Kriti 2024

Automated Research Paper Categorization

Team Kapili

10 February, 2024

Introduction

- The aim was to make an automated system that provides intelligent category suggestions based on the paper's content.
- We had a multi-classification problem in which we had to classify a given title and abstract into a single or possibly multiple categories.
- So, the model we mainly tried out most of the things with was BERT(Bidirectional Encoder Representations for Transformers). BERT is a deep learning model that has given state-of-the-art results on a wide variety of natural language processing tasks. It has been pre-trained on Wikipedia(so basically a large corpus of text) and requires task-specific fine-tuning.
- We chose this specific model because this model was relatively smaller than other LLMs and BERT is fully bidirectional. BERT gives incredible accuracy and performance on smaller data sets.

Data Overview

- We had a train dataset of 50k rows and a test dataset of 10k rows. Each row had columns namely Id, Title, Abstract and Categories. There were in total 57 distinct categories.
- We one hot encoded the categories into 57 different columns. Also, we decided to entirely drop abstract from the training process and only train with title.
- Then we removed all the stopwords like me, you, our, has, have, etc. and also converted every uppercase to lowercase so that they get tokenized as the same word.
- Then we split the training dataset into train, val and test data in the ratio of 45k, 2.5k and 2.5k respectively.
- We then checked that the number of rows in which the title had more than 30 words was a very small percentage so we kept max length for tokenization to be 30.

BERT

- BERT, which stands for Bidirectional Encoder Representations from Transformers, is based on Transformers, a deep learning model in which every output element is connected to every input element, and the weightings between them are dynamically calculated based upon their connection.
- This is contrasted against the traditional method of language processing, known as word embedding, in which previous models like GloVe and word2vec would map every single word to a vector, which represents only one dimension.
- As opposed to directional models, which read the text input sequentially, 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.

Training Process

- Now, we import BERT from local files and tokenize the title. Then, we use training and testing data loader.
- We create a custom loss function and Adam optimizer and then create a function for training the model. We run 5 epochs for the same. Took around 20 minutes to run all epochs. Also experimented with number of epochs and this was the ideal one as the validation loss increased after 5 epochs.

Result

- Then we try out the prediction on our split test dataset and get an F1 score(micro) of around 0.63 and F1 score(macro) of around 0.56. Similarly, now we do the same for the actual test dataset and convert the final dataframe in to a csv file for submission.

- We could have utilised abstract for training as well but since it was taking too much time to train (an hour for a single epoch) and the results were not that great (less score than what we got using only title for training), we decided to drop abstarct for training.

Discussion

- We also faced some challenges while solving the problem. We couldn’t experiment more with training the Abstract or Title+Abstract as a single column as the training times were too high.
- Also, there were a few shortcomings with BERT too. The BERT model cannot process texts which are longer than 512 tokens. There were majority of examples which had more than 512 words such that the average length of text taking all examples was around 800.
- Furthermore, there were like 15 categories which were appearing in the dataset for less than 1% but we couldn’t remove them because they were one-fourth of the total categories. So, we made a custom loss function which deals with imbalanced datasets significantly and it remarkably increased the score.