Testing BERT for Generality in Cross-dataset Question Answering Performance

Jelle Bootsma
Eindhoven University of Technology
Eindhoven, The Netherlands
j.h.bootsma@student.tue.nl

Délano Gaasbeek
Eindhoven University of Technology
Eindhoven, The Netherlands
a.c.g.gaasbeek@student.tue.nl

Jeroen 't Lam
Eindhoven University of Technology
Eindhoven, The Netherlands
j.t.lam@student.tue.nl

Harish Sekar
Eindhoven University of Technology
Eindhoven, The Netherlands
h.sekar@student.tue.nl

Kjell Weijts
Eindhoven University of Technology
Eindhoven, The Netherlands
k.weijts@student.tue.nl

ABSTRACT

We adopt the existing pre-trained model called BERT, which stands for **B**idirectional **E**ncoder **R**epresentations from **T**ransformers, to create state-of-the-art models to work on a specific downstream task. The BERT model is a transformer-based machine learning technique for Natural Language Processing, pre-trained using unlabeled text on deep bidirectional representations. We fine-tune BERT-based models for Question Answering using different industry-standard datasets. Afterwards, we evaluate these models using evaluation sets of the other datasets, to test generality in cross-dataset Question Answering performance.
We find that a model trained on a specific dataset outperforms other models on that specific evaluation set by a significant margin, even in very similar datasets.

KEYWORDS

datasets, neural networks, machine learning, Natural Language Processing, Question Answering

ACM Reference Format:
Jelle Bootsma, Délano Gaasbeek, Jeroen 't Lam, Harish Sekar, and Kjell Weijts. 2021. Testing BERT for Generality in Cross-dataset Question Answering Performance. *ACM Trans. Graph.* 37, 4, Article 111 (August 2021), 8 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Natural Language Processing using machine learning is a subject that has been researched extensively. One of the main tasks in this subject is machine reading comprehension and answering questions based on the gained comprehension. An example of such a Question Answering task is shown in table 1. To research this task, multiple large-scale datasets have been created over the years [Cambazoglu et al. 2020]. With these datasets, a great number of varying model architectures and architecture improvements have been made [Wolf et al. 2020].
The model systems produced in recent years continuously improve their accuracy when tested on the created datasets. Even so, there are still issues with creating a model that can reliably answer questions using reading comprehension. One of these issues is the fact that many models have been specifically designed and trained for one task particularly on one dataset. This means that when the model is tested another task or on another dataset the accuracy decreases. To solve this issue, general model structures have been created [Gillioz et al. 2020].
One language representation model that was designed with this problem in mind, is BERT [Devlin et al. 2019]. BERT is designed in such a way that it can easily be fine-tuned to create models for a wide range of tasks. Because BERT works very well and is easy to adapt, it is often used in other research as the base of their architecture. So, BERT works well for multiple different tasks, but do the models it creates also perform well when tested on another dataset? In this paper, we fine-tune BERT on multiple different datasets and test these fine-tuned models on different datasets. The test results are then transformed such that a fair comparison can be made.
The goal is to find out if the models created by BERT work well on Question Answering datasets they have not been trained for. Furthermore using the results the most suited dataset for BERT can be found. The datasets not suited for BERT lead to possible improvements that can be tested.

2 METHODS

Before BERT could be fine-tuned on several QA datasets to measure cross-dataset QA performance, a BERT architecture that allowed for fine-tuning was needed. For this, the training examples[Schneider 2021b][Schneider 2021a] by Nikita Schneider were used to base our model on.

2.1 Model

The complete model architecture is built around the **BERT$_{\text{BASE}}$ EN Uncased** model. This pre-trained model is the smaller of the

Authors' addresses: Jelle Bootsma, Eindhoven University of Technology, Eindhoven, The Netherlands, j.h.bootsma@student.tue.nl; Délano Gaasbeek, Eindhoven University of Technology, Eindhoven, The Netherlands, a.c.g.gaasbeek@student.tue.nl; Jeroen 't Lam, Eindhoven University of Technology, Eindhoven, The Netherlands, j.t.lam@student.tue.nl; Harish Sekar, Eindhoven University of Technology, Eindhoven, The Netherlands, h.sekar@student.tue.nl; Kjell Weijts, Eindhoven University of Technology, Eindhoven, The Netherlands, k.weijts@student.tue.nl.

0730-0301/2021/8-ART111 $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

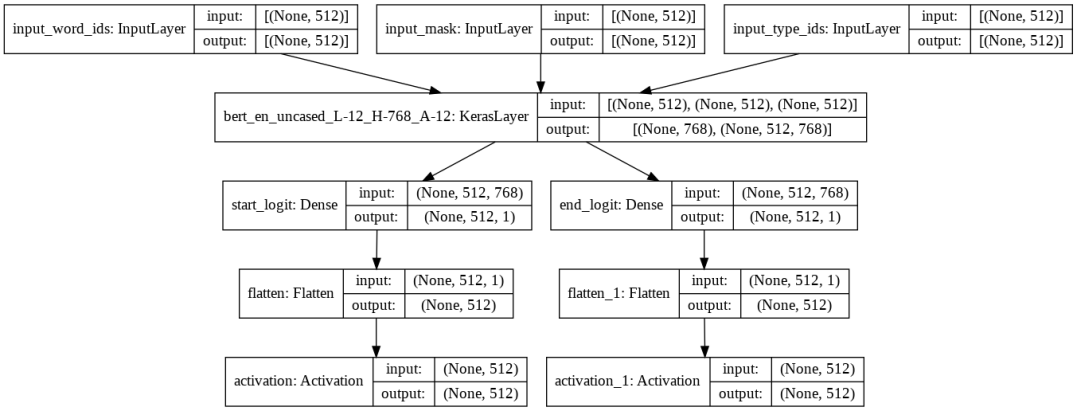

Figure 1: Overview of the model architecture

Article	Project Apollo
Paragraph	The Apollo program, also known as Project Apollo, was the third United States human spaceflight program carried out by the National Aeronautics and Space Administration (NASA), which accomplished landing the first human on the Moon from 1969 to 1972.First conceived during Dwight D. Eisenhower's administration as a three-man spacecraft to follow the one-man Project Mercury which put the first Americans in space,
Question 1	What project put the first Americans into space?
Answer	Project Mercury
Question 2	What program was created to carry out these projects and missions
Answer	Project Apollo

Table 1: An example of reading comprehension questions and answers about a given paragraph

two official BERT sizes, consisting of 12 hidden layers, a hidden size of 768, 12 attention heads, and 110 million parameters[Devlin et al. 2019]. The larger pre-trained model consists of 24 hidden layers, a hidden size of 1024, 16 attention heads, and a total size of 340 million parameters.

The $\textbf{BERT}_{\textbf{BASE}}$ was used to reduce the time required for fine-tuning. This will hurt the absolute performance of our model for QA tasks, but this should not be an issue for our research, as the same model architecture is used for all fine-tuning.

Around the BERT layer, there are several other layers, all of which can be seen in Figure 1.

On the input side of the BERT layer, there are 3 input layers, all with a shape size of 512, which corresponds to the BERT maximum sequence length. These layers feed the pre-processed data into BERT. These layers are:

(1) `input_word_ids`
The ids of the tokenized context and question. Padded with 0 to the maximum sequence length.
(2) `input_mask`
The attention mask tokens. This equals 1 for the length of the tokenized context and question, and is padded with 0 to the maximum sequence length.
(3) `input_type_ids`
The ids of the token types of the tokenized context and question. Also padded with 0 to the maximum sequence length.

