1.Task: Sentiment Analysis of SST2 Dataset.

Our Goal is to make a model that take sentence as input and give output as 1(good) or 0 (Bad)

1. **Life Cycle Of Projects:**

    1. **Dataset:** I downloaded the dataset from Github You can find it https://github.com/clairett/pytorch-sentiment-classification/tree/master/data/SST2

    2. **Data Preprocessing:**

        - Tokenization: Split the text into individual words or tokens.
        - Stop words Removal: Remove common words (e.g., "the," "and") that don't carry much sentiment.
        - Removing URL, HTML tags, and punctuation.
        - Lemmatization or stemming: Reduce words to their base or root form.
        - Lowercasing: Convert all text to lowercase for consistent analysis.

    3. **Why Convert Text to a Vector?**

        - We convert words into vectors to apply mathematical concepts like linear algebra.
        - It ensures that similar texts are closer geometrically (distance between vectors should be less).

    4. **Feature Extraction:**

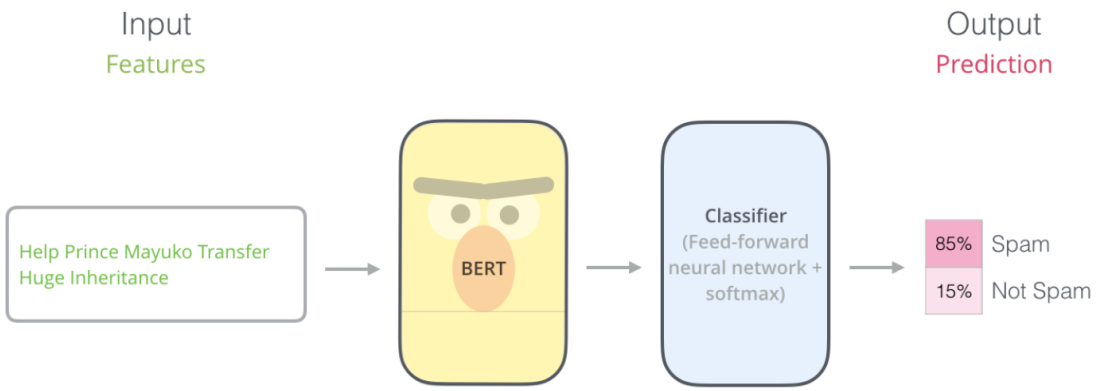
        - TF-IDF Vectorization: Transform text into numerical feature vectors using TF-IDF (Term Frequency-Inverse Document Frequency) or Count Vectorization.
        - Word Embeddings: Use pre-trained word embeddings like Word2Vec, GloVe, or fast Text for semantic understanding of words.

Model Architecture

Bidirectional Encoder Representations from Transformers (BERT)

1. **BERT** is a system that understands language more than any other tool in human history, though not as well as humans do.
2. If you've used Google Search, you've used BERT. It powers almost every query in the English language and many other languages.
3. BERT helps with tasks such as text encoding, document retrieval based on similarity to the query, summarization, highlighting relevant parts in a text, and response selection.
4. BERT enables a search engine to understand the context and how words are related to each other, which is very important for a search engine.
5. BERT takes in language and outputs a table where each column corresponds to one of the words. For many use cases, we tend to look at just the first column which represents the whole sentence.
6. BERT can be used to build a semantic search engine by focusing on the first column of the output table.

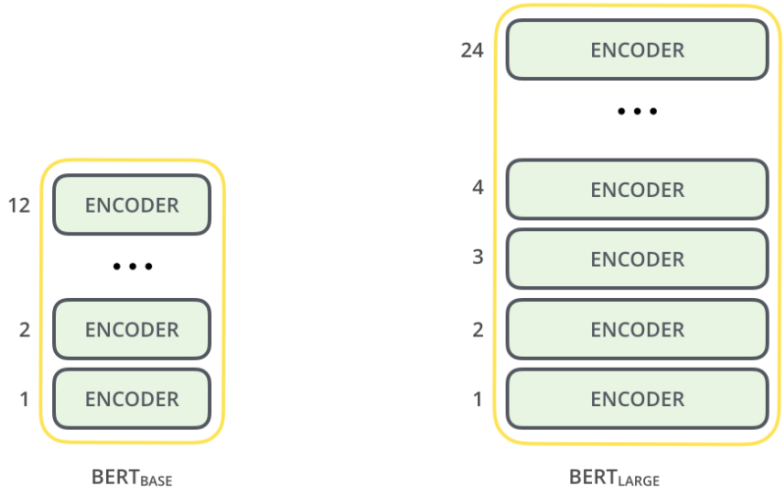
Example: Overview of BERT using Spam Classification



BERT can be thought of as a feature extraction method for text.
It takes in text data and outputs a numerical representation (features) that captures the semantic meaning of the text.
These features can then be used in various downstream tasks such as text classification, sentiment analysis, question answering, and more.
For Classification Task we can pass these features to any classifier(Logistic Regression) model and get output
The key advantage of BERT is that it considers the context of words in a sentence, making its features more expressive and accurate compared to traditional methods.

