NLP & Finance

By: Vanessa, Georgia, Daniel, Vinayak, and Prajwal.

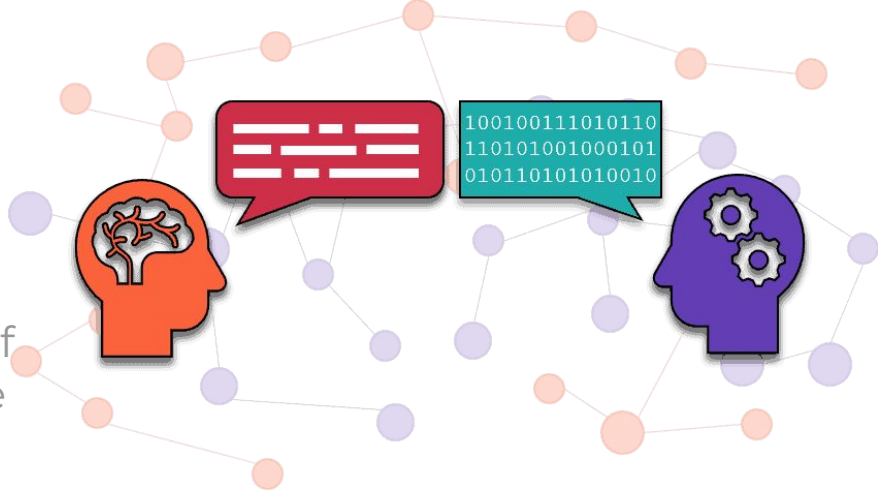
Instructor: Judah Engel

What is NLP ?

What is NLP ?

Natural Language Processing (NLP) is a branch of Computer Science. Specifically, NLP refers to the branch of Artificial Intelligence (AI).

What is NLP ?



Natural Language Processing (NLP) is a branch of Computer Science. Specifically, NLP refers to the branch of Artificial Intelligence (AI).

=> helps computers to understand human languages.
=> breaks down human input, text data so that computers can analyze and process.

NLP & STOCK MARKET

Analyze dataset

Finance Train: raw input of sentences from financial news.

- Train data to recognize the sentiments behind the texts.

```
df_train = get_finance_train()
print(df_train.head())
```

```
                                                                                                Sentence
0                                                                  Autotank Group is part of Aspo 's Systems Division .
1  The contract includes design , construction , delivery of equipment , installation and commissio...
2  Rapala said it estimates it will make savings of 1-2 mln eur a year by centralising its French o...
3  The share capital of Alma Media Corporation (business ID 1944757-4)is EUR 45,031,513.80 and it i...
4  The financial impact is estimated to be some 1.5 MEUR annual improvement in the division 's resu...

   Label
0      1
1      1
2      2
3      1
4      2
```

Label: gives sentiment label after learning data
1: Neutral
2: Positive

Testing dataset

Finance Test: after training, we tested the validity of our model using Finance Test data set. We tested to see if our model can recognize the sentiment behind text.

```
3  df_test = get_finance_test()
4  print(df_test.head())
```

```
                                                                                Sentence  \
0  In the third quarter of 2010 , net sales increased by 5.2 % to EUR 205.5 mn , and operating prof...
1  Foundries division reports its sales increased by 9.7 % to EUR 63.1 mn from EUR 57.5 mn in the c...
2                                        Financing of the project will come mainly from China .
3  Sukhraj Dulai , of the 2900 block of Boni Sue Court , a cul-de-sac on the city 's north side , s...
4  Finland 's leading metals group Outokumpu said its fourth-quarter net profit more than tripled o...

   Label
0      2
1      2
2      1
3      1
4      2
```