On the output side of the BERT layer, there are two dense layers, one corresponding to the start of the answer location in the text, and one corresponding to the end location of the answer.
Both of these then feed into their own flatten layer, and finally a softmax activation layer. The start and finish of the answer in the context can be computed using the output of these activation layers, and the answer can then be retrieved.

2.2 Pre-processing

To handle all training data simply and understandably, a `Sample` class was used, as seen in Figure 2. This object represents a single question/answer pair. It also contains the context, location of the answer in the context, and the id of the question.

This means that all information required to train the model is contained in this model, which makes pre-processing easier, as the `Sample` class has a method `preprocess`, also from the training example[Schneider 2021a]. This method takes the variables contained in the class instance and pre-processes this information for the model.

In the pre-processing method, the location of the end of an answer is calculated, based on the start

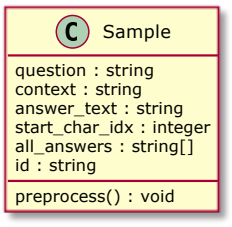

Figure 2: Class diagram of the Sample class

position and the length of the answer. Then both the question and the context are converted to tokens, using the `BertWordPieceTokenizer` from the `tokenizers` [Huggingface 2019] python package.
This tokenizer requires a vocabulary, to tokenize the inputs. For this vocabulary, the vocabulary included in the `BERT-Base, EN Uncased` model was used. After the context and question are tokenized, all tokens that contain a part of the answer are marked. If no tokens are marked the question is skipped, as no answer was included for this sample, and the question was also not marked as unanswerable.
After this, the context and question tokens are concatenated and truncated to the maximum sequence length of BERT.
The link to GitHub repository containing the implementation is given in appendix A

3 EXPERIMENTS

To test the assumption that the BERT model is general, the following experiment was performed: The model was fine-tuned on multiple different datasets and then each fine-tuned model was tested on each dataset. In this case the model was fine-tuned for three different datasets: SQuAD 1.1 [Rajpurkar et al. 2016], SQuAD 2.0[Rajpurkar et al. 2018] and CoQA [Reddy et al. 2019]. These datasets were chosen because of their use in multiple papers and the fact that some datasets contain unanswerable questions. The three fine-tuned models are then each tested against the dev-set of each dataset, which results in 9 different tests having been performed. Because different datasets use different evaluation scripts, the choice was made to evaluate all the results using the SQuAD 2.0 evaluation script. Both because of ease of access and the inclusion of unanswerable questions. This means that to allow the evaluation to be performed correctly, CoQA had to be converted into the SQuAD format. Additionally, any samples that exceeded the maximum sequence length of 512 tokens were removed from the dev-sets. This was done, because as clean a comparison as possible was desired, and questions that the models could not understand, as a result of their length, were undesired.
The results produced by the SQuAD evaluation script are in the F1 and EM form. The F1 (macro-averaged) score is calculated by measuring the average overlap between a prediction's answer to a question and the related answer in the ground truth. The EM (Exact Match) score is calculated by measuring the percentage of predictions whose answer exactly matches any answer in the ground truth. Both scores are shown in the results as values between 0 and 100.
During the fine-tuning process for all three datasets, the following settings for the hyperparameters are used:

- 2 epochs. BERT uses 3 and 2 epochs for SQuAD 1.1 and SQuAD 2.0 respectively
- A batch size of 8, as opposed to 32 (for SQuAD 1.1) and 48 (for SQuAD 2.0) used in BERT, due to time and resource constraints. E.g. limited GPU computing power
- A learning rate of $1e^{-5}$