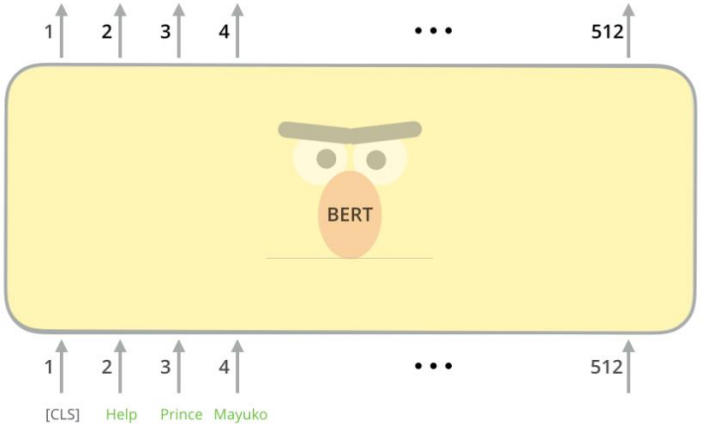
How Bert Works??

Bert is a transformer model which only consist of Encoder Only .



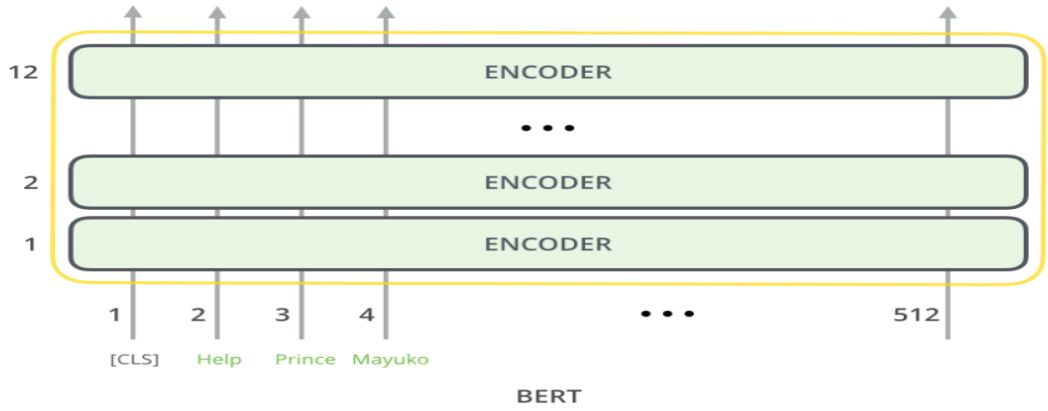
The BERT model comes in two sizes, each with a significant number of encoder layers, referred to as Transformer Blocks in the original paper.
The Base version has twelve of these layers, while the Large version has twenty-four. Additionally, both versions have larger feedforward networks (with 768 hidden units for Base and 1024 for Large) and more attention heads (12 for Base and 16 for Large).

Model Inputs:



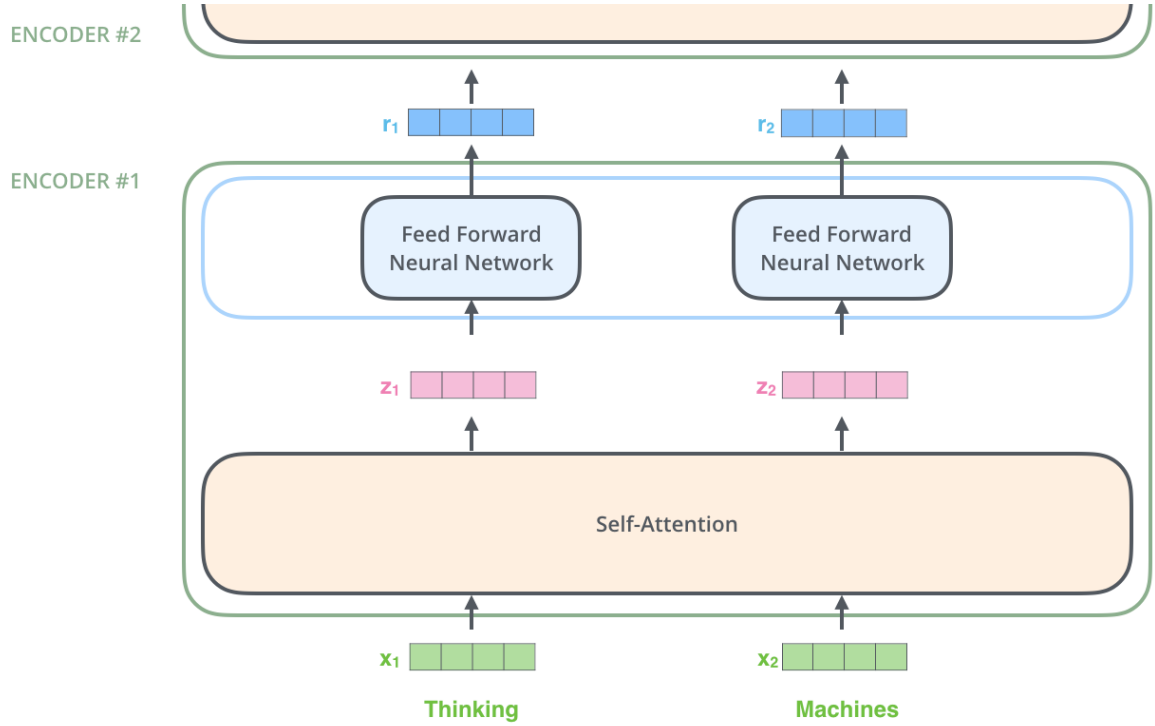
Here CLS stands for classification.

Similar to the standard encoder in the Transformer model, BERT processes a sequence of words as input that continues to move up through the layers of the model.
Each layer applies self-attention, processes the results through a feed-forward network, and then passes it on to the next encoder layer.



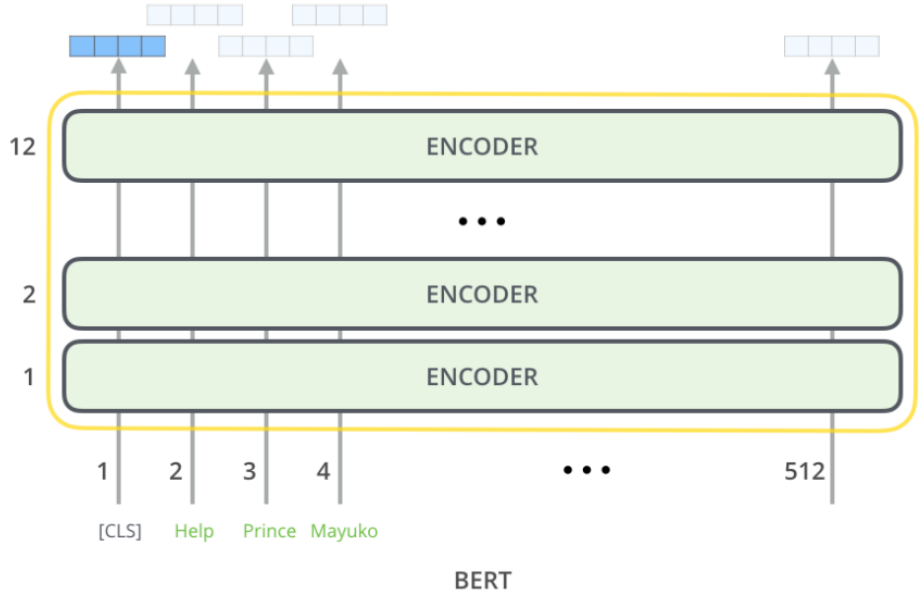
Encoder:

Encoder has two main module one is Self-Attention and other one is simple FNN. Basically, Each words is first converted into vector using Word2vec and then passed through encoder modules and at output of Encoder contain all the information about inputs and then outputs of encoder passed to all decoder in Transformers but for BERT only encoders are required.



In this process, X1 and X2 represent the vectorized forms of the words “Thinking” and “Machines” respectively.
These vectors are created using the Word2Vec model.
These vectors are then passed through a Self-Attention mechanism, which generates new vectors, Z1 and Z2.
These vectors are subsequently passed through a Feed-Forward Neural Network (FNN), resulting in the output vectors r1 and r2. These output vectors then serve as the input for the second encoder layer.

Model output:
Each position in the process generates a vector of a specific size, which is the “hidden_size”. In BERT Base, this size is 768. For the sentence classification example we’ve been discussing, we concentrate on the output from the first position only, which corresponds to the special [CLS] token



Model Accuracy:

1.I have used BERT for Feature Extraction and then on Top I have used Logistics Regression,

SVM, XGBOOST for classification
I got accuracy of 83 percentage through SVM.
We can get more accuracy if we train model on our datasets
2. Hence I have trained BERT model also and I am getting 90 percent accuracy .

Challenges I faced:
I donot have much resources to train transformer architecture