Cyberbullying Detection on Social Media

Anurag Rajendra Watane
11013614

[Slide 2]

A brief introduction about cyberbullying.

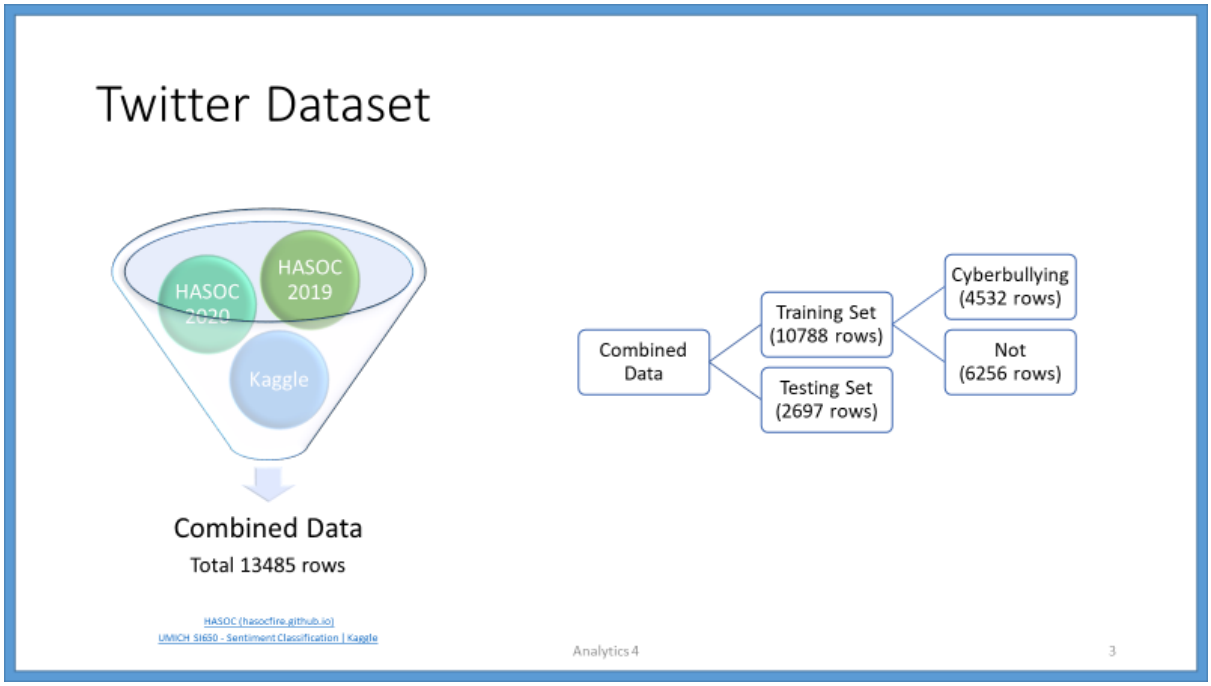
What is Cyberbullying?

- The term "Cyberbullying" means, use of Information Technology to harm or harass other people in a deliberate, repeated, and hostile manner
- Mean text messages, spreading rumours, posting messages, and sharing embarrassing photos and videos on social networking sites are all examples of cyberbullying

[slide 3]

I collected the Twitter data from various sources and combined them. The combined dataset has total number of 13485 sentences, which I further split into training and testing set.

The training set has a target column with two classes, cyberbullying, and not-cyberbullying.

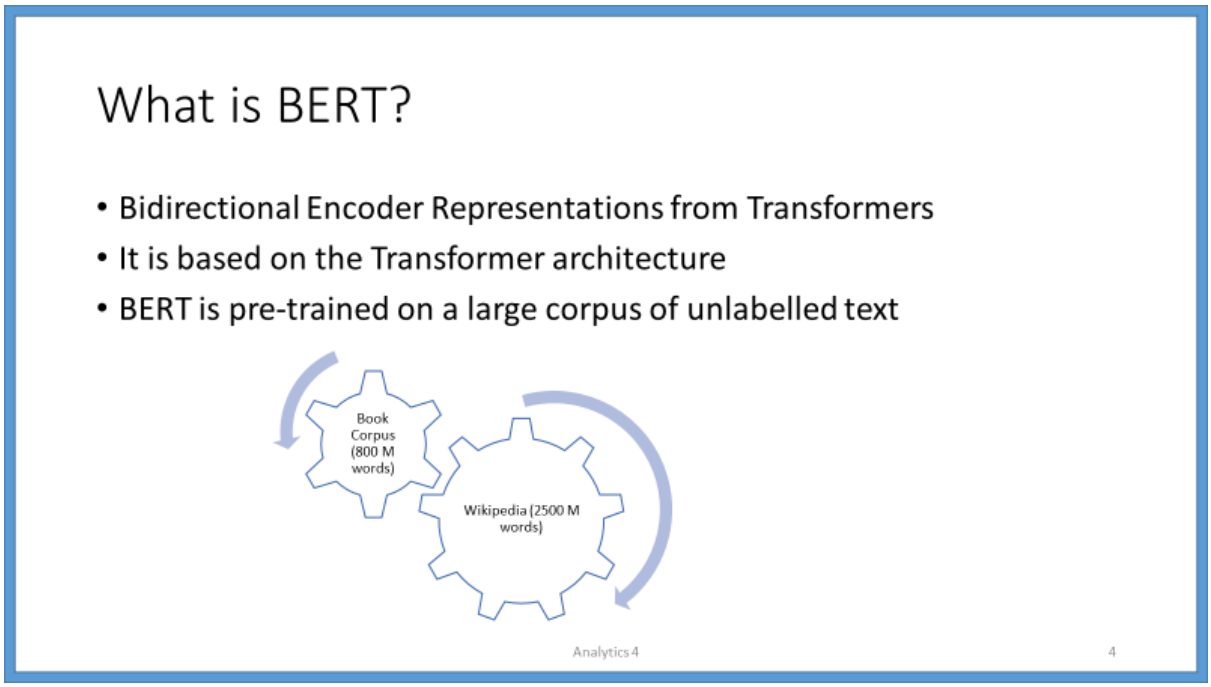


[Slide 4]

BERT stands for Bidirectional Encoder Representations from Transformers.

It is based on the Transformer architecture.

BERT is pre-trained on a large corpus of unlabelled text including the entire Wikipedia (containing 2,500 million words) and Book Corpus (containing 800 million words).



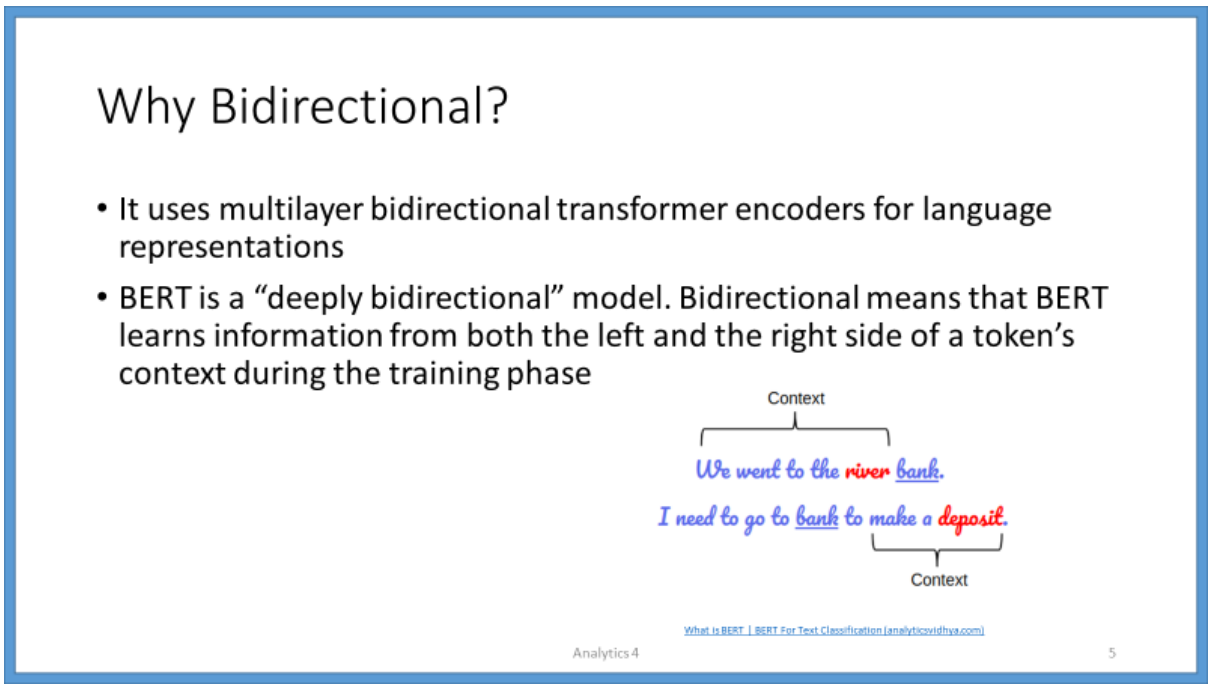
[Slide 5]

BERT uses multilayer bidirectional transformer encoders for language representations.

