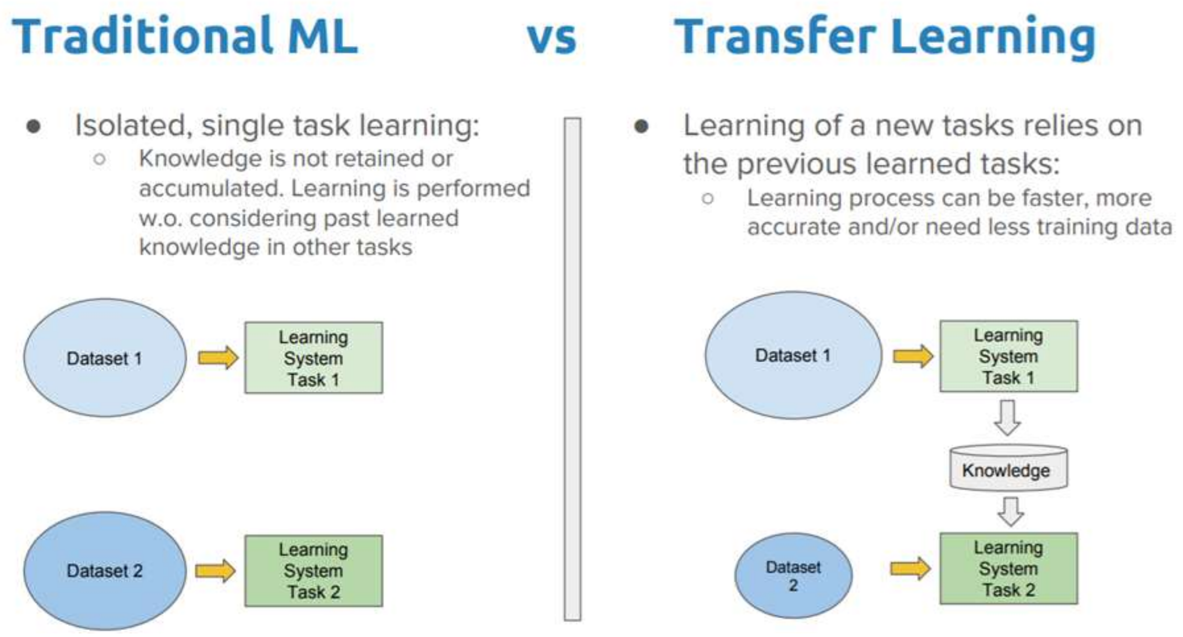
Fine Tuning Pretrained BERT Model for Named Entity Recognition

Name - Kanishka Parganiha

Date – 12/07/2020

Transfer Learning



Traditional ML vs **Transfer Learning**

- Isolated, single task learning:
  - Knowledge is not retained or accumulated. Learning is performed w.o. considering past learned knowledge in other tasks

- Learning of a new tasks relies on the previous learned tasks:
  - Learning process can be faster, more accurate and/or need less training data

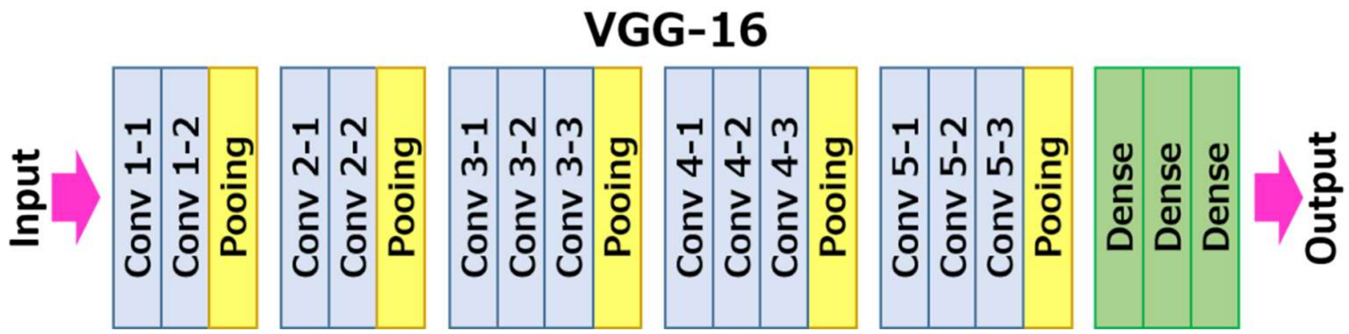
Pretrained Model

A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem similar to the one that we want to solve.

Examples:

