On the application of Transformers for estimating the difficulty of Multiple-Choice Questions from text

Luca Benedetto and **Paolo Cremonesi**
Politecnico di Milano, Milan, Italy
{name.surname}@polimi.it

Giovanni Aradelli and **Andrea Cappelli** and **Andrea Giussani** and **Roberto Turrin**
Cloud Academy Sagl, Mendrisio, Switzerland
{name.surname}@cloudacademy.com

Abstract

Classical approaches to question calibration are either subjective or require newly created questions to be deployed before being calibrated. Recent works explored the possibility of estimating question difficulty from text, but did not experiment with the most recent NLP models, in particular Transformers. In this paper, we compare the performance of previous literature with Transformer models experimenting on a public and a private dataset. Our experimental results show that Transformers are capable of outperforming previously proposed models. Moreover, if an additional corpus of related documents is available, Transformers can leverage that information to further improve calibration accuracy. We characterize the dependence of the model performance on some properties of the questions, showing that it performs best on questions ending with a question mark and Multiple-Choice Questions (MCQs) with one correct choice.

1 Introduction

Question Difficulty Estimation (QDE), also referred to as "calibration", is a crucial task in education. Indeed, since question difficulty can be leveraged to assess the skill level of students under examination, wrongly calibrated questions are cause of erroneous estimations. Also, an accurate calibration enables to detect and discard questions that are too easy or too difficult for certain students. Traditionally, QDE is performed manually or with pretesting. Manual calibration is intrinsically subjective and inconsistent. Pretesting leads indeed to an accurate and consistent calibration, but i) introduces a long delay between question generation and when the question can be used to score students, and ii) requires to deploy the new questions before actually using them for scoring. To address the issues of the traditional approaches to calibration, recent research tried to leverage Natural Language Processing (NLP) techniques to perform QDE from question text: the idea is to use some pretested questions to train a model that performs QDE from text and thus eliminates (or at least reduces) the need for pretesting of new questions. Although such works showed promising results, none of them experimented with the latest NLP research, such as Transformer-based models (Vaswani et al., 2017). In this work, we aim at filling that gap, studying how different Transformers, also trained with different approaches, compare with the current state of the art. We evaluate several models built upon the pre-trained BERT (Devlin et al., 2019) and DistilBERT (Sanh et al., 2019) language models on the publicly available *ASSISTments* dataset and the private *CloudAcademy* dataset. By using data from two different domains, we add to the growing body of evidence showing that item text can be used to perform QDE. More precisely, we show that - after being fine-tuned on the task of QDE from text - Transformer models are capable of calibrating newly generated questions more accurately than previously proposed approaches. On top of that, we explore the possibility of leveraging additional textual information which might be available (e.g. transcript of video lectures) to perform an additional pre-training of the Transformers before fine-tuning them for QDE, and show that such approach can be used to further improve the accuracy. Overall, Transformer-based models are capable of reducing the RMSE by up tp 6.5%, with respect to previous approaches. Lastly, we perform an analysis of the best performing model to understand whether some question characteristics (e.g. text length, question type) particularly affect its performance. Code is available at `https://github.com/aradelli/transformers-for-qde`.

2 Related Work

There is a large interest in understanding how textual features affect item difficulty (El Masri et al., 2017; Hickendorff, 2013), and this is not limited to the educational domain: for instance Wang et al. (2014) focus on difficulty estimation in community question answering systems. The first works addressing QDE from text focused on MCQs and used deterministic approaches based on bag of words and similarities between question, answer, and distractors (Alsubait et al., 2013; Yaneva et al., 2018; Kurdi et al., 2016). Recent works mostly use machine learning approaches.

Huang et al. (2017) proposed a neural network for QDE of "reading" problems, in which the answer has to be found in a text provided together with the question. The model receives as input both the text of the question and the document, thus it actually estimates the difficulty of finding the correct answer in the provided document. This is a major difference from all other works, in which the difficulty depends on the questions only.

Yaneva et al. (2019) introduced a model to estimate the *p-value* of questions from text. The *p-value* of a question is defined as the fraction of students who correctly answered it and does not account for different skill levels. This model was trained using the text of questions and a large dataset of medical documents, thus it cannot be used if an analogous dataset is not available. Similarly, the model proposed in (Qiu et al., 2019) requires for training a dataset of medical documents in addition to the question texts. The model is made of two neural architectures to estimate the *wrongness* (i.e. $1 -$ p-value) of newly-generated MCQs, considering it as made of two components which indicate i) how difficult it is to choose between the possible choices, and ii) how difficult it is to recall the knowledge required to answer the question.

