**Unsupervised Learning Using BERT**

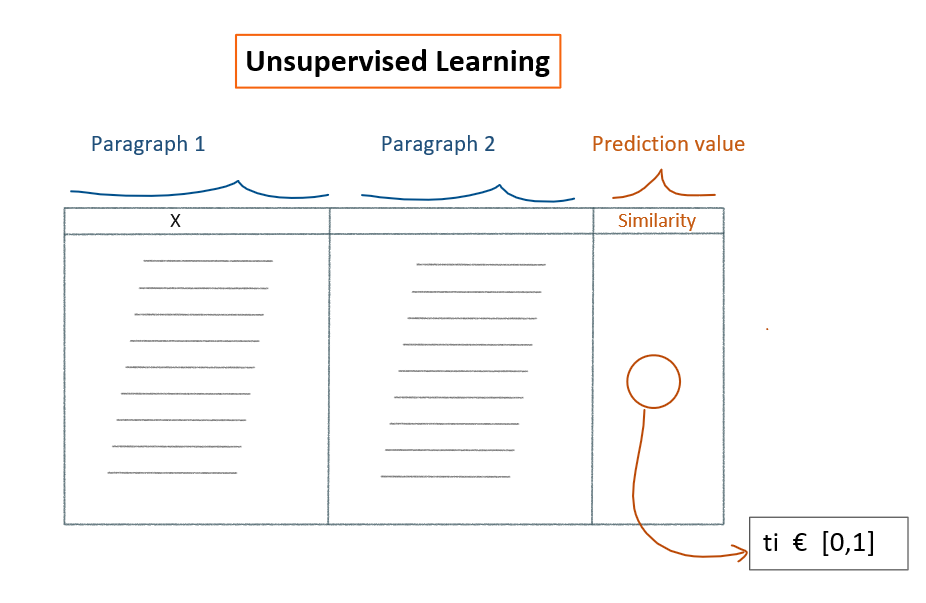
1. **Problem Statement**

Given two paragraphs, quantify the degree of similarity between the two text-based on Semantic similarity. Semantic Textual Similarity (STS) assesses the degree to which two sentences are semantically equivalent to each other. The STS task is motivated by the observation that accurately modelling the meaning similarity of sentences is a foundational language understanding problem relevant to numerous applications including machine translation (MT), summarization, generation, question-answering (QA), short answer grading, semantic search.

STS is the assessment of pairs of sentences according to their degree of semantic similarity. The task involves producing real-valued similarity scores for sentence pairs.

1. **Machine Learning Formulation**

The given dataset does not contain any label. Therefore, can be treated as an unsupervised learning problem. So, the task is to find the degree of similarity between the pair of text paras i.e. predict a value between 0-1.



1. **Approaches**

**Approach 1: Using Avg Word2Vec of text1 and text2 and find similarity**

1 Find the word embeddings of each word and average it on total words of text.

2. Find Cosine Similarity between embeddings of text1 and text2.

3. Scale the Cosine Similarity result between [0,1]

**Approach 2: Using Avg TfIDF Word2Vec of text1 and text2 and find similarity**

1. Fit TfIDF on combination of text1 and text2 and create dictionary of vocabulary and idf\_scores

2. Find the word embeddings of each word and multiply the embedding of that word with its corresponding idf\_score and average the embeddings of total words present in text.