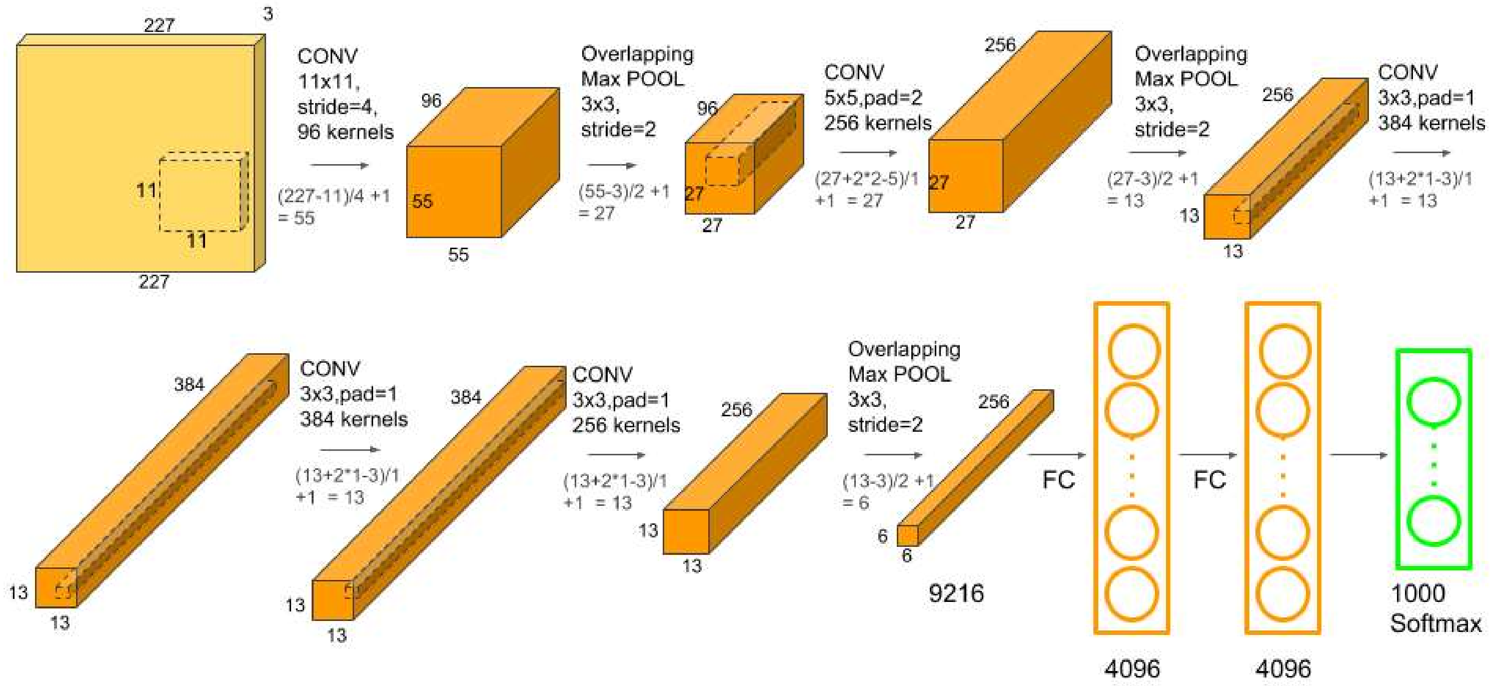
1. Very Deep Convolutional Networks for Large-Scale Image Recognition(VGG-16):

- Trained on 14 million images belonging to 1000 classes.
- VGG16 was trained for weeks using NVIDIA Titan Black GPU’s.



2. AlexNet:

- Trained over 15 million labeled high-resolution images in over 22,000 categories
- Consists of 5 Convolutional Layers and 3 Fully Connected Layers
- One network takes five and six days to train on two GTX 580 3GB GPUs

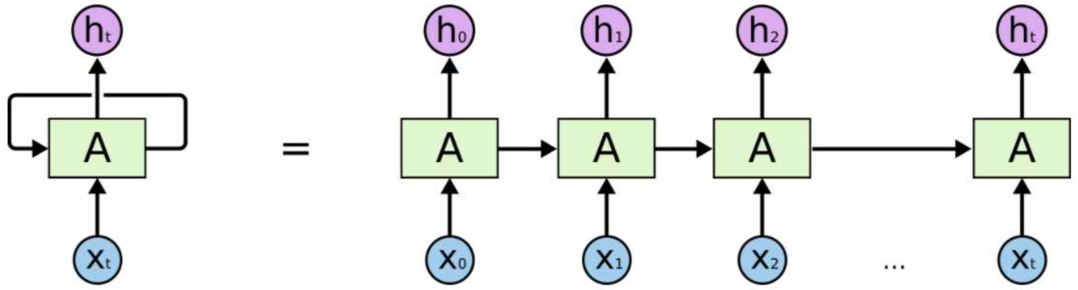


What is Named Entity Recognition?

- NER is a task of extracting information from the sequence of words and sentences and classifying them into pre-defined categories.

Sequence Modeling

- Task of predicting the output where the current output is dependent upon the previous input.



An unrolled recurrent neural network.

Drawbacks:

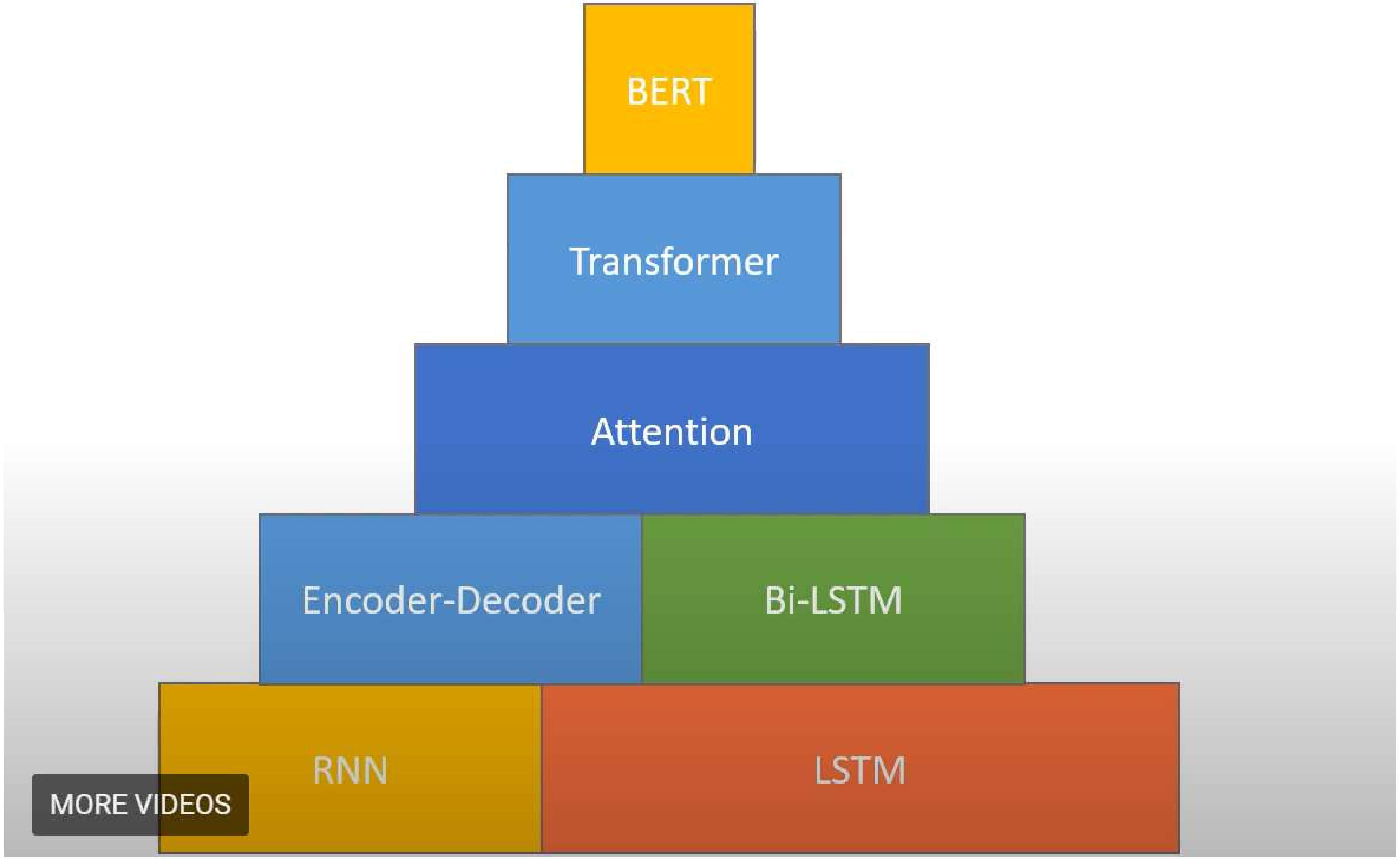
- Don’t support parallel computation
- Larger the sequential data, larger the training time
- Larger the data more the time steps of backpropogation
- Problem of Vanishing Gradient

Bi-Directional Encoder Decoder Transformers (BERT)

- Introduced by Google, BERT is a PreTrained Model that consists of layers of Transformers and uses Attention Mechanism
- Trained BooksCorpus (800M words) and English Wikipedia (2,500M words)

Pros:

- Supports Parallel computation
- Computation per layer requires 1 single step
- No problem of Vanishing gradient
- Uses Positional Encoding to capture the position information data


BERT
Transformer
Attention
Encoder-Decoder
Bi-LSTM
RNN
LSTM
MORE VIDEOS

Model Architecture of BERT

- Multi-layer bidirectional Transformer encoder

BERTBASE (L=12, H=768, A=12, Total Parameters=110M)

BERTLARGE (L=24, H=1024,A=16, Total Parameters=340M).