3.1 SQuAD 1.1

SQuAD or **S**tanford **Qu**estion **A**nswering **D**ataset v1.0 was released around June 2016, and was one of the first massive high quality reading comprehension dataset. SQuAD consists of 100.000+ questions posed by crowd workers based on a set of Wikipedia articles. Each question can be answered by using a segment of text from the corresponding Wikipedia article. More specifically SQuAD 1.1 contains 107.785 question-answer pairs on 536 articles. Which is almost two orders of magnitude larger than previous manually labeled reading comprehension datasets. Furthermore, instead of providing a list of answers, the model can choose from SQuAD 1.1 only provides all possible spans in the passage. This means that SQuAD was larger and more open-ended than most other reading comprehension datasets from and before its time.
Because of this, and since BERT was originally tested on SQuAD, this dataset was chosen for the experiments in this paper.
There is a large amount of information about SQuAD, its format, and how to convert other datasets into the SQuAD format. With this information, our experiments used the SQuAD format as the base format when fine-tuning and testing the model. An example of the SQuAD format can be seen in Table 2. This format is then split into one `Sample` object, described in Figure 2, for each question-answer pair. Which are used as input for the model. The parameters of the `Sample` class correspond mostly to the SQuAD format, the only changes are `answer_text` = `text`, `start_char_idx` = `answer_start`, and `all_answers` = an array containing all text string values in answers. The predictions later consist of a dictionary containing the id of the question and the predicted answer made by the model. These predictions are then compared against the actual answers from the original dataset in the evaluation script v2.0 provided on the SQuAD website[Stanford 2018].

3.2 SQuAD 2.0

Similar to the SQuAD 1.1 dataset, SQuAD 2.0 is a very massive and high-quality reading comprehension dataset, released in February 2021. It takes the ~100.000 questions from SQuAD 1.1 and adds over 50,000 unanswerable questions written by crowd workers that look similar to answerable ones. Because of this, the system should now not only be able to answer questions correctly but also determine which questions are unanswerable based on the given paragraph.
This dataset was chosen for the experiments both since BERT was originally tested on SQuAD 2.0 and because this version includes the addition of unanswerable questions.
SQuAD 2.0 has the same format as SQuAD 1.1 and as such the parser for the `Sample` objects is the same. The only addition is that an unanswerable question has an `answer_text` and `all_answers` parameter containing an empty string and an array containing an empty string, while `start_char_idx` = 0.

3.3 CoQA

CoQA is a large-scale dataset for building **Co**nversational **Q**uestion **A**nswering systems, released in August 2018. The goal of the CoQA challenge is to measure the ability of machines to understand a text passage and answer a series of interconnected questions that appear in a conversation. CoQA contains 127.000+ questions and answers

Title	University of Notre Dame
Paragraphs:	{context, qas}
context:	Architecturally, the school has a Catholic character. Atop the Main Building's gold dome is a golden statue of the Virgin Mary. Immediately in front of the Main Building and facing it, is a copper statue of Christ with arms upraised with the legend Venite Ad Me Omnes. Next to the Main Building is the Basilica of the Sacred Heart. Immediately behind the basilica is the Grotto, a Marian place of prayer and reflection. It is a replica of the grotto at Lourdes, France where the Virgin Mary reputedly appeared to Saint Bernadette Soubirous in 1858. At the end of the main drive (and in a direct line that connects through 3 statues and the Gold Dome), is a simple, modern stone statue of Mary.
qas:	{answers, question, id}
answers:	{answer_start, text}
question:	To whom did the Virgin Mary allegedly appear in 1858 in Lourdes France?
id:	5733be284776f41900661182
answer_start:	515
text:	Saint Bernadette Soubirous

Table 2: The format of the SQuAD 1.1 dataset

that were collected from 8000+ conversations. Due to this format, the dataset has some unique features that other datasets do not have. 1) The questions are all conversational; 2) the answers can be free-form text; 3) each answer also comes with an evidence subsequence highlighted in the passage; 4) the passages are collected from seven diverse domains. Since this dataset is set up in a uniquely, it poses a lot of challenging phenomena that are not present in other existing comprehension datasets.

Because of the large range of questions as well as the unique features CoQA provides and the fact that BERT had not yet included CoQA in their paper, this dataset was chosen for the experiments.