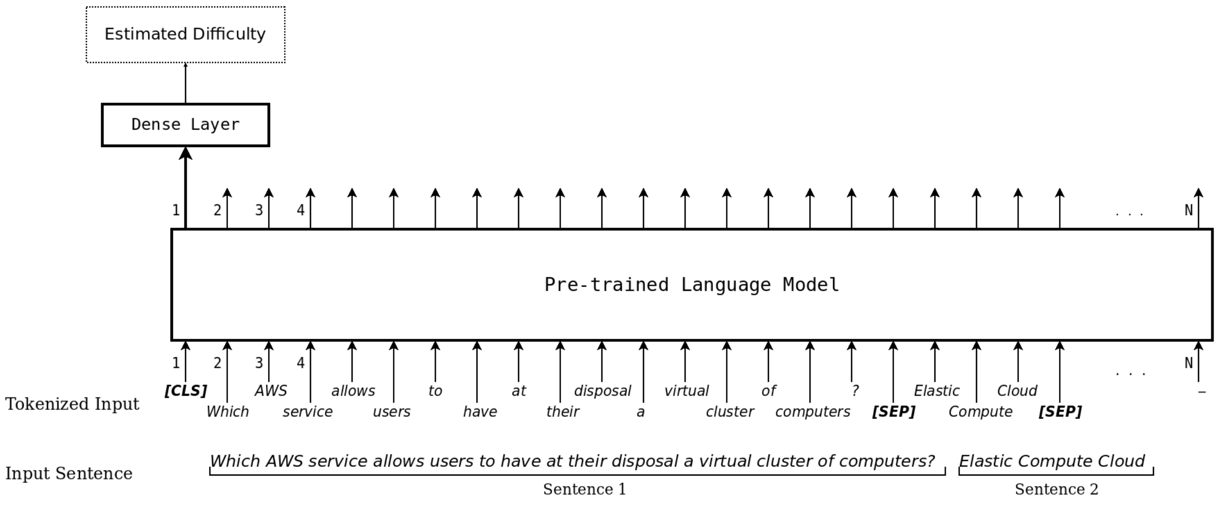
Benedetto et al. (2020b) proposed R2DE, a model that estimates the difficulty and discrimination, as defined in Item Response Theory (IRT) (Hambleton et al., 1991), of newly generated MCQs. R2DE computes TF-IDF features from the text of the questions and the text of the possible choices, and feeds two Random Forest regressors with such features. The performance of R2DE was improved in (Benedetto et al., 2020a), with the addition of readability and linguistics features.

To the best of our knowledge, (Xue et al., 2020) is the first work that explored the effects of transfer learning for QDE from text. Specifically, the authors fine-tune pre-trained ELMo embeddings (Peters et al., 2018) for the task of response time prediction, and subsequently perform a second fine-tuning for the task of *p-value* estimation. Differently from all the other models, this one and R2DE can be trained using only question text, without needing an additional dataset of related documents.

Our approach differs from previous research in several ways. First of all, we adopt transfer learning on Transformers, which are yet to be explored for QDE from text. Secondly, similarly to (Benedetto et al., 2020b,a), we perform the estimation of the IRT difficulty which, differently from *wrongness* and *p-value*, models students' skill levels as well. Specifically, we consider the one-parameter model (Rasch, 1960), a logistic model that associates a skill level to each student and a difficulty level to each question (both represented as scalars). A brief introduction to the one-parameter IRT model is given in Appendix A. Thirdly, the Transformer models presented here do not necessarily require an additional dataset of documents from the same topics as the questions, and they can be trained using only question texts; however, if such dataset is available, it can be leveraged to further improve the accuracy of calibration. Lastly, each previous work experimented on one private dataset; we experiment on two datasets (one being publicly available), showing that Transformers can be successfully used for QDE in different domains.

3 Introduction to BERT and DistilBERT

BERT (Devlin et al., 2019) is a pre-trained language model that reached state of the art performance in many language tasks. Its key technical innovation was the application of the Transformer, a popular self-attention model (Vaswani et al., 2017), to language modeling. BERT is originally trained to address two tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). MLM consists in removing one word from the input text and asking the model to fill the gap, while in NSP the model is asked - given two input sentences - to tell whether the second sentence is a reasonable continuation to the first one. Crucially,



Estimated Difficulty
Dense Layer
Pre-trained Language Model
1 2 3 4 . . . N
Tokenized Input
[CLS] Which AWS service allows users to have at their disposal a virtual cluster of computers ? [SEP] Elastic Compute Cloud [SEP] . . . –
Input Sentence
Which AWS service allows users to have at their disposal a virtual cluster of computers?
Elastic Compute Cloud
Sentence 1
Sentence 2

Figure 1: Fine-tuning of the pre-trained language model for the task of question difficulty estimation.

BERT can be used for many different downstream tasks, as we do here for QDE: starting from the pre-trained model, it is sufficient to stack an additional layer on top of the original network and then retrain it on the desired task (process named "fine-tuning"). During fine-tuning, the internal weights of the pre-trained model are updated and adapted to the desired task (together with the weights of the added layer), which is both more efficient than training the whole network from scratch and capable of better results, as the knowledge of the pre-trained model is not discarded.

BERT is a large model and therefore requires many resources for training and fine-tuning. For this reason, we also experiment with DistilBERT (Sanh et al., 2019), which is a language model obtained by distilling BERT. Knowledge distillation is a compression technique in which a small model is trained to reproduce the full output distribution of a larger model (Hinton et al., 2015). With this approach, DistilBERT is able to retain 95% of BERT's performance on several language understanding tasks using about half the number of parameters of BERT. Similarly to BERT, DistilBERT can be fine-tuned on downstream tasks and it is therefore worth exploring for QDE from text.

4 Models

This section describes how we build the different models which are compared with the current state of the art of QDE from text. These models are built upon the two pre-trained language models, fine-tuning them with two different approaches. The first approach consists in directly fine-tuning the pre-trained model for the task of QDE from text. The second approach is made of two steps: we i) further pre-train the pre-trained model on the task of Masked Language Modeling (MLM) to improve domain knowledge, and subsequently ii) fine-tune it on the task of QDE from text. This is all done separately for the two datasets: we do not perform any experiments across the two datasets.