Where L=Number of Layers, H = Hidden Size and A = Number of Attention

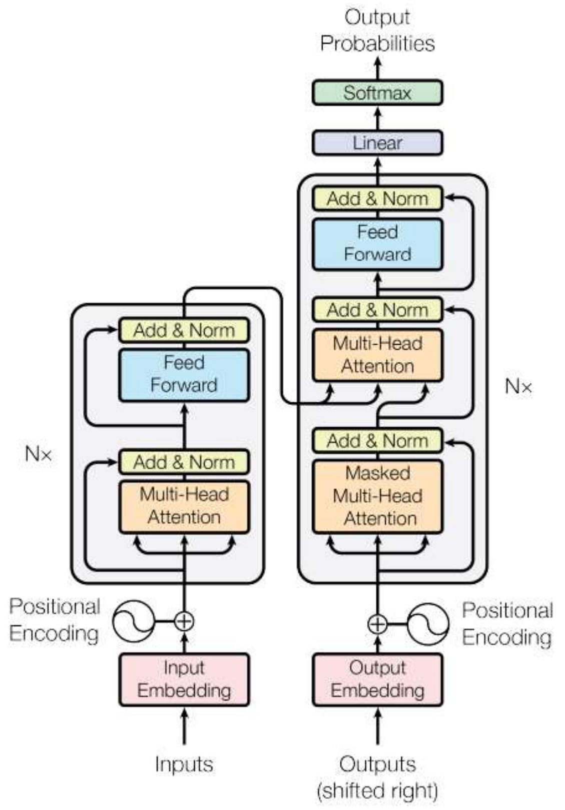
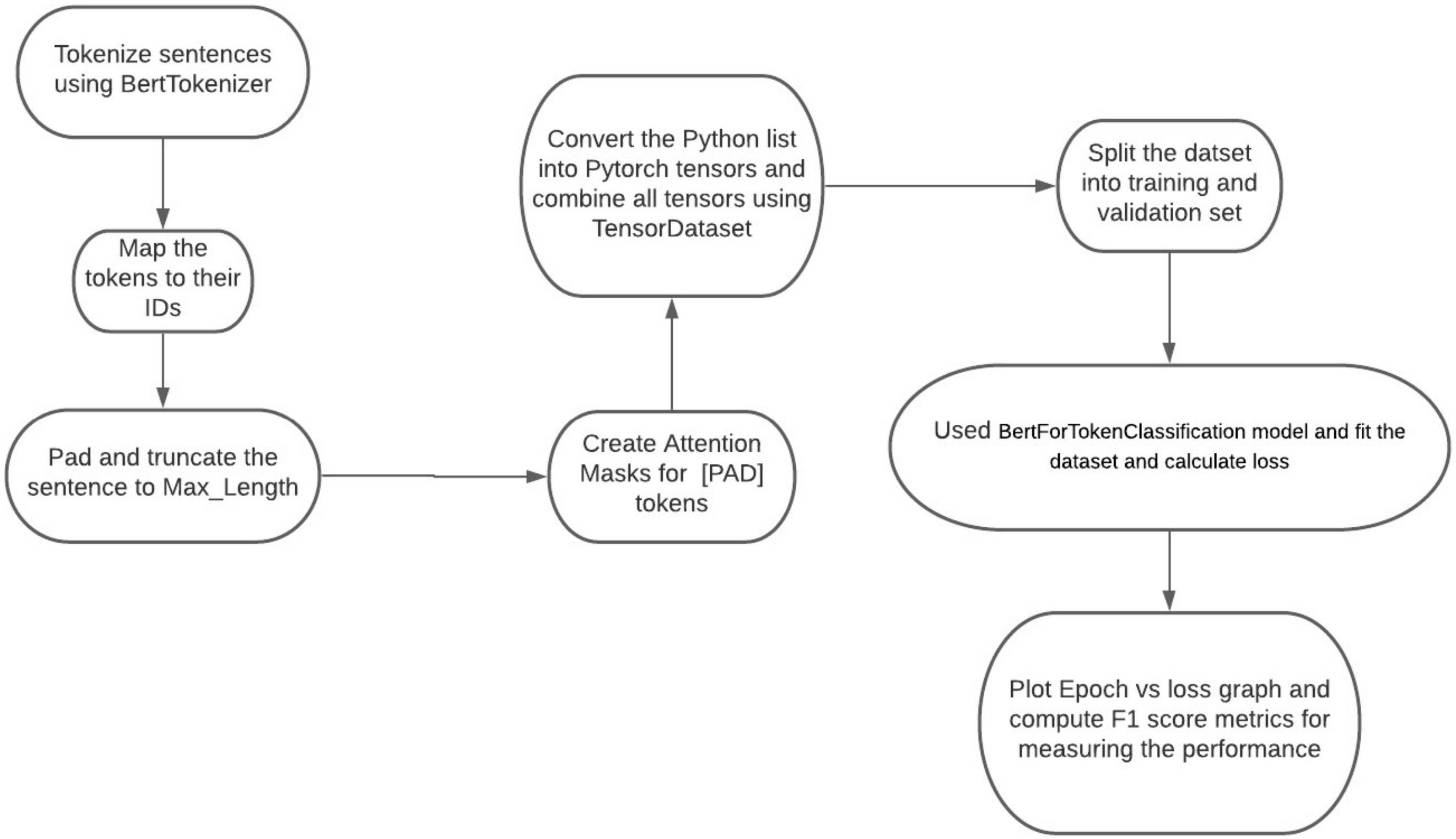
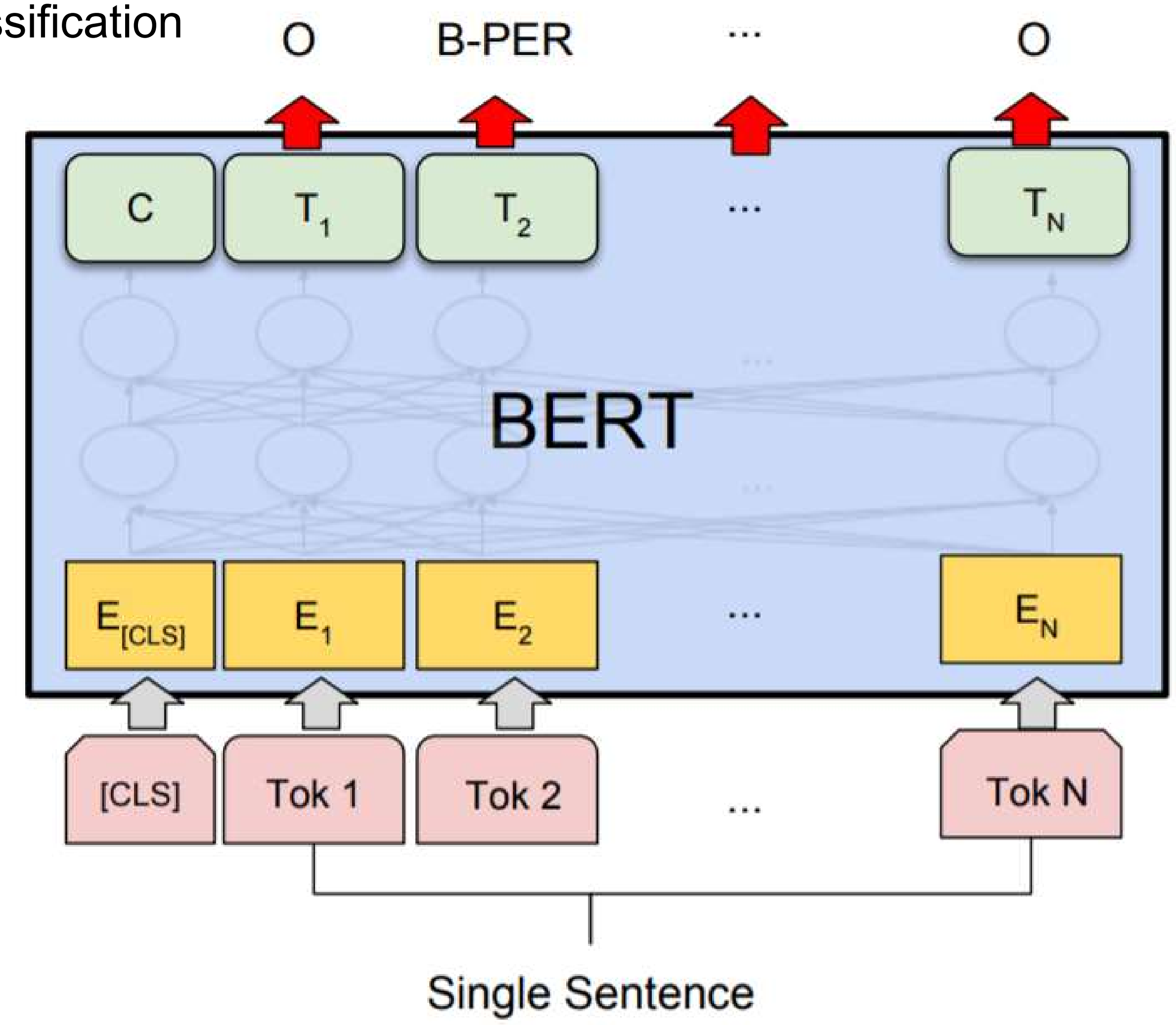


