Machine Learning Engineer Nanodegree

Capstone Proposal

Bhavesh Laddagiri

Domain Background

Teachers are one of the most important people for a child in his school days. Teachers nurture and shape the perspective of a child. However, every child is different and so is the learning pace and learning way. Due to this it becomes very important for a teacher to spend time with a student on a personal level. But for this to work, the teacher needs time which isn't available a lot with the current education system in most countries. They spend most of their time checking tests and analyzing assignments. A teacher's foremost responsibility is to address the student and understand his learning structure and not focus on how well he/she performed in a test but rather how well he could potentially perform. I have personally experienced the lack of time that teachers face due to the mechanical task of checking tests and assignments. It was in my school days that the my teacher always had piles of paper on her desk to check and because of that I was sometimes hesitant to bother her for asking doubts as I knew that even though she won't refuse to clear my doubts, checking those papers was important for her job because the education system works that way.

In order to solve this problem, I propose a Deep Learning tool which takes care of "how well the student performed" so that the teacher can focus on "how well the student can potentially perform". Current Deep Neural Networks are good at the task of analyzing data and humans are great at connecting to people at the personal level. In this particular case, Deep Learning will be used to find the contextual similarity of the student's answer with the original solution of the teacher. Pretrained Language Models (BERT) will be used for better context representation of the text. This problem is essentially a textual similarity problem and BERT based models have achieved SOTA in such natural language tasks

Problem Statement

Tests are currently the primary way of assessing the performance of a student in most education systems around the world. And usually the tests are quite frequent and require the teacher to spend a lot of their time in checking them. Instead this time can be used by the teacher to better plan the lessons and optimize the lessons for those who get left behind. With this capstone project I aim to use Machine Learning to take out the burden of tests assessment from the hands of teachers and allow them to spend more of their time with their students. It also makes them stress-free as going through many papers can be tiresome. This problem can primarily be quantified by the time spent in the task by a human.

Datasets and Inputs

This problem will primarily be solved by semantic similarity of the answer by the student and the teacher’s solution. As this problem is in the education domain the, I would be considering collecting the dataset from the real world by using the answer scripts of past exams in my college and also using the Plagiarism Detector dataset which is based on computer science answers. As it's not a public dataset, to protect the student's privacy, I have taken the appropriate permissions from each individual and the dataset will not contain any information which reveals the identity of the student in any form. If there is any information in the answer script about the student, it will be either anonymized or removed. The dataset will consist of 3 columns- Original Solution, Student's Answer and a binary label of 0 or 1 i.e. incorrect and correct respectively. The train dataset consists of 100 samples and the test dataset consists of 30 new samples. As the dataset is manually collected, it will be ensured that the dataset is balanced and contains quality samples.
I think that getting the data from an educational institute will generalize the resulting model to real world scenarios. Using a semantic similarity dataset like the Quora Question Pairs dataset is not suitable for this problem. That is why I am getting the data from a real school.

Solution Statement

In the real world, answer scripts are mostly handwritten so, my solution will consist of two stages, OCR and Semantic Similarity. For OCR on the handwritten text, I will be making use of Google Vision API. For semantic similarity, I will be fine-tuning ALBERT using PyTorch and the Transformers library by Hugging Face. ALBERT and similar models are great at retaining context over long lengths of text and performing semantic similarity with it will allow the model to consider not only the word similarity but also the context for comparison as checking test answers requires context similarity more than the word similarity.

Every teacher has some level of lenience towards students. Some teachers might be strict and some might be liberal. To account for that, a teacher can setup some liberty conditions like adding a bias value to the predicted scores or rounding off the predicted scores to the nearest multiple of 0.5 or 0.25. For example, if a particular question is of 10 marks and the model predicts a value of 0.82 (predictions will be between 0 and 1 because of sigmoid), then the score will be 8.2 / 10. In this case the teacher can set the liberty conditions to round off to the next whole number. So, the final score becomes 9 / 10.

The performance of the model will primarily be measured by the Precision, Recall and F1 score so as to reduce false negatives as it will be unfair to students as well as reduce false positives so as to not give marks more than they deserve to get.

Benchmark Model

To evaluate the performance of the fine-tuned ALBERT model on my custom dataset, I will be using a classical NLP technique i.e. Word2Vec. Word2Vec basically gives a vector embedding of a given input text. To perform semantic similarity I will be passing the Euclidean distances between the pairs of text through a fully connected neural network with a sigmoid layer to get the binary output of 0 or 1. As there is no previous benchmark for my particular dataset, I will be using the Word2Vec model's architecture and **train** it from scratch on my custom dataset and then evaluate it on my test set for Precision, Recall and F1. Doing so will allow me to measure my fine tuned model architecture against a baseline model i.e. Word2Vec. If my model performs better than the baseline, then it's a validation that my model is better than classical approaches and usable in the real world.

Evaluation Metrics

To evaluate such a model which checks whether a given answer is correct or not based on its semantic similarity with the original solution, taking into account both the number of false positives and false negatives is important. False negatives might result in unfair marks to students and false positives might result in more marks than a student deserves to get.

False Positives are measured by the Precision which is equal to TP / (TP + FP). Lesser the number of false positives, more the precision.

Similarly, False Negatives are measured by the Recall which is equal to TP / (TP + FN). Lesser the number of false negatives, greater the recall.

For having the optimal balance between both precision and recall, I will be evaluating the benchmark model and my model using the F1 Score. F1 is equal to twice the product of precision and recall divided by the sum of precision and recall.

Project Design

The technology stack for this capstone project will consist of- AWS Sagemaker or Google Colaboratory, API Gateway + Lambda or Google Cloud Functions, Google Vision API and React as a frontend. The project can be divided into many stages.

1. Data Preprocessing Stage:

   This stage will deal with preprocessing the pairs of text as segment IDs, attention masks, integers and fixed length vectors as required by BERT based models for finetuning.

2. Training and Model Building:

At this stage, the pretrained ALBERT model will be used as an encoder to generate the BERT embeddings and fully connected layers will be used to make a binary prediction. The training will be carried out in Sagemaker using the PyTorch SDK or Google Colab depending on the price constraints.

3. Evaluating the Model:

The built model will be evaluated on the test set for the following metrics: Precision, Recall and F1. The benchmark model of Word2Vec will also be built, evaluated and compared to the Fine-tuned ALBERT model.

4. Deploying the Model:

I plan to test the deployment in two ways: Sagemaker Endpoint or Google Cloud Functions. If the size of the final model and its associated libraries is less than 500 MB, I will be deploying the model as an API in Google Cloud Functions or else will be
deploying it as an Endpoint in Sagemaker. If a Sagemaker Endpoint is used, API Gateway and Lambda will also be used for creating a REST API.

5. React Frontend:

A React Web App will be developed for easy access to teachers and educators who can utilize the descriptive type test checking ability of the model and also use the integrated Google Vision API's OCR for converting the handwritten answers of students to text for the model to perform inference on. This React App will include interface for setting the liberty conditions and setting up the necessary post processing for getting the final marks/score. A topic wise report of the test performance of the student can also be generated using D3.js. The React App will be hosted on Netlify as a serverless application.