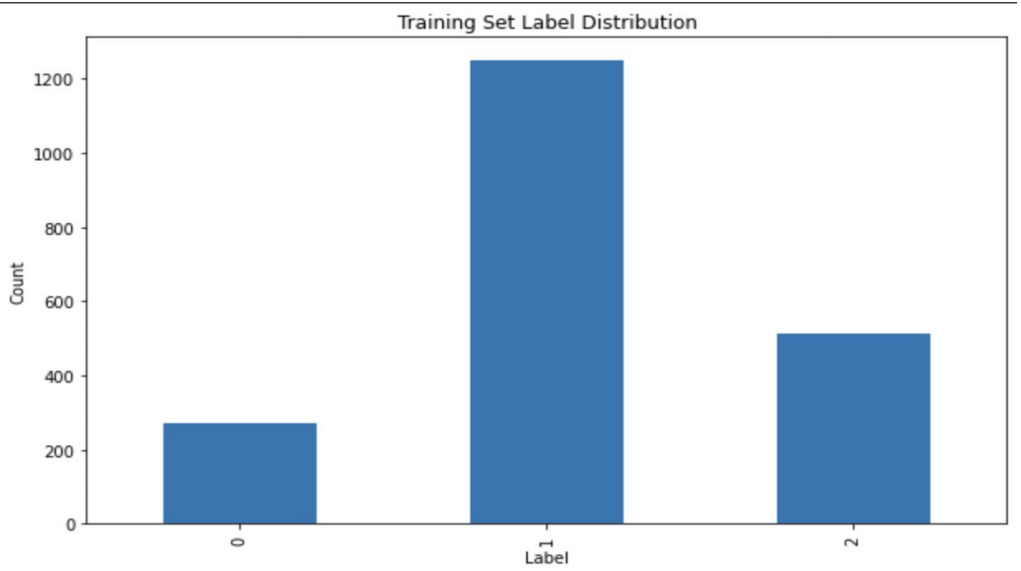
Label: gives sentiment label after learning data
1: Neutral
2: Positive

Sentiment Analysis

- We can use Sentiment Analysis to analyze text data and give appropriate prediction and evaluate the stock market

Visualize data through different methods

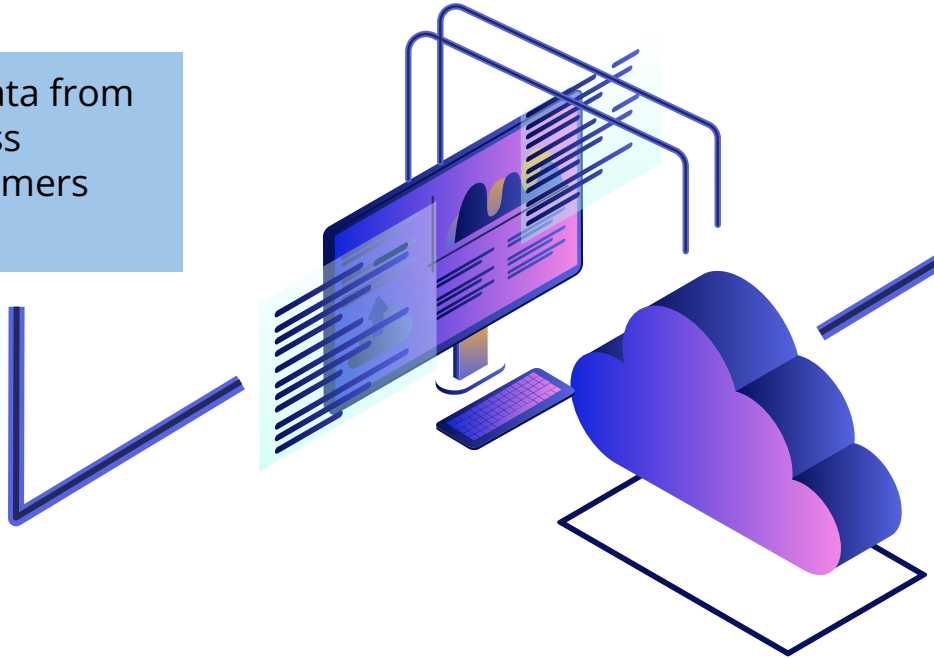
1. Making bar plot with data



Helps us to observe features of the data, such as the distribution of sentiment labels as seen in the plot above.

2. Word Cloud

Collect text data from news, business reports, customers feedback,etc



Text Preprocessing

★ Clean and prepare data before processing.

★ Filter out the unnecessary part of data input.

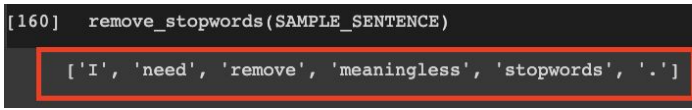
machine learning model can process more effectively, save more time and space.

