Utrecht University

Bachelor Thesis
Artificial Intelligence
7.5 ECTS

Towards automated communication training:

Fine-tuning deep contextualized embeddings

Author:
Reinier Bekkenutte
(6218318)

Supervisor:
Dr. Denis Paperno

Second reader:
Dr. Frans Adriaans

June 2020

Abstract

TrainTool is a web-app where trainees can train their communication skills. This training is done by recording the trainee's response to a video and evaluate that recording on a certain criterion. Automating the evaluation of these responses would make the system more efficient. To effectively run an automated communication training system, a classifier to evaluate criterion-user-input-pairs is necessary. As deep neural networks enabled text classification to reach new heights, this research aims to test if Google's pre-trained neural model BERT can be fine-tuned to effectively classify the criterion-transcription-pairs. This novel task is called criterion-transcription-evaluation. Since this task is inherently different than tasks in previous studies, this task is a novel application of text classification. A multilingual BERT model as well as a pre-trained Dutch BERT model called BERTje were fine-tuned for this task. Results show that both models outperformed the baselines. Next to that, BERTje has a slightly better performance than the multilingual BERT. A larger dataset and more computing power is needed to further fine-tune the model and gather results that are more representative of the possibilities of this classifier.

Table of Contents

1 Introduction

In automated answer grading, an answer to a certain question is evaluated immediately by a classifier in the background. In this research, the extent to which text classification can get transferred to an automated answer grading problem is examined. The classification of text plays a prominent role in Natural Language Processing. Whether it is the classification of sentiment or the spam detector in an email-inbox, countless applications can be found for this NLP task. In the scope of answer grading, text classification could be beneficial. Since the auto-evaluation of answers would be of value for the automation and digitalization of education. While applying a new task to text classification, this research will contribute to the discussion surrounding short answer grading. As is it will build on existing literature and methods for text classification and will broaden its possibilities.

A company in Utrecht, called Faculty of Skills, has developed a (web-)app called TrainTool. This app gives users the opportunity to train their skills regarding communication and is currently being used by all kinds of companies, universities and other institutions. The trainings are based on the following concept: Firstly, a trainee sees instructions, including certain criteria on which he or she will be evaluated. Secondly, a video is shown, for example a colleague asking something. Finally, the trainee has to record a response, keeping the given criteria in mind when reacting. This concept aims at mimicking the situation that is described, so the trainee can feel as if the conversation is really happening.
These responses are evaluated in two ways. On the one hand, the criteria each get marked with either a pass or a fail. On the other hand, more general feedback is given, where the trainee learns what he or she did right or wrong and in what way they could improve their communication. Currently, both of these parts of feedback are being given by a human coach, hired by Faculty of Skills or by the institution that is running the program on TrainTool.

The aim of this research is to investigate the way in which the grading of each criterion could be automated, consequently, the task is called criterion-transcription-evaluation. The automation of this feedback would be both beneficial for the company, as they need less manual labor to grade the inputs, as well as for the trainees, as they get instant feedback after recording their answers. I did this by fine-tuning an existing neural model such that it was able to classify a criterion-user input pair, similar to how a human coach would do. In other words, the application of known methods for deal-

ing with text-pair classifiers to this novel application of text classification was tested. To capture the essence of this research, the following research question had been chosen:

Can this novel application of criterion-transcription-evaluation be considered as a text classification task and can known methods for dealing with text-pair classification be used to solve this problem?

Next to that, two sub-questions, each focusing on a smaller scope within the problem at hand, have been formulated:

- What will the effect of training the classifier on the entire dataset be when comparing it to only training it on data corresponding to a single appraisal question?

This question is relevant because each appraisal question has a different domain in which it operates. The extent in which the classifier is able to generalize over the data is measured by this question. This is tested by choosing three different datasets with different levels of complexity and compare the results after training the classifier on each dataset.

- Is Dutch represented well enough in existing pre-trained models such that the classifier is able to outperform the baselines?

