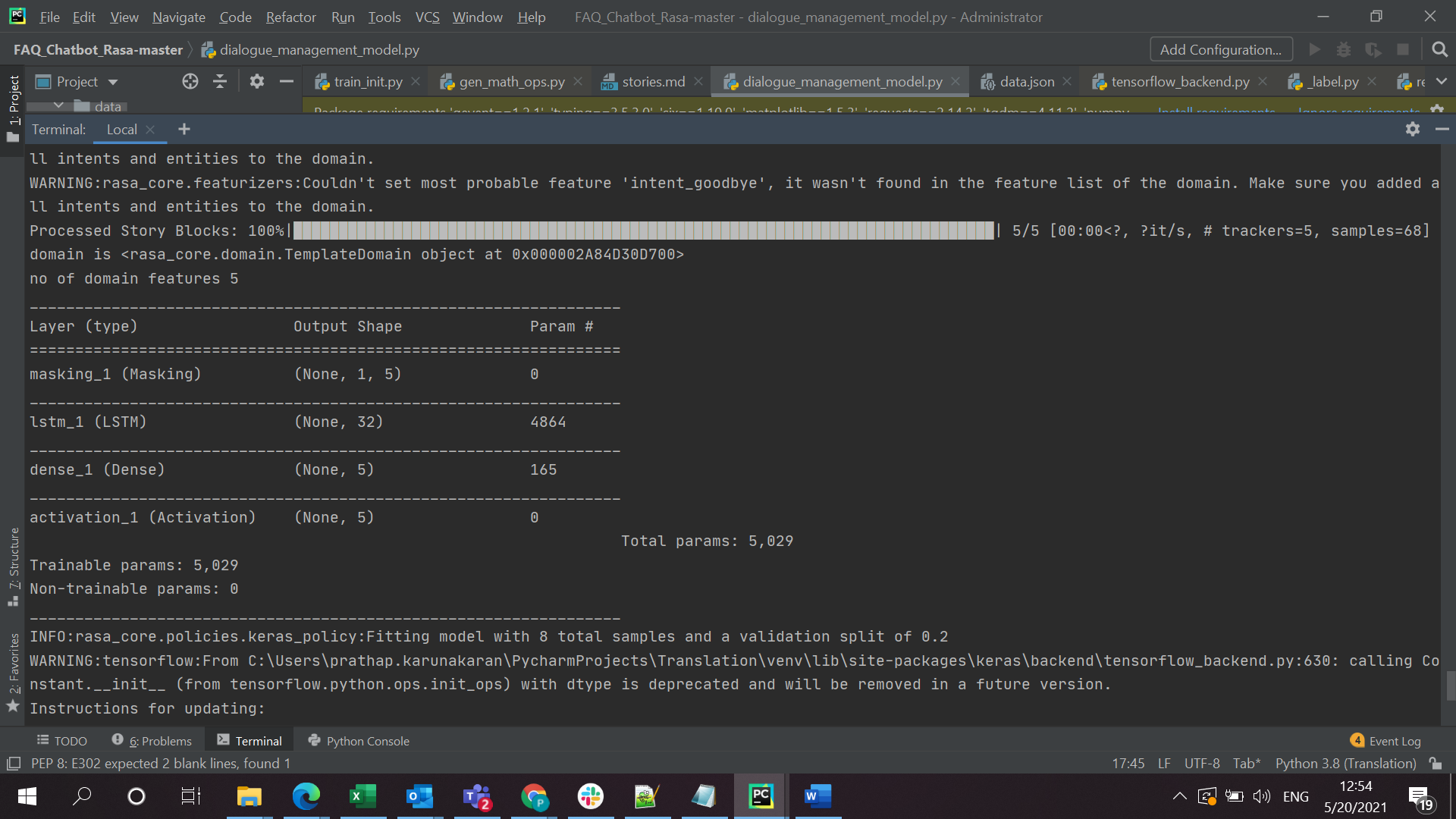
**Classification Report:**



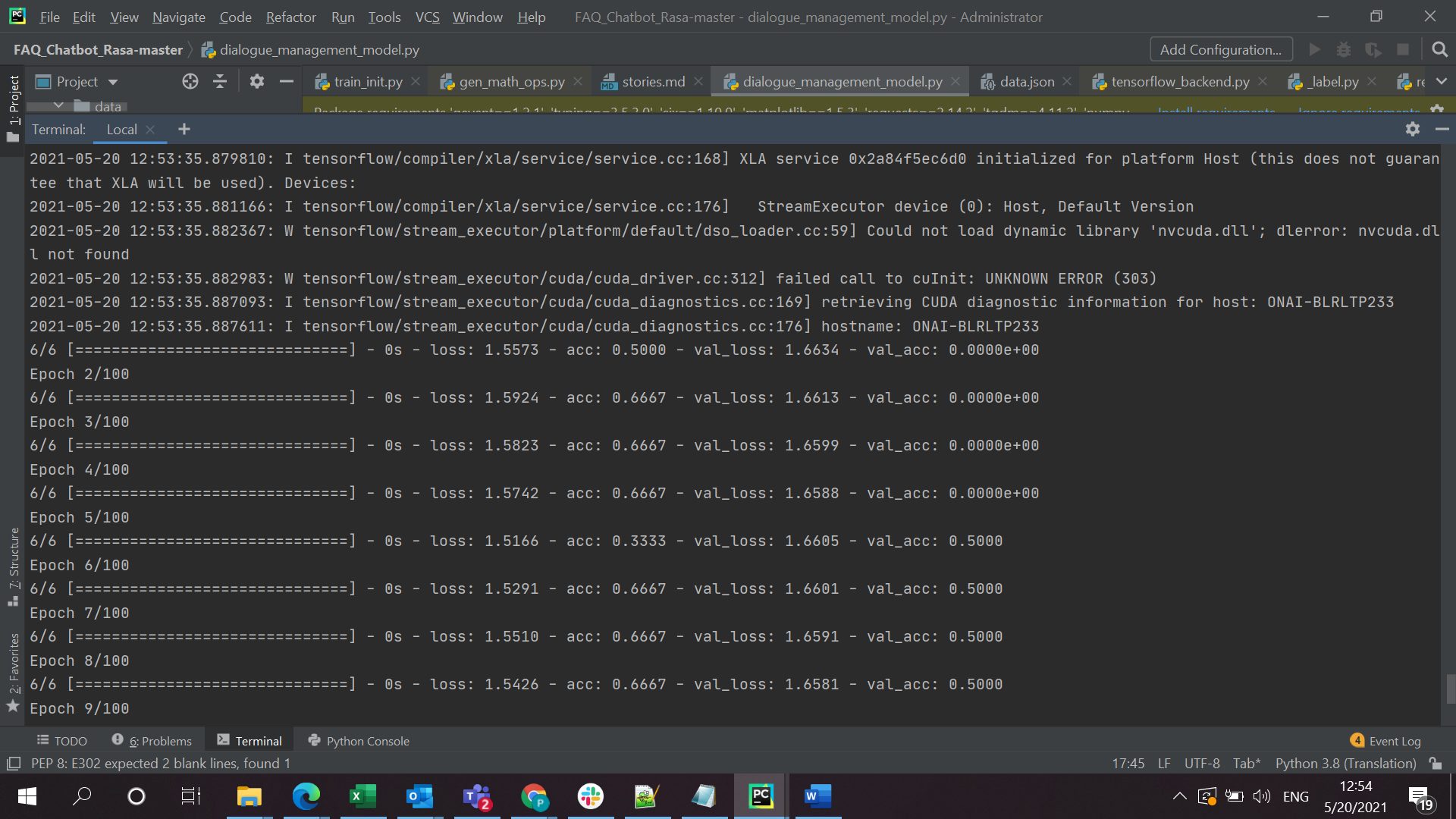
2) Using any existing framework for designing a chatbot (like RASA) implement a chatbot that can answer basic user questions posed in Natural Language. Data for training the system can be found [here](https://drive.google.com/file/d/1Afme0Tf6TJUoUqkdGzFgkATxhHSDLnbE/view?usp=sharing). Your chatbot should imbibe the features of an intelligent conversation. A set of test questions will be released before the submission - for which you will have to submit the answers generated by your chatbot - along with your own reasoning about the quality of the answer. Your reasoning for the answer generated, right or wrong, will be as significant for judging your implementation, as the answers themselves. Your final submission should include documentation about your framework used, how it resolved questions and finds answers as well as the design of the Knowledge base.