BERT is a “deeply bidirectional” model. Bidirectional means that BERT learns information from both the left and the right side of a token’s context during the training phase.

For example, in the first sentence, the word ‘bank’ is associated with river, and it’s meaning can be understood with the context on left side.

In the second sentence, the word ‘bank’ is associated with money deposit, which can be understood with the help of context on right side.



[Slide 6]

The pre-trained BERT model weights already encode a lot of information about our language.

Because of the pre-trained weights this method allows us to fine-tune our task on a much smaller dataset.

It takes much less time to fine-tune the model.

Why pretrained model?

- The pre-trained BERT model weights already encode a lot of information about our language
- Because of the pre-trained weights this method allows us to fine-tune our task on a much smaller dataset
- It takes much less time to fine-tune the model

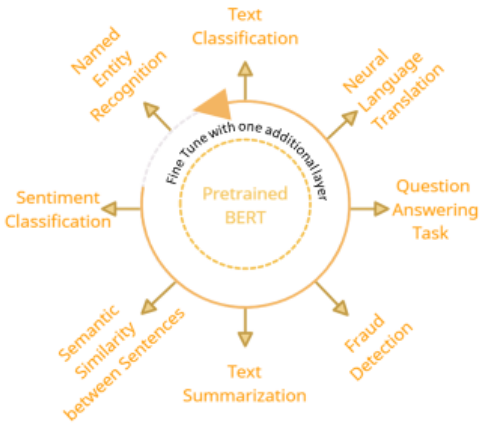
[Slide 7]

The pre-trained BERT model can be fine-tuned with just one additional output layer, to create state-of-the-art models for a wide range of NLP tasks such as Text Classification, Neural Language Translation, Question Answering Task.

BERT uses the concept of fine-tuning, and the final model for any task is almost the same as BERT.

Advantages of pre-trained model

- BERT uses the concept of fine-tuning, and the final model for any task is almost the same as BERT.

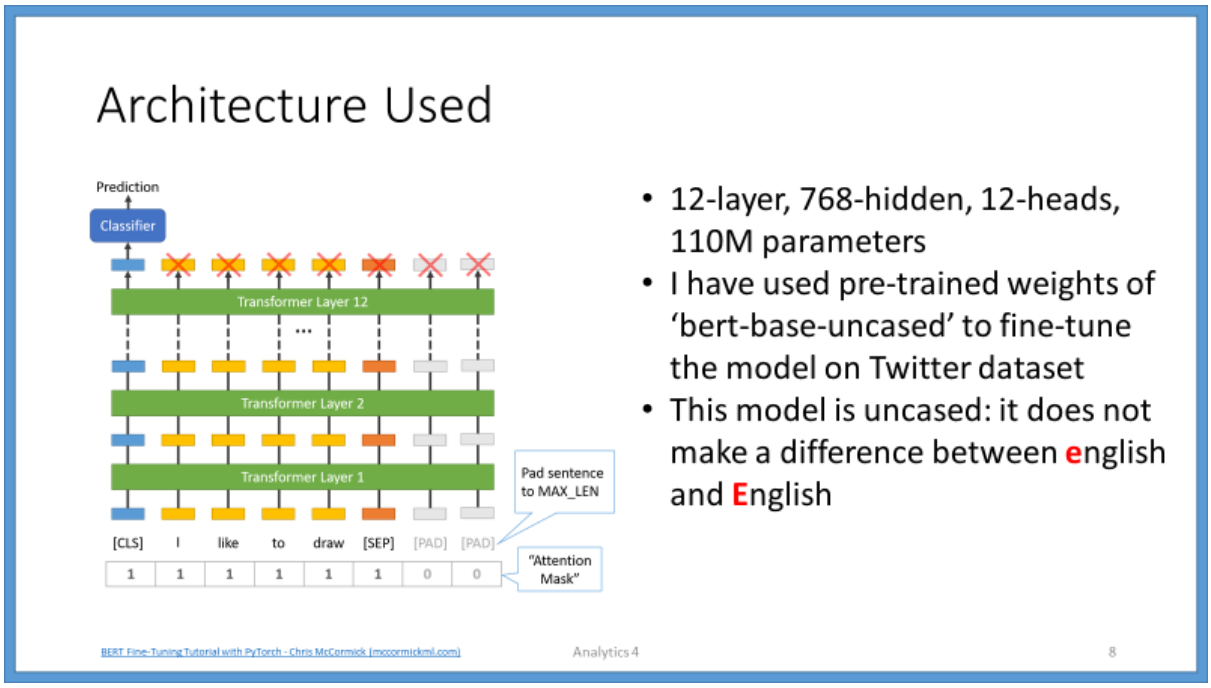


[Slide 8]

The BERT-Base model uses 12 layers of transformers block, with a hidden size of 768, and number of self-attention heads as 12, and has around 110M trainable parameters.

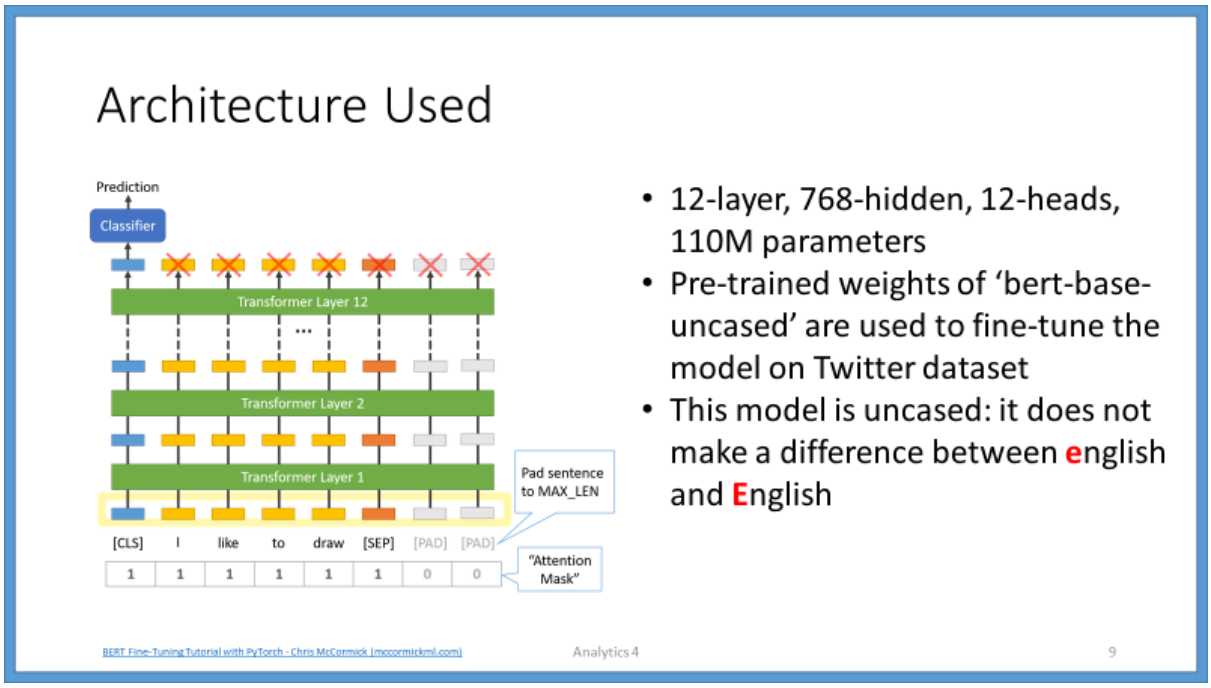
Pre-trained weights of ‘bert-base-uncased’ are used, to fine-tune the model on Twitter dataset.