Considering that the available dataset is in Dutch, this could be an obstacle for the construction of this classifier. In comparison with, for example, English, where there are a lot more resources avaiable. This research will show, how well existing Dutch pre-trained models will perform.

Relevance for Artificial Intelligence

Text classification is an important type of task in Natural Language Processing. The introduction of neural networks and deep learning techniques broadened the possibilities of this part of NLP. This enabled computers to handle massive amounts of data and enabled them to learn about language with significantly less human labor needed. This research will add something of value to the discussion about text classification and to this field of AI. This value comes from the fact that this is a new application, as it differs from simple answer grading, or answer-reference answers comparison, which is very prevalent in the literature. Unlike those applications, trainees

are not answering a question, but will have to keep certain criteria in mind, that will be used for evaluation in combination with their transcribed spoken input. The following example shows how trainees process the criterion and give their input:

Note that this example is translated from Dutch

- *Criterion:* I show that I am aware of the situation of the client.
- *User input classified as positive:* Hi Guido! Good of you to call. If I am not mistaken you want to know if the cost for the dermatologist is covered, is that right?

It is clear that the user input is not directly related to the criterion. However, there is a higher level of semantic overlap, which the classifier will have to learn. The fact that spoken language is being classified, also gives rise to the novelty of this task, since that changes the form and meaning of the text that has to be classified. This is different from 'normal' short answer grading while in that case written data is classified, which results in different grammar, word order and use of words. Next to that, the data for this new application is transcripted, which is not the case in short answer grading. This makes it a very interesting case, while setting the fundamentals for constructing such a classifier enables this classifier to be applied to similar problems like the one at hand. Next to the fact that it is a novel application, it also fits well as a continuation of the research done in this field, as the analysis of related work is an important part of this research.

Structure

In the next section, the findings in related research in NLP, specifically in the scope of short answer grading, are described. In section three, I go into more detail about the specifications of the problem, the data and about the formulation of the desired classifier in technical terms. Section four is used to explore what implementations of the classifier have been tried previously by the company. After that, in chapter five, I formulate the implementation that I will test. In section six, the results are described. Section seven, consists of an analysis and interpretation of the results. Finally, in section eight, I will conclude with regard to my research questions, discuss its limitations and present my recommendations for future research.

2 Related Work

A part of NLP research closely related to this project, is Short Answer Grading. The biggest difference between these systems and the one at hand, is that the data consists of question-answer-pairs, whereas in the case of this research, the data is made up of criterion-transcription-pairs. This difference is important to note, as it changes the way in which the combination of the two parts of the input get processed and evaluated.

Most of the SAG-systems that will be addressed in this section have the following aspects that are necessary per classification:

- A question which the students must answer;
- one or more reference answers;
- the answer the student has put in;
- the grade, often binary, but in some cases scalar.

The majority of these systems are built on the idea of comparing the given answer to the reference answers, so these classifiers can be seen as reference-input-pair classifiers. A typical example [1] of the problem such a SAG must deal with, goes as follows:

- *Question:* You used several methods to separate and identify the substances in mock rocks. How did you separate the salt from the water?
- *Ref. Answer:* The water was evaporated, leaving the salt.
- *Student -1 Response:* By letting it sit in a dish for a day. – Classification: (Incorrect)
- *Student-2 Response:* Let the water evaporate and the salt is left behind. – Classification: (Correct)

There have been a number of approaches towards this problem. A study that tried to implement an automated Short Answer Grader has been performed at the University of Texas by Mohler and Rada [2]. Since this is a study done before the breakthrough of neural networks, they went for a different approach. Mohler et al. experimented with multiple measures of semantic similarity between student's short answers on a Computer Science assignment and the related correct answer. In the study they compared

different measures of similarity, but also the difference in knowledge-based measures and corpus-based measures. Although I went for a different approach in this research, the approach of Mohler et al. has similarities with the implementation currently in use by Faculty of Skills (see section 4).

