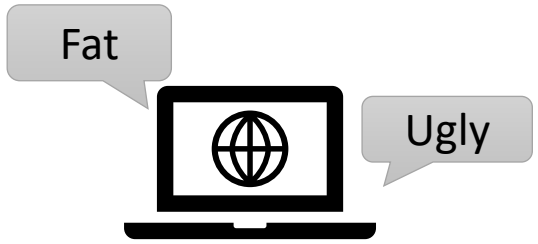
Cyberbullying Detection on Social Media

Anurag Rajendra Watane

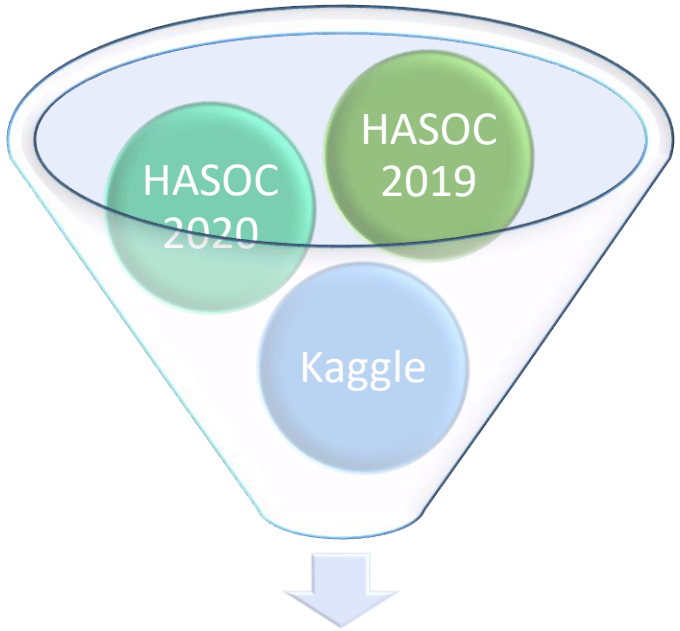
11013614

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

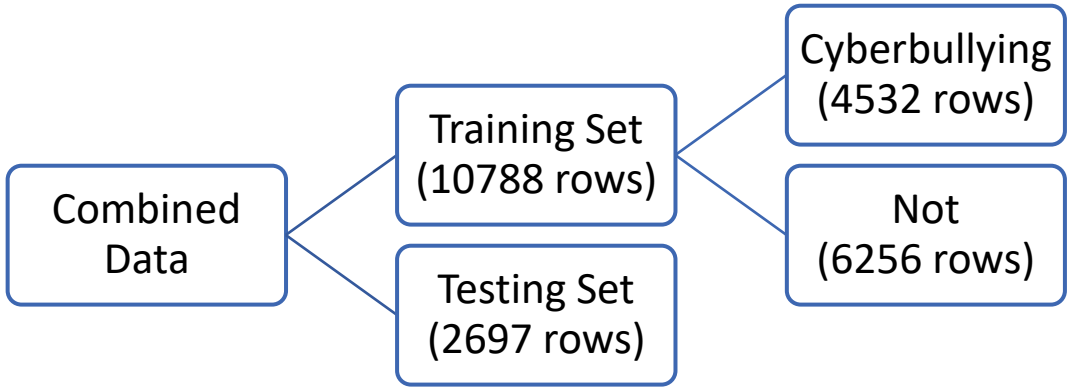


Twitter Dataset



Combined Data

Total 13485 rows

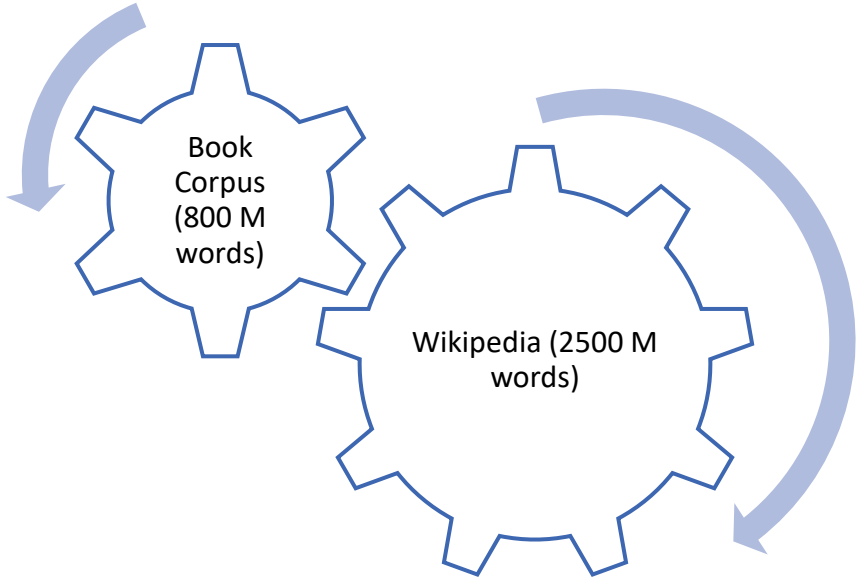


HASOC (hasocfire.github.io)

UMICH SI650 - Sentiment Classification | Kaggle

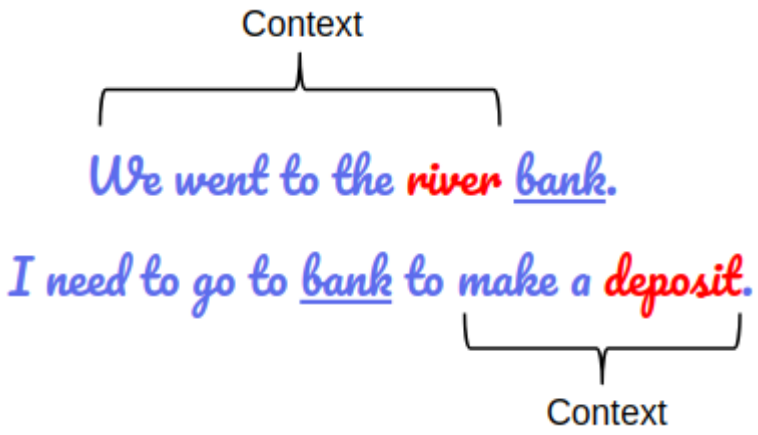
What is BERT?

- Bidirectional Encoder Representations from Transformers
- It is based on the Transformer architecture
- BERT is pre-trained on a large corpus of unlabelled text



Why Bidirectional?

- It 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



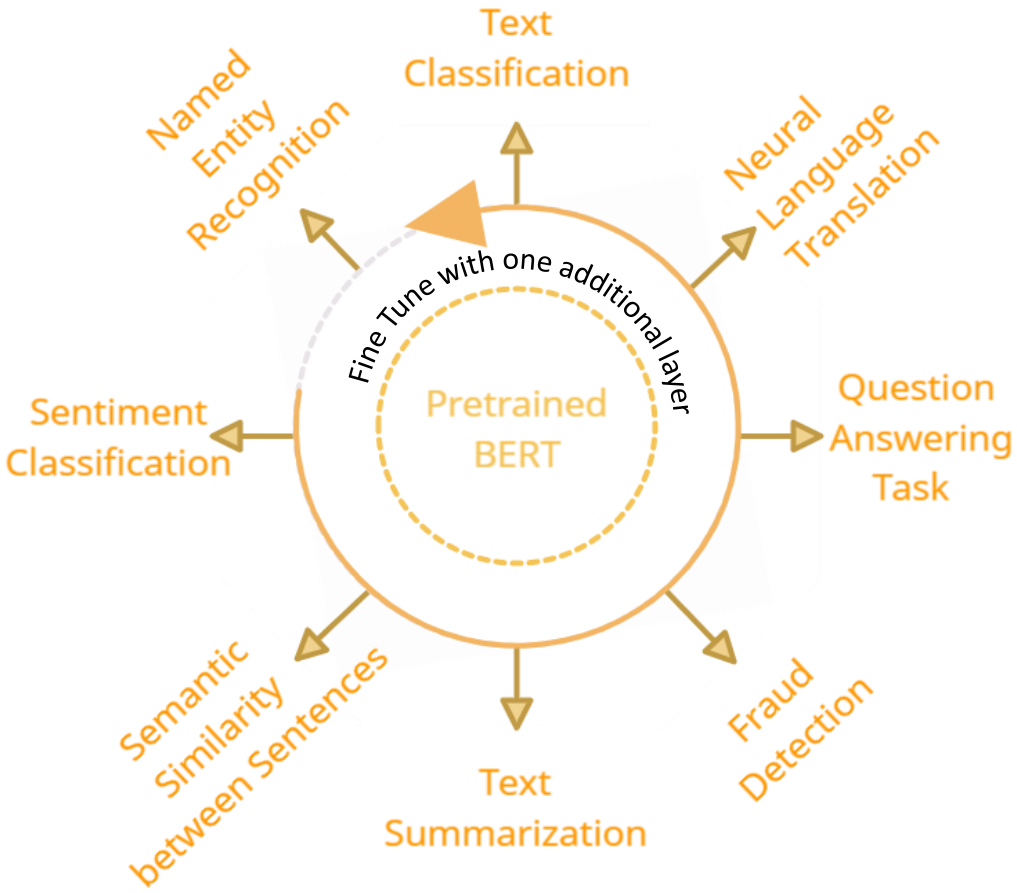
What is BERT | BERT For Text Classification (analyticsvidhya.com)

Why pretrained model?