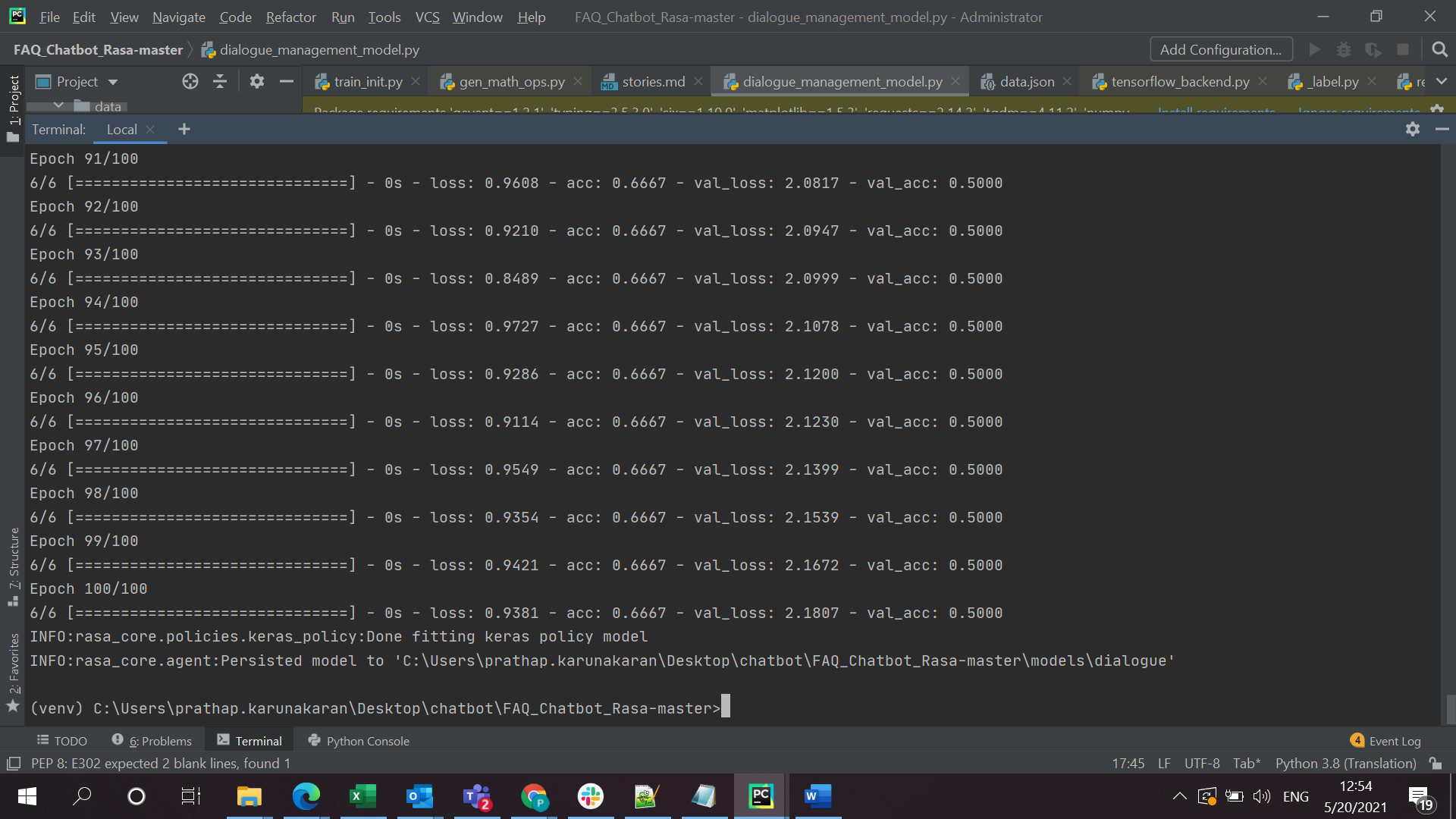
Built the chatbot using rasa, rasa-nlu and rasa-core. It is a simple but intelligent bot, which is trained based on the covid data given as training set.

Simple LSTM model is used to build the intelligent bot, model is below



Trained the bot for 100 Epochs.