Figure 1: The Transformer - model architecture.

Approach:


Tokenize sentences using BertTokenizer
Map the tokens to their IDs
Pad and truncate the sentence to Max_Length
Create Attention Masks for [PAD] tokens
Convert the Python list into Pytorch tensors and combine all tensors using TensorDataset
Split the datset into training and validation set
Used BertForTokenClassification model and fit the dataset and calculate loss
Plot Epoch vs loss graph and compute F1 score metrics for measuring the performance

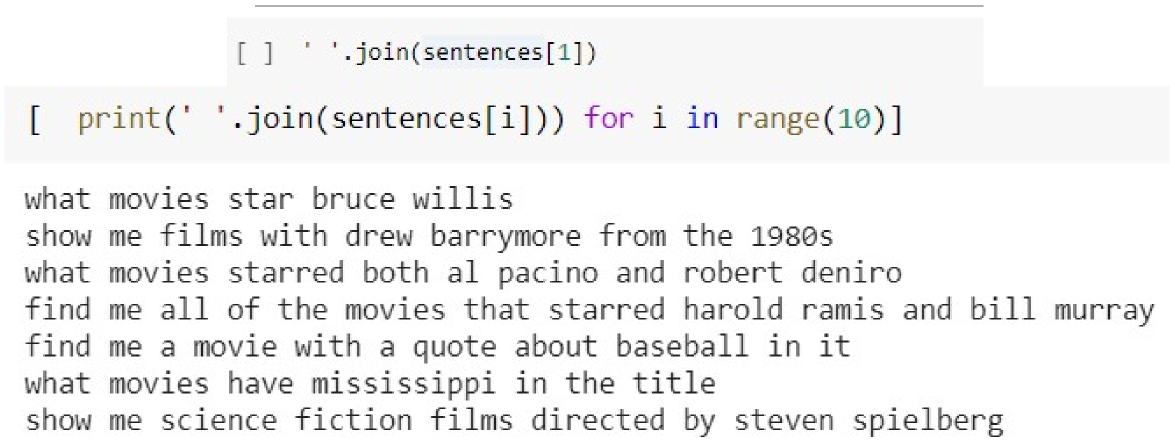
BertForTokenClassification



Dataset Used: MIT Movie dataset

url_train='https://groups.csail.mit.edu/sls/downloads/movie/engtrain.bio'
url_test='https://groups.csail.mit.edu/sls/downloads/movie/engtest.bio'

