**BERT:** Pre-training of Deep Bidirectional Transformers for Language Understanding (Bidirectional Encoder Representations from Transformers)

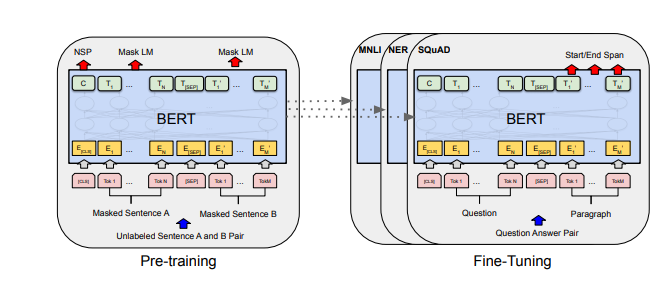
BERT (Bidirectional Encoder Representations from Transformers) is a new model by researchers at Google AI Language, which was introduced and open-sourced in late 2018, and has since caused a stir in the NLP community. The key innovation of the BERT model lies in applying the bidirectional training of Transformer models to language modeling.

BERT and other Transformer encoder architectures have been wildly successful on a variety of tasks in NLP (natural language processing). They compute vector-space representations of natural language that are suitable for use in deep learning models. The BERT family of models uses the Transformer encoder architecture to process each token of input text in the full context of all tokens before and after, hence the name: Bidirectional Encoder Representations from Transformers.

BERT models are usually pre-trained on a large corpus of text, then fine-tuned for specific tasks.

**How BERT Works**

As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it’s non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word).

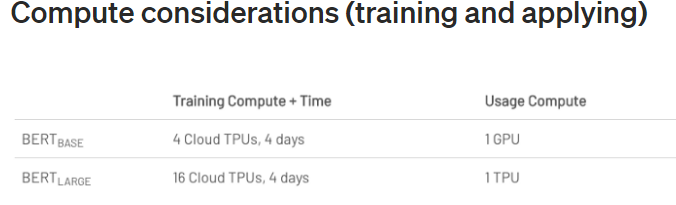


**BERT Applications**

Search engine, recommendation system, text classification ,name entity recognition, Text summarization, machine translation ,Question and answering and many NLP tasks. Many companies like Google and facebook using BERT for NLP and search engines and Recommendations.

* The process we follow is simple example text classification
* **Load the IMDB dataset**
* **Load a BERT model from TensorFlow Hub**
* **Build your own model by combining BERT with a classifier**
* **Train your own model, fine-tuning BERT as part of that**
* **Save your model and use it to classify sentence**

We can fine tune BERT model and even GLUE. Fine-tuning takes **a model that has already been trained for a particular task and then fine-tuning or tweaking it to make it perform a second similar task**. For example, a deep learning network that has been used to recognize cars can be fine-tuned to recognize trucks



Reference: https://arxiv.org/pdf/1810.04805.pdf

## 1. Masked LM (MLM)

## 2. Next Sentence Prediction (NSP)

Refer link both models

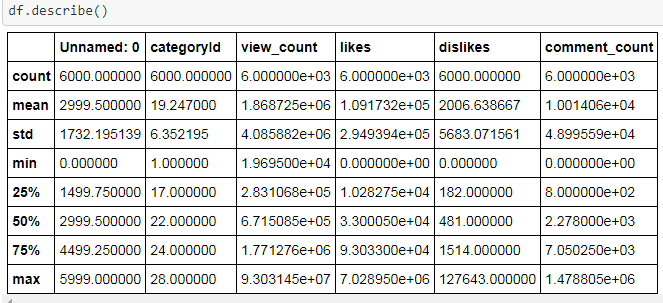
<https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270>

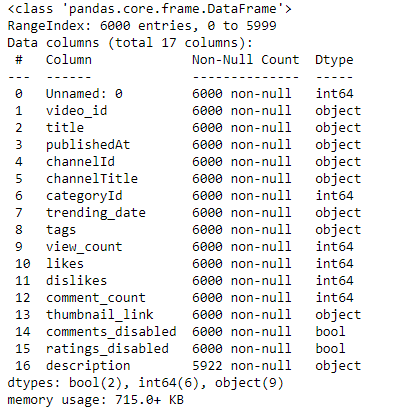
Now let see my work i have implemented basic **Content based filtering .**The recommendations by this method are highly in correlation with your subject of interests and their attributes.