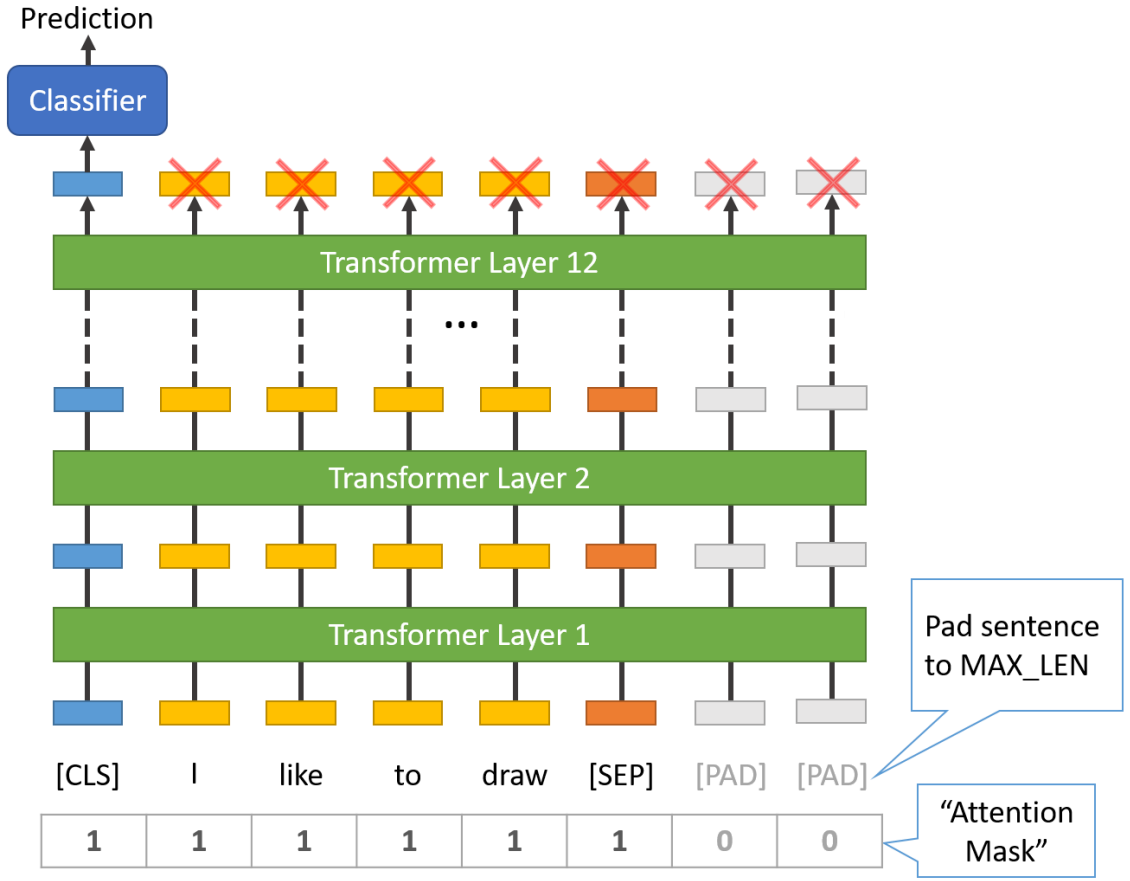
- The pre-trained BERT model weights already encode a lot of information about our language
- Because of the pre-trained weights this method allows us to fine-tune our task on a much smaller dataset
- It takes much less time to fine-tune the model

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.

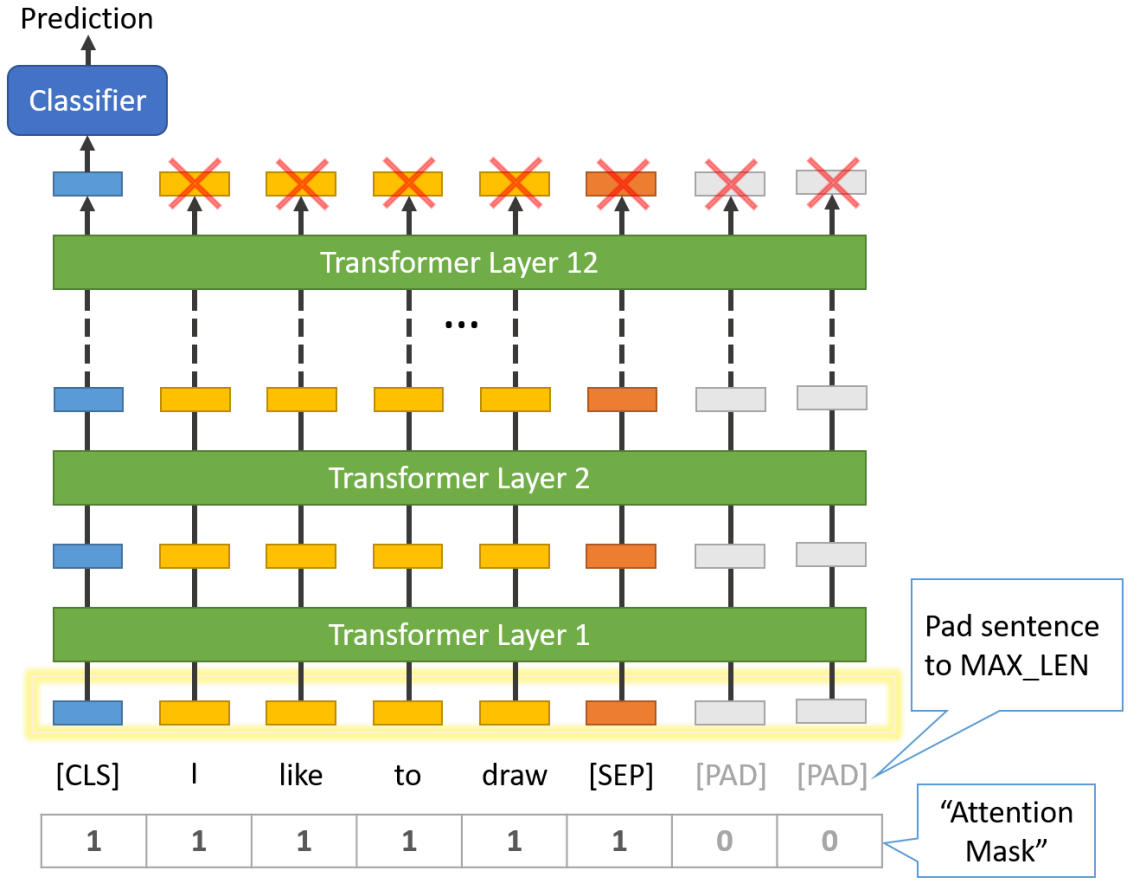


Architecture Used



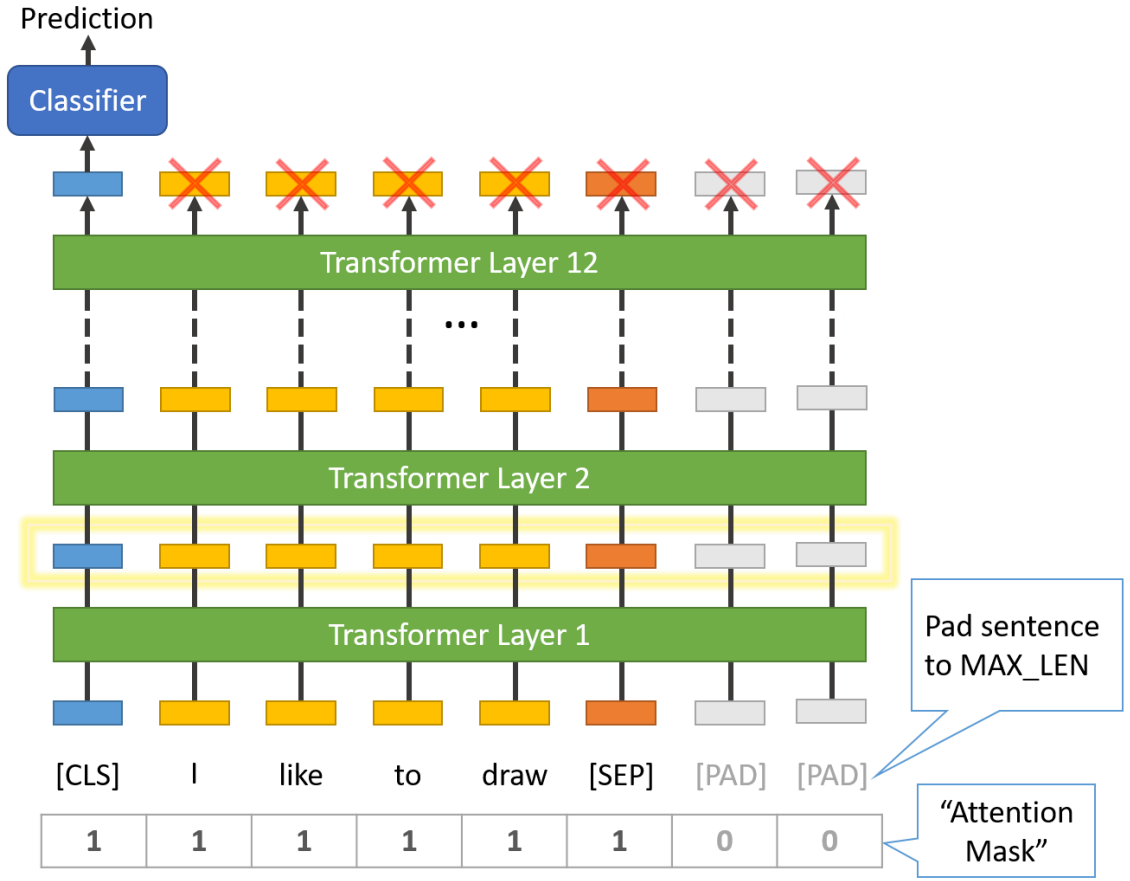
- 12-layer, 768-hidden, 12-heads, 110M parameters
- I have used pre-trained weights of 'bert-base-uncased' to fine-tune the model on Twitter dataset
- This model is uncased: it does not make a difference between **e**nglish and **E**nglish