A study where neural networks were used was conducted by Riordan et al. who were inspired by the success neural models were showing on the task of automated essay scoring [3]. However, they also noted some important differences between essay- and short answer grading, such as the length of the inputs, as well as the aspects the input has to be graded on. In the scope of essays, grammar and the organization of text is important, while for short answers, the focus is almost entirely on the content. The latter is also the case in my research, especially since it is spoken text that has to be classified, which means that grammar and organization have an even lesser role to play. Rioardan et al. found that, consistent with findings concerning neural models for other text classification tasks, using bi-directional LSTMS and making use of attention layers produced the best results. Next to that, this study found that certain word embeddings performed better than others and confirmed the belief that word embeddings are able to yield a good representation of the input for neural models.

Wang et al. proposed a version of a short answer grader that used both implicit and explicit representations of language[4]. They did this by constructing a neural model driven by knowledge-based features of the short texts concerning syntax and semantics. Their reasoning behind this 'hybrid' form came from the idea that using implicit and explicit representations separately would not yield good results. Firstly, because short texts do not contain enough information to enable standard NLP approaches to gather their explicit contents. Secondly, implicit representations tend to be harder to grasp for smaller inputs. Their Knowledge Powered Convolutional Neural Network yielded better accuracy than state-of-the-art methods on a number of tasks, proving that combining both representations is an interesting approach to classifying short texts.

Similarly to Mohler et al., Patil and Agrawal aimed at computing the semantic similarity between a number of reference answers and the student's answer [1]. Unlike the former study, they did make use of neural networks to compute this similarity. The embeddings of the reference answers, the student's input and two correct responses from other students go through a number of layers, most importantly a bi-directional LSTM (which enables the model to include information about the context of a word or an entire sentence). Later in the model, all the modeled sentences come together in a fully connected layer and finally, the output of that layer goes as input

in a logistic regression layer, that outputs the probability of the student's answer being correct. An important finding in this study is that using correctly classified students' responses that are correct as input, really helped the model in understanding what are important features for a good answer. This is also the goal of the reference answer, however those are often stated in general terms, not representative of how a student would say it. Having actual student input as reference seems to be of significant positive effect on the performance of the model.

The studies mentioned above differ on a number of aspects with the research in this paper:

- The size of the available dataset is very small, so that means that it is hard for the classifier overall to achieve good generalization over the data, but it also means that a pre-trained model is needed in order to obtain embeddings that can capture the contents of the text in a good way.
- The data for my research consists of transcriptions of spoken language, which differs from written language in vocabulary, grammar and also context. The latter point refers to the fact that all the transcriptions used in this research are recorded responses to a certain video the user has seen. This means that that video influences the user's answer.
- All the above studies are focussed on question answering, while the task in this research is not answering a question, but rather reacting in a way that is in line with a given criterion.

It can be concluded that criterion-transcription-evaluation is a novel task, as it differs on the points above with short answer grading. This means that other methods than the ones in previous studies have to be used.

3 Specifying the Problem

In this section I will elaborate on what the task consists of, both for a user of the platform as well as for the classifier that I am constructing. Next to that, the dataset that has been made available to me will be specified and an explanation will be given of the way in which the data is split between the train-, validation- and test sets.

3.1 The Task

As stated in the introduction, TrainTool is a webapp that lets its users improve their level of communication. This is done by letting trainees (as the users are called) record their response to a video the application shows them. Before they respond, they see a number of criteria or 'appraisal questions', as they are called in the data, which they have to keep in mind when responding. Afterwards, the trainees will be evaluated on whether or not they met those criteria. This research aims at automating that evaluation, specifically by fine-tuning and expanding a pre-trained model that is proven to give state-of-the-art results on other NLP tasks. This means that the classifier will have to take a criterion-user-input-pair as input and put out a value that indicates whether or not the related criterion is met.

3.2 The Data

For this research, I had access to two datasets from two different companies that have ran programs in TrainTool in the past. Every datapoint consists, most importantly, of the following data:

- An appraisal question, which is the criterion trainees have to meet in their recorded response;
- a transcription of the trainee's response, which is the transcribed audio of the recorded response;
- the label that was given to the criterion-transcription-pair by a human coach. This is considered as the ground truth for the classifier.