This model is uncased; therefore, it does not make a difference between english (in small case) and English (in sentence case).



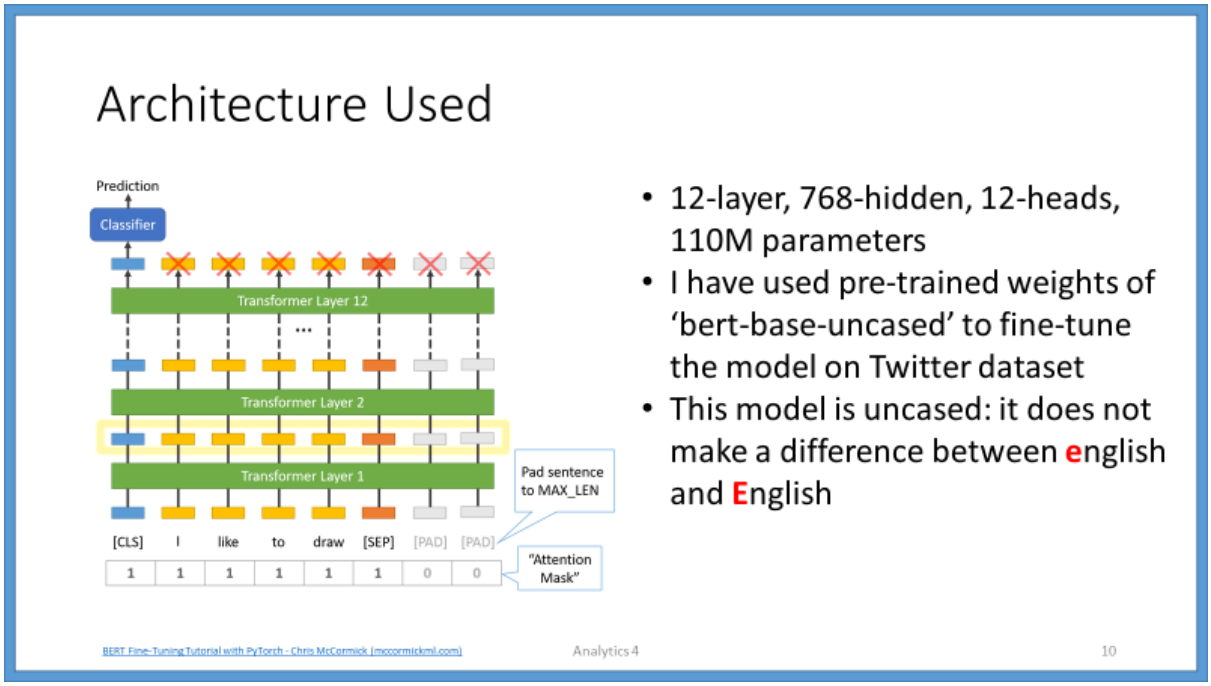
[Slide 9]

BERT consists of 12 Transformer layers. Each transformer takes in a list of token embeddings,



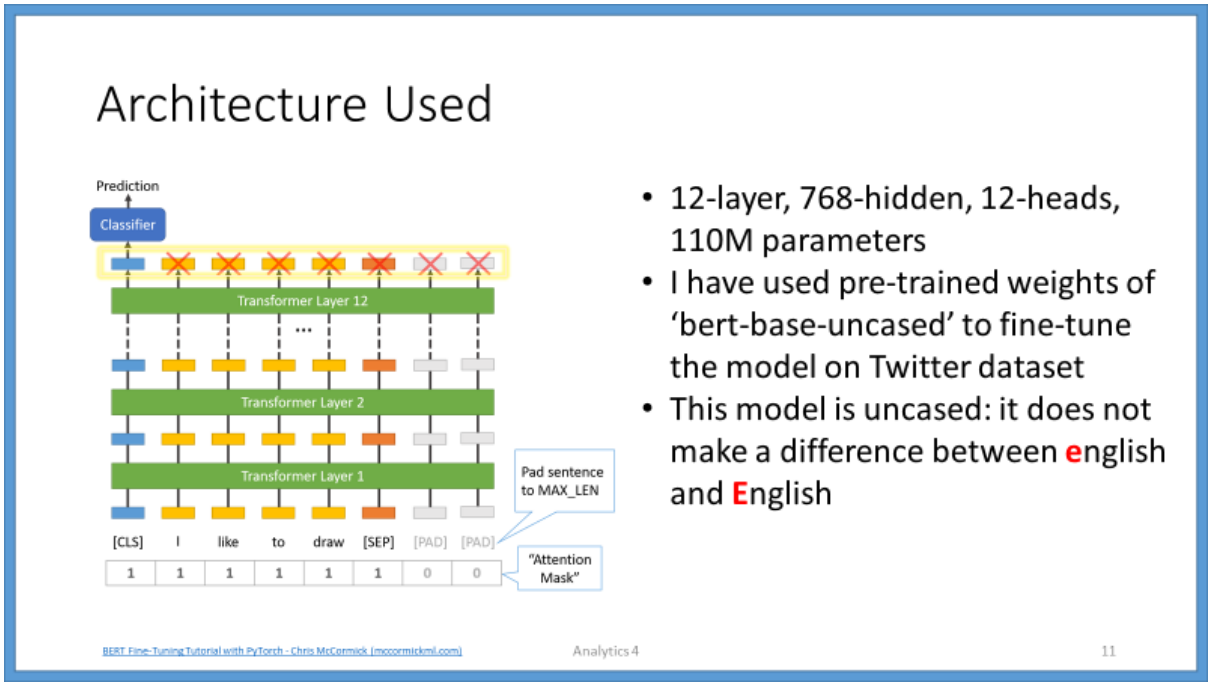
[slide 10]

and produces the same number of embeddings on the output, but with the feature values changed.



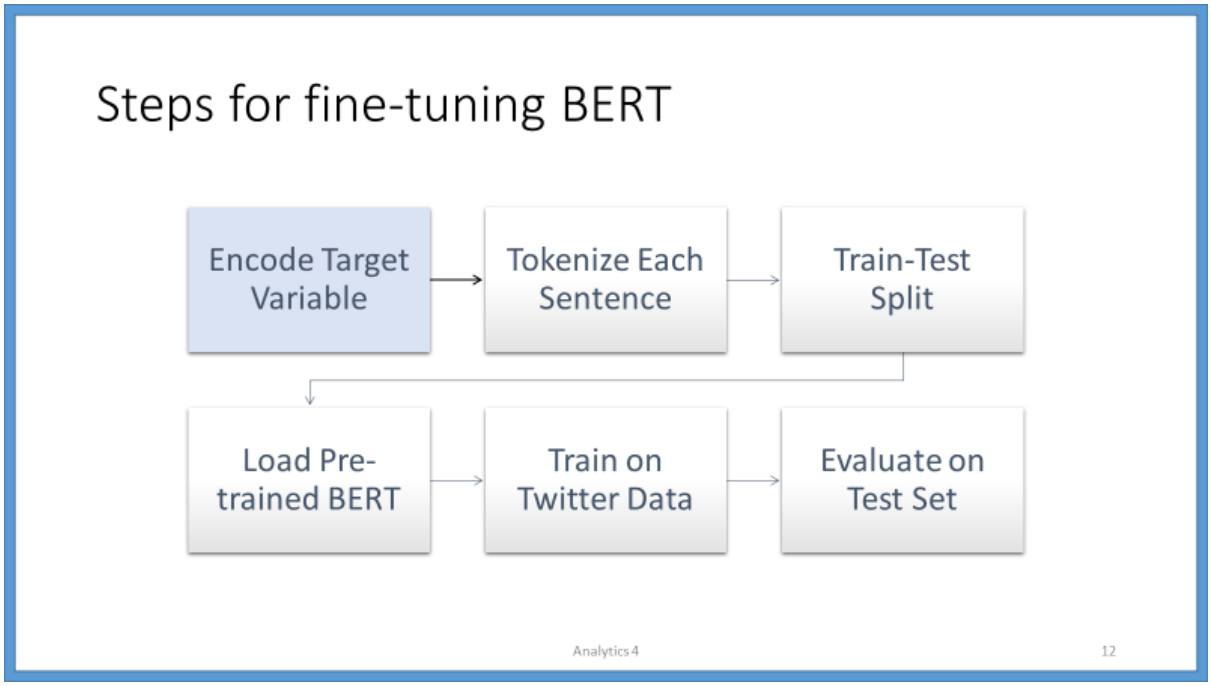
[slide 11]

On the output of the final (12th) transformer, only the first embedding (corresponding to the [CLS] token) is used by the classifier.



