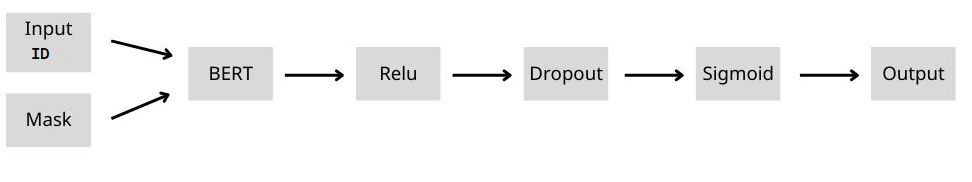
**Methodology**

BERT model

BERT, Bidirectional Encoder Representations from Transformers, is a transformer-based machine learning technique for NLP pre-training developed by Google. It is designed to pre-train deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. In modern NLP area, a popular solution for NLP problem is two stage transfer learning, which first pre-train a model that has a basic “understanding” about all natural language, which is what BERT do, and then use our dataset to fine tune the model.

Fine tune the model is not difficult, add some additional layer after the output of the pretrain model’s output, even a single linear layer can complete the mission. However, pretrain a model needs a lot of dataset, calculation resource, and time to complete. To conquer this, there are many researchers in NLP domain release their pretrained BERT model on the Internet, let other people easier to solve problems in NLP.

In this part, we use BERT to solve the problem, the structure of our model is as below:



First, we use official tokenization script created by the Google team to help us tokenize the input text and convert to input id, and let input id and mask be the input of BERT layer.

For the setting of the BERT Model, we use Adam Optimizer, and the learning rate is set to 1e-5, and the lost function is binary cross entropy. The other settings are the same as default, and the output will send to the next hidden layer.

The next layer is a dense layer with activation function Relu, and the output will be sent to dropout layer, which randomly ignore part of feature detection. This is helpful for avoiding overfitting. Then the output will be sent to a dense layer with activation function Sigmoid, and finally output the result.

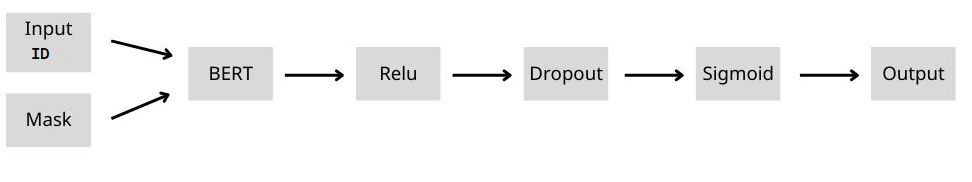
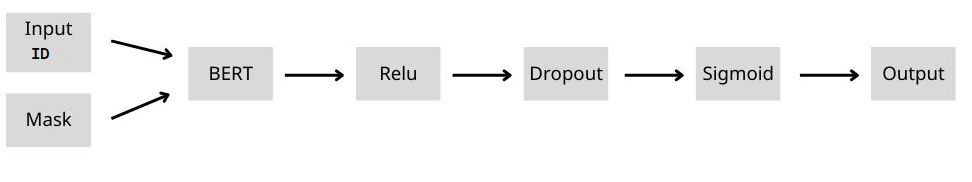
We split 1/5 training data to validation set to test the performance while training, and use training set to train our model. After training done, we use the model to predict the test data, and upload the result to Kaggle server to test the performance. The score we get on Kaggle is 0.82378.

**Experiment**

BERT

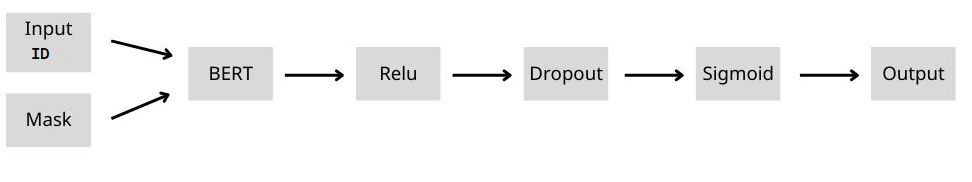
In the experiment, we try two architectures of neural network, and test how the number of epochs affect the performance of model. The first architecture is we only add a single sigmoid layer after BERT layer, the second one is the architecture mentioned above.

First architecture:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | Training loss | Training accuracy | Validation  loss | Validation accuracy |
| 3 | 0.1401 | 0.9478 | 0.5569 | 0.7991 |
| 5 | 0.0572 | 0.9773 | 0.7455 | **0.8083** |
| 10 | 0.0327 | 0.9824 | 1.0348 | 0.8043 |

Second architecture:



|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Epoch | Training loss | Training accuracy | Validation  loss | Validation accuracy |
| 3 | 0.2542 | 0.9090 | 0.6696 | 0.7919 |
| 5 | 0.1195 | 0.9548 | 0.6594 | 0.7997 |
| 7 | 0.0535 | 0.9767 | 1.1333 | 0.8096 |
| 9 | 0.0452 | 0.9765 | 1.2501 | **0.8135** |