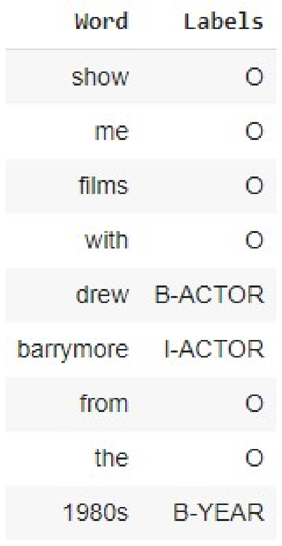
Dataset format: BIO / IOB format (short for inside, outside, beginning) is a common tagging format for tagging tokens in a chunking task in computational linguistics



```
[ ]  ' '.join(sentences[1])
```

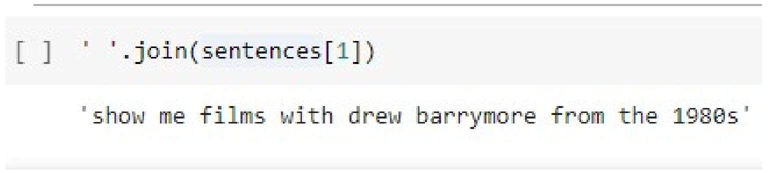
```
[  print(' '.join(sentences[i])) for i in range(10)]
```

```
what movies star bruce willis
show me films with drew barrymore from the 1980s
what movies starred both al pacino and robert deniro
find me all of the movies that starred harold ramis and bill murray
find me a movie with a quote about baseball in it
what movies have mississippi in the title
show me science fiction films directed by steven spielberg
```

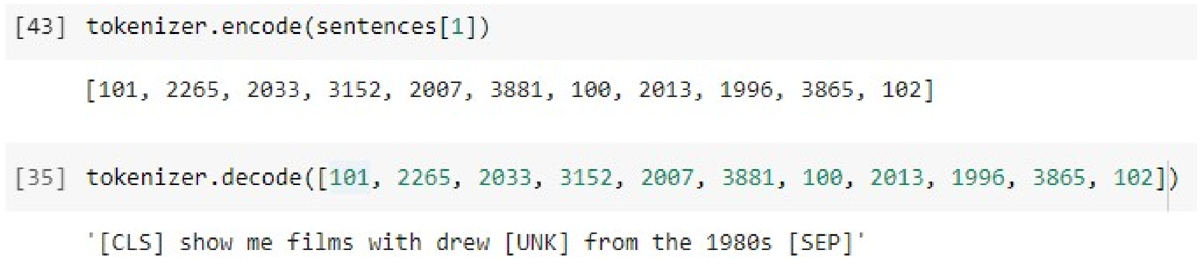


Word	Labels
show	O
me	O
films	O
with	O
drew	B-ACTOR
barrymore	I-ACTOR
from	O
the	O
1980s	B-YEAR

Tokenization of Input text

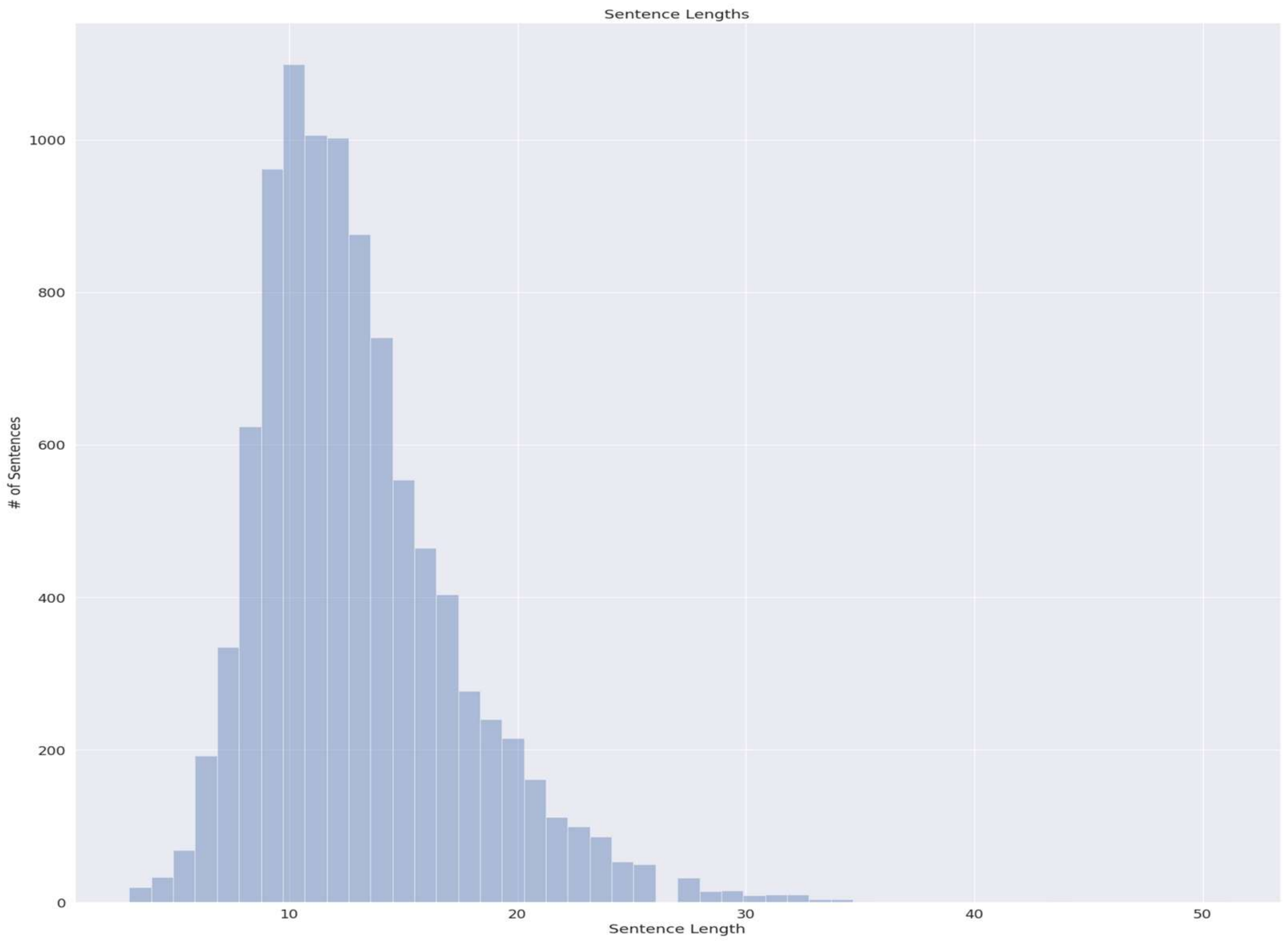


```
[ ]  ' '.join(sentences[1])

     'show me films with drew barrymore from the 1980s'
```