Tokenization

```
['We',
 'are',
 'presenters',
 'of',
 'the',
 'NLP',
 'and',
 'Finance',
 'project.',
 '```']
```

Stemming

Stopword



Logistic Regression & Bag of Words Review

Logistic Regression: a simple method that estimates the probability of an event occurring

Bag of Words: transforms a string of text into computational data (in this case, the frequency of words)

We can use both of these ideas to convert each of our sentences into an array that designates each individual word into 0, 1, and 2 (positive, neutral, or negative). This will make our algorithm much more accurate as seen in the final accuracy result.

How It’s Done:

```
all_sentences = ["Google AI made remarkable achievements in 2019.", "Google Stock was at its all time high"]
vectorizer = CountVectorizer()
vectorizer.fit(all_sentences)
```

- Initialized CountVectorizer and applied .fit() to convert the sentence to a vector of word counts

- Then converted to an array and printed to be visualized
- 1’s and 0’s means neutral and positive words, so the statement is likely positive or neutral

```
bag_of_words_matrix = vectorizer.transform(all_sentences).toarray()

print(bag_of_words_matrix)
```

```
[[1 1 1 0 0 1 0 1 0 1 1 0 0 0]
 [0 0 0 1 1 1 1 0 1 0 0 1 1 1]]
```

(These are example sentences to learn how this works on the small scale)

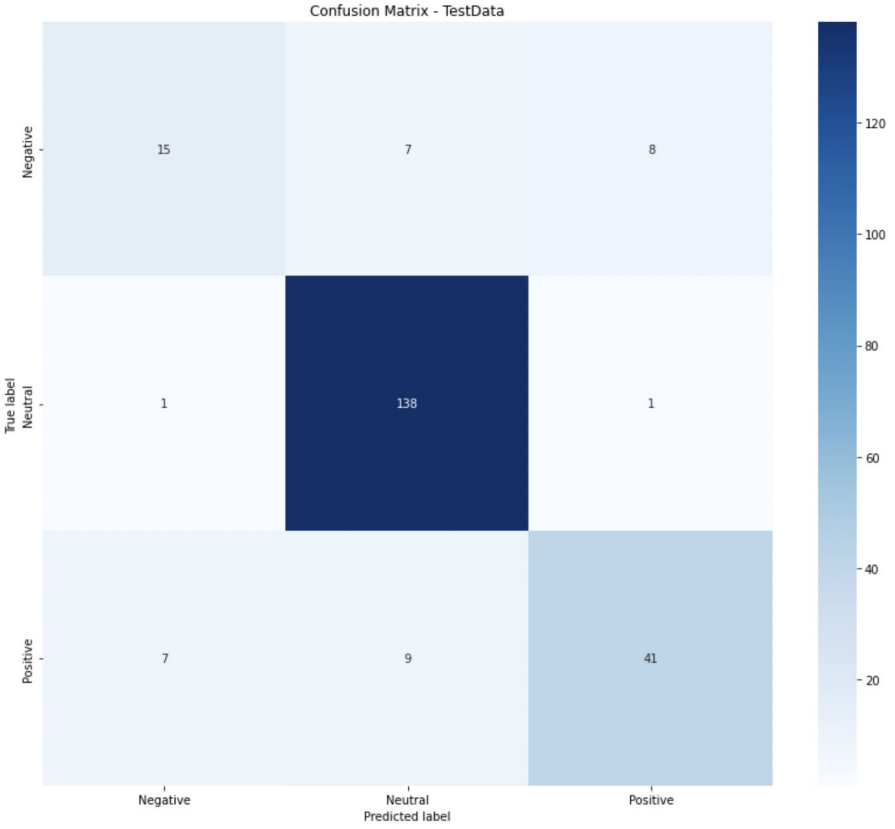
How It’s Done (cnt.)

```
def train_model(train_sentences, train_labels):
  train_sentences = [" ".join(t) for t in train_sentences]
  train_labels = [l for l in train_labels]

  # vectorizer = initialize CountVectorizer
  vectorizer = CountVectorizer()

  # train_vect = get the vector representation of train_sentences using the .fit() and .transform() methods
  vectorizer.fit(train_sentences)
  train_vect = vectorizer.transform(train_sentences)

  # model = initialize a Logistic Regression model
  model = LogisticRegression()

  # Train with train_vect and train_labels using the .fit() method of LogisticRegression
  model.fit(train_vect, train_labels)

  return model, vectorizer
  train_model(train_sentences , train_labels)
```