Architecture Used



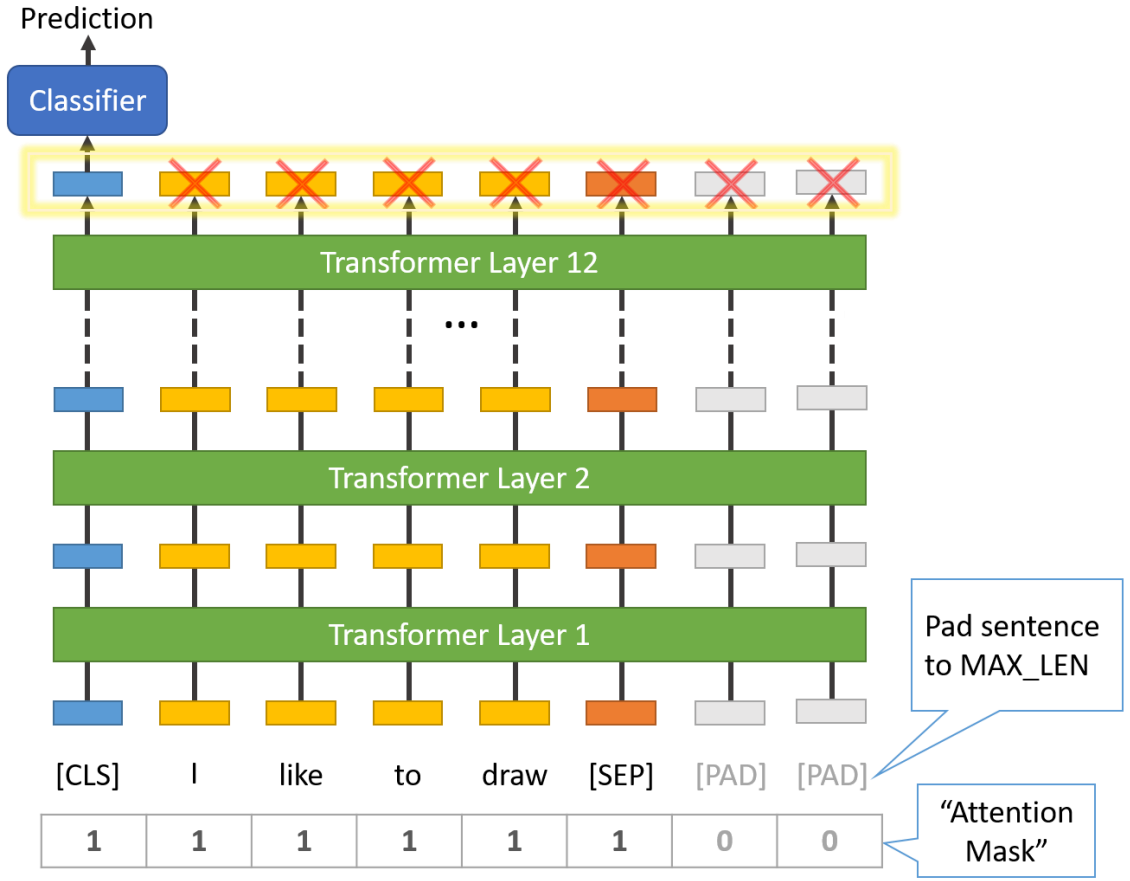
- 12-layer, 768-hidden, 12-heads, 110M parameters
- Pre-trained weights of 'bert-base-uncased' are used to fine-tune the model on Twitter dataset
- This model is uncased: it does not make a difference between **e**nglish and **E**nglish

Architecture Used



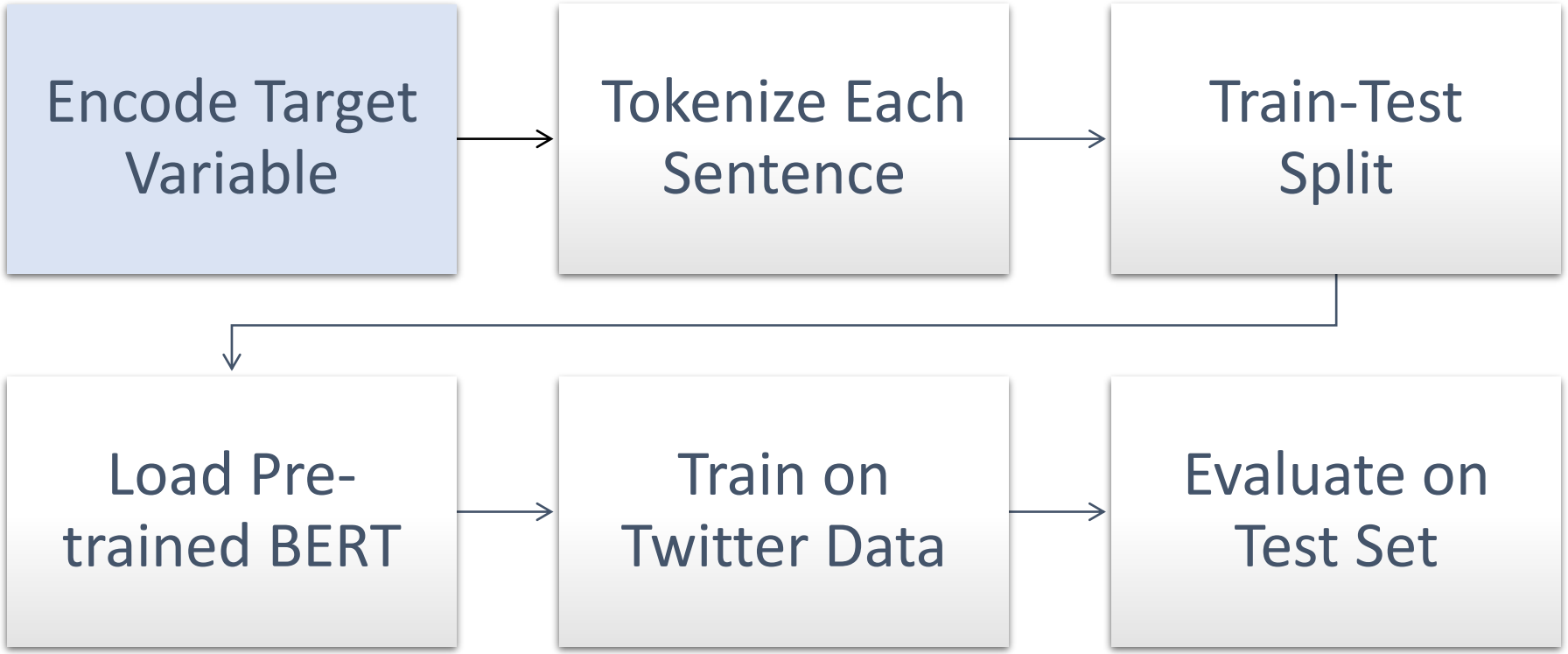
- 12-layer, 768-hidden, 12-heads, 110M parameters
- I have used pre-trained weights of 'bert-base-uncased' to fine-tune the model on Twitter dataset
- This model is uncased: it does not make a difference between **e**nglish and **E**nglish

Architecture Used

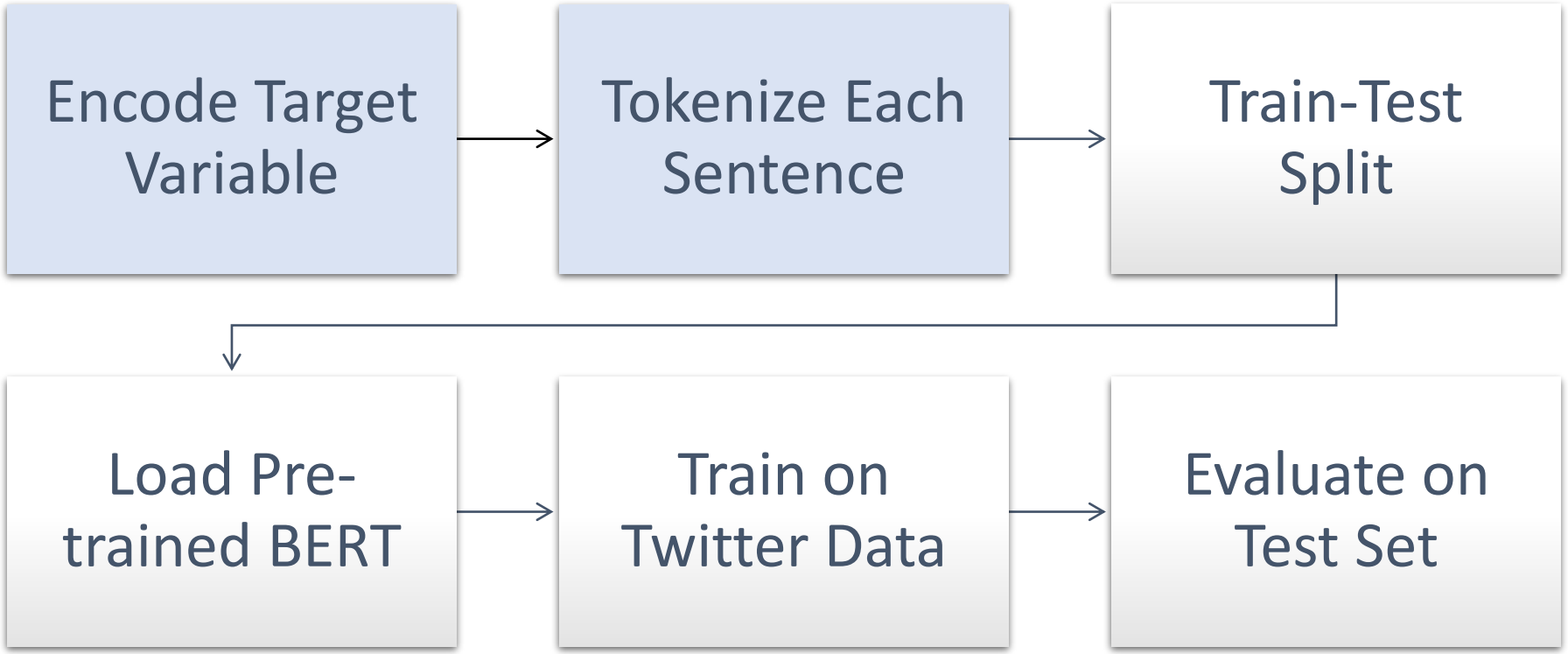


- 12-layer, 768-hidden, 12-heads, 110M parameters
- I have used pre-trained weights of 'bert-base-uncased' to fine-tune the model on Twitter dataset
- This model is uncased: it does not make a difference between **e**nglish and **E**nglish

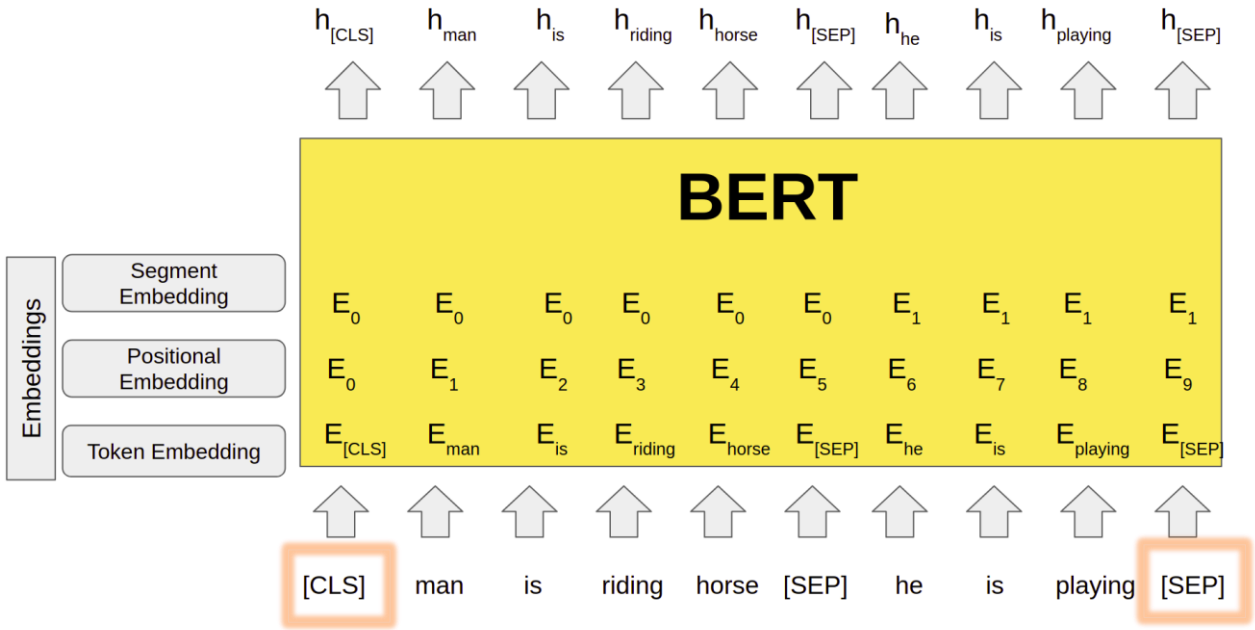
Steps for fine-tuning BERT



Steps for fine-tuning BERT



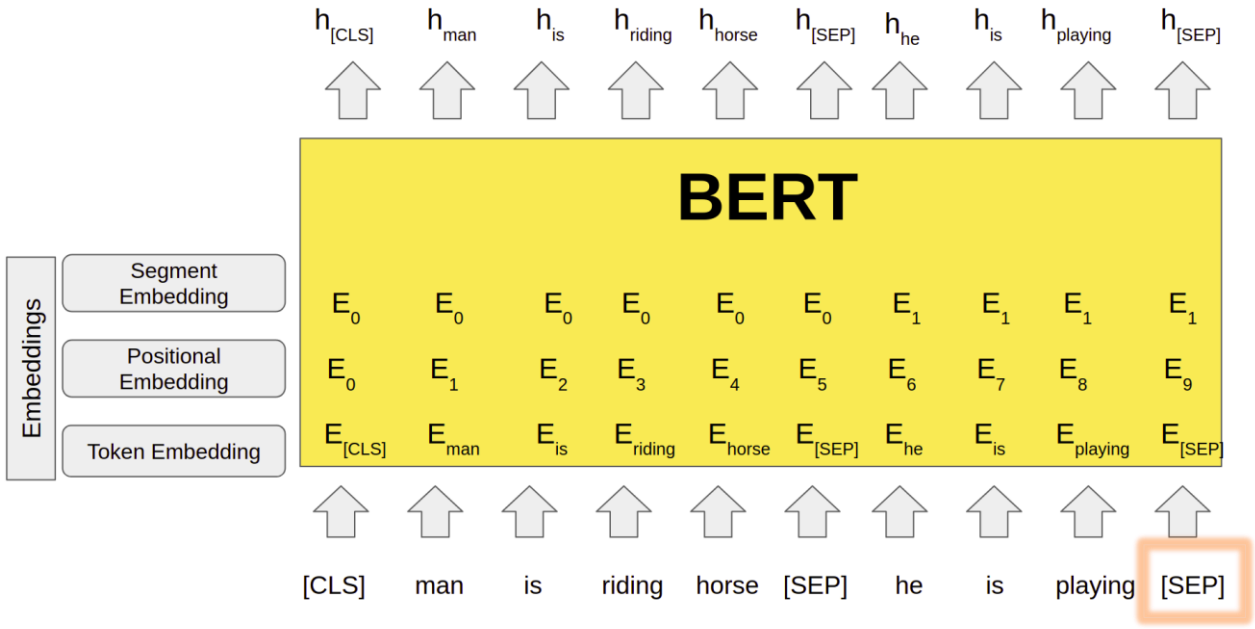
Tokenization for BERT



- 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

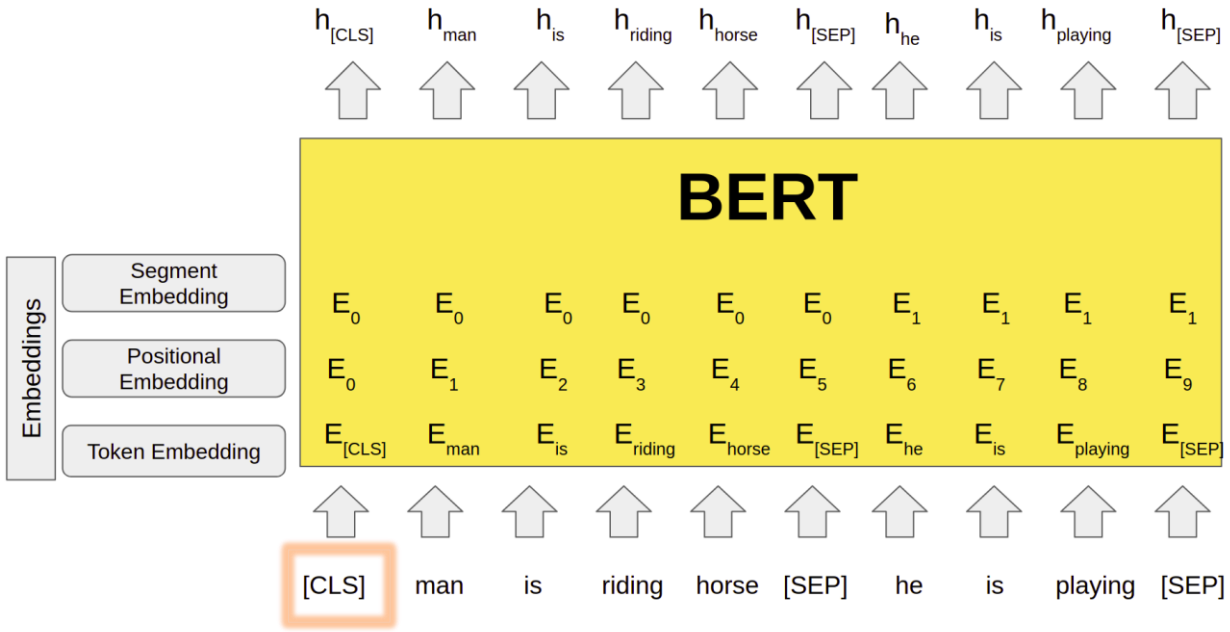
BERT for Natural Language Processing |All You Need to know about BERT (analyticsvidhya.com)

Tokenization for BERT



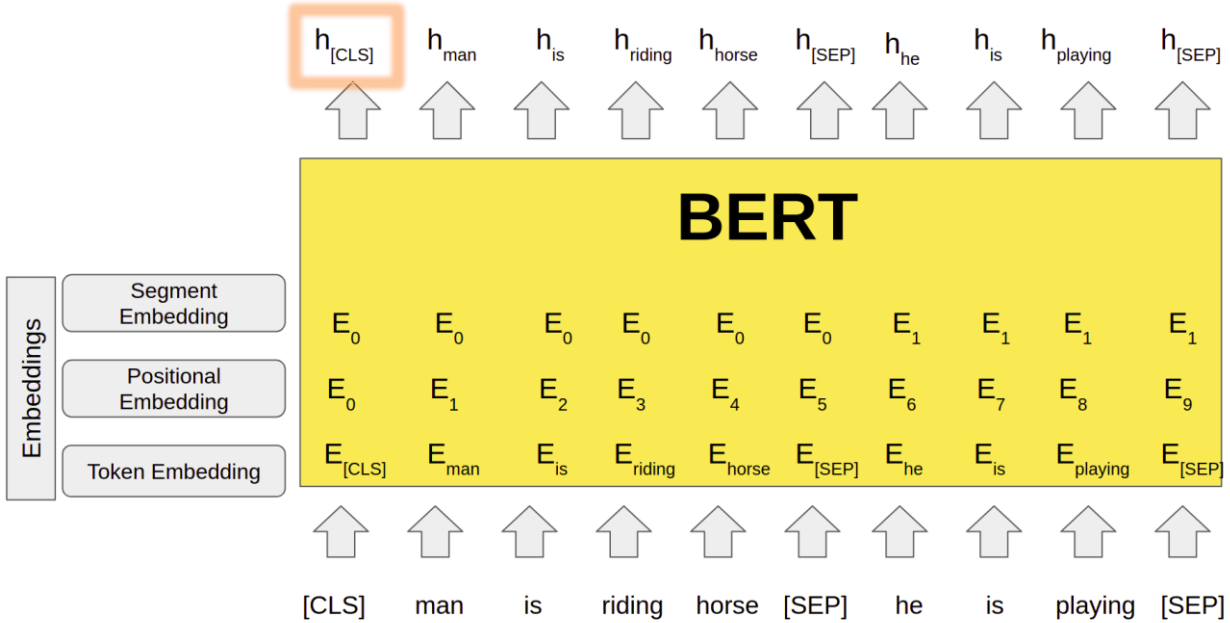
- BERT uses special tokens [CLS] and [SEP] to understand input properly
- [SEP] token has to be inserted at the end of every sentence
- The first token of every input sequence is the special classification token – [CLS]
- The last hidden state of BERT, corresponding to this token (h[CLS]) is used in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

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

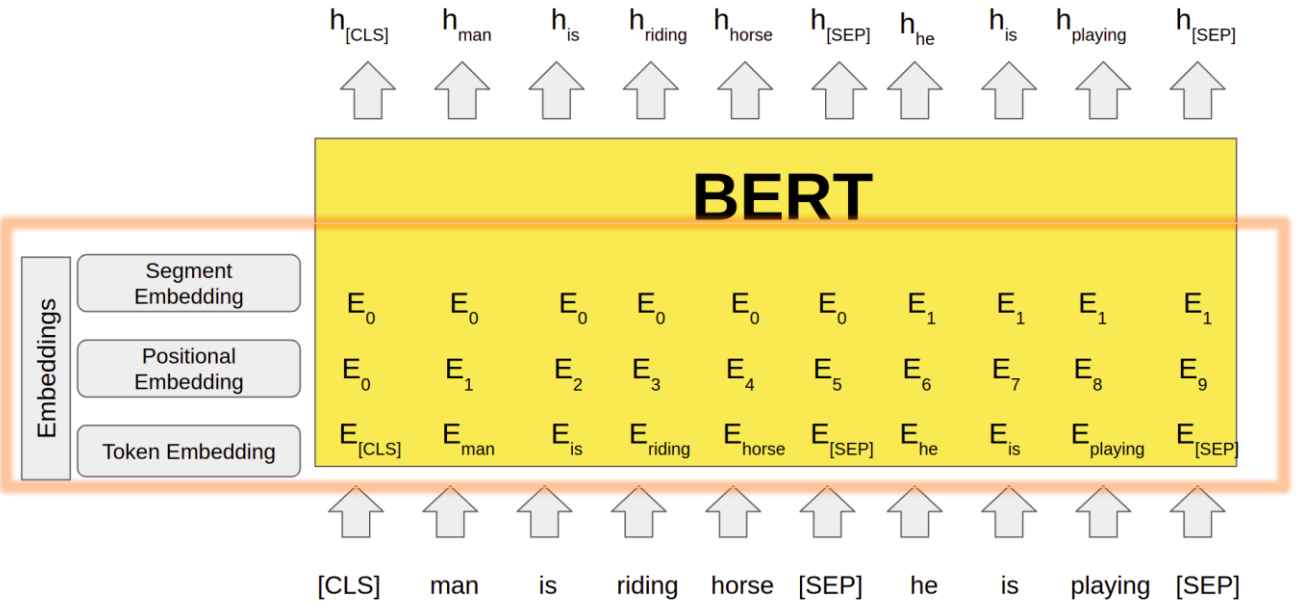
Tokenization for BERT



- 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

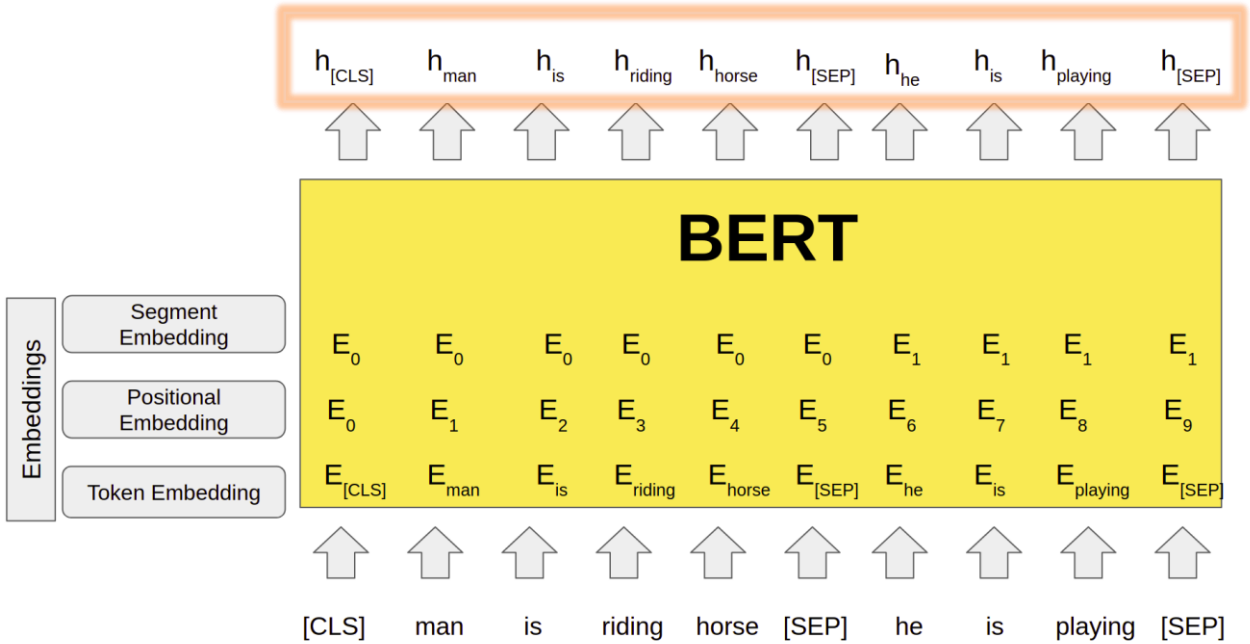
BERT for Natural Language Processing |All You Need to know about BERT (analyticsvidhya.com)