For clarity, I will give an example of an appraisal question with two corresponding transcriptions, evaluated as correct and incorrect, respectively. The example will also be translated. Note that the original Dutch transcriptions are altered for privacy reasons.

- *Appraisal question (original):* Ik vat het verhaal van de klant of de klantvraag samen.
- *Appraisal question (translation):* I give a summary of the client's question or issue.
- *Correctly evaluated transcription (original):* Dat kan ik zeker voor u nakijken als ik het goed begrijp is wat je wil weten wat er wel en niet vergoed wordt mijn vraag is alleen nu welke behandelingen wilt u graag ondergaan
- *Correctly evaluated transcription (translation):* I could certainly check that for you if I understand correctly you want to know whether or not you are covered my question now is what treatments would you like to undergo
- *Incorrectly evaluated transcription (original):* Goeiemiddag meneer vlissingen goed dat u belt we gaan dit even voor u uitzoeken
- *Incorrectly evaluated transcription (translation):* Good afternoon mister Vlissigen good of you to call we will check this for you

Next to the data noted above, each datapoint has metadata that is not used in this research. The original data consisted of videos of trainees. These videos were transcribed by an automated system and were not changed for the purpose of this research. These transcriptions are present in the dataset. It is important to note that the transcriptions were not done manually and contain mistakes, which could have effect on the performance of the model. However, it is also a good test for the classifier, since it would also have to deal with imperfect transcriptions should it be used as an autograder in TrainTool.
The two datasets I had access to came from a number of programs run by two different companies in TrainTool, each with different appraisal questions. For privacy reasons, the names of the companies are not stated, but they both operate in a different domain of work, which is beneficial for the level of representation the results show.

- The dataset from company 1 consisted of 12 appraisal questions, each with 50 responses.
- The dataset from company 2 consisted of 24 appraisal questions, each with a number of responses varying between 40 and 55.

The combination of the two datasets resulted in a dataset with 36 different appraisal questions and a total number of 2309 datapoints. However, some datapoints needed to be removed because the transcription was missing. I decided to experiment with this classifier on the following three setups:

- Train, validate and test on the entire dataset.
- Train, validate and test only on the data from company 1.
- Train, validate and test only on data from a single appraisal question.

These three setups would give a good insight as to what extent the classifier would be able to generalize over the data. For splitting the data in train-, validation- and test-sets, a 60%-20%-20% ratio was used. The split for the entire dataset and the data from company 1 was done by randomly selecting appraisal questions for validation and appraisal questions for testing, until the numbers were close to the ratio stated above. The rest of the data was left for testing. The choice to split the data bu appraisal questions was made to prevent overfitting on the appraisal questions, making the results more representative.

The dataset corresponding to the third setup stated above, came from appraisal question 87909, which was the appraisal question with the highest number of datapoints. To split this data in train-, validation- and test-sets, both the positively graded datapoints and the negatively graded datapoints were distributed following the same ratio as the two other datasets. This was done to make sure the negative and the positive datapoints were evenly distributed over the three sets. The following table shows the exact number of datapoints for every dataset, after the datapoints with missing transcriptions were removed:

	Train	Val.	Test	Total
Entire dataset	1313	471	482	2266
Company 1 dataset	1015	371	315	1701
87909 only dataset	60	24	29	113

Table 1: Overview of the number of datapoints in the datasets used in this research.

4 Previous Implementations

Before this research, employees at Faculty of Skills had already tried a number of implementations to automate the criterion-transcription-pairs. Among those implementations were two machine learning techniques, one simple, sequential neural network and one technique that involves calculating the semantic similarity between the user-input to a number of model sentences that are manually put in for every appraisal question. In this section I will elaborate on these previously implemented classifiers, making a distinction between the latter and the rest. This was done to emphasize that for the first three implementations, no manual labor or analysis of the data was needed, while for the fourth the model sentences need to be constructed.