An example of the CoQA format is shown in table 3. Similar to the SQuAD dataset, the data is converted into a `Sample` object. The parameters for `Sample` are created as follows: `question = [questions][i]`, `context = story`, `answer_text = answers[i]`, `start_char_idx = span_start`, `all_answers = {span_text, input_text}` and `id = id + "_" + turn_id`

source	mctest
id	3dr23u6we5exclen4th8uq9rb42tel
filename	mc160.test.41
story:	Once upon a time, in a barn near a farm house, there lived a little white kitten named Cotton. Cotton lived high up in a nice warm place above the barn where all of the farmer's horses slept. But Cotton wasn't alone in her little home above the barn, oh no. She shared her hay bed with her mommy and 5 other sisters. All of her sisters were cute and fluffy, like Cotton. But she was the only white one in the bunch. The rest of her sisters were all orange with beautiful white tiger stripes like Cotton's mommy. Being different made Cotton quite sad. She often wished she looked like the rest of her family. So one day, when Cotton found a can of the old farmer's orange paint, she used it to paint herself like them. When her mommy and sisters found her they started laughing. "What are you doing, Cotton?!" "I only wanted to be more like you". Cotton's mommy rubbed her face on Cotton's and said "Oh Cotton, but your fur is so pretty and special, like you. We would never want you to be any other way". And with that, Cotton's mommy picked her up and dropped her into a big bucket of water. When Cotton came out she was herself again. Her sisters licked her face until Cotton's fur was all all dry. "Don't ever do that again, Cotton!" they all cried. "Next time you might mess up that pretty white fur of yours and we wouldn't want that!" Then Cotton thought, "I change my mind. I like being special"
questions	{input_text, turn_id}
answers	{span_start, span_end, span_text, input_text, turn_id}
additional_answers	{0,..., n}
input_text	What color was Cotton?
turn_id	1
span_start	59
span_end	93
span_text	a little white kitten named Cotton
input_text	white
turn_id	1
0,...,n	{span_start, span_end, span_text, input_text, turn_id}

Table 3: The format of the CoQA dataset

4 RESULTS

In this section, the paper presents the results obtained using a model trained on 3 different QA datasets and evaluated on the same dev dataset to measure the cross-dataset performance which counts to a total of 9 NLP tasks. There are two metrics used to evaluate the performance of the fine-tuned BERT model.

(1) The F1 score which captures the precision and recall of the words that are being chosen as part of the answer are truly belonging to the part of the answer.

(2) The exact match EM score which calculates the number of answers that are exactly correct with the same start and end index.

The scores obtained are based on the predictions made from the dev set as the actual test set is preserved from the reach of the public to maintain its integrity. The corresponding F1 and EM scores are

presented in Table 4.
It is notable that every model outperformed all other models, when using the evaluation set corresponding to their training set, by a significant margin, with a single EM score exception. The performance results of our model is presented in Table 4. Our SQuAD 1.1 model tested on the same dev set achieved an EM score of 78.8 and an F1 score of 86.3, which is comparable to the results seen in [Devlin et al. 2019] which on evlauation achieved an EM score of 80.5 and an F1 score of 88.5. This paper is based on the model architecture described in [Devlin et al. 2019]. More specifically it uses the $\text{BERT}_{\text{BASE}}$ model but fine-tuned with different hyperparameters, thus achieving the previously mentioned scores. Since the results in this paper are very close to the results of the original implementation, the assumption can be made that our implementation is correct and the difference can be explained by different hyperparameters.

4.1 SQuAD 1.1 model

Looking at the results of the SQuAD 1.1 model, it can be seen that it performed the best on the SQuAD 1.1 evaluation set, while achieving lower scores on the SQuAD 2.0 and CoQA evaluation sets. In Table 5, the EM and F1 values are split across the answerable and unanswerable questions in the evaluation set, which clarifies the source of the worse result of the SQuAD 1.1 model on the SQuAD 2.0 evaluation. The decrease of the overall EM and F1 values are the consequence of the unanswerable questions, as here results are only around one twentieth of the answerable question results.
The results of the SQuAD 1.1 model on the CoQA evaluation set are further reduced from the SQuAD 2.0 evaluation set results. The F1 and EM scores have also diverged, and the F1 score is now a lot higher than the EM score. A possible explanation for this could be the conversational nature of the CoQA dataset. The longer answers in this dataset could make an exactly matching answer more unlikely, while the predicted answer is still part of the conversational correct answer, thus impacting the F1 less than the EM score.

