Automatic answer grading using transformers

Juraj Martiček Mário Harvan Patrik Tiszai
Faculty of Information Technology
Brno University of Technology
Brno, Czech Republic
{xmarti97,xharva03,xtisza00}@vutbr.cz

***Abstract*— Automatic answer grading using transformers with attention**

***Index terms*—Convolutional networks, GPT, attention, transformers**

I. Introduction

This is an introduction to be completed when final results are achieved.

A. Paper overview

In this paper we made a system for automatic grading of hand-written answers with help of OCR and transformers with attention.

II. Task

Main task of this paper is to create a system, which is able to grade evaluate tests

III. Dataset

We used students assigments from Brno University of Technology class Artificial Intelligence and Machine Learning [https://www.fit.vut.cz/study/course/SUI/.en]. Questions and answers were scanned by pero-ocr (Hradiš et al.). Scanned page contains student's name, login, assigment header, and questions followed by hand-written answers. First step was to prepare these assigments by hand. We created spreadsheet, which mapped every scanned page to a login. Logins were paired with their grades by lecturers, and anonymized afterwards. Our raw dataset contained all the scanned pages in pero-ocr XML format, hashed logins paired with every page of mentioned scans, and each login hash paired with grades from each answer of assigment.

After all these preparations, there was still one thing left to do. We didn't have any way of knowing which scanned region with its transcription [OCR - regions and transcriptions] was question or answer. We didn't even know what number or year of assigment it belonged to. So we decided to pair regions to questions-answer pair. Because OCR scanner was fine-tuned to understand Czech hand-written texts, even the printed question didn't have the best accuracy. But questions from last years are available online in PDF format. There was a slight catch, because unicode letters were embedded in fonts, and we didn't have any way of decoding these letters from PDF. We decided to take broken questions from these assigments, and used openai API [openai api], with prompt starting with: "Please, correct spelling errors in this sentence: " followed by our faulty copied question. This approach helped us later with correcting spelling errors in OCR readings, especially Slovak ones (pero-ocr is made for Czech language, and university accepts answers in Slovak language as well).

A. Extracting text answers from OCR scan

Process of extracting question and answer from exam sheet was pretty straightforward. We made fixed templates with regions where different parts of text should be, and extracted transcriptions to their respective data structures. Each exam contains about 7 pages of scans. Most of them are in order, but we needed to check it afterwards. Pairing region templates to scan pages done by hand. Then we went through each of the page scans, and checked their correctness. We implemented script for viewing these regions with template. This provided us with a possibilty to move template around the screen, because some scans were offset.

Next part of data preparation process was to group all of the regions (and their transcriptions) to their respective places in our data sctructure. This meant to collect all the transcriptions from our fixed regions, and divide them into questions and answers, and assign them question number, and assigment year. One of the side processes was to pair same/similar questions within the different assigment groups, and years, so we could treat them as same questions.

Resulting data structure consisted of metadata a transcripted text with each question number within assigment.

Then we corrected all of the OCR read handwritten answers via openai's API analogously as mentioned earlier. Finally, we had all of the needed data prepared.

IV. Baseline solution

As our baseline solution, from which we are able to compare our solution, we chose BERT (Devlin et al.) architecture using cosine similarity for comparing answer (to be graded) with other students answers of known grade. With this process we were able to find few most similar answers with ther percentage similarity, and interpolate the resulting grade for new answer. 70% of results were withing of 1.0 grade (on 0-5 scale) using this technique.

V. Training

A. OpenAI model fine-tuning

We used openai API for fine-tuning language models with a purpose to prepare our model for a classification task. Grading system was on 0-4 point scale. We also enriched our dataset with mixed answers to wrong questions, and added them a grade of 0. When training via openai API, dataset was of the following format: prompt and completion. Completition was single token consisting of resulting point grade. We experimented with different approaches of prompt formatting. The best results gained following prompts:

- For question *n*, grade answer: *answer text*
- For question *question text*, grade answer: *answer text*

All prompts were suffixed by `\n\n###\n\n` sequence, and grades were normalized into 5 classes. At the time of writing this article, there were four available models dedicated for fine-tuning (sorted by model size, starting with smallest): *Ada*, *Babbage*, *Curie* and *Davinci*.

B. SBERT similarity grading

References