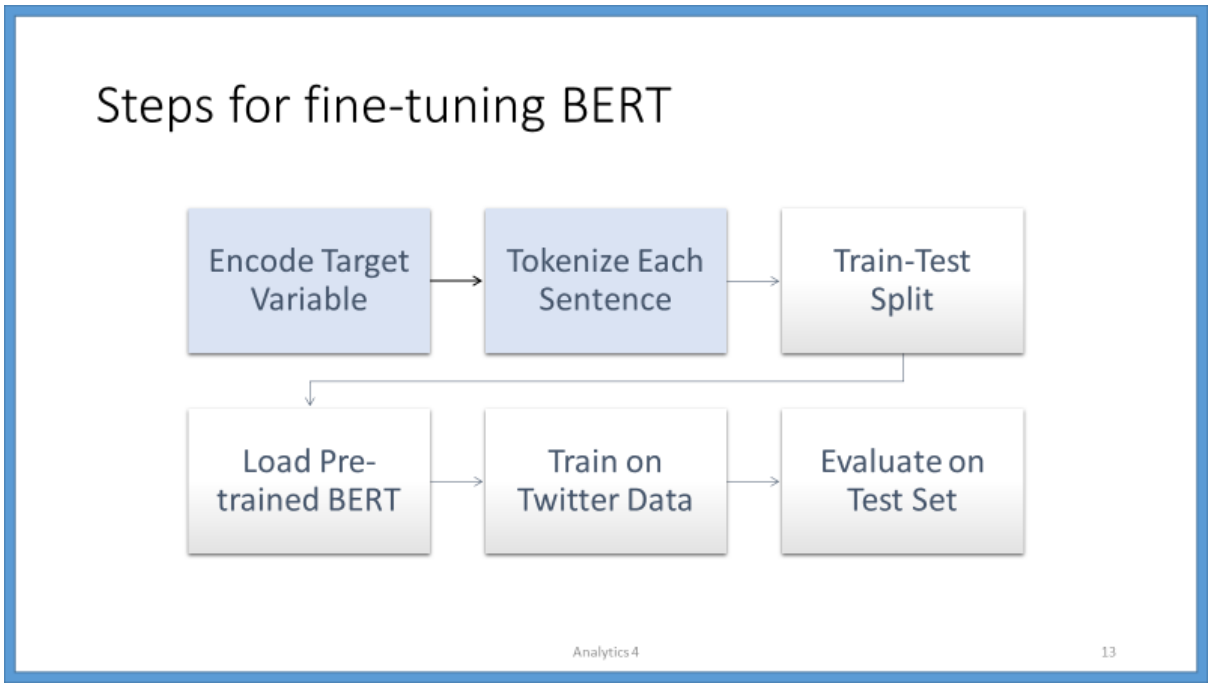
[slide 12]

The first step for fine tuning the BERT model is to encode the labels in target variable, by converting each label into number.



[slide 13]

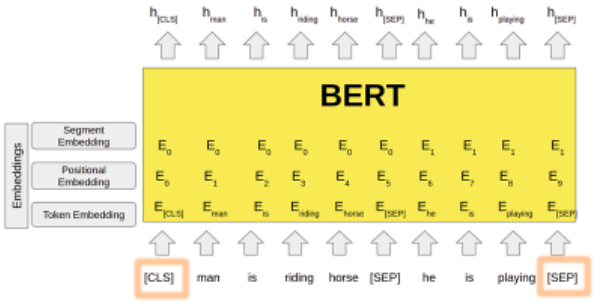
The second step is to tokenize each sentence.



[slide 14]

BERT uses special tokens [CLS] and [SEP] to understand input properly.

Tokenization for BERT

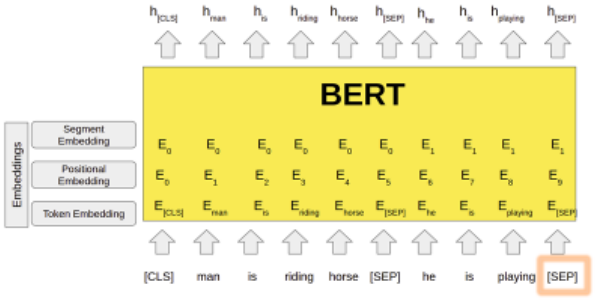


- BERT uses special tokens [CLS] and [SEP] to understand input properly
- [SEP] token has to be inserted at the end of every sentence
- The first token of every input sequence is the special classification token – [CLS]
- The last hidden state of BERT, corresponding to this token (h[CLS]) is used in classification tasks as an aggregate of the entire sequence representation

[slide 15]

[SEP] token must be inserted at the end of every sentence.

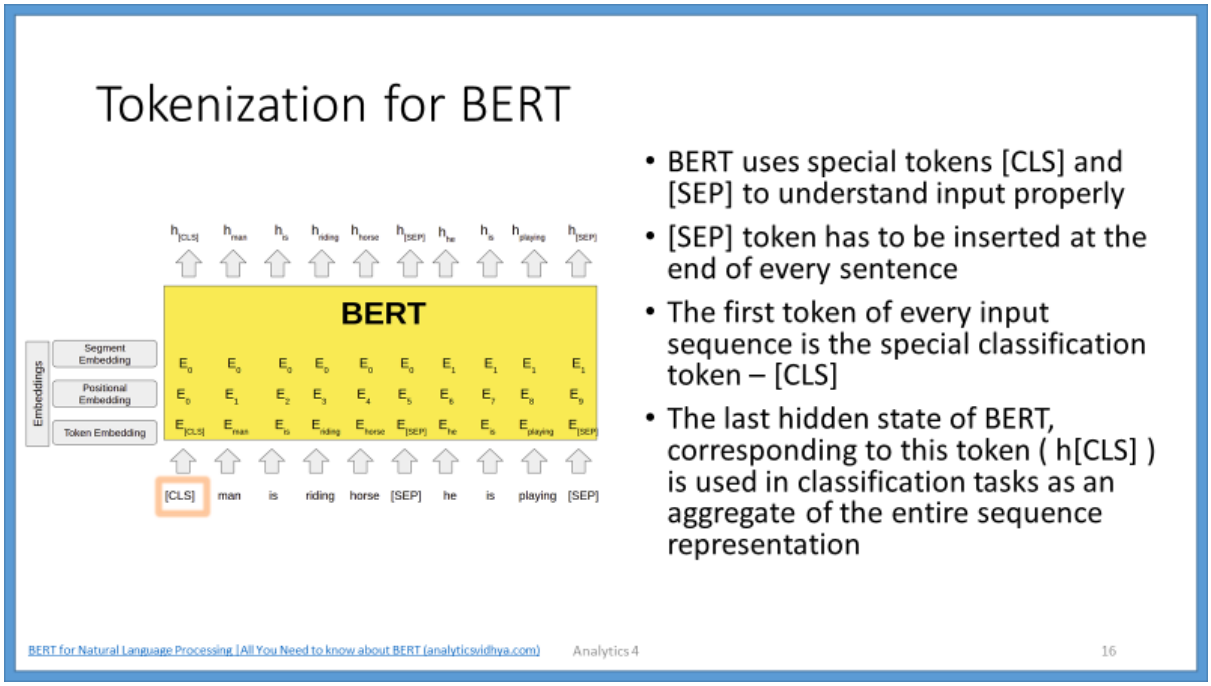
Tokenization for BERT



- BERT uses special tokens [CLS] and [SEP] to understand input properly
- [SEP] token has to be inserted at the end of every sentence
- The first token of every input sequence is the special classification token – [CLS]
- The last hidden state of BERT, corresponding to this token (h[CLS]) is used in classification tasks as an aggregate of the entire sequence representation

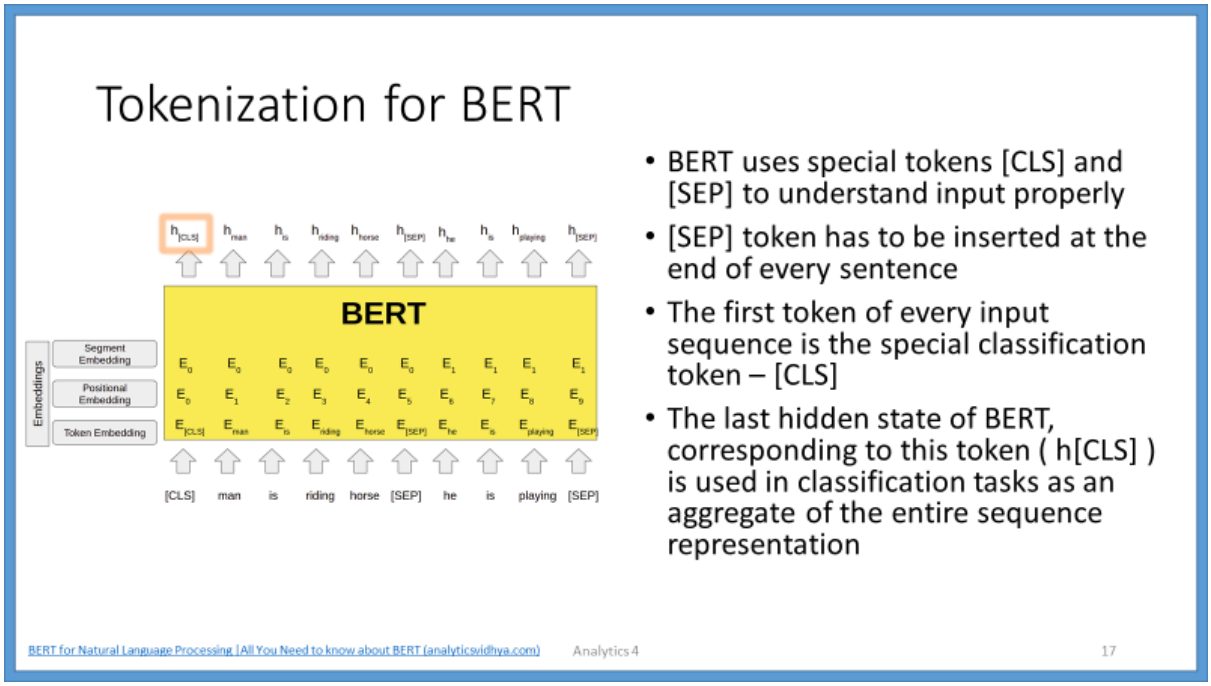
[slide 16]

The first token of every input sequence is the special classification token – [CLS].



[slide 17]

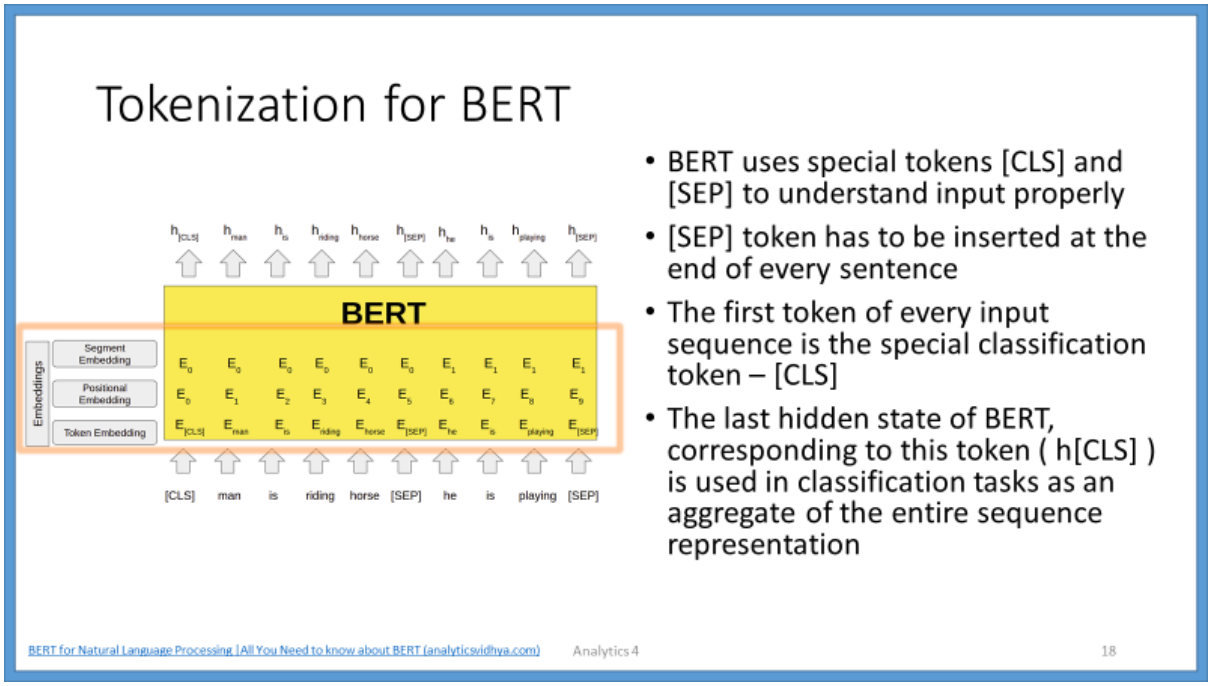
The last hidden state of BERT, corresponding to this token (h[CLS]), is used in classification tasks, as an aggregate of the entire sequence representation.



[slide 18]

Final Embeddings, used by model architecture are, the sum of token embedding, positional embedding as well as segment embedding.

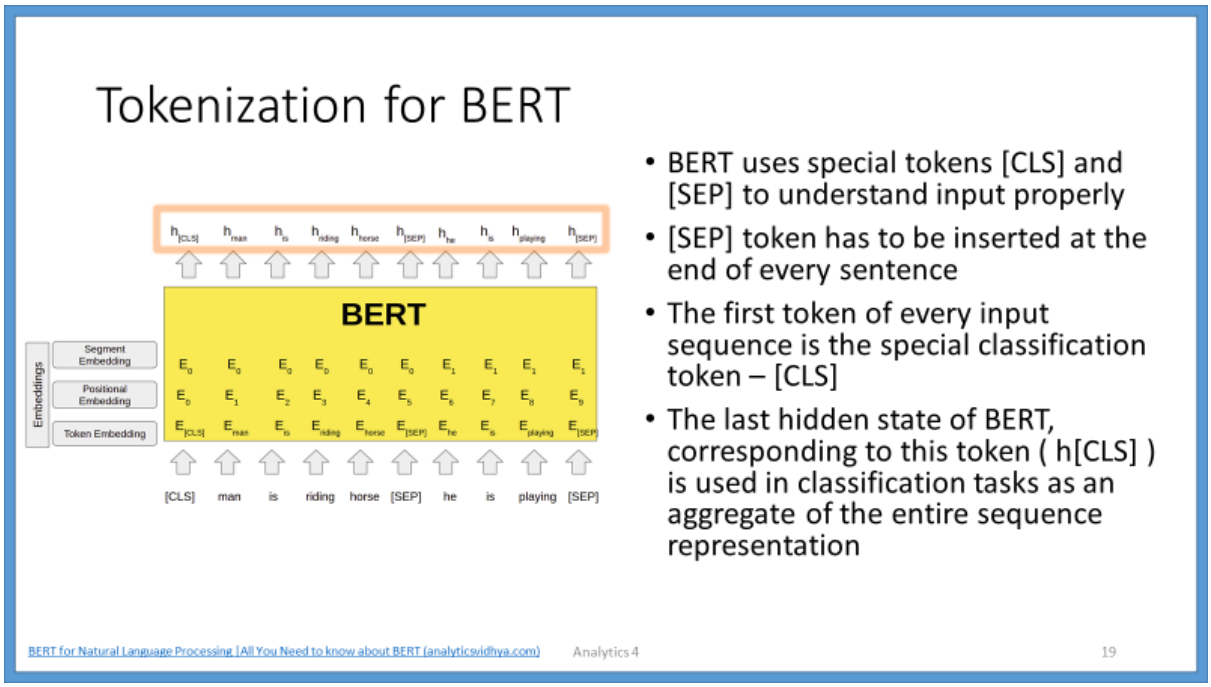
The final embeddings are, then fed into the deep bidirectional layers, to get output.



[slide 19]

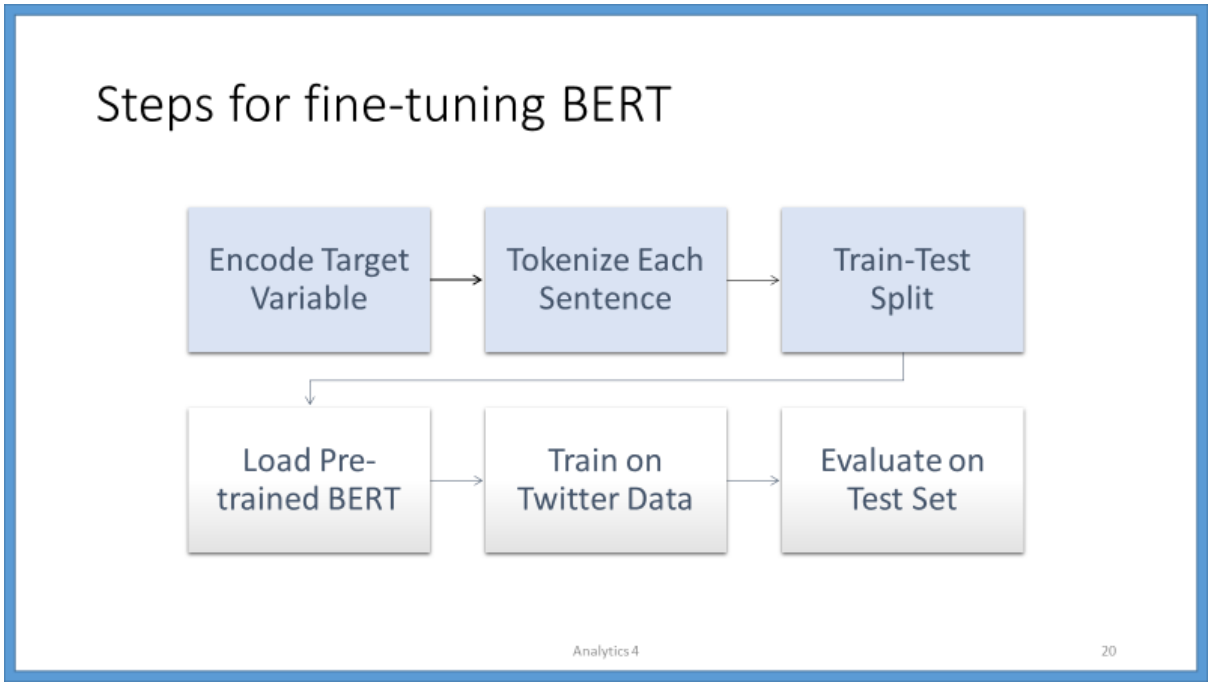
The output of the BERT is the hidden state vector, of pre-defined hidden size, corresponding to each token, in the input sequence.

These hidden states, from the last layer of the BERT, are then used, for various NLP tasks.



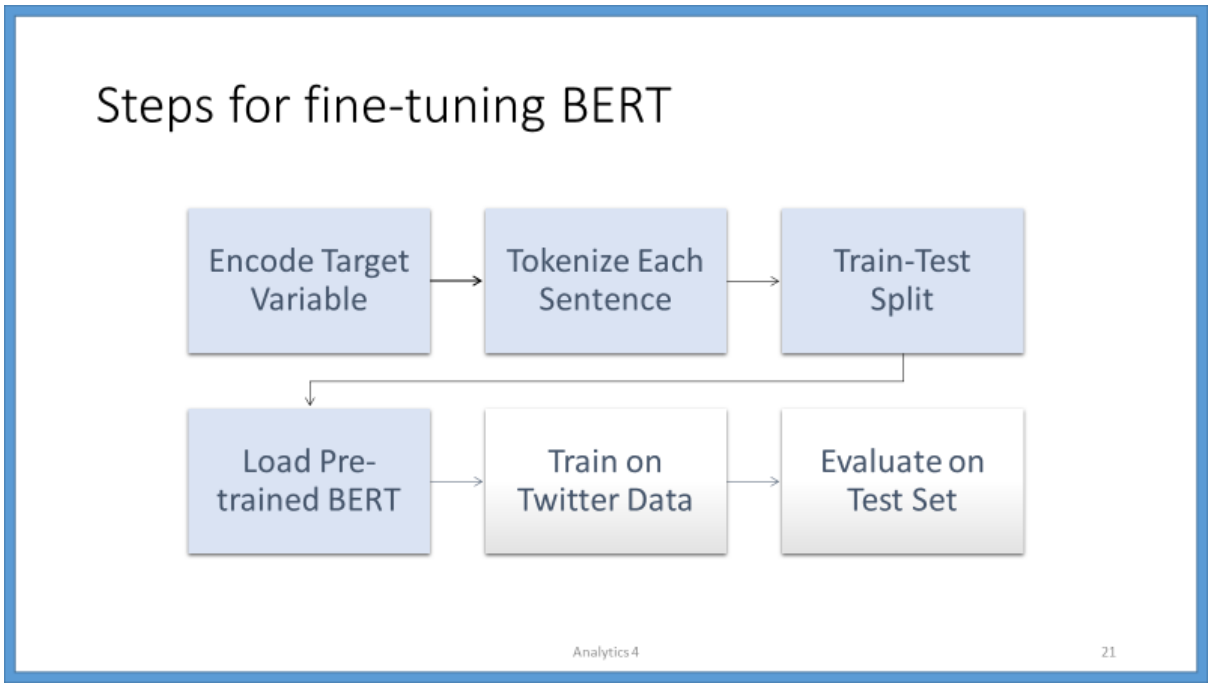
[slide 20]

The third step is to divide the dataset into training set and testing set.



[slide 21]

The fourth step is to load the pre-trained BERT model.



[slide 22]

BertForSequenceClassification is used to load the pre-trained BERT model from 'transformers' library developed by company Hugging Face.

The BertForSequenceClassification is used to fine-tune the BERT model.

BertForSequenceClassification is the normal BERT model with an added single linear layer on top for classification.