4.1 The Baselines

The three methods previously implemented that require no manual labor beforehand will be used as baselines for the new implementation that will be introduced in the next section. These methods are the following:

- Bag of Words Logistic Regression
- Bag of Words Naive Bayes
- Bag of Words Sequential Model

It is important to note that all three implementations above take only the transcriptions as input, unlike the proposed implementation in this paper that uses criterion-transcription-pairs. Next to that, all three convert the input transcription to a Bag of Words vector, which simply makes a vocabulary of the entire input and counts the occurrences of each word, for each transcription. Word order and any higher level linguistic features are not modeled. Also note that the Logistic Regression and Naive Bayes methods do not make use of the validation dataset. In the Results section, the performance of these baseline methods is compared with the performance of the newly proposed implementation.

4.1.1 BoW Logistic Regression

For this implementation, the *LogisticRegression* linear model included in the *SKLearn* Python package was used. To calculate the accuracy, the number of correctly classified datapoints is divided by the total number of datapoints.

4.1.2 BoW Naive Bayes

This classifier was constructed using the *MultinomialNB* classifier included in the *SKLearn* Python package. To calculate the accuracy, the number of correctly classified datapoints is divided by the total number of datapoints.

4.1.3 BoW Sequential Model

This neural model consists of two fully connected neural network layers. The first layer consists of ten nodes, with the $ReLU$ activation function. That output gets sent to the second layer, which consists of one node with the *Sigmoid* activation function. During training, the Adam optimizer was used, the binary cross-entropy loss function was minimized and the accuracy was the ratio of the number of correctly classified transcriptions and the total number of transcriptions. The model was trained for 20 epochs, as the model did not improve when training for a larger amount of epochs.

4.2 Smart Sentence Matching

Since the above implementations only use Bag of Words as an input, the structure and semantic properties of the input gets lost. To combat this, a non-machine learning technique was proposed, which was called Smart Sentence Matching. This is also the implementation that is currently used in TrainTool. Using Facebook's multilingual LASER library, Smart Sentence Matching utilizes the distance between parts on the input and pre-constructed 'model sentences' in the semantic space. This implementation yielded better results than the implementations mentioned above, but did require manual labor. For this reason I decided not to use this implementation as a baseline.

5 BERT-based Model

In this section I will dive deeper into the model that I constructed and explain the choices I made regarding the (hyper)parameters.

Considering the earlier implementations of dealing with the criterion-transcription-evaluation task, I concluded that a Bag of Words representation of the input was not able to sufficiently grasp the semantics of the transcriptions. The input needs to be represented in a more sophisticated way than just counting the occurence of words, which is why Bag of Words is not a suitable representation of the transcriptions. Previous studies showed that the use of neural models yields state-of-the-art results on short answer grading, however for this research there was not much data available. This data sparsity meant that training a model on the task end-to-end is not realistic. Using a pre-trained model would help overcome this problem, as it already has a reasonable representation of multiple levels of the language used.

5.1 Google's BERT

One pre-trained language representation model that was introduced recently, is Google's BERT, or Bi-directional Encoder Representation of Transformers [5]. The creators of BERT stated that: '*the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications*'. I used the multi-lingual version of BERT, which included support for Dutch, as a starting point of my neural network. This meant that I would have a set basis for my neural network and only needed to fine-tune the (hyper)parameters. Next to the multilingual version of BERT, De Vries et al. developed a BERT model that was only trained on Dutch corpora, called BERTje [6]. While the Dutch part of the multilingual vesion of BERT was trained on just Dutch Wikipedia, BERTje was trained on several, high quality Dutch corpora with different domains. I decided to use both the multilingual BERT and BERTje and compare the results, as that would give me insight into whether or not the more specific training would yield better results.

The idea of the BERT models is that it is pre-trained by the authors and afterwards fine-tuned for a specific downstream task. The BERT models are pre-trained with two specific tasks:

- Masked Language Modelling, which means that part of the input tokens for training get masked in order for the model to predict those tokens. This approach to training results in a deep bidirectional representation of the language of the input tokens.
- Next Sentence Prediction, which means that the model is pre-trained on a binarized next sentence prediction task. This way, the model can be trained to understand sentence relationships, which is an important feature for a number of tasks the BERT model can be used for. In this research, understanding of a sentence relationship is important, since the input is a pair of sentences.