4.1 Fine-tuning for QDE from text

This is the simplest of the two approaches; the architecture used for fine-tuning is shown in Figure 1. Given the pre-trained language model, we stack an additional fully connected layer on top of the network, in order to use that as the new output. Following the fine-tuning guidelines in (Devlin et al., 2019), we use only the first output of the pre-trained language model. This works since the first output correspond to the special token `[CLS]` which is added at the beginning of the input text and is the only one used for regression and classification. Since question calibration is a regression task, the additional output layer has one neuron, and the weights of the connections with the previous layer are randomly initialized. During fine-tuning, both the weights of the additional layer and the internal weights of the pre-trained language model are updated. For the input, we use the same tokenization and the same encoding as the original models. That is, all the input samples start with the special token `[CLS]` and contain two sentences, separated by the `[SEP]` token (another special token): the first sentence is the (tokenized) question while the second one contains the (tokenized) choices' text[1].

[1]It is possible to have a question made of several sentences. In that case, the whole question is considered as "Sentence 1",

Specifically, we experiment with three different encodings for the second sentence: i) *Q only*, we leave it empty, thus considering only the text of the question, ii) *Q+correct*, we use the text of the correct choice(s) (as in Figure 1), iii) *Q+all*, we use the text of all the possible choices, concatenating them in a single sentence. For the *ASSISTments* dataset, only the first encoding is possible as the text of the choices (correct answer and distractors) is not available. We also experimented a fourth approach, considering all the possible choices using several `[SEP]` tokens between each choice; however, this approach performed largely worse that the others, thus we do not report it here. We believe that the model, in that case, does not have enough training questions to learn the meaning of the additional separators (BERT and DistilBERT are pre-trained to use only one `[SEP]` token).

4.2 Pre-training for MLM

Masked Language Modeling (MLM) is a fill-in-the-blank task, where a word of the input text is substituted by a `[MASK]` token and the model is trained to use the surrounding words to predict the word that was masked. We leverage MLM to perform an additional pre-training of the pre-trained language models before the fine-tuning on QDE from text. Our goal is to let the model learn the questions' topics more accurately than how it would do with the fine-tuning on QDE only. In order for MLM to be effective, though, we need an additional dataset of documents about the same topics that are assessed by the questions: this is available only for the *CloudAcademy* dataset, which contains the transcript of some of the video-lectures on the e-learning platform. In practice, we perform pre-training with MLM as follows. We randomly mask 15% of the words of the available lectures, then train the language model to predict the masked words sentence by sentence. The actual prediction is performed by stacking a fully connected layer and a softmax layer on top of the original pre-trained model: for each masked sentence, this additional layer consumes as input the contextual embedding corresponding to the `[MASK]` token, and tries to predict the word that should be inserted in its place. After pre-training the model on the task of MLM, the additional dense and softmax layers are removed from the network, thus leaving us with a pre-trained model which has the same architecture as the original one, with the only difference that all the internal weights were updated during the additional MLM pre-training. The architecture for the final fine-tuning for QDE from text is the same as the one shown in Figure 1.

and the `[SEP]` token is still used only to indicate the end of the question. We use this naming ("Sentence 1" and "Sentence 2") since it is the one used in the original paper.

5 Experimental Datasets

In this work we use the publicly available data collection provided by *ASSISTments* and the private *CloudAcademy* data collection. Both data collections are made of two datasets: i) the *Answers* dataset (referred to as *A*) and ii) the *Questions* dataset (*Q*). It is important to remark that *A* and *Q* are abstract names: we have one *A* dataset for *CloudAcademy* and one *A* dataset for *ASSISTments* (similarly for *Q*). *A* contains the students' answers: for each one, it stores the user ID, the question ID, the correctness of the answer and a timestamp. Importantly, *A* contains only "first timers", meaning that we consider only the first interaction between a student and a question. *Q* contains the textual information about the items: question ID, question text and, in the case of *CloudAcademy*, the text of the possible choices. For the experiments on *CloudAcademy* data, we also have access to an additional dataset - referred to as *Lectures* (*L*) - which contains the transcripts of some online lectures available on the platform and is used for the additional MLM pre-training.

5.1 *ASSISTments* dataset

ASSISTments[2] is an online tutoring system that provides instructional assistance while assessing students (Feng et al., 2009). In practice, this means that questions - called *problems* - can be broken down into steps: if the student does not get the *original* problem correctly, he has to answer a sequence of *scaffolding* questions that break the problem down into steps. In the current work, we consider both original and scaffolding problems for QDE from text. An example problem and the corresponding scaffolding questions are shown in Appendix B. We filter the dataset to keep only questions that are answered by at least 50 students, to improve the reliability of the estimation of ground truth latent traits with IRT; on average, each item is answered by 151 students and each student answers to 64 different items. We also remove the questions that require external resources and the system messages

[2]https://new.assistments.org/

(e.g. "*Submit your answer from the textbook.*", "*Sorry, that is incorrect. Let's go to the next question!*"). After removal of the unsuitable questions, the final dataset used for QDE from text contains 11,393 different items. A is publicly available for download[3]; Q is publicly available under request[4].