Bot is very simple that it has only very few intents, a sample story would look like

## Generated Story 8234137205479123135

\* greet

- utter\_greet

\*

- action\_get\_answer

\* inform

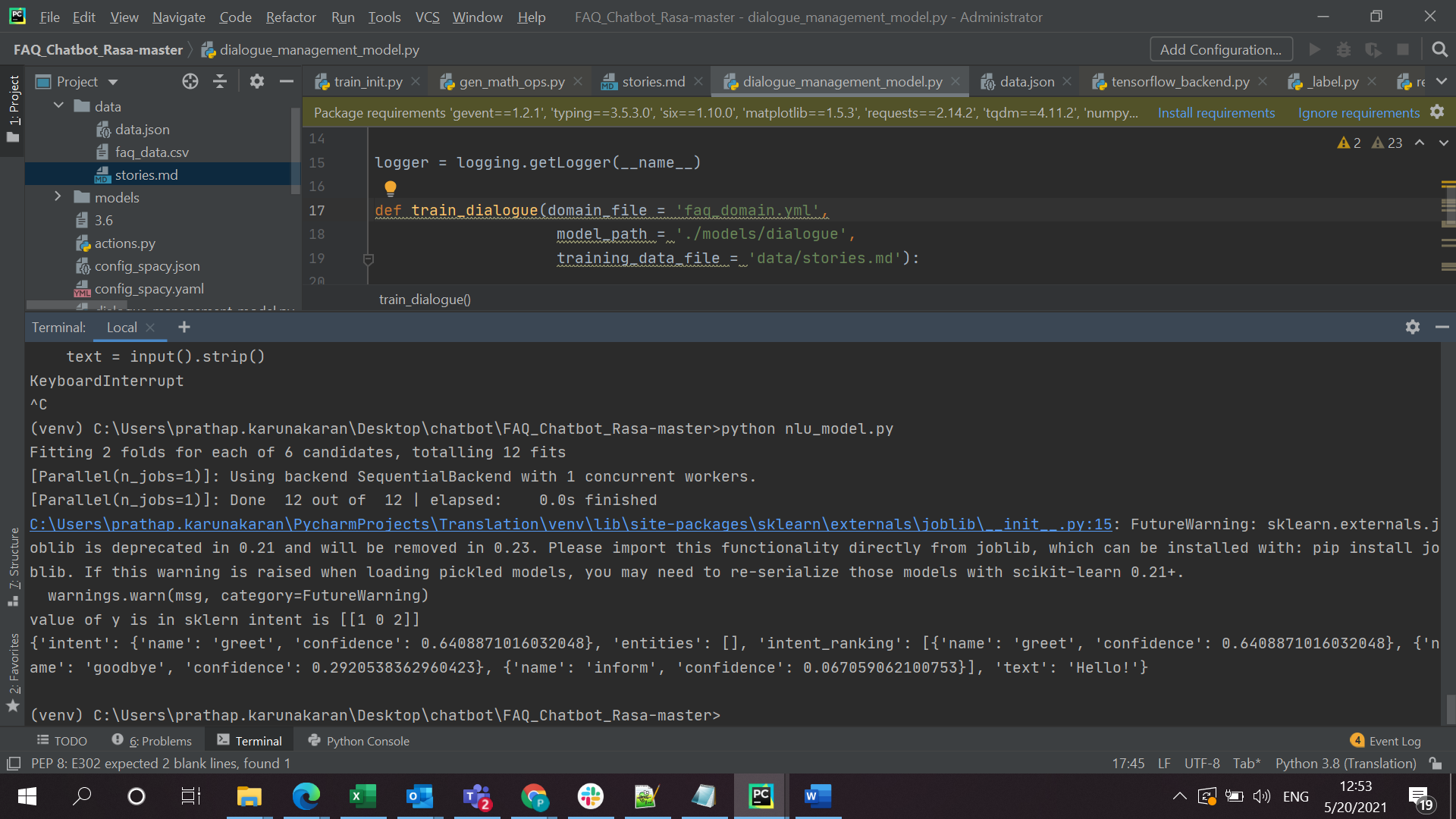
- action\_get\_answer

\* goodbye

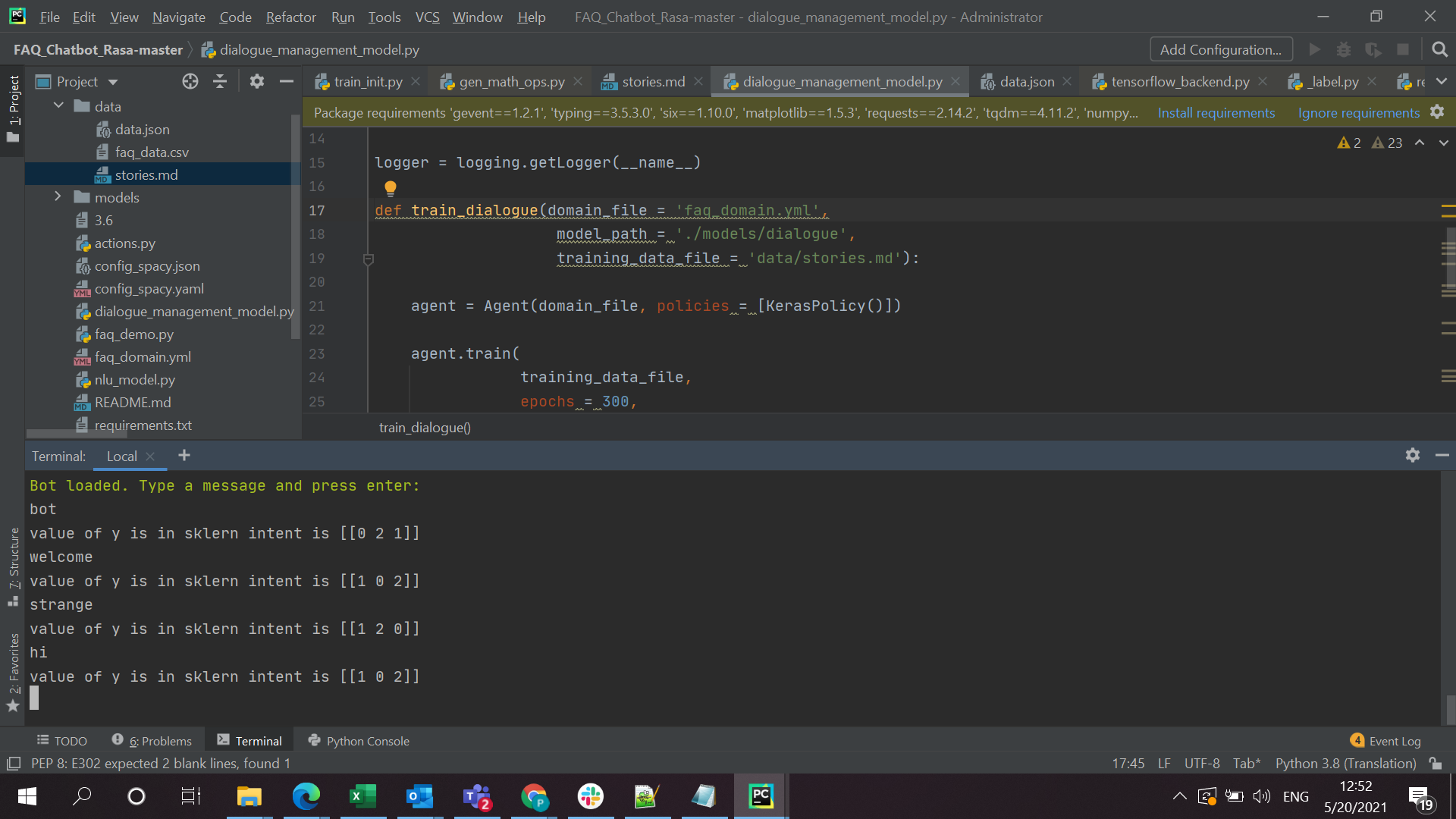
- utter\_goodbye

- export

