Pre training of Deep Bidirectional Transformers

for Language Understanding

Introduction

This paper introduces a model called BERT (Bidirectional Encoder Representations from Transformers). It deals with pre-training deep bidirectional representations from unlabeled text and fine tuning natural language processing tasks which includes natural language interference, paraphrasing which involves the machine understanding the relationship between the sentences and token-level tasks like question and answering.

Pre-trained language representation can be done in two ways. In the feature based, task specific model is taken and pre-trained features are added. In fine tuning, task specific features are less while the model is trained by fine tuning all pre-trained parameters.

Fine tuning has been difficult on present unidirectional NLP tasks with left to right architecture where the tokens respond to only the tokens on the left side of them. This is not at all efficient in NLP tasks like question answering. Hence BERT is introduced to makes these bidirectional and establish relationship between the tokens on both sides by fusing the right and left side tokens which allows to pre-train a deep bidirectional Transformer. It is also pre- train sentence prediction tasks. BERT is very efficient on token and sentence level tasks.

Approach

Here three types of approaches are considered, unsupervised feature-based, unsupervised fine tuning and Transfer learning from supervised data.

Unsupervised feature-based approach is mainly used for sentence embeddings and paragraph embeddings where left-to-right sentences are produced from prior sentences. While training, context-sensitive features are extracted from both left-to-right and right-to-left models. Every token is represented as concatenation of left-to-right and right-to-left tokens. This approach is used in NLP models like sentiment analysis, question answering.

In unsupervised fine tuning, encoders which produce token representations are pre-trained from unlabeled texts and are fine tuned. Main advantage of this approach is that few parameters have to be learned from scratch. The architecture and parameters of unsupervised feature-based and unsupervised fine tuning are similar.

Another approach which is considered is transfer learning from supervised models with huge dataset where few layers from the pre-trained supervised models are incorporated. Transfer learning has proven to be effective in both NLP and Computer Vision tasks

Model

BERT model is both pre-trained and fine tuned. The model is pre-trained with unlabeled data over different pre-trained tasks. In fine tuning, the parameters are initialized from pre-trained tasks and are then fine tuned using labeled data from tasks which have separate fine tuned models even if they are initialized with same pre-trained parameters. The model is a bi-directional transformer encoder and is released in tensor2tensor library. Transformers are widely used nowadays.

Its experimented on WordPiece embeddings containing 30,000 tokens. The first token of every sentence is taken as a unique identification token. The pair of sentences are grouped to a sequence. The sentences are divided into A or B group in two ways, firstly they are separated with a special token. Secondly, learned word embeddings are added to tokens to differentiate them. Input embeddings are denoted as E and final hidden special token vector as C. The input representation of a token is done by adding the corresponding token, segment and position embeddings.

Pre-Training

Pre-training of BERT is done in two ways, Masked LM and Next Sentence Prediction(NSP).

A bi-directional model is supposed to be more powerful than the general left-to-right and right-to-left architecture but standard conditional language models can only be trained by left-to-right or right-to-left approach. Hence to incorporate bi-directional approach in the standard language models, masking technique is used. In masked LM procedure, random tokens are masked and predicted. Here 15% of WordPiece embeddings tokens are masked. However there is a downside to this approach because there would be a mismatch between pre-training and fine-tuning as there would be no masked tokens present during fine-tuning. This problem is overcome by not replacing all masked tokens with actual tokens. The ith token is replaced with masked token 80% of the time, random token 10% of the time and original token for another 10% of time.

Its very important in standard language models to establish relationship between the sentences. This can be done by pre-training the sentence prediction tasks that can be generated from any corpus. When choosing the sentences, 50% of the time B is followed by A with B being next sentence of A and 50% of the time its random sentence from corpus.

For pre-training corpus, BooksCorpus containing 800M and Wikipedia containing 2500M words is used. For Wikipedia, only the texts are extracted and lists, headers, tables are ignored.

Fine-Tuning

BERT uses self attention mechanism which combines two stages where swapping of outputs and inputs, and encoding of text pairs independently by a common pattern takes place. In the output, the token representations are fed into token level tasks such as sequence matching, question answering and CLS representation is fed into classification tasks such as sentiment analysis. Comparetively, pre-training is more expensive than fine tuning.

BERT is tested on GLUE, SQuaD v1.1, SQuAD 2.0 and SWAG

GLUE (General Language Understanding Evaluation) is a collection of diverse natural language understanding tasks. Its fine tuned with 3 epochs of batch size 32. best fine-tuning learning rate on the dev set is calculated. During fine-tuning, hidden vector C is used and new parameters like classification layer weights W and number of labels K is used. Standard classification loss is calculated with C and W.

SQuAD (Stanford Question Answering Dataset is a collection of 100k crowd-sourced question and answer pairs and Situations with Adversarial Generations dataset contains 113k sentence pair completion examples