Tokenization for BERT



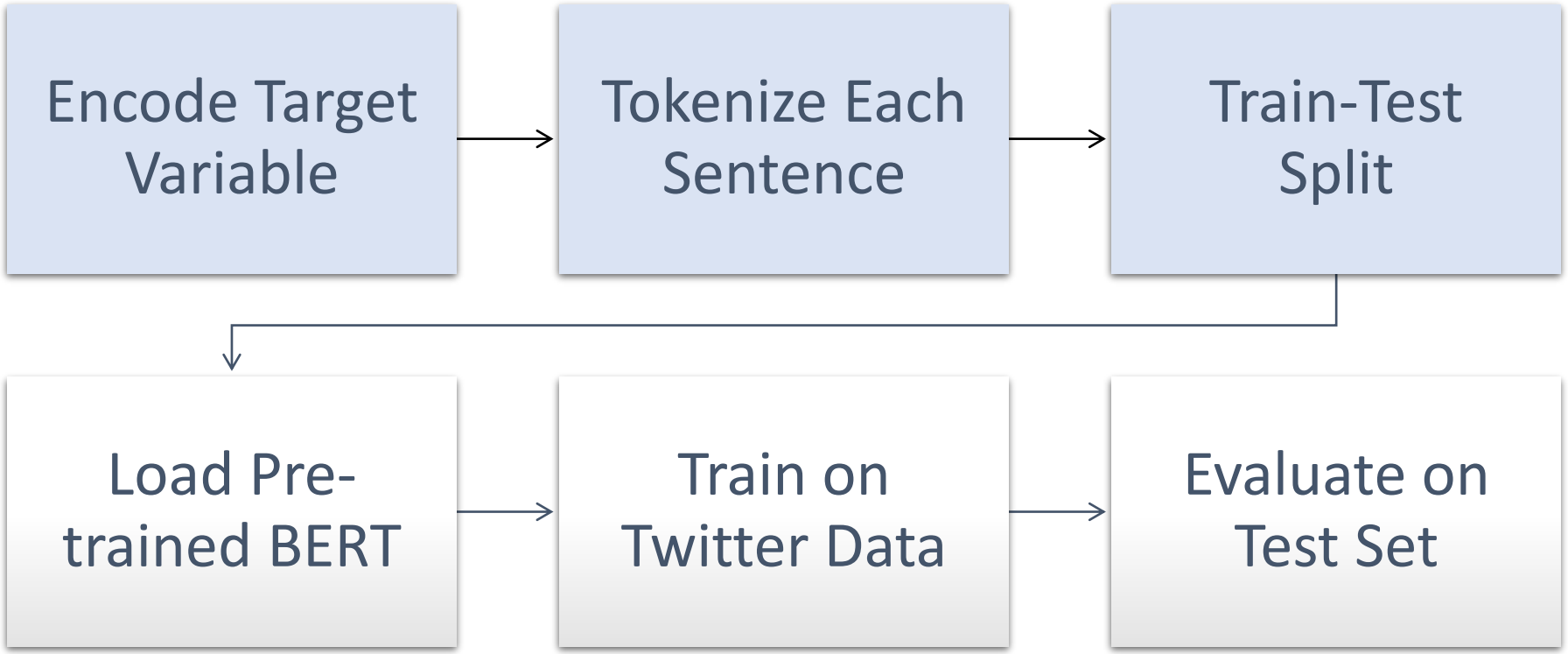
- BERT uses special tokens [CLS] and [SEP] to understand input properly
- [SEP] token has to be inserted at the end of every sentence
- The first token of every input sequence is the special classification token – [CLS]
- The last hidden state of BERT, corresponding to this token (h[CLS]) is used in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

Tokenization for BERT

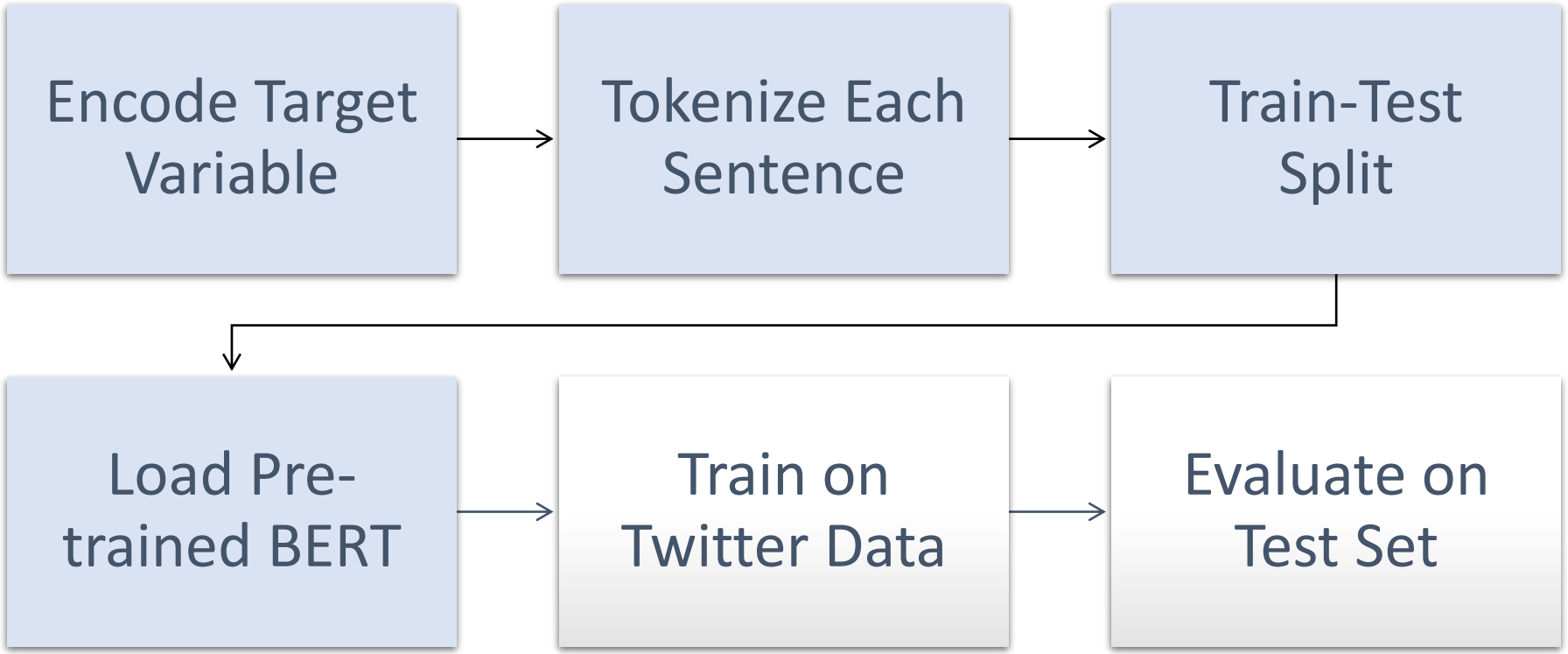


- BERT uses special tokens [CLS] and [SEP] to understand input properly
- [SEP] token has to be inserted at the end of every sentence
- The first token of every input sequence is the special classification token – [CLS]
- The last hidden state of BERT, corresponding to this token (h[CLS]) is used in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

Steps for fine-tuning BERT



Steps for fine-tuning BERT

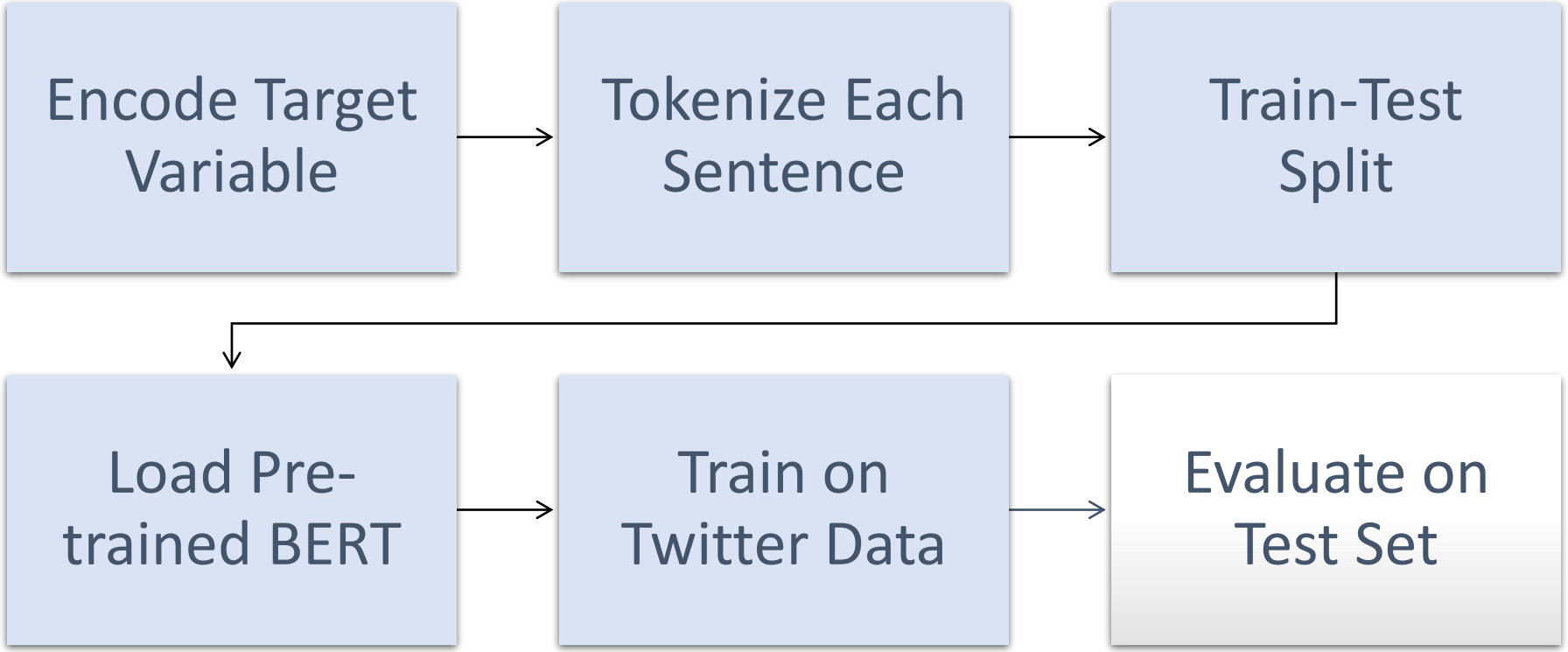


Load pretrained BERT

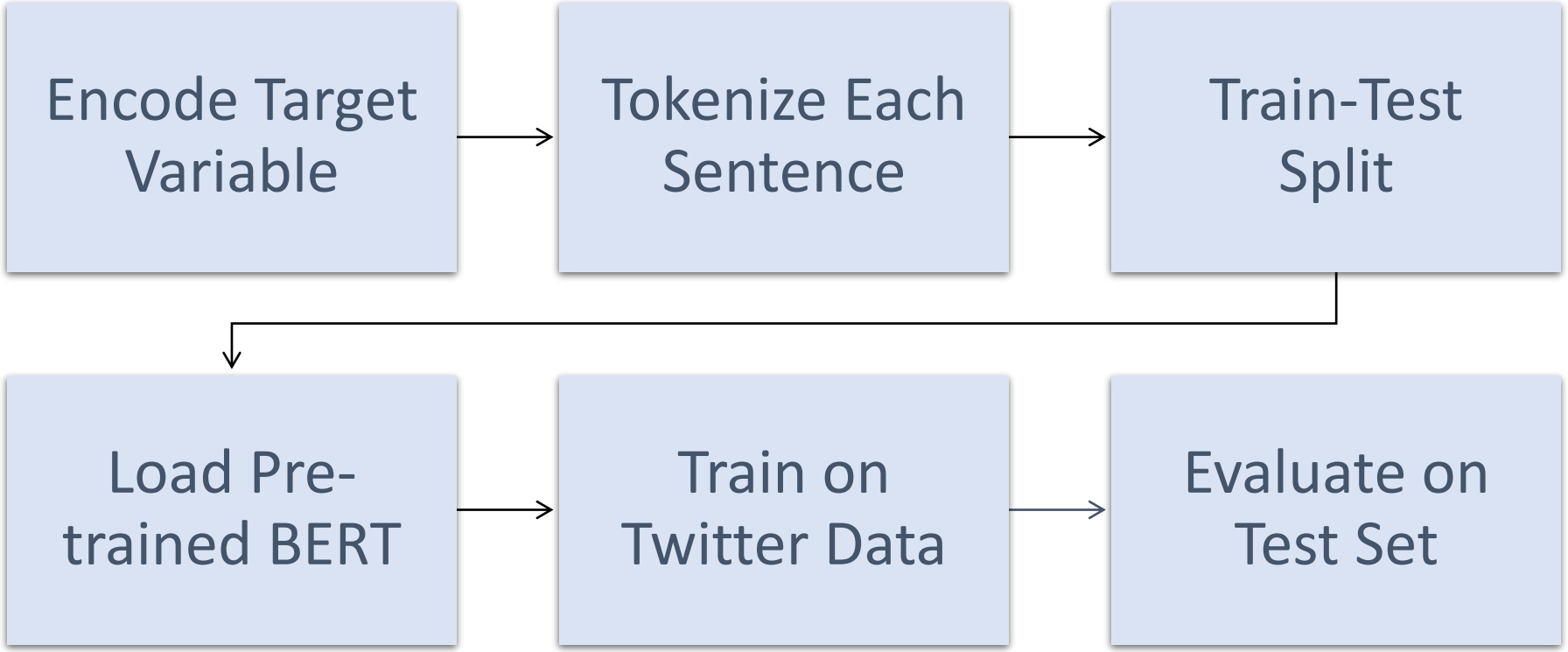
```
from transformers import BertForSequenceClassification
model = BertForSequenceClassification.from_pretrained(
    "bert-base-uncased",           # Use the 12-layer BERT model, with an uncased vocab
    num_labels = 2,                # The number of output labels--2 for binary classification
    output_attentions = 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

Steps for fine-tuning BERT



Steps for fine-tuning BERT



F1 Score: BERT

```
accurate = 0
for (i,j) in zip(flat_predictions, flat_true_labels):
    if i==j:
        accurate += 1
accurate/len(flat_predictions)
```