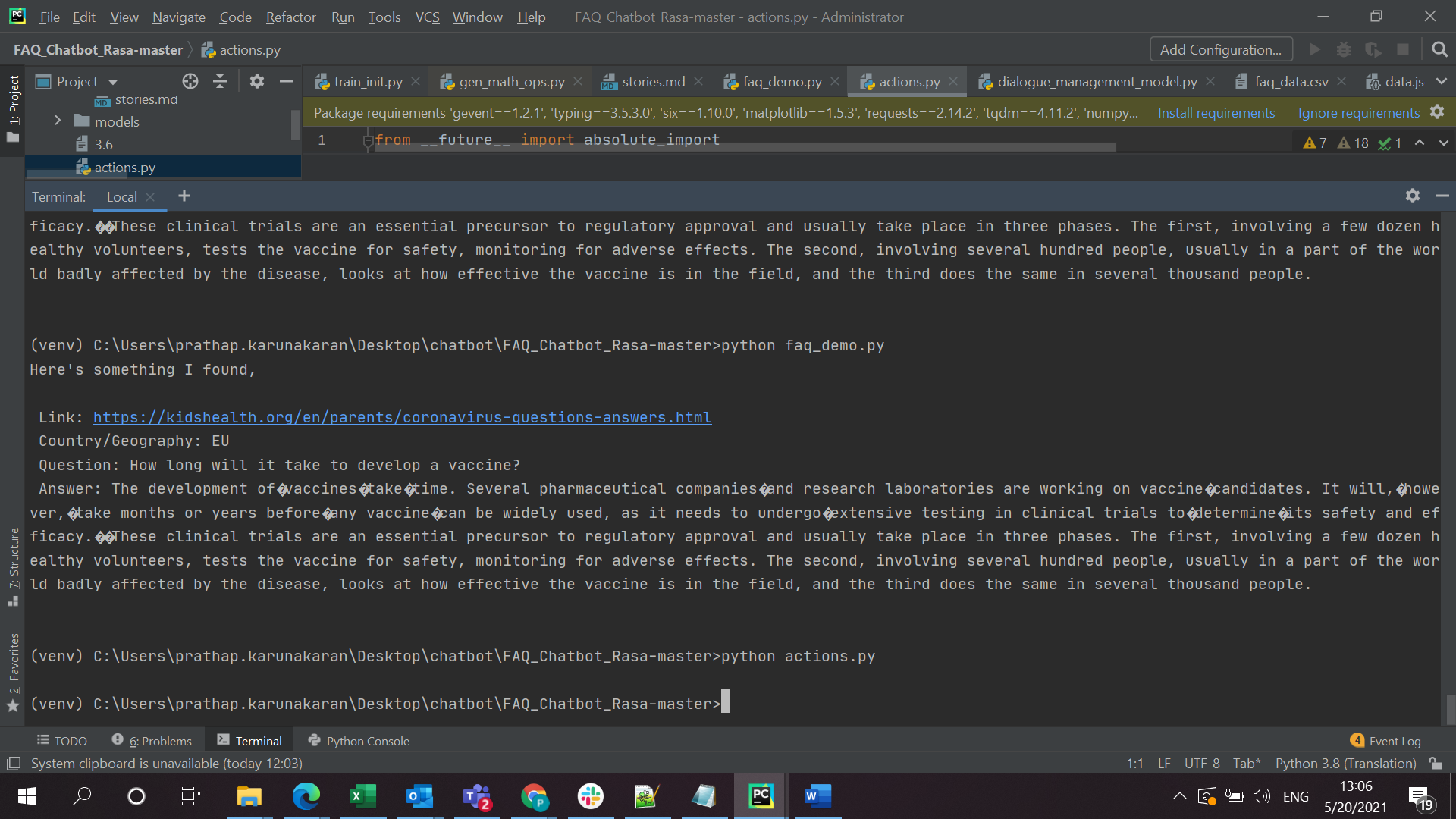
After getting trained, model is smart enough to identify the intents:



We have even tested to check if it is very good in identifying the intents:



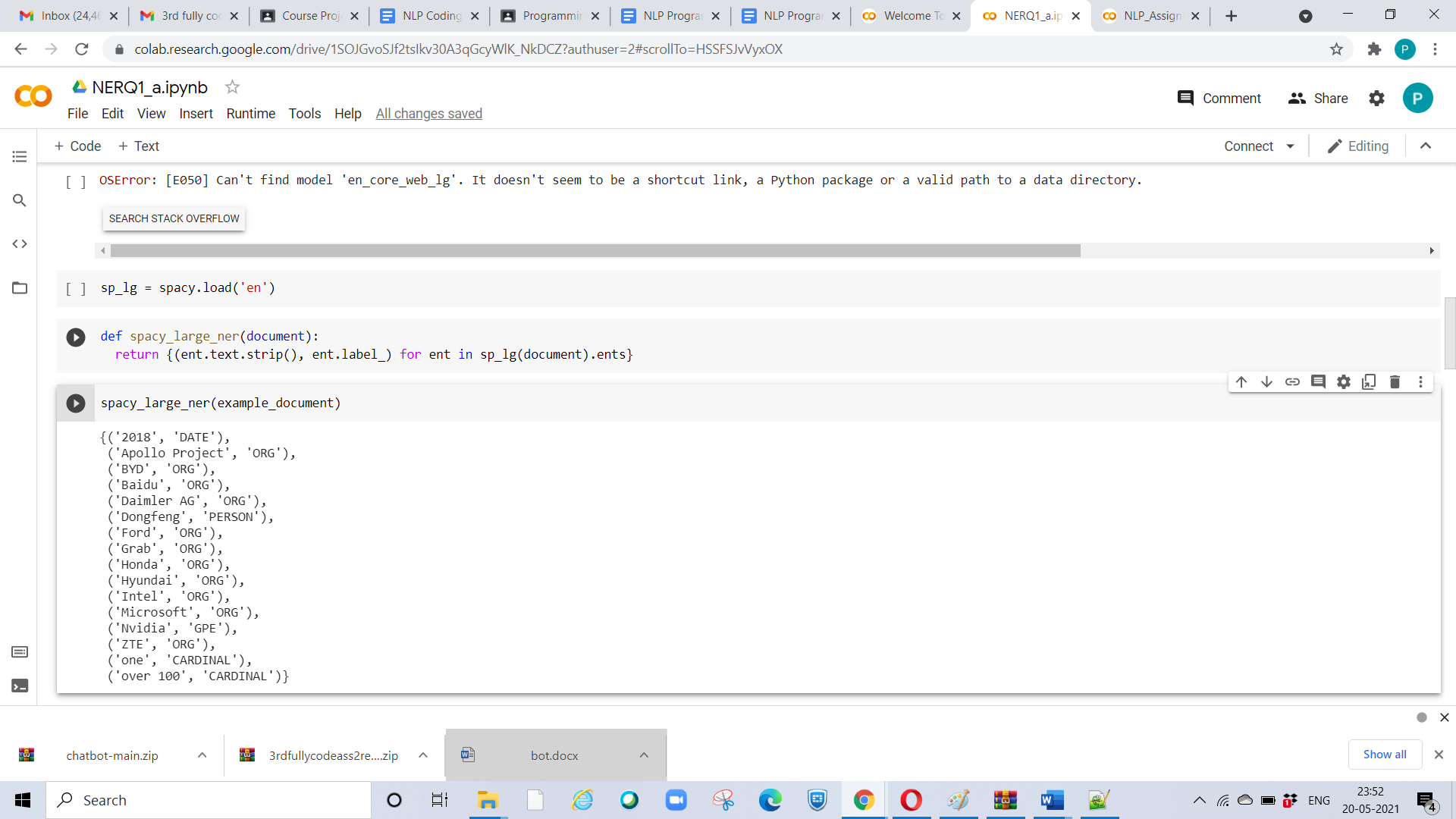
When testing, it has given the below result:



Attached the code, also code in github <https://github.com/prathapk123/chatbot1>

1. Use 20Newsgroups dataset [[link](http://qwone.com/~jason/20Newsgroups/)] and do the following tasks
   1. Run Stanford NER and Spacy NER and extract Named Entities from the data.
   2. Find the top 100 LOC and PERSON entities from the data set. What is the degree of correlation between the two systems? Consider partial and full matches.
   3. Generate Word2vec, Glove, Fasttext and BERT word vectors for the above corpus.
   4. Use the NERs detected in Step (a). to create annotated documents for NER detection. Divide the document collection into training, validation and test data sets.
      1. Implement a custom NER system (for all 4 vector embedding techniques mentioned in (c)) using LSTM. Compare the results of all models obtained (namely, (i) LSTM with word2vec, (ii) LSTM with glove, (iii) LSTM with fast text, (iv) LSTM with BERT embeddings.
      2. Implement a BERT based NER model. Compare the result of this model with those obtained in (d- (i))
   5. Submit full documentation of the system - along with comparative performances of all the systems.
2. Ran Stanford NER and Spacy NER and extracted Named Entities

Spacy



Stanford NER