5.2 *CloudAcademy* dataset

CloudAcademy[5] is an e-learning provider offering online courses about IT technologies. All the questions are MCQ and we have access to the text of the possible choices. An example question is shown in Appendix B. The dataset used in our experiments is a sub-sample of the *CloudAcademy* data collection and it was generated in order to have only questions answered by at least 50 students. A contains 7,323,502 interactions, involving 34,696 students and 13,603 unique questions; on average, each item is answered by 304 students and each student answers to 115 different items. The overall correctness is 66%. L contains the transcript of some of the online lectures offered by *CloudAcademy* about the same topics (i.e. cloud technologies) assessed by the questions. L contains a total of 159,563 sentences and 3,228,038 words.

6 Experimental Setup

As displayed in Figure 2, training is performed in two steps, repeated for the two datasets: i) the IRT model is trained in order to calibrate the questions and obtain the ground truth difficulties, then ii) these ground truth latent traits are used as target values to train the model on QDE from text.

The first step consists in using A to estimate with IRT the target difficulty of all the questions. Specifically, we use *pyirt*[6] for the estimation and consider $[-5; 5]$ as the possible range of difficulties. Difficulties estimated at this stage will later be used as ground truth and therefore are inserted as target values in Q. Then, Q is split into a train dataset (Q_{TRAIN}), used to train our model on QDE from text, and a test dataset (Q_{TEST}), which is used for the final evaluation of the model; we keep 80% of the questions for training and 20% for testing. At training time, we keep a portion of Q_{TRAIN} (10%) as development set, for hyperparameter tuning.

When L is used for pre-training the model, the setup is very similar. The only difference is that the regression model, before being fine-tuned on the task of QDE from text using Q_{TRAIN}, is pre-trained on the task of MLM on L. Being an unsupervised task, we use the whole L for this.

Transformers are implemented with the *transformers*[7] library from *HuggingFace*; fine-tuning and pre-training are performed with *TensorFlow*[8]. Hyperparameters are shown in Appendix C.

7 Results

7.1 Evaluation of the input configurations

Before comparing the Transformer models with the state of the art, we show here the performance of the different configurations, both with and without the additional pre-training on MLM. Table 1 displays the Mean Absolute Error (MAE) and the Root Mean Squared Error (RMSE) of the different configurations; the error is the difference between the IRT difficulty and the estimation of the Transformer model. The input configurations are the ones presented in Section 4: i) *Q only*, ii) *Q+correct*, iii) *Q+all*. We show the mean of the errors, measured over three independent runs with different random initializations. The results shown here are obtained on Q_{TEST} for *CloudAcademy*; indeed, for *ASSISTments* the text of the possible choices is not avail-

Table 1: Comparison of different Transformer models on *CloudAcademy*, with (Y) and without (N) additional MLM pre-training.

Model (MLM)	Input	MAE	RMSE
DistilBERT (N)	Q only	0.805	1.017
DistilBERT (N)	Q + cor.	0.799	1.019
DistilBERT (N)	Q + all	0.794	1.013
BERT (N)	Q only	0.807	1.022
BERT (N)	Q + cor.	0.789	0.999
BERT (N)	Q + all	0.811	1.027
DistilBERT (Y)	Q only	0.801	1.016
DistilBERT (Y)	Q + cor.	0.788	0.998
DistilBERT (Y)	Q + all	0.795	1.009
BERT (Y)	Q only	0.809	1.023
BERT (Y)	Q + cor.	0.774	0.981
BERT (Y)	Q + all	0.794	1.005

[3]https://sites.google.com/site/assistmentsdata/home/2012-13-school-data-with-affect

[4]https://sites.google.com/site/assistmentsdata/home/assistments-problems

[5]https://cloudacademy.com/

[6]https://pypi.org/project/pyirt/

[7]https://huggingface.co/transformers

[8]https://www.tensorflow.org/

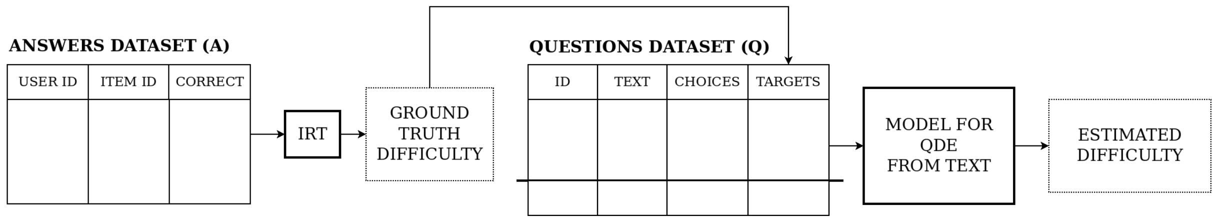


Figure 2: Experimental setup.

able and therefore the only possible input configuration is *Q only*. Even though there is not one model configuration which clearly outperforms all the others, it can be seen that both the additional pre-training and the textual information of the possible choices are helpful in improving the accuracy of the estimation. For DistilBERT without the additional MLM pre-training the best configuration is *Q+all*, while in all the other cases the best input configuration is *Q+correct*.

7.2 Comparison with state of the art

As baselines, we use: i) ZeroR: predicts for all the questions the average difficulty of the training questions. ii) R2DE (Benedetto et al., 2020b): performs difficulty estimation in two steps: the input texts are converted into feature arrays with TF-IDF and then the feature arrays are given as input to a Random Forest regression model, that performs the actual estimation. We implemented R2DE using the available code[9]. iii) ELMo (Xue et al., 2020): we re-implement this model to adapt it to our experimental datasets, and show here the results obtained with all the input configurations (on our dataset, the best performing input configuration is different than the one in the original paper).