5.2 Preprocessing

In order for the BERT models to be able to handle the criterion-transcription-pairs and to start fine-tuning the model, some preprocessing was needed. Both the criterion and the transcription had to be tokenized by the tokenizer included with the respective BERT model. Before that, some tags needed to be added in front, in between and at the end of a criterion-transcription-pair, as both items were concatenated. The tokenizing resulted in an array of ID's corresponding to each token that was present in the vocabulary, named *input_ids*. In the process of tokenizing and adding those tags, the longest input sequence was found, which was saved as the *max_seq_len*. All concatenated pairs were padded until their length was equal to the *max_seq_len*. Finally, a *segment_ids* array had to be constructed, which for every token in the tokenized string indicated whether it was part of the transcription or the appraisal question. In the end, for every criterion-transcription-pair, the *input_ids* and *segment_ids* were the input to the neural network.

5.3 Architecture of the neural network

The basis of the neural network used for this task, was one of the BERT models, multilingual BERT or BERTje. Both models consisted of 12 layers and a hidden representation size of 768 [6] [5]. These models could be loaded into my *Keras* model with the *bert-for-tf2* package, using their respective configuration files, where the pre-trained parameters are stored.
As described in the subsection above, every criterion-transcription-pair starts with a tag. The final hidden vector of this starting tag, corresponds to the aggregate representation of the entire input-pair. For this reason, a *Lambda layer* was added after the BERT layer in my model. This simply takes the entire BERT-output as input, and returns the hidden vector that contains

the aggregate representation. As stated above, this hidden vector contains 768 nodes.

Even though Devlin et al. stated that only adding a classification layer after the BERT model would be sufficient, I opted to add four fully connected *Dense layers*. Through comparing the performance of the model on the validation set, I found that adding these extra layers yielded better results, which could be due to the fact that this enabled the model to grasp higher level features of the representation as put out by the BERT model. All four layers used *Rectified Linear Unit* as their activation function and had regularizers to apply penalties on their respective outputs, because this yielded the best results. Finally, a classification layer with two nodes was added, since the classification is binary. The activation function for this final layer was *Softmax*. Note that a *Dropout layer* was added between every pair of layers. This was done to combat overfitting during training.

The loss function that had to be minimized during training was the *Sparse Categorical Cross-Entropy* function. For accuracy, the *Sparse Categorical Accuracy* was used. Finally, *Adam* was used as the optimizer.

5.4 (Hyper)parameters of the model

After testing a large number of configurations, the following parameters yielded the best results on the validation dataset:

- **Dropout**: 0.2, equal for every *Dropout layer* in the network
- **Regularization rate**: 0.0001, equal for every fully connected *Dense layer* in the network
- **Learning rate**: 0.00001
- **Batch size**: 3, this was the maximum batch size that the available GPU was able to handle
- **Number of epochs**: 10. Note that the weights of the epoch with the best performance was saved.

6 Results

In the previous section, I explained the architecture and parameters of the proposed model. In this section I will compare the performance of the model I implemented with the performance of the simple machine learning implementations that are used as a baseline. Next to that the performance of the model will be evaluated on both the Multi-Lingual BERT version as well as the pre-trained Dutch BERT version called BERTje. I will do the comparisons in the three setups described in section 3. By comparing the models on these three different levels of complexity, I will be able to see whether or not the classifier is able to generalize over the entire dataset, or that it performs better on datasets where the data is more alike.

To be able to handle the random factors of the model, the performance is measured by taking the average training, validation and testing accuracy of running the model five times for ten epochs. Of those ten epochs, the one with the highest validation accuracy is saved and evaluated on the test set. I chose to do this because the accuracy of the model sometimes deteriorated on later epochs, which could be tied to overfitting on the training data. Taking the average also accounts for outliers in performance and makes sure the results show a clearer view on the actual capabilities of the classifier.