Dataset: <https://www.kaggle.com/jyotmakadiya/top-trending-videos-youtube-2021?select=GB_videos_data.csv>

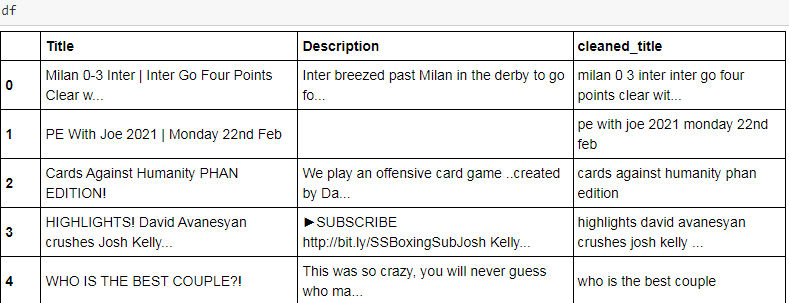
About dataset

The below image shows about total 17 columns and

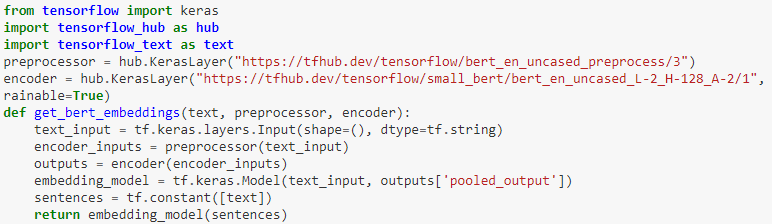




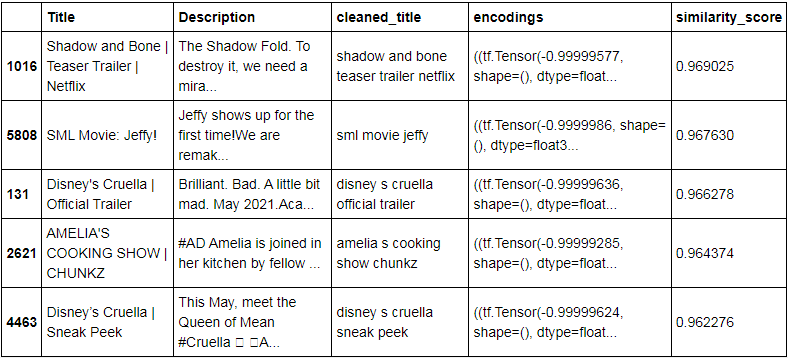
After preprocessing dataset the dataframe looks like



This is how i created layers and bert embedding



Then finally recommendations using cosine similarity



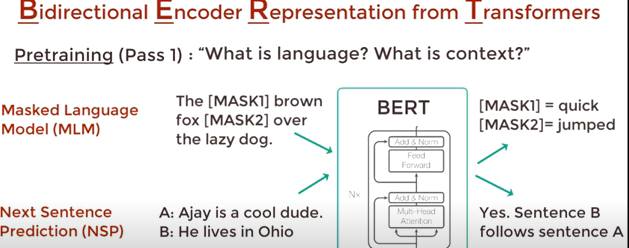
**Full code:**

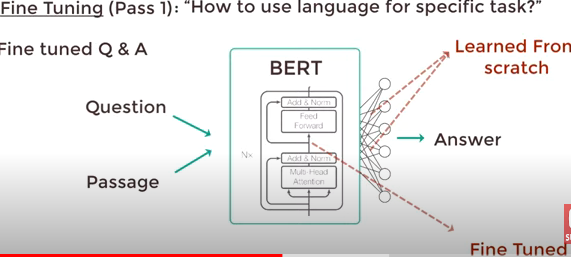
<https://github.com/rakeshrohan-123/BERT/tree/main/BERT>

**Result and observation:**

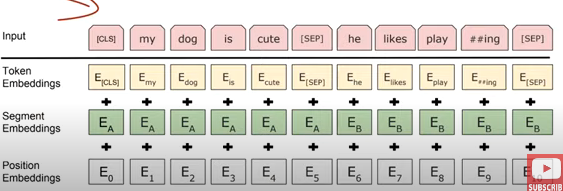
Let us generalize NLP tasks major challange in NLP task is as context spread the RNN fails and next coming to LSTM which solves that problem but in case of Machine Tanslation has failed and role of transformer comes to picture and finally bidirectional transformer achitecture is BERT.

Pretraining and fine tuning BERT



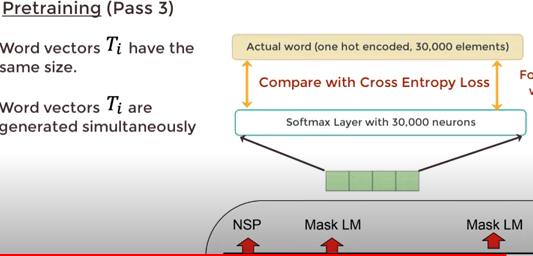


Input vector for BERT



Finally coming to output side

Length of output vector tokens is equal to number of neurons in the layer and we use activation function in these way we convert vector to distribution and actual label for this distribution is one hot encoded for the actual word so we compare this two distribution and train the network using loss function. Note that output representation also contain the masked one predicting those word from contex.



## Our recommendation based on cosine similarity that is basically the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction