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*— This paper focuses on automatic answer grading using tranformers with attention. Our system grades handwritten tests using OCR text recognition and training dataset, which consist of answers graded by a teacher. We compared two methods of answer grading, one using answer similarity method with sBert model and second using OpenAI API.**

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

I. Introduction

Automatic answer grading is a technique for assigning grades to student answers. Most solutions are working only on test answers that were written on a computer. One of the most popular technique is having a pre-graded dataset of answers, then using a transformer neural network to compare the similarity of pre-graded answers to the new answer. Then they assign grades according to the grade of the most similar group of answers. The authors of [1] showed that this solution works great for less complex test questions. If the question is too complex, a more extensive dataset of pre-graded answers is needed to successfully grade new answers because there are more possible correct solutions for each question.

In our experiments, we tested the automatic grading of handwritten test answers. We wanted to test whether using OCR to convert handwritten answers to text is feasible. Because of the rising popularity of OpenAI models, we wanted to show if it is possible to fine-tune their models for answer grading. We compared using both methods on our test dataset which we created from OCR test scans. Our tests showed great results in grading hand-written test answers using OCR and OpenAI models, which we fine-tuned on a few pre-graded answers in our dataset.

II. Task

Main task of this paper is to create a system, which is able to evaluate tests with grading system consisting of 0-4 points for each answer.

III. Dataset

We used **students assigments** from Brno University of Technology class Artificial Intelligence and Machine Learning[2]. Questions and answers were scanned by pero-ocr[3][4][5]. **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 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.

IV. Baseline solution

As our baseline solution, from which we are able to compare our solution, we chose BERT[6] 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 create the resulting grade for new answer by using weighted average.

V. Training

A. *OpenAI model fine-tuning*

We used openai's API[7] for fine-tuning language models with a purpose to prepare our model for a **classification task**. Grading system was on a 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, a **dataset** was of the following format: prompt and completion. **Completion** was a single token consisting of the 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*. Our first experiments were using *Ada* model, then we switched to *Babbage*, which yielded us even better results.

We created five **fine-tuned models** named *v1*, *v2*, *v3*, *v4*, *v5*. **First model** was using created *Ada* engine, and its performance was noticably lower.

So we focused on others. ***v2* model** was firstly trained on `2022 term number 1` assigment, then we decided to choose five random tests from each of the other tests, and fine-tuned it afterwards. This approach was very successful, and we discuss it later in results. For rest of the models, we used **unified dataset** consisted of all tests from year 2022, divided into training, validation, and test dataset (test dataset was made by taking 10% of the questions choosen by random from training dataset). **Models *v3* and *v4*** are identical in terms of dataset content. Only difference was usage of the OpenAI's tool to prepare dataset for training. In first case, we tried deleting repeating prefix "For question number ___", also we used space before completion to make better results. *v4* didn't use these improvements. Finally, for the ***v5* model**, we removed questions, which were needed to answer in graphical form, not text (and OCR wasn't able to extract any data from these). To summarize above mentioned models:

- ***v1*** - *Ada*
- ***v2*** - *Babbage* trained using base 2022-1 dataset, and then fine-tuned by choosing 5 question-answers of each test.
- ***v3*** - *Babbage* optimized by removing prefixe, and adding completion space.
- ***v4*** - *v3* but without dataset optimizations.
- ***v5*** - Dataset optimized by removing graphical answers

All of the above mentioned models were examined, and we measured their performace by using **test dataset**. For each entry from this dataset, we made openAI completion request, and evaluated this result. **Result evaluations** are discussed in `Results` section.

B. SBERT similarity grading

We used pre-trained multilingual models for SBERT[8] to achieve good results even for Czech and Slovak languages in tests. We first selected four answers for each grade from our dataset. We use those answers to compare similarity score to asnwer that we want to grade. Our program selects answers that have low similarity scores, which helps to better capture all possible answers that have the same grade. After that we compare each new answer with the set of graded ones and calculate cosine simularity with each group of graded answers. We then assign score to the answer, for which the similarity score is highest. We experimented with different models from Hugging face model library[9]. We then selected two models, which performed the best on our dataset. The best performing models where *paraphrase-multilingual-MiniLM-L12-v2* and *distiluse-base-multilingual-cased-v2*

VI. Results

Initially, each model underwent assessment using the dataset comprising the second test conducted in 2022, as it yielded the most favorable outcomes. The method we used is based on the meaned squared error. Two factors were selected for assessment: the MSE for the score of each individual question in the test, and the MSE for the overall test score. Throughout the process of fine-tuning, all models underwent consistent evaluation procedures. Finally, we compared the performance of all models to identify the one that achieved the most optimal outcome. In addition, the models were compared to two baseline models: one assigning a fixed score of 2 points to each question, and the other assigning a fixed score of 3 points to each question. Results are shown in Figure 1.

Model_v4 emerged as the top-performing model, and subsequent tests were conducted using this model on each 2022 test dataset, comparing its performance to the baseline models that assigned fixed points to each question.

While **model_v4** performed exceptionally well overall, its performance on certain tests fell slightly below expectations. Showed in Figure 2. Nevertheless, it still outperformed the other models, achieving the best results. The mean squared error (MSE) for the entire test datasets set below 6, indicating that the scoring errors for the complete tests were within a range of 2.5 points.

Figure 1 - MSE for all models on all quesions for year 2022 test 2

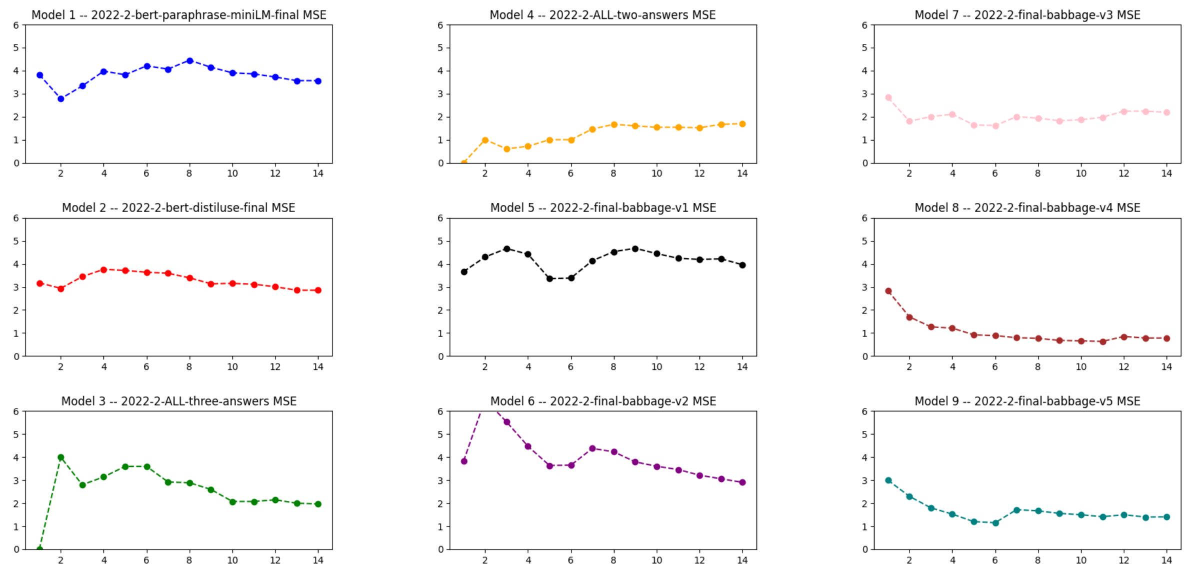
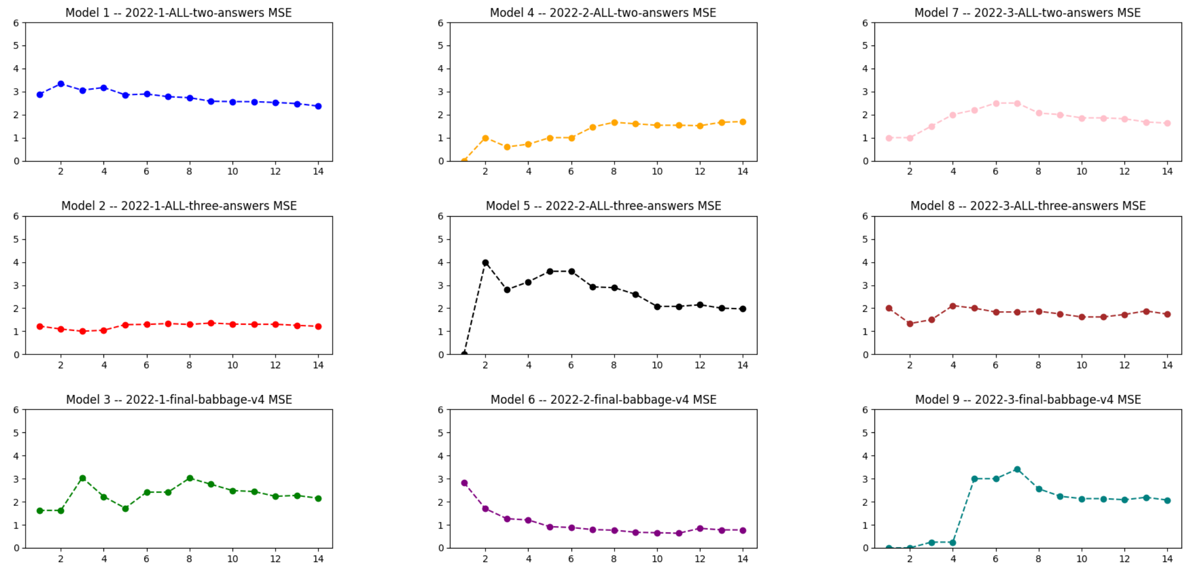
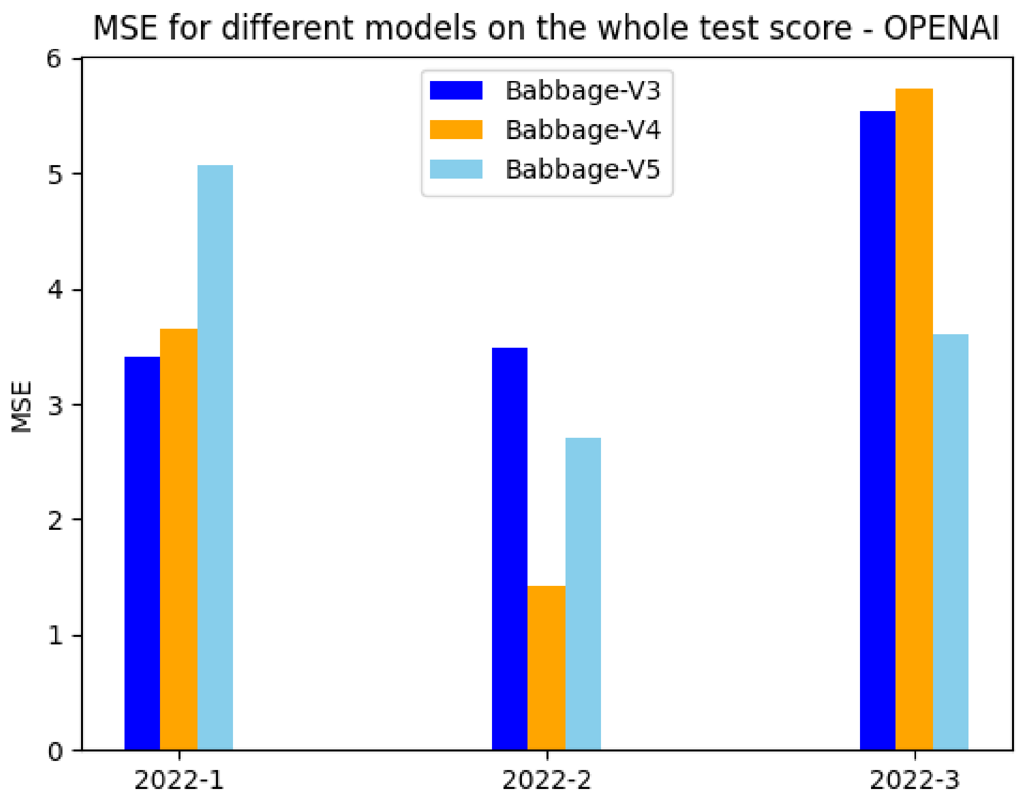