Table 2 and Table 3 show the results of the experiments on QDE for *CloudAcademy* and *ASSISTments*, by displaying the MAE and the RMSE obtained on Q_{TEST} by the Transformer models and the chosen baselines (all the possible input configurations are considered for the baselines). For each Transformer model, only one input configuration is shown, as obtained in subsection 7.1. In order to understand how well the models estimate the difficulty of very easy and very challenging questions, we also show the MAE and RMSE measured on the questions whose difficulty b is such that $|b| > 2$ (also referred to as "extreme" questions). The results shown are the mean and the standard deviation of the errors over three independent runs.

[9]https://github.com/lucabenedetto/r2de-nlp-to-estimating-irt-parameters

Table 2 shows that also R2DE and ELMo are able to leverage the text of the possible choices to improve the accuracy of the estimation; indeed, the best performing input configuration is *Q+all* for R2DE and *Q+correct* for ELMo. The table shows that the Transformer models generally outperform the proposed baselines on both metrics, both with and without the additional pre-training on MLM. The model that consistently and significantly outperforms all the others is BERT with *Q+correct* and the additional MLM pre-training. It is also interesting to remark that ELMo seems to perform estimations a bit biased towards high and low difficulties: indeed, considering overall MAE and RMSE it performs at the same level of R2DE but it is better for the estimation of "extreme" questions. This might also be the reason why ELMo performs better than DistilBERT without MLM on "extreme" questions but worse overall. All models have larger errors on "extreme" questions than on general ones, but the increase is different for each of them: the best performing model has an increase in the MAE of 1.19, which is lower than the other Transformers (from 1.28 to 1.41), ELMo (1.37) and R2DE (1.52). Results are similar for the RMSE: the increase is 1.19 for the best model, between 1.19 and 1.30 for the other Transformers, 1.23 for ELMo (*Q+correct*) and 1.37 for R2DE (*Q+all*).

Table 3 shows results similar to Table 2: BERT is the best performing model and both Transformer models outperform the baselines. However, we can see that the errors are larger than in the previous experiment, thus suggesting that all the models are less capable at estimating the difficulty of the questions in *ASSISTments*. There could be several reasons for this, but we believe that this limitation is mainly due to two aspects: i) the platform allows the creation of question with images (not available to us); ii) the language used in the dataset is "less natural" than in *CloudAcademy* (e.g. many questions are equations with no additional text).

Table 2: Comparison with the state of the art on *CloudAcademy*.

Model	Input	MLM	MAE	MAE, $\|b\| > 2$	RMSE	RMSE, $\|b\| > 2$
ZeroR	-	-	0.845 ± 0.000	2.527 ± 0.000	1.069 ± 0.000	2.568 ± 0.000
R2DE	Q only	-	0.826 ± 0.001	2.397 ± 0.004	1.051 ± 0.001	2.468 ± 0.005
R2DE	Q + cor.	-	0.819 ± 0.001	2.320 ± 0.005	1.033 ± 0.002	2.391 ± 0.005
R2DE	Q + all	-	0.813 ± 0.001	2.331 ± 0.008	1.034 ± 0.001	2.405 ± 0.008
ELMo	Q only	-	0.833 ± 0.002	2.286 ± 0.032	1.053 ± 0.002	2.373 ± 0.025
ELMo	Q + cor.	-	0.831 ± 0.008	2.184 ± 0.033	1.048 ± 0.010	2.276 ± 0.018
ELMo	Q + all	-	0.839 ± 0.004	2.213 ± 0.025	1.057 ± 0.007	2.308 ± 0.015
DistilBERT	Q + all	N	0.794 ± 0.005	2.203 ± 0.044	1.013 ± 0.007	2.309 ± 0.036
BERT	Q + cor.	N	0.789 ± 0.010	2.118 ± 0.130	0.999 ± 0.017	2.222 ± 0.110
DistilBERT	Q + cor.	Y	0.788 ± 0.005	2.067 ± 0.074	0.998 ± 0.007	2.187 ± 0.061
BERT	Q + cor.	Y	**0.774** ± 0.011	**1.962** ± 0.042	**0.981** ± 0.015	**2.079** ± 0.047

Table 3: Comparison with the state of the art on *ASSISTments*.

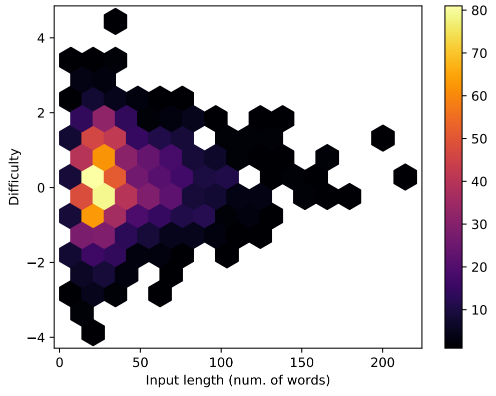
Model	Input	MLM	MAE	MAE, $\|b\| > 2$	RMSE	RMSE, $\|b\| > 2$
ZeroR	-	-	1.066 ± 0.000	2.882 ± 0.000	1.424 ± 0.000	3.033 ± 0.000
R2DE	Q only	-	0.966 ± 0.001	2.408 ± 0.005	1.304 ± 0.001	2.700 ± 0.003
ELMo	Q only	-	0.933 ± 0.013	2.025 ± 0.052	1.255 ± 0.017	2.375 ± 0.036
DistilBERT	Q only	N	0.919 ± 0.009	1.864 ± 0.059	1.239 ± 0.010	2.259 ± 0.037
BERT	Q only	N	**0.911** ± 0.003	**1.849** ± 0.074	**1.228** ± 0.003	**2.243** ± 0.038

7.3 Analysis of the best performing model

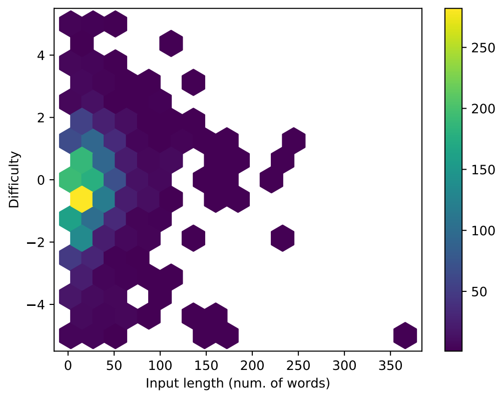
We analyze here some of the characteristics of the best performing model (i.e. BERT), trying to understand whether there are some question properties which particularly influence its accuracy. We perform the same analysis for R2DE, to understand whether such characteristics are a peculiarity of BERT or are shared among different models; the choice of R2DE is motivated by the fact that it is the second best performing model on the *CloudAcademy* dataset (excluding the other Transformer models) and it uses TF-IDF to create the features, thus a non-neural approach. We report here the results of three analyses, studying possible differences in the accuracy of QDE depending on i) input length and question difficulty, ii) number of correct choices, and iii) whether it is a cloze question.

First of all, in Figure 3 we show for the two datasets the distribution of questions depending on the input length and the true difficulty; the number of questions in each bin is represented by its color. The two figures show that the distribution is far for uniform and in many areas there are not enough questions to obtain significant results from this analysis. We do not show here the distribution of *CloudAcademy* for the *Q+all* input configuration, which is the one used by R2DE, but it is qualitatively similar to the ones displayed. Considering this distribution, we decided to focus only on some of the questions while analyzing the error depending on the input length and the true difficulty. Specifically, we keep questions i) with $|b| < 3$ and len $<= 110$ for *CloudAcademy* and BERT (96.4% of the questions); ii) with $|b| < 3$ and len $<= 110$ for *CloudAcademy* and R2DE (96.9%); iii) with $|b| < 4$ and len $<= 80$ for *ASSISTments* (94.3%), there is no difference between BERT and R2DE because they use the same input configuration.

Figure 4 shows how the estimation error (represented by the color) of BERT and R2DE depends on the input length and the target difficulty of the questions. The error heavily depends on the target difficulty: this suggests that the two models tend to estimate difficulties closer to 0 than the target values (especially R2DE). Indeed, we have observed that both the target difficulties and the estimated difficulties follow a Gaussian distribution, with higher variance for the target difficulties. There is no clear correlation between error and input length, but in some cases it seems that the error increases with the input length (e.g. row $b = -1$ in BERT). Figure 5 shows the same analysis as Figure 4 for *ASSIST-*

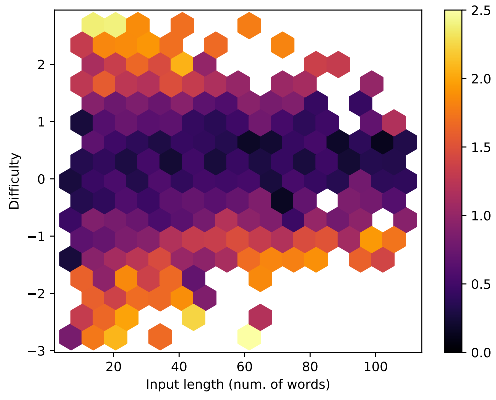


(a) *CloudAcademy*, *Q+correct* configuration.

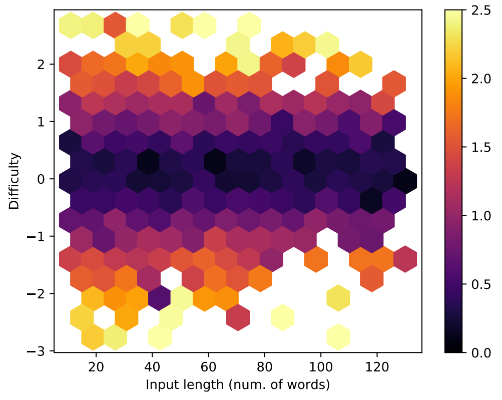


(b) *ASSISTments*.

Figure 3: Distribution of questions per input length and difficulty.



(a) BERT.



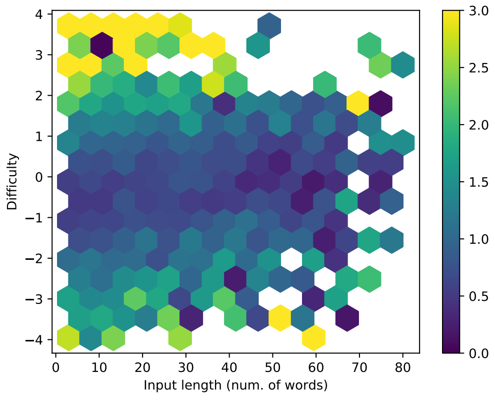
(b) R2DE.

Figure 4: *CloudAcademy*, estimation error depending on input length and target difficulty.

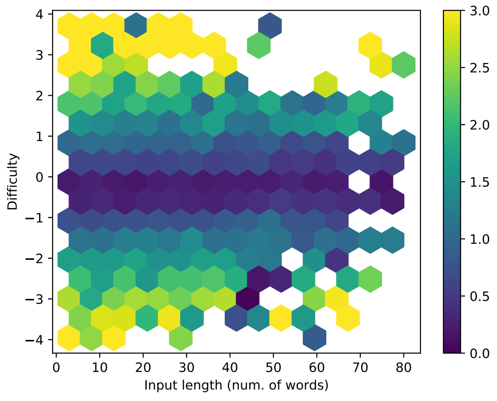
ments. The findings are very similar, but it is even more evident the fact that R2DE tends to perform predictions close to 0; the error of BERT depends less heavily on the difficulty. Again, there are no clear correlations between the input length and the accuracy of the estimation.

In *CloudAcademy*, there are i) *cloze* questions, in which the correct choice goes in place of an underscore in the text, and ii) questions with a question mark at the end. Of the 1259 test questions, 222 are cloze questions. From the average errors for the different types of questions, we observed that both BERT and R2DE perform slightly worse on cloze questions: BERT's MAE is 0.804 on cloze questions and 0.756 on the other questions; similarly, R2DE's MAE is 0.893 on cloze questions and 0.794 on the other questions.

Lastly, we looked at the average error depending on the number of correct choices: we compared questions with multiple correct choices (there are 141 of them in the test set) and the questions with one correct choice. For BERT, the overall MAE is 0.774 and on the questions with multiple correct choices it is 0.764; in the case of R2DE, the MAE is 0.813 overall and 0.750 for questions with multiple correct choices. This difference between the two models might be due to the nature of R2DE itself: indeed, it uses a bag of words approach thus it does not care about the position of each word. Instead, BERT uses contextual embeddings which depend on the position of each word and the encoding of multiple correct choices we performed (i.e. concatenation) might not be the better choice, especially considering that probably there are not enough questions to learn the encoding.



(a) BERT.



(b) R2DE.

Figure 5: *ASSISTments*, estimation error depending on input length and target difficulty.

8 Conclusions

In this paper we have performed a study of how Transformer models perform in the task of QDE from text, and have proposed a model which outperforms previous approaches. Specifically, the proposed model is built upon a pre-trained BERT language model, which is fine-tuned for the task of QDE from text. Previous approaches either require an additional dataset of documents about the same topics assessed by the questions or cannot leverage such information; differently from them the proposed model is capable of outperforming state of the art approaches being trained only on the text of the questions, and can be further improved if such additional dataset is available. As an outcome from our analysis, we can say that: i) if an additional dataset is available, BERT with MLM pre-training seems to be the best performing model; ii) if the only available data is the text of the questions, DistilBERT might be a better option, as it has basically the same performance as BERT but at a fraction of the computational cost. Furthermore, we studied the effect of some questions characteristics on BERT and R2DE, comparing the two models. We have observed that the magnitude of the error naturally increases with the magnitude of the difficulty (especially for R2DE), but there is not a clear correlation between the input length and the accuracy of the estimation. We have also observed that both models are less accurate in estimating the difficulty of cloze questions, compared to questions that end with a question mark, and that the decrease in accuracy is lower for BERT. We believe that this happens because underscores are not frequent in natural language and thus the model has a chance of learning them only during the fine-tuning on QDE, not during MLM pre-training. This is probably not enough data for learning (from scratch) the meaning of underscores in exam questions. Lastly, BERT performs better on questions with only one correct choice than on questions with multiple correct choices (for the latter, it is also outperformed by R2DE). This might be due to the encoding we used for multiple correct choices, and it is worth exploring in future research. Future works could continue to dig deeper into the analysis of the model, as we believe that the accuracy of the estimation could be further improved by using an ensemble model in which different sub-models are used depending on some characteristic of the question under calibration. Also, future works will try to explore the attention layers of the proposed model, as it might provide useful information about the reasons why the model works better on some questions.

References

Tahani Alsubait, Bijan Parsia, and Ulrike Sattler. 2013. A similarity-based theory of controlling mcq difficulty. In *2013 second international conference on e-learning and e-technologies in education (ICEEE)*, pages 283–288. IEEE.

Luca Benedetto, Andrea Cappelli, Roberto Turrin, and Paolo Cremonesi. 2020a. Introducing a framework to assess newly created questions with natural language processing. In *International Conference on Artificial Intelligence in Education*, pages 43–54. Springer.

Luca Benedetto, Andrea Cappelli, Roberto Turrin, and Paolo Cremonesi. 2020b. R2de: a nlp approach to estimating irt parameters of newly generated questions. In *Proceedings of the Tenth International Conference on Learning Analytics & Knowledge*, pages 412–421.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Yasmine H El Masri, Steve Ferrara, Peter W Foltz, and Jo-Anne Baird. 2017. Predicting item difficulty of science national curriculum tests: the case of key stage 2 assessments. *The Curriculum Journal*, 28(1):59–82.

Mingyu Feng, Neil Heffernan, and Kenneth Koedinger. 2009. Addressing the assessment challenge with an online system that tutors as it assesses. volume 19, pages 243–266. Springer.

Ronald K Hambleton, Hariharan Swaminathan, and H Jane Rogers. 1991. *Fundamentals of item response theory*. Sage.

Marian Hickendorff. 2013. The language factor in elementary mathematics assessments: Computational skills and applied problem solving in a multidimensional irt framework. *Applied Measurement in Education*, 26(4):253–278.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

Zhenya Huang, Qi Liu, Enhong Chen, Hongke Zhao, Mingyong Gao, Si Wei, Yu Su, and Guoping Hu. 2017. Question difficulty prediction for reading problems in standard tests. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Ghader Kurdi, Bijan Parsia, and Uli Sattler. 2016. An experimental evaluation of automatically generated multiple choice questions from ontologies. In *OWL: Experiences And directions–reasoner evaluation*, pages 24–39. Springer.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Zhaopeng Qiu, Xian Wu, and Wei Fan. 2019. Question difficulty prediction for multiple choice problems in medical exams. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 139–148. ACM.

Georg Rasch. 1960. *Probabilistic models for some intelligence and attainment tests*. Danish Institute for Educational Research.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Quan Wang, Jing Liu, Bin Wang, and Li Guo. 2014. A regularized competition model for question difficulty estimation in community question answering services. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1115–1126.

Kang Xue, Victoria Yaneva, Christopher Runyon, and Peter Baldwin. 2020. Predicting the difficulty and response time of multiple choice questions using transfer learning. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 193–197.

Victoria Yaneva, Peter Baldwin, Janet Mee, et al. 2019. Predicting the difficulty of multiple choice questions in a high-stakes medical exam. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 11–20.

Victoria Yaneva et al. 2018. Automatic distractor suggestion for multiple-choice tests using concept embeddings and information retrieval. In *Proceedings of the thirteenth workshop on innovative use of NLP for building educational applications*, pages 389–398.

A Introduction to IRT

We model question difficulty as defined in the one-parameter IRT model (also named Rasch model (Rasch, 1960)), which associates a skill level θ to each student and a difficulty level b to each question. For a given question j, its latent trait b_j define the item response function (i.r.f.), which indicates the probability ($\mathrm{P_C}$) that a student i with skill level θ_i correctly answers the question. The formula of the i.r.f. is as follows:

$$\mathrm{P_C} = \frac{1}{1 + e^{-1.7\cdot(\theta_i - b_j)}} \tag{1}$$

According to the intuition, a student with a given skill θ_i has a lower probability of correctly answering more difficult questions. Also, if a question is too difficult or too easy (i.e. $b_j \to \infty$ or $b_j \to -\infty$), all the students will answer in the same way (i.e. $\mathrm{P_C} \to 0$ or $\mathrm{P_C} \to 1$) regardless of θ_i. Given some students' answers to a set of questions, with IRT it is possible to estimate both the

Table 4: Example questions from *ASSISTments*.

ID	Question	Type
330	*The computer game Peter wants to buy will cost at least $50 and not more than $70. He earns $3 an hour running errands for his grandmother. Which inequality shows the number of hours, n, he will have to work to pay for the game?*	Original
326	*What is the minimum cost of the game?*	Scaffolding
327	*What is the maximum cost of the game?*	Scaffolding
328	*Write an expression that represents the amount of money Peter earns in n hours.*	Scaffolding
329	*Which inequality shows the number of hours, n, Peter will have to work to pay for the game?*	Scaffolding

Table 5: Example question from *CloudAcademy*.

Role	Text
Question	*A user has launched an EBS backed EC2 instance in the US-East-1 region. The user wants to implement a disaster recovery (DR) plan for that instance by creating another instance in a European region. How can the user accomplish this?*
Correct choice	*Create an AMI of the instance and copy the AMI to the EU region. Then launch the instance from the EU AMI.*
Distractor	*Use the "Launch more like this" option to copy the instance from one region to another.*
Distractor	*Copy the instance from the US East region to the EU region.*
Distractor	*Copy the running instance using the "Instance Copy" command to the EU region.*

skill levels of the students and the difficulty of the questions via likelihood maximization, by selecting the configuration (i.e.the θs and bs) that maximizes the probability of the observed results. Also, it is possible to assess the knowledge level $\tilde{\theta}_i$ of a student i from the correctness of its answers to a set of calibrated assessment items $Q = q_1, q_2, ..., q_{N_q}$. This is done by maximizing the results of the multiplication between the i.r.f. of the questions that were answered correctly and the complementary (i.e. $1 - \mathrm{P_C}$) of the i.r.f. of the questions that were answered erroneously.

B Example questions

An example problem from *ASSISTments* and the corresponding scaffolding questions are shown in Table 4.

An example question from *CloudAcademy*, with its correct answer and distractors, is given in Table 5.

C Hyperparameters

For fine-tuning on QDE from text we select the hyperparameters from the following pool of candidates:

- *batch size = 16, 32, 64*;
- *learning rate = 1e-5, 2e-5, 3-5*;
- *patience early stopping = 10 epochs*;
- *dropout additional layer = 0.1, 0.2, 0.3, 0.4, 0.5*;
- *internal dropout = 0.1, 0.2, 0.3, 0.4, 0.5.*

For the additional pre-training on MLM, the hyperparameters are selected between the following candidates:

- *batch size = 64*;
- *learning rate = 1e-5*;
- *number of epochs = 4, 12, 24, 36*;
- *dropout = 0.1.*

In both cases we use *sequence length = 256* and the *Adam* optimizer.