The following table shows the sparse categorical accuracy of the respective models for the training, validation and test sets of the entire dataset. The way the data is divided between training, validation and testing is discussed in section 3. The bold percentages are the highest values in each column.

My implementation	Train	Val.	Test
Multilingual BERT	84.44 %	71.68 %	67.51 %
BERTje	**93.70 %**	**73.34 %**	**68.31 %**

Baseline	Train	Val.	Test
BoW Logistic Regression	86.06%	68.37 %	63.70 %
BoW Naive Bayes	78.06%	69.43 %	61.62 %
BoW Sequential model	85.90%	70.50 %	64.90 %

Table 2: Sparse Categorical Accuracy of the entire dataset (appraisal questions and user inputs from both companies combined)

The following table shows the sparse categorical accuracy of the respective models for the training, validation and test sets of the dataset which only contains appraisal questions and corresponding user inputs from the single-company dataset. The bold percentages are the highest values in each column.

My implementation	Train	Val.	Test
Multilingual BERT	82.62%	67.82%	64.05%
BERTje	**87.45%**	**69.54%**	**67.11%**

Baseline			
BoW Logistic Regression	85,91%	59.57%	59.47%
BoW Naive Bayes	79.41%	63.61%	54.15%
BoW Sequential model	86.00%	64.70%	56.80%

Table 3: Sparse Categorical Accuracy of the dataset from Company 1

The following table shows the sparse categorical accuracy of the respective models for the training, validation and test sets of the data from only appraisal question 87909. The bold percentages are the highest values in each column.

My implementation	Train	Val.	Test
Multilingual BERT	86.67%	**75.00%**	**58.62%**
BERTje	86.67%	**75.00%**	**58.62%**
Baseline			
BoW Logistic Regression	**100%**	70.83%	**58.62%**
BoW Naive Bayes	96.66%	**75.00%**	**58.62%**
BoW Sequential model	95.00%	**75.00%**	**58.62%**

Table 4: Sparse Categorical Accuracy of the data from appraisal question 87909

7 Analysis and interpretation of the results

As table 1 reports, it immediately becomes clear that the implementation I brought forward in this paper outperforms the baselines on each of the training, validation and test sets, albeit the gain in accuracy is marginal. Next to that, the Dutch pre-trained version of the BERT model (BERTje) performs better than the Multilingual BERT version, even though the difference between them is small. It is also important to note that the training accuracy of the Multilingual version of BERT is significantly lower compared to BERTje, which means that BERTje is able to gather a better representation of the data.

The results on single-company dataset (table 2), show similar results as described above, the new implementation outperforms the baseline and BERTje outperforms the multilingual version of BERT. However, the difference in accuracy of both the BERT implementations versus the baseline is larger than when the models are trained on the entire dataset. Furthermore, the performance on only the single-company dataset overall, is lower than when the models are trained on the entire dataset. This could be due to the fact that, even though all the data is from the same company, there could still be a difference between the appraisal questions. This shows that, taking only data from one company does not guarantee that the inputs are more similar, which would make it easier for the model to understand the way in which the data should be evaluated.

Investigating the numbers in the third column, it is apparent that the newly introduced model as well as the baselines, give almost identical re-

sults. Not surprisingly, the accuracy percentages in the test are equal to the percentage of datapoints in the test set that are evaluated positively. In other words, all the implementations converge to a model where every datapoint is classified positively. One possible explanation for this outcome is the low number of training data. Since only data from one appraisal question could be used, there was not much data available for models to train on. The fact that the BERT implementations as well as the baseline give the same results, suggests that training a separate classifier per question is not a promising approach.