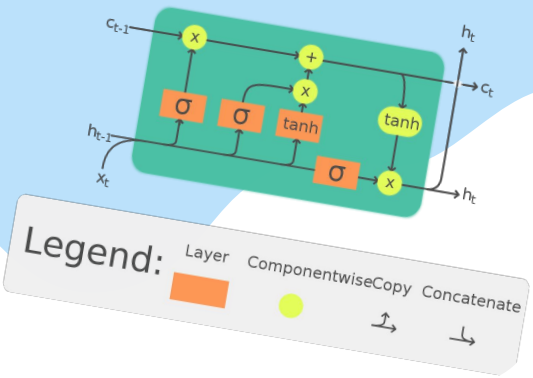
- Function to use logistic regression & fit the vectorized data to the model
- Pass in the training data to model.fit to train the model

The result: 85.46% accuracy

Analyzation Of Accuracy Through a Confusion Matrix



- Shows predicted vs actual through 9 colored squares (the darker the square, the more frequent the instance)



Building an long short-term memory (LSTM) Model

Run this to load the datasets and setup the environment.

Show code

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
finance_test.csv    100%[===================>]  28.58K  --.-KB/s    in 0s
[nltk_data] Downloading package wordnet to /root/nltk_data...
finance_train.csv   100%[===================>] 252.53K  --.-KB/s    in 0.002s
Train & Test Files are loaded
```

We first create an environment with the appropriate libraries for building the LTSM model

Data Preprocessing

Data Preprocessing

To effectively utilize an LSTM, we need to carry out some more data preprocessing. In addition to the preprocessing in the previous notebook, we want to carry out these additional filters to properly create our representations:

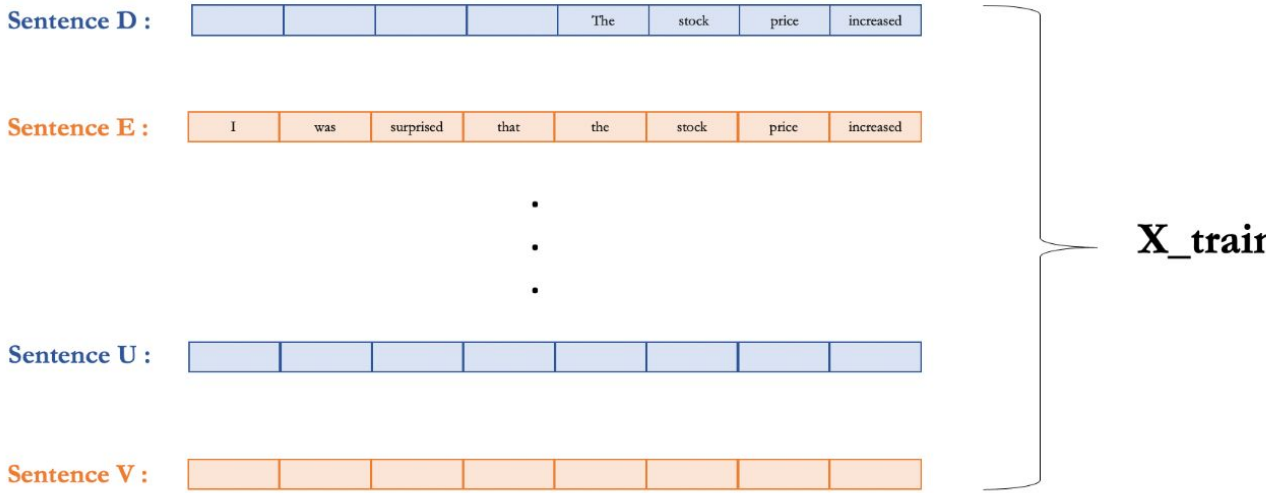
- Clean our text by removing punctuation marks.
- Encoded our text into vectors of *uniform length*. To do this we will *pad* our sentences.

Before we build our LTSM there are certain preprocessing process which are required to maximize efficiency of the system. These include

- Removal of punctuation marks
- Padding

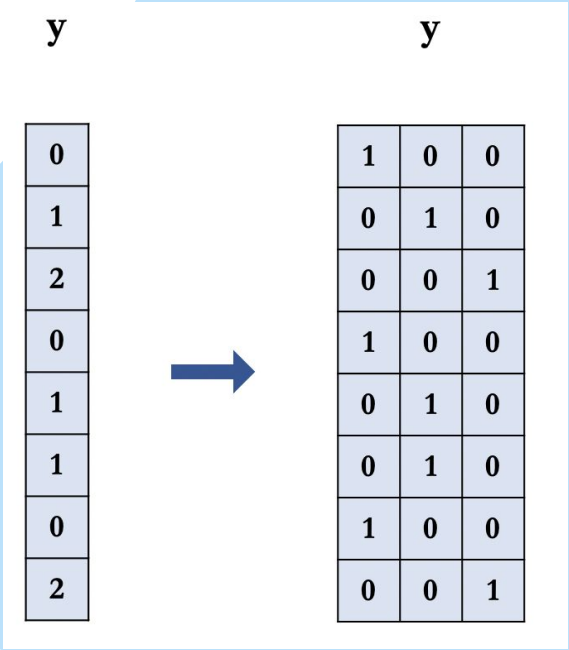
Padding

Padding is an essential process when building LTSM as it ensures that all inputs are of the same length. This is done by filling the input with empty elements or features till all given input reach a maximum length.



Creating y dummy variables

The original y labels are given as integers. However, we want to convert these to one-hot encoded vectors of length 3 to signify which of the labels for each of the examples we would design. This will allow us to learn probabilistic interpretations of the output predictions to best classify the sentences with our LSTM model.



Finally!!!

We can setup the model