```
0.8894348894348895
```

```
from sklearn.metrics import f1_score
f1_score(flat_true_labels, flat_predictions, average='macro')
```

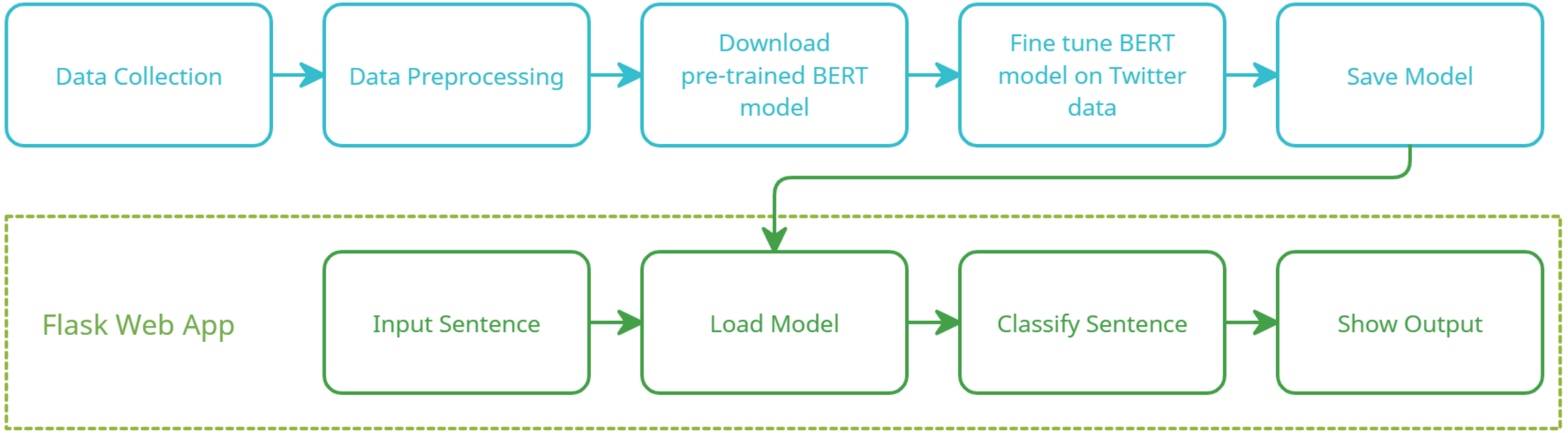
```
0.8894182002608317
```

F1 Score: Ensemble Model

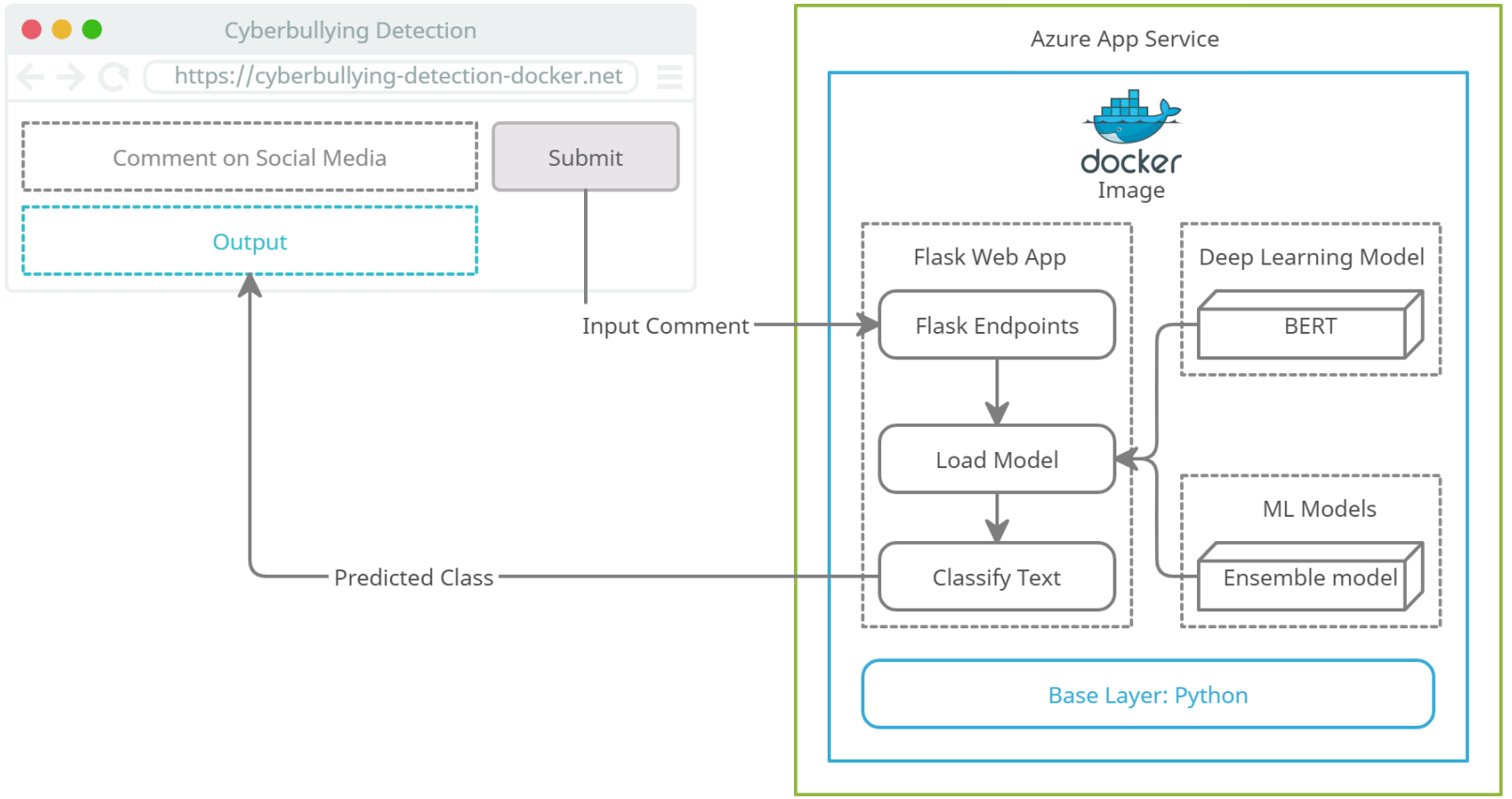
```
#Train and evaluation
for clf in (log_clf, svm_clf, mnb_clf, voting_clf):
    clf.fit( X_train_counts, y_train )
    y_pred = clf.predict(X_test_counts)
    print(clf.__class__.__name__, f1_score(y_test, y_pred, average = 'macro'))
```

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

Pipeline


Data Collection
Data Preprocessing
Download pre-trained BERT model
Fine tune BERT model on Twitter data
Save Model
Flask Web App
Input Sentence
Load Model
Classify Sentence
Show Output

Framework


Cyberbullying Detection
https://cyberbullying-detection-docker.net
Comment on Social Media
Submit
Output
Input Comment
Predicted Class
Azure App Service
docker
Image
Flask Web App
Flask Endpoints
Load Model
Classify Text
Deep Learning Model
BERT
ML Models
Ensemble model
Base Layer: Python

Challenges

- Loading the BERT model failed with Flask Web App on Azure Linux
- I tried three different approaches to load the BERT model, but none of them worked on Azure Linux Machine. On the other hand, all three ways work on the local machine
- Flask App on Azure: [Cyberbullying Detection (cyberbullying-detection.azurewebsites.net)]

```python
import torch
from transformers import BertForSequenceClassification, BertTokenizer, BertConfig

# -------------Method 1---------------#
# Save Model
model_to_save = model.module if hasattr(model, 'module') else model
model_to_save.save_pretrained(folder_path)
tokenizer.save_pretrained(folder_path)
# Load Model
model = BertForSequenceClassification.from_pretrained(folder_path)
tokenizer = BertTokenizer.from_pretrained(folder_path)

# -------------Method 2--------------#
# Save Model
torch.save(model, folder_path)
# Load Model
model = torch.load(folder_path)

# -------------Method 3--------------#
# Save Model
model_to_save = model.module if hasattr(model, 'module') else model
torch.save(model_to_save.state_dict(), "pytorch_model.bin")
model_to_save.config.to_json_file("config.json")
tokenizer.save_vocabulary("vocab.txt")
# Load Model
config = BertConfig.from_json_file("config.json")
model = BertForSequenceClassification(config)
state_dict = torch.load("pytorch_model.bin")
model.load_state_dict(state_dict)
tokenizer = BertTokenizer("vocab.txt", do_lower_case=True)
```

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)

Future Scope

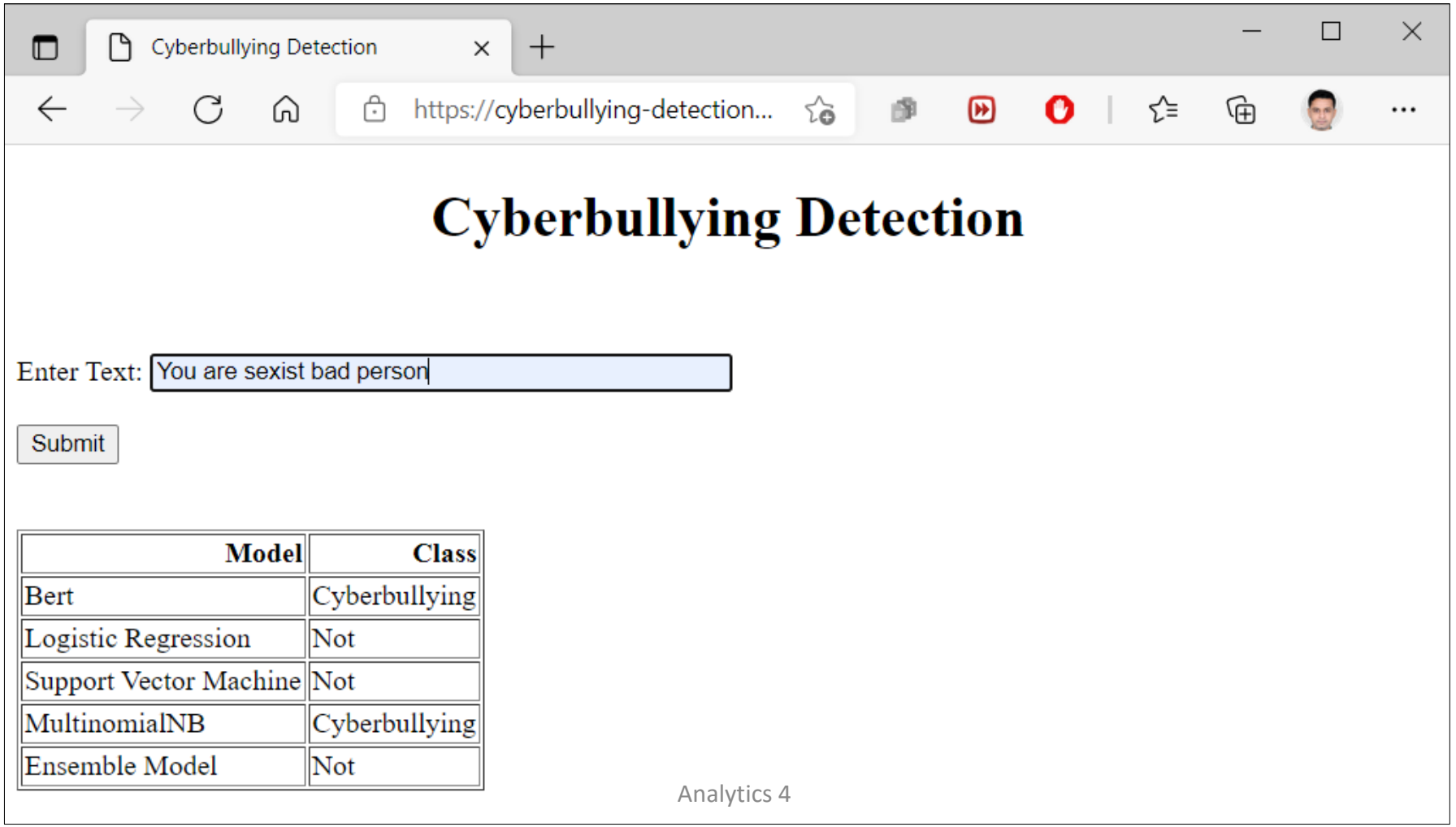
- Training BERT on data with more than two classes
- Comparison of BERT with Ensemble model for more than 2 classes

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)

Demo

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



Thank You!