Figure 2 - MSE for model_v4 on all quesions for year 2022




MSE for different models on the whole test score - OPENAI
Babbage-V3
Babbage-V4
Babbage-V5
MSE
0
1
2
3
4
5
6
2022-1
2022-2
2022-3

References

[1] X. Zhu, H. Wu, and L. Zhang, "Automatic short-answer grading via bert-based deep neural networks," *IEEE Trans. Learn. Technologies*, vol. 15, no. 3, pp. 364–375, 2022, doi: 10.1109/TLT.2022.3175537.

[2] Department of Computer Graphics, Multimedia (UPGM). [Online]. Available: https://www.fit.vut.cz/study/course/SUI/.en

[3] O. Kodym, and M. Hradis, "Page layout analysis system for unconstrained historic documents," *Corr*, 2021. [Online]. Available: https://arxiv.org/abs/2102.11838

[4] M. Kiss, K. Benes, and M. Hradis, "AT-ST: self-training adaptation strategy for OCR in domains with
limited transcriptions," *Corr*, 2021. [Online]. Available: https://arxiv.org/abs/2104.13037

[5] J. Kohút, and M. Hradis, "Ts-net: OCR trained to switch between text transcription styles," *Corr*, 2021. [Online]. Available: https://arxiv.org/abs/2103.05489

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language
understanding," *Corr*, 2018. [Online]. Available: http://arxiv.org/abs/1810.04805

[7] OpenAI, "Openai api," 2023. (\url{https://openai.com/api/})

[8] N. Reimers, and I. Gurevych, "Sentence-bert: sentence embeddings using siamese bert-networks," *Corr*, 2019. [Online]. Available: http://arxiv.org/abs/1908.10084

[9] [Online]. Available: https://huggingface.co/models