Load pretrained BERT

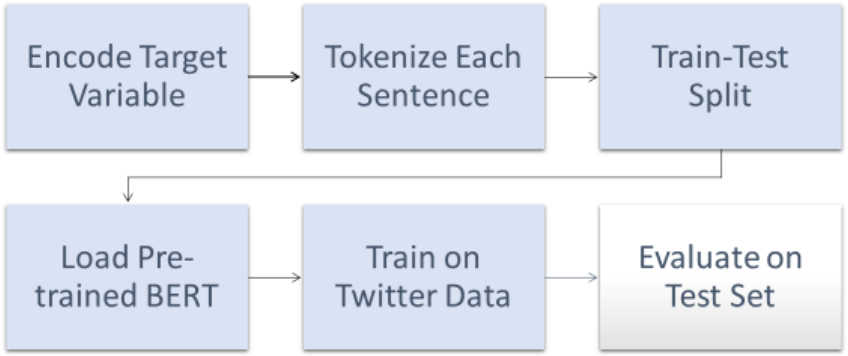
```
from transformers import BertForSequenceClassification
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",          # Use the 12-layer BERT model, with an uncased vocab
    num_labels = 2,               # The number of output labels--2 for binary classification
    output_attentions = False,    # Whether the model returns attentions weights
    output_hidden_states = False, # Whether the model returns all hidden-states
)
```

- BertForSequenceClassification is used to load the pre-trained BERT model from 'transformers' library developed by company Hugging Face
- The BertForSequenceClassification is used to fine-tune the BERT model
- BertForSequenceClassification is the normal BERT model with an added single linear layer on top for classification

[slide 23]

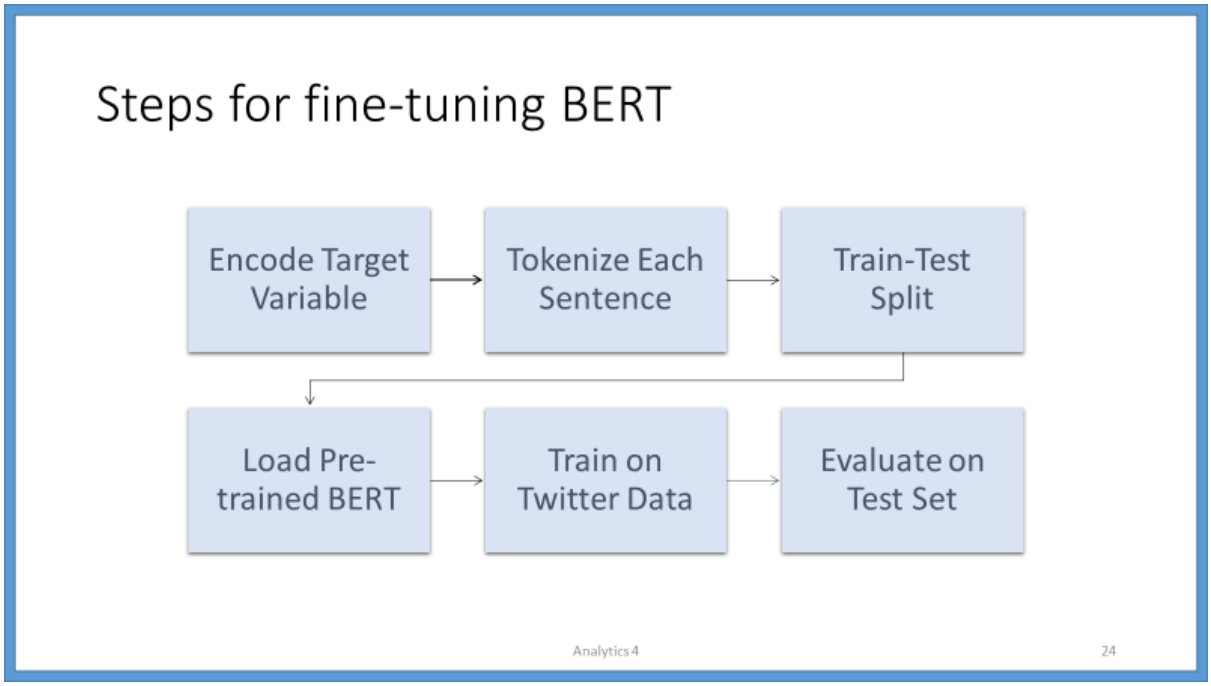
The 5th step is to train the model on Twitter data.

Steps for fine-tuning BERT



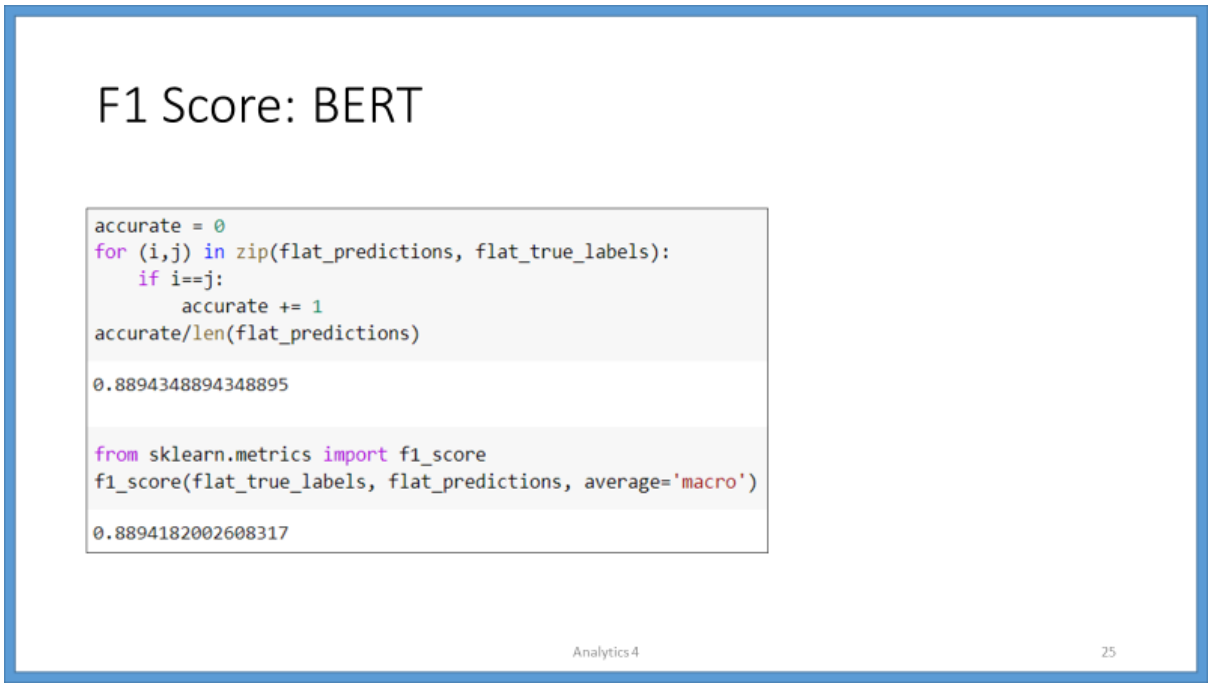
[slide 24]

The final step is to evaluate the fine-tuned model on test set.



[slide 25]

The F1-score obtained for BERT model is 0.88.



[slide 26]

I also created an Ensemble model to compare the results, and the F1 score obtained for Ensemble model is 0.78.

F1 Score: Ensemble Model

```
#Train and evaluation
for clf in (log_clf, svm_clf, mnb_clf, voting_clf):
    clf.fit( X_train_counts, y_train )
    y_pred = clf.predict(X_test_counts)
    print(clf.__class__.__name__, f1_score(y_test, y_pred, average = 'macro'))
```

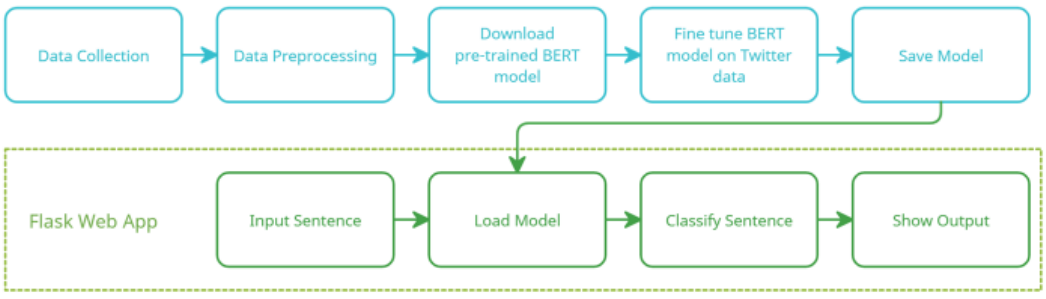
```
LogisticRegression 0.7774494137250483
SVC 0.7791690204159132
MultinomialNB 0.7148849401810784
VotingClassifier 0.7848905890192499
```

Analytics 4 26

[slide 27]

After completing the training part, I saved the model, and used in Flask Web App as a sentence classifier.

Pipeline



Analytics 4 27

[slide 28]

This is the complete framework, where I am using Flask to create endpoints.

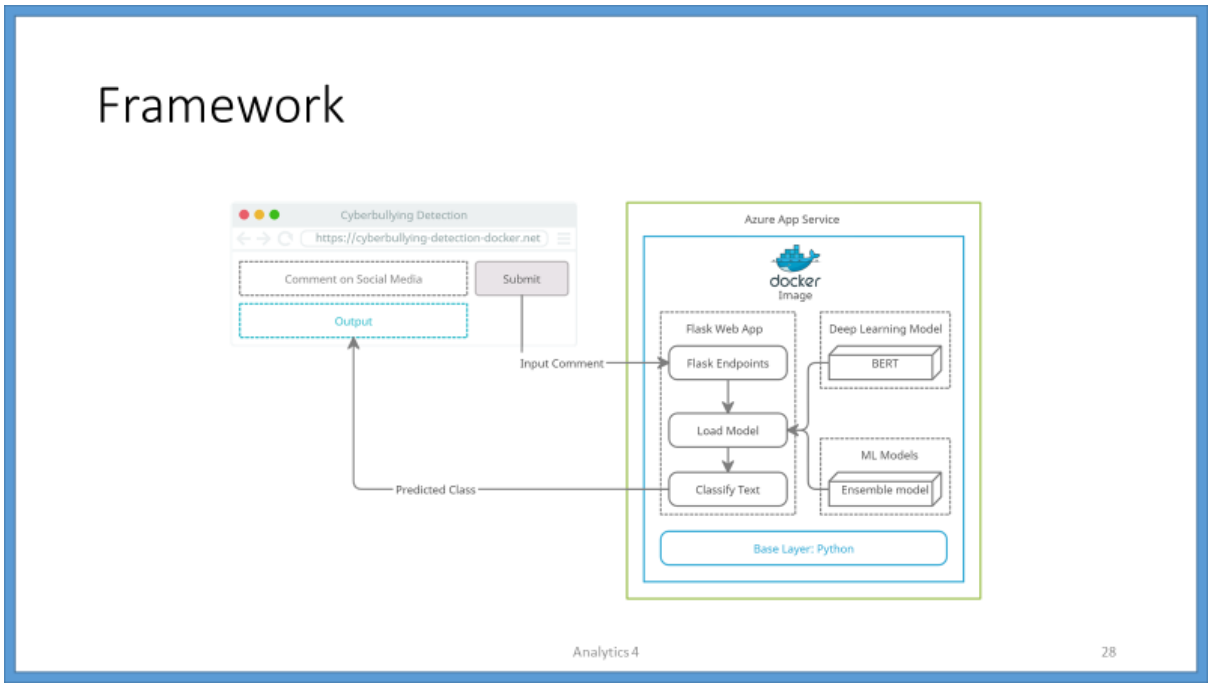
The endpoints are used to send and receive GET and POST requests to the Web browser.

The Flask Web App contains a class which loads the model.

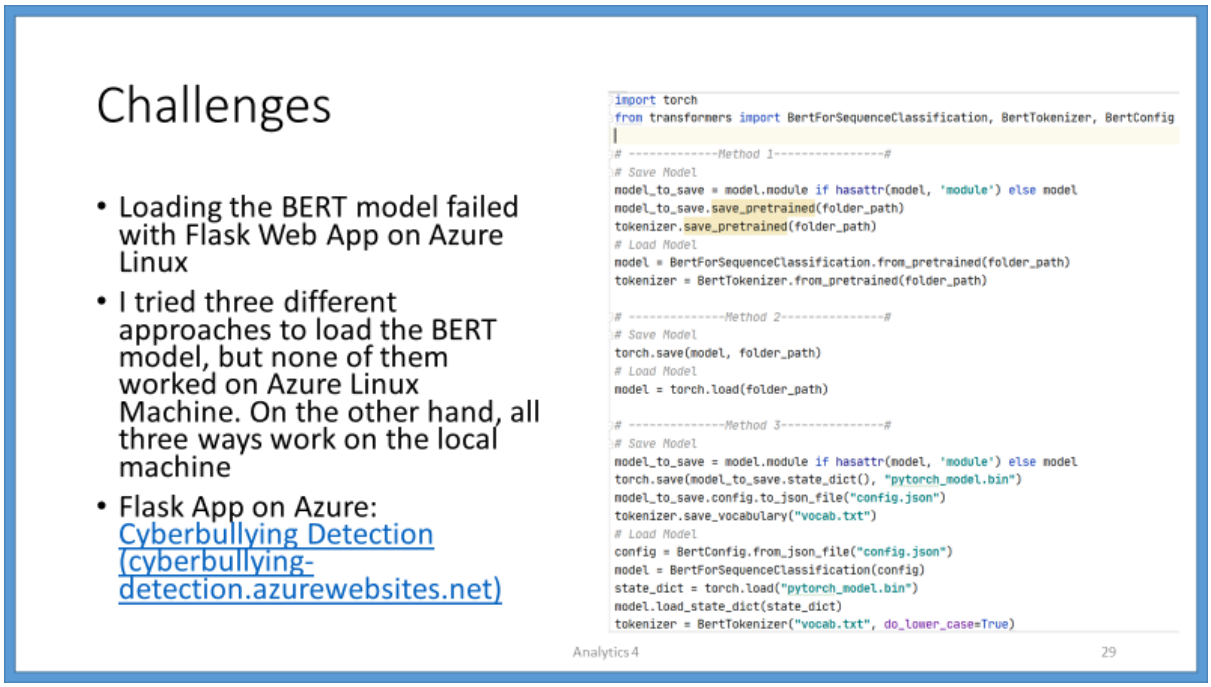
And another class use those models to detect cyberbullying text.

The complete Flask Web App is then containerize using Docker. And this container is running on a Python base layer.

And I deployed this complete docker image on Azure.



[Slide 29]



[Slide 30]

Solution

- Deployed the Flask App to Azure as a Docker container
- Docker image of Flask App on Azure: Cyberbullying Detection (cyberbullying-detection-docker.azurewebsites.net)

[Slide 31]

Future Scope

- Training BERT on data with more than two classes
- Comparison of BERT with Ensemble model for more than 2 classes

[Slide 32]

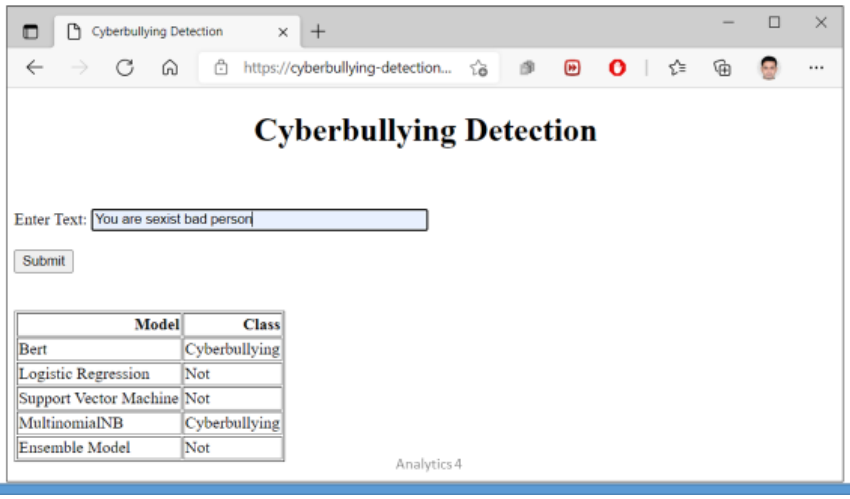
References

- Dataset 1: HASOC (hasocfire.github.io)
- Dataset 2: UMICH SI650 - Sentiment Classification | Kaggle
- What Is Cyberbullying | StopBullying.gov
- What is BERT | BERT For Text Classification (analyticsvidhya.com)
- BERT Fine-Tuning Tutorial with PyTorch · Chris McCormick (mccormickml.com)
- BERT for Natural Language Processing |All You Need to know about BERT (analyticsvidhya.com)

[Slide 33]

Demo

https://cyberbullying-detection-docker.azurewebsites.net/



[Slide 34]

Thank You!