3. Following above step find the embeddings of text1 and text2.

4. Find Cosine Similarity between embeddings of text1 and text2.

5. Scale the Cosine Similarity result between [0,1]

**Approach 3: Using Spacy NLP pipeline to find similarity**

1. Find the Similarity between embeddings of text1 and text2 using spacy NLP pipeline.

2. Scale the Similarity result between [0,1]

**Approach 4: Using Bert pretrained model as feature extraction (pytorch framework GPU)**

1. Format the text into desired format of Bert model.

2. Find the word 784 dim embeddings for each token and from the embedding of [cls] token.

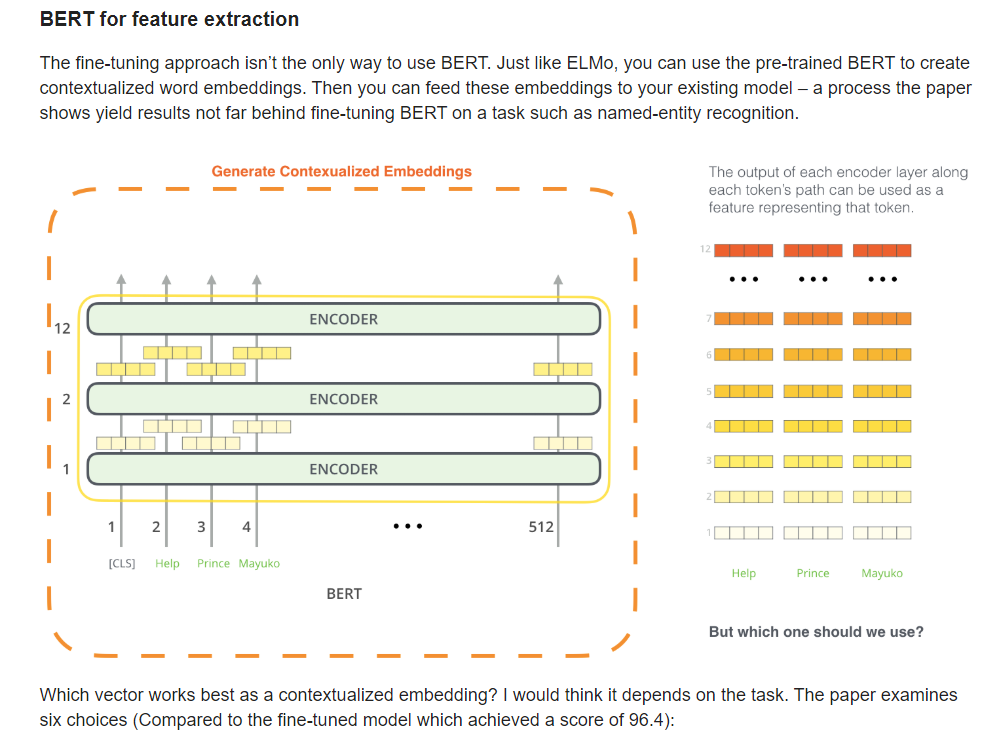
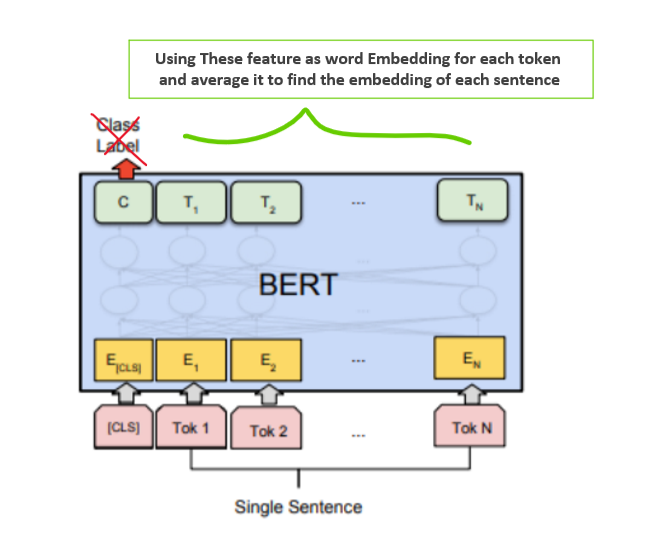
2. Average the embeddings of total tokens passed to BERT model.

3. Following above step find the embeddings of text1 and text2.

4. Find Cosine Similarity between embeddings of text1 and text2.

5. Scale the Cosine Similarity result between [0,1]

**Graphical explanation of approach 4:**



1. **Reason to choose these approaches**

1. As embedding of pre trained models hold depth semantic and context of neighbour words. It would be a decent idea to use embeddings to find similarity.

2. I have tried to do experiment using glove as embedding and tfidf scores in combination to find the similarity and results were quite decent.

1. **Conclusion**

1. These approaches are just some first cut solution to the problem.

2. We can see using mentioned 4 approaches we are getting quite decent results; some approaches are obviously better than others because of better embeddings.

3. TfIDF avg W2V has done really well to find the similarity scores.

4. Results could be definitely improved using some addition techniques like dimension reduction techniques and matrix factorisation techniques.

5. Need to explore more "how to tackle unsupervised Learning problem? "