When looking at the bigger picture, it becomes clear that the implementation with the BERT models performs best on the entire dataset, which is not unexpected, when looking at the number of datapoints. Neural networks tend to work better when there is more data to train on. My findings support that statement, as performance lowered when the available data to train on decreased. It also becomes apparent that the Dutch pre-trained version of the BERT model (BERTje) performs better than the multilingual version of BERT. This happened in both the entire dataset as well as the dataset containing data from only one of the two companies. This is what is to be expected, as BERTje is trained on more Dutch corpora than multilingual BERT. The latter is only trained the Dutch version of Wikipedia [6]. Since this is spoken language, a model only trained on Wikipedia is likely to get worse results. However, the difference is not as big as I had expected. This could be due to the fact that, since the transcriptions are not entirely accurate, a lot of words are still misunderstood by BERTje.

8 Conclusion

It can be concluded that, when the model is trained, validated and tested on the entire dataset available, it outperforms the baselines set out for this research. The model even performs at its best when trained on the largest dataset, even though the appraisal questions differ significantly. Thus can be concluded that the model is indeed able to handle a large number of different appraisal questions and is able to capture those differences. It can also be concluded that, when the model is trained on just one appraisal question, the performance drops significantly.

Next to the above point, it is evident that the model was able to handle Dutch text. The results show that BERTje outperfomed the multilingual version of BERT, due to the fact that BERTje was trained on more Dutch corpora. This made the fact that the data was in Dutch less of an obstacle than I had expected beforehand. I was worried that Dutch would not be represented sufficiently in existing pre-trained models, but this turned out not to be the case. However. there are English versions of BERT that are at least twice the size of the BERT models used in this research and have been trained on a larger number of corpora. If this was the case for BERTje or any other Dutch model, it could allow for further performance gains.

After the findings in this research, the conclusion can be drawn that this novel task of criterion-transcription-evaluation can be considered as a text-classification task. Since BERT models have been fine-tuned in order to construct this classifier, a known method for text-classification was used to solve this classification problem. The findings in this research also add to the list of tasks that BERT models can be fine-tuned for.

After the conclusions above, it is important to note that the generalizability of this research is limited by the small amount of data available for this research. More data would yield better results, as the model can be trained on more criterion-transcription-pairs and have a better understanding what makes a good transcription. This is also a trend that can be seen in the results, as the largest dataset has the highest results.

Moreover, the imperfections in a number of transcriptions makes it hard for the model to effectively grasp the meaning of those transcriptions. This could have a negative effect on the results. To test this, the transcriptions could manually be corrected. However, this means that this model could not be used in the actual application, because there would be no possibility to alter the transcriptions in that case. Another possibility would be to either investigate better ways to gather the transcriptions, or by letting

users check their own transcriptions, which both have their own advantages and disadvantages.

Another limitation to this research was the limited amount of computing power that was available. Training the model took a significant amount of time and if more computing power was available, more testing could have been done. Next to more testing, larger architectures could have been tested and a larger batch size could have been used.

Finally, it is important to note that, unlike most text-classification tasks, the task uses transcriptions of spoken language. This means that results could be even better, if there was a model available that was pre-trained on corpora that consist of spoken language. As the BERT model is ever-evolving, chances are that such a model will become available in the future.

9 Appendices

9.1 Code of the implementation

See the attached *.py* files for the code of the baselines as well as the newly proposed implementation.

References

[1] Pranjal Patil and Ashwin Agrawal. Auto grader for short answer questions. 2018.

[2] Michael Mohler and Rada Mihalcea. Text-to-text semantic similarity for automatic short answer grading. pages 567–575, 2009.

[3] Brian Riordan, Andrea Horbach, Aoife Cahill, Torsten Zesch, and Chungmin Lee. Investigating neural architectures for short answer scoring. pages 159–168, 2017.

[4] Jin Wang, Zhongyuan Wang, Dawei Zhang, and Jun Yan. Combining knowledge with deep convolutional neural networks for short text classification. 350, 2017.

[5] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[6] Wietse de Vries, Andreas van Cranenburgh, Arianna Bisazza, Tommaso Caselli, Gertjan van Noord, and Malvina Nissim. Bertje: A Dutch Bert model. *arXiv preprint arXiv:1912.09582*, 2019.