PROGRESS REPORT(PROJECT DIARY) :

Harshit Modi
Un-Scripted
190100075

1. Read about supervised and unsupervised learning in ML

    - Supervised learning works upon classification. We already know that there is a defined output. As if there is a teacher who is training/teaching the model
    - Unsupervised learning works on clustering. We have a distribution of data points and then we try to find out how closely these data points are related to each other. These closely related data points are divided as clusters and the test data point is evaluated based on to which cluster it resembles the most.

2. Explored and understood various classification methods in supervised learning-
    a. K-NN:

        Let us understand this by considering that the output is binary (let say positive and negative) output. Now let us consider that we have a datapoint whose output is unknown. Consider K nearest neighbours (the number K is usually an odd number). Now the distance of unknown data points is measured from the K-nearest neighbors and the final value is given to it based on which type of data points (positive or negative) it is closest to.

    b. Linear Regression:

    c. Logistic Regression
    d. SVM
    e. Decision Tree and Random Forest
3. Explored and understood various clustering techniques in unsupervised learning-
    a. K-means clustering
    b. DBSCAN
    c. HCA
4. Explored and understood optimization methods-

    a. Maximum likelihood function
    b. Cost function
    c. Gradient Descent method
    d. Stochastic Probability
5. Explored and understood various method to convert text to vector in NLP-
    a. One-hot encoding
    b. Bag of words
    c. TF-IDF
    d. Word2Vec
       Also, explored a bit how feature embedding is done using Tensorflow and keras.

Comment:

- I had implemented a scikit library to train models based on supervised learning. Although, I had not explored how to code for models based on Unsupervised learning.
- I have understood the algebra used for optimization methods and how words are converted into vectors, but haven't implied them in code yet.

Journalism Team

21/06 - 26/06

Explored various topics for the project. Came across a startup (called otter.ai) which works over a similar idea as ours. It makes notes/minutes of the meet vy classifying the speakers, highlighting the keywords, and by also mentioning the time stamp. I also found an open source github repo which claims to have similar code as of otter.ai. But, the code is in javascript. Also found some dataset for the journalism model (video + audio + text) but they weren't quite helpful as we had to extract the text of the video using some extensions.

Hugging Face Course

28/06 - 04/07
6 hr/week

Transformers

They are used to solve all kinds of NLP tasks. One of the main transformer libraries is **pipeline.** *Pipeline* connects a model with its necessary preprocessing and postprocessing steps, allowing us to directly input any text and get an intelligible answer. Some uses of pipelines are:

1. "Sentiment-analysis"
2. "zero-shot-classification"
3. "summarization"
4. "fill-mask"
5. "ner (named entity recognition)"

There are three main steps involved when you pass some text to a pipeline:

- The text is preprocessed into a format the model can understand.
- The preprocessed inputs are passed to the model.
- The predictions of the model are post-processed, so you can make sense of them.

A. *Zero-shot-classification pipeline* is very powerful; it allows you to specify which labels to use for the classification, so you don't have to rely on the labels of the pretrained model.

Transformers, how do they work?

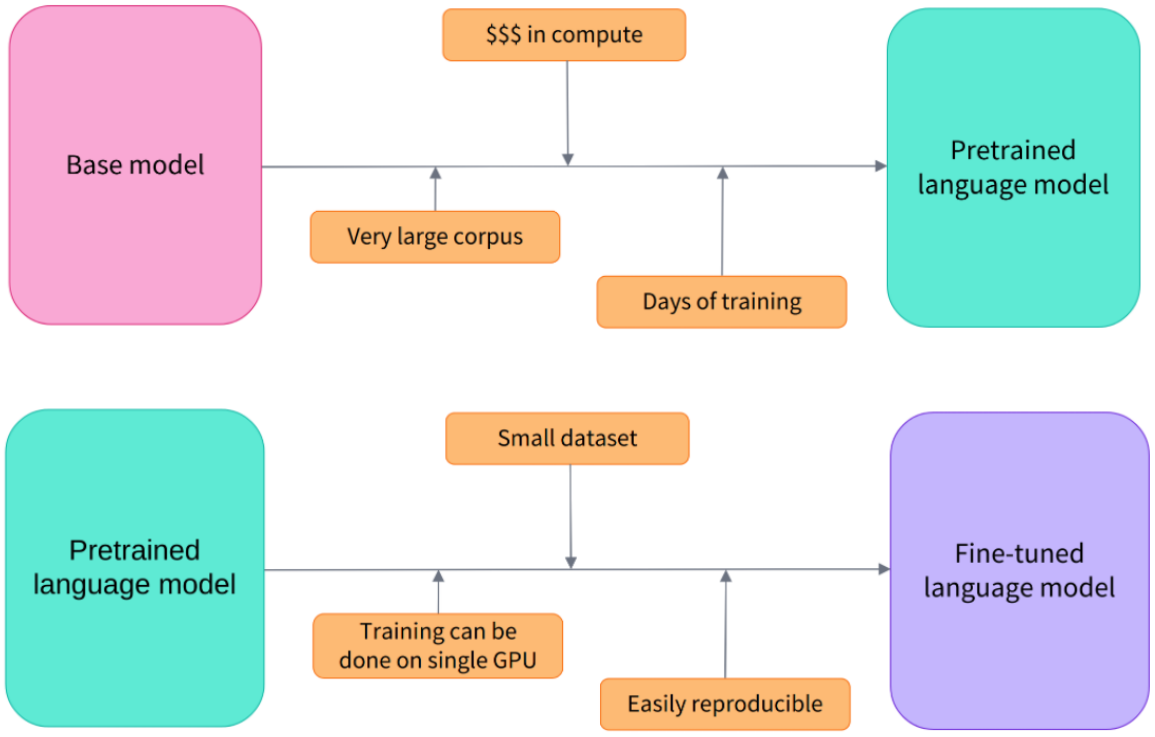
There are two ways to make a transformer model

1. Pretrain the model from scratch
2. Fine tune the pretrained model

In Pre Training from scratch, lots of environmental and computational cost are required whereas in fine-tuning the pretrained model, we just have to train the pretrained model on the additional dataset because of which environmental and computational time and cost are reduced significantly.

In case of fine-tuning, the method is called transfer learning.

Below is the pictorial difference between the two:



A general architecture of the model can be classified as:

- Encoder-only models: Good for tasks that require understanding of the input, such as sentence classification and named entity recognition.
- Decoder-only models: Good for generative tasks such as text generation.

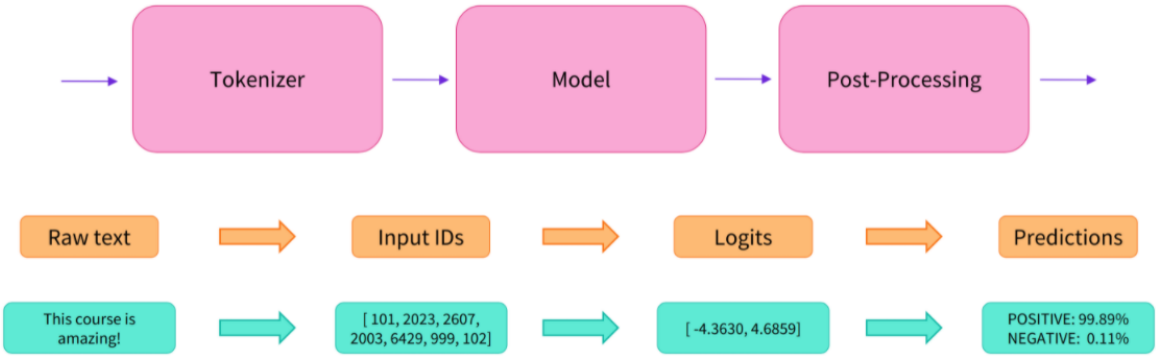
- Encoder-decoder models or sequence-to-sequence models: Good for generative tasks that require an input, such as translation or summarization.

The following snippet shows some of the examples of these models-

Model	Examples	Tasks
Encoder	ALBERT, BERT, DistilBERT, ELECTRA, RoBERTa	Sentence classification, named entity recognition, extractive question answering
Decoder	CTRL, GPT, GPT-2, Transformer XL	Text generation
Encoder-decoder	BART, T5, Marian, mBART	Summarization, translation, generative question answering

What is behind the Pipelines?

Till now we have understood various pipeline models. Now let's dive into how they work. Following flow chart explains about the mechanism behind the pipeline.



Firstly, a raw text is passed through the pipeline. The text is then broken and each text is assigned some ID. The tokenizing can be done in either of the 3 ways-

- Word-based tokenizer: It gives every word a unique token ID
- Character-based tokenizer: It gives every character a unique token ID
- Subword-based tokenizer: It gives every word a token ID but group up words having similarity. Ex. dogs/dog are given the same token ID

How to fine-tune the model?

Hugging face hubs already have a pretrained model on a very large dataset.For our use, we need to fine tune the model on our dataset (in our case the journalism dataset) to bias the model as per our requirements. We use *Trainer API* for transfer learning (fine tuning our model)

Named Entity Recognition Tagging (NER)-

04/07 - 11/07
6 hr/week

In NER, a sentence (or a text sequence) is classified into various components such as person, organization, time, location, etc.
To evaluate the quality of a NER system's output, several measures have been defined. The usual measures are called Precision, recall, and F1 score. However, several issues remain in just how to calculate those values.

- Precision is the number of predicted entity name spans that line up exactly with spans in the gold standard evaluation data. I.e. when [Person Hans] [Person Blick] is predicted but [Person Hans Blick] was required, precision for the predicted name is zero. Precision is then averaged over all predicted entity names.
- Recall is similarly the number of names in the gold standard that appear at exactly the same location in the predictions.
- F1 score is the harmonic mean of these two

NER Dataset:

Link to various dataset types for NER
Link for Indian Language annotated dataset
Link to dataset of Indian Languages. Although it is password protected :(
Link Open source Entity Recognition for Indian Language based NER
Link Kaggle repository with many datasets

Automatic Summarization

Automatic summarization is the process of shortening a set of data computationally, to create a subset (a summary) that represents the most important or relevant information within the original content.

There are two general approaches to automatic summarization: extraction and abstraction.

1. Extraction-based summarization: Here, content is extracted from the original data, but the extracted content is not modified in any way. Examples of extracted content include key-phrases that can be used to "tag" or index a text document, or key sentences (including headings) that collectively comprise an abstract, and representative images or video segments.
2. Abstraction-based summarization: Abstractive methods build an internal semantic representation of the original content, and then use this representation to create a summary that is closer to what a human might express. Abstraction may transform the extracted content by paraphrasing sections of the source document, to condense a text more strongly than extraction.

Keywords Extraction

Using keyword extraction functionality of spaCy. "en_core_web_sm" model is used for the same

Final:

11/07 - 18/07
6 hr/week

1. Finding the datasets for fine tuning the google speech recognition model
2. Writing the code for cleaning the text-
    a. Removing non-english words
    b. Removing extra whitespaces
    c. Grammatically correcting the sentence- I used Ginger It library for this objective

~~~ The End ~~~
~~~