```
n_labels = 3
label_map = {0 : "negative",
             1 : "neutral",
             2 : "positive"}
```

We first create our number of labels(sentiments) and assign these labels to numbers for easy machine translation.

Run this cell to instantiate your model!

Show code

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 256, 300)          300300

 spatial_dropout1d (SpatialD  (None, 256, 300)         0
 ropout1D)

 lstm (LSTM)                 (None, 100)               160400

 dense (Dense)               (None, 3)                 303

=================================================================
Total params: 461,003
Trainable params: 461,003
Non-trainable params: 0
_________________________________________________________________
None
```

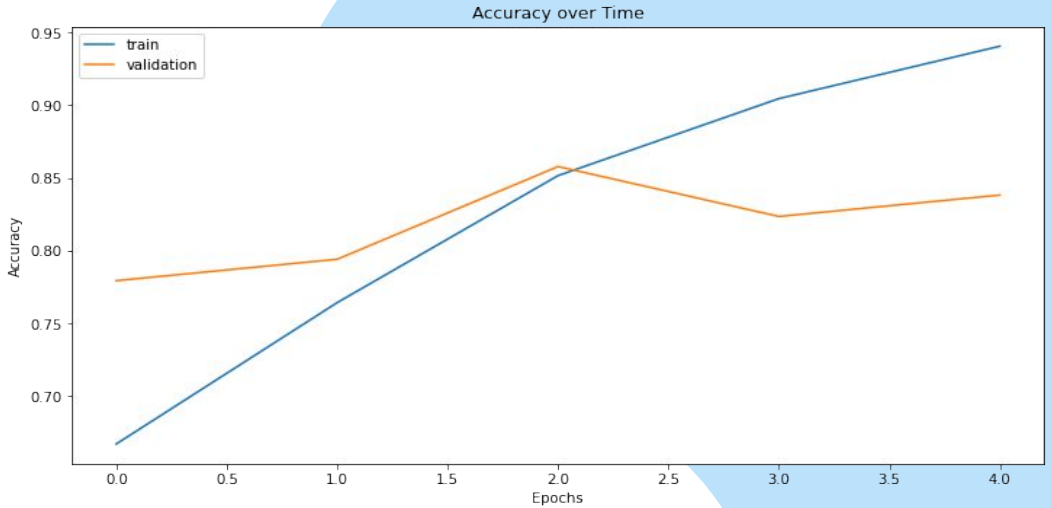
We can now assemble our model

Evaluating our model

We can now evaluate our model carrying out simple calculation and verifying it's accuracy and calculating its deviation.

```
test_loss, test_accuracy = model.evaluate(X_test, y_test)
print(test_loss, test_accuracy)
### YOUR CODE HERE

8/8 [==============================] - 1s 108ms/step - loss: 0.5266 - accuracy: 0.8238
0.5265951156616211 0.8237885236740112
```

We can even plot graphs



BERT

What is BERT ?