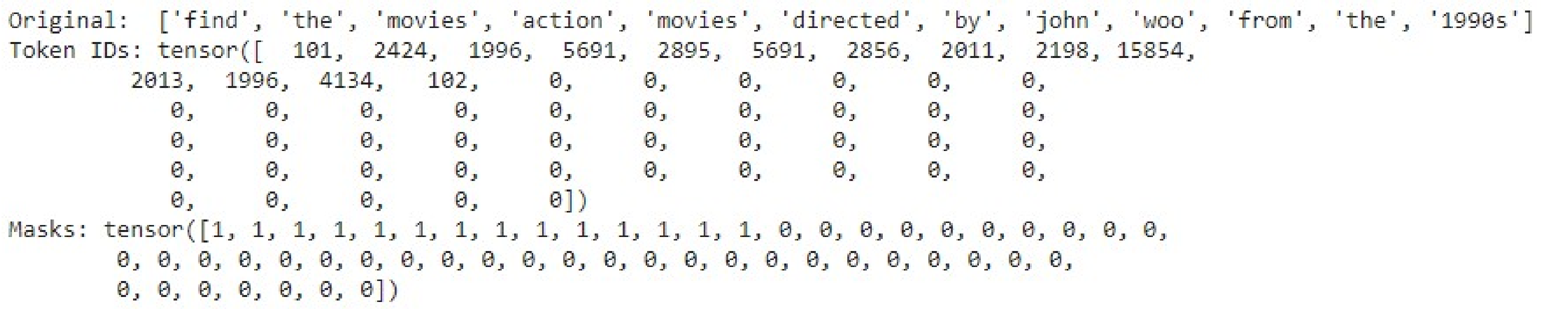
```
[43] tokenizer.encode(sentences[1])

     [101, 2265, 2033, 3152, 2007, 3881, 100, 2013, 1996, 3865, 102]

[35] tokenizer.decode([101, 2265, 2033, 3152, 2007, 3881, 100, 2013, 1996, 3865, 102])

     '[CLS] show me films with drew [UNK] from the 1980s [SEP]'
```