What is BERT ?

- BERT stands for Bidirectional Encoder Representations from Transformers. It is an advanced, pre-trained NLP model that understands language by looking at the context of each word.

What is BERT ?

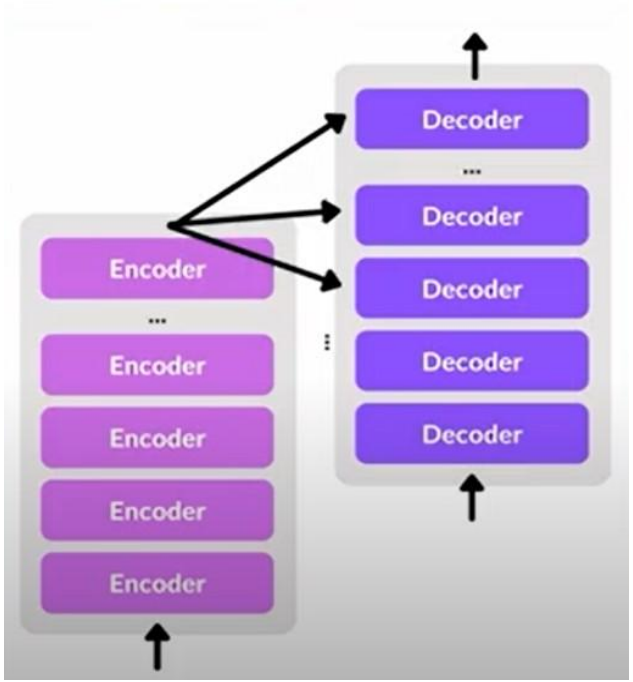
- BERT stands for Bidirectional Encoder Representations from Transformers. It is an advanced, pre-trained NLP model that understands language by looking at the context of each word.

It is a pretrained model

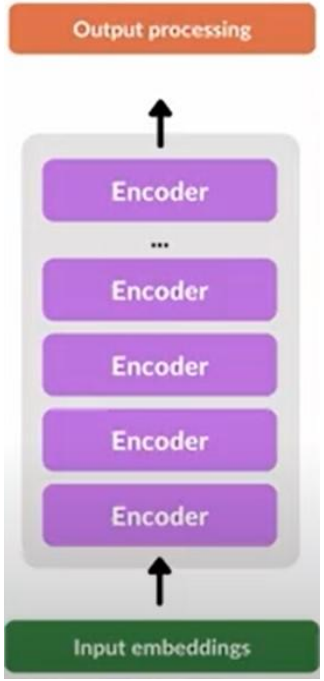
- So we can train it on our data very quickly
- We don't need as much training data
- BERT works on more than just English!
- An excellent NLP model.

Architecture of BERT

TRANSFORMER



TRANSFORMER IN BERT



How we did it ?

How we did it ?

1. We preprocessed our data and trained our model
   - We used Bert tokenizer for tokenization.
   - Text Formatting

     Pad and cut all sentences to a single constant length.

     Differentiate real tokens from padding tokens with the “attention mask”.

“Attention mask” - It is an array of 0s and 1s showing which tokens are padding and which are not.

We made attention masks for each sentence in our project by using this code.

```
attention_masks = []

###YOUR CODE HERE###
attention_masks = []
for sequence in input_ids:
  mask = [float(i > 0) for i in sequence]
  attention_masks.append(mask)
print (attention_masks[0])

###END CODE ###
```

```
1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0, 1.0,
```

How we did it ?

2. We set up our data for BERT
    - Split our Dataset into training and validation models.
    - Split our attention mask.
    - Converting data to tensors and creating Data loaders.

```
X_train, X_val, y_train, y_val = train_test_split(input_ids,  labels, test_size=0.15,  random_state=RND_SEED)
```

```
train_masks, validation_masks, _, _ = train_test_split(attention_masks,  input_ids,  test_size=0.15,random_state=RND_SEED)
```

How we did it?

3. Fine Tuning our model
    - We used the pytorch Hugging face transformer library for this.

Results - *We got 100% test accuracy for 2270 test sentences*
.

Thank you for listening! :)