Histogram for length of sentences

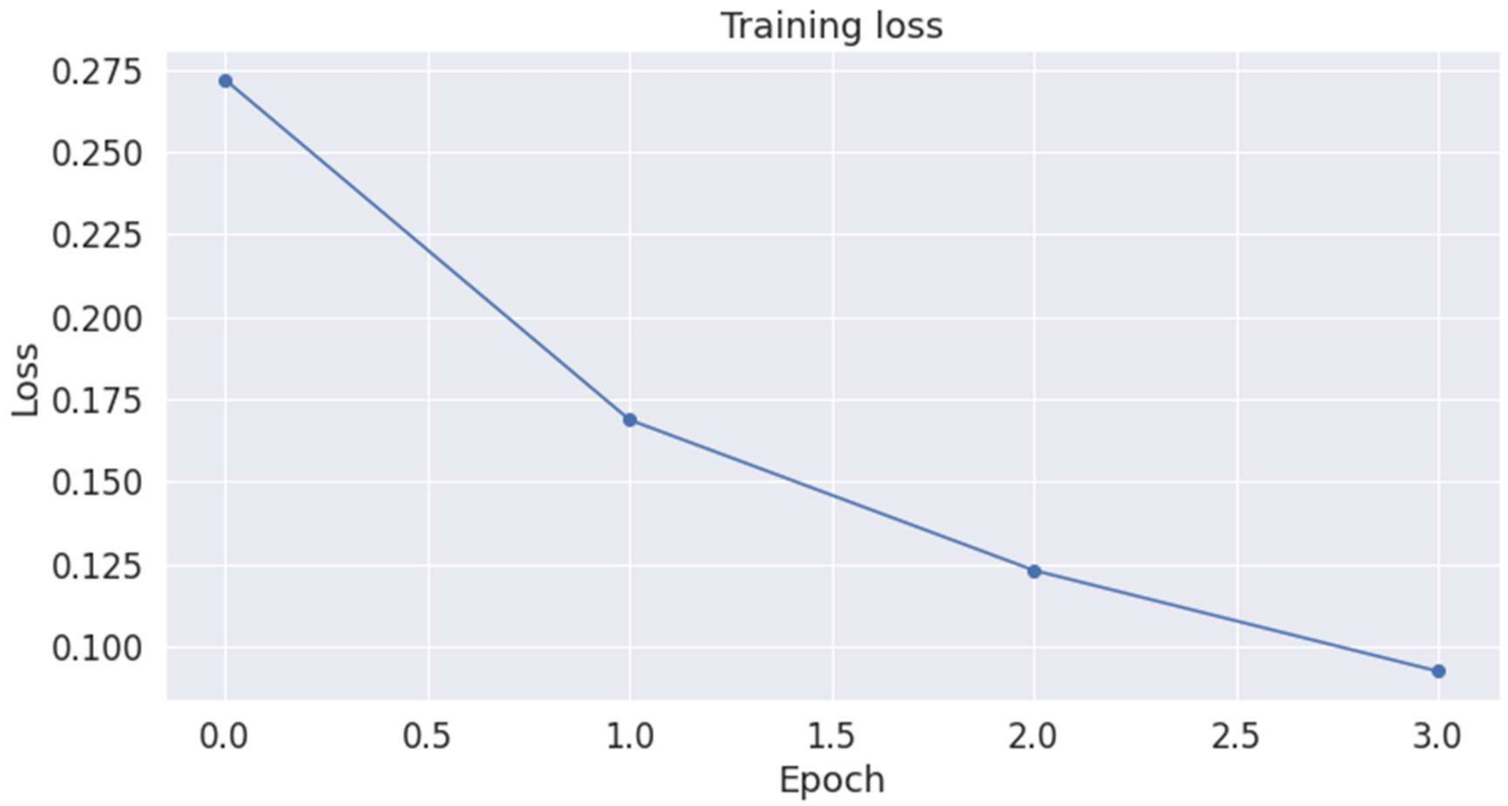


Attention Mask:

It's a mask to be used if the input sequence length is smaller than the max input sequence length in the current batch. It's the mask that we typically use for attention when a batch has varying length sentences.

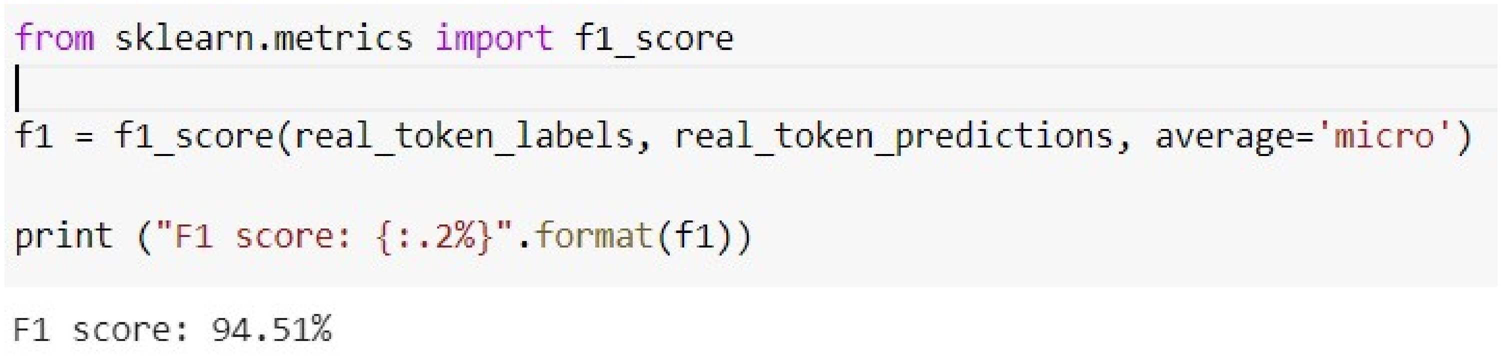


```
Original:  ['find', 'the', 'movies', 'action', 'movies', 'directed', 'by', 'john', 'woo', 'from', 'the', '1990s']
Token IDs: tensor([  101,  2424,  1996,  5691,  2895,  5691,  2856,  2011,  2198, 15854,
         2013,  1996,  4134,   102,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
            0,     0,     0,     0,     0])
Masks: tensor([1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
        0, 0, 0, 0, 0, 0, 0])
```


Training loss
Loss
0.275
0.250
0.225
0.200
0.175
0.150
0.125
0.100
0.0
0.5
1.0
1.5
2.0
2.5
3.0
Epoch

Conclusions:

- For this multi-class problem F1 score is calculated for measuring the performance of the model.
- F-1 Score – 94.51%



```
from sklearn.metrics import f1_score

f1 = f1_score(real_token_labels, real_token_predictions, average='micro')

print ("F1 score: {:.2%}".format(f1))
```

```
F1 score: 94.51%
```