4.2 SQuAD 2.0 model

The performance of the SQuAD 2.0 fine-tuned model is by far the best performer on the SQuAD 2.0 evaluation set. Table 5 again indicates that the unanswerable questions are the cause of this result. Even though the SQuAD 2.0 model performs worse than the SQuAD 1.1 model on answerable questions, the unanswerable performance of the SQuAD 2.0 model makes it far outperform the other models.
Keeping this in mind, the performance of answerable questions was degraded significantly when compared to the SQuAD 1.1 model performance, decreasing by ~ 8 points on both F1 and EM. This drop in performance pertains to both the SQuAD 1.1 and the answerable part SQuAD 2.0 evaluation sets.
This pattern continues for the CoQA results, as the the performance of answerable questions is lower than the SQuAD 1.1 model, but the the performance of unanswerable questions is greatly improved. Here the overall score of the SQuAD 2.0 model is not greater the SQuAD 1.1 model, as there are far fewer unanswerable questions.

System	Dev datasets					
$\text{BERT}_{\text{BASE}}$	SQuAD 1.1		SQuAD 2.0		CoQA	
	F1	EM	F1	EM	F1	EM
Model trained on SQuAD 1.1	86.3	78.8	45.1	41.2	33.3	13.6
Model Trained on SQuAD 2.0	77.3	70.8	74.4	71.0	25.1	10.3
Model Trained on CoQA	44.1	11.6	23.9	7.6	48.2	12.9

Table 4: Performance results of the model fine-tuned on SQuAD variants and CoQA datasets

4.3 CoQA model

For the CoQA fine-tuned model, the divergence of F1 and EM scores noted in subsection 4.1, can be seen on all evaluation sets. This makes sense, as a model fine-tuned on CoQA would predict more conversational answers more often, thus not scoring as many exact matches, but leaving the F1 score less affected, as the correct answer is given by the CoQA model.
This is most likely a major reason to the low performance of CoQA when compared to the other models on the non-CoQA evaluation sets. In terms of F1 score, the CoQA model outperforms the other models on its own evaluation set. The SQuAD 1.1 model outperforms it somewhat in terms of EM scores. This difference is only 0.7 points, and the difference between F1 scores of these two models is far larger, so overall the SQuAD 1.1 model is still outperformed by the CoQA model
The SQuAD 2.0 model also outperforms the CoQA model on the CoQA evaluation set, but only with regards to unanswerable questions. However, the amount of unanswerable questions is not enough to compensate for the notably lower performance of SQuAD 2.0 on answerable questions in the CoQA evaluation set. Notably, the cross-dataset evaluation scores of the CoQA model are far closer than other models, all F1 scores being between 44 and 49, and the EM scores falling between 10 and 13, as long as we limit the scores to answerable questions.

5 DISCUSSION

5.1 HotPotQA

Originally, the intention was to include HotPotQA as a fourth dataset to test. However, we decided to not include HotPotQA, as the fine-tuned model has exceedingly bad performance. As this would not represent HotPotQA, and we would not be able to draw any meaningful conclusions, as our architecture is not suited for training HotPotQA.
The main reason for this is the structure of HotPotQA data. In the HotPotQA dataset, a question has a definitive answer, with a set of supporting facts. These facts do not include a location in the context, and neither does the answer. SQuAD and CoQA do include locations of answers. As our architecture predicts the specific location of answers in a given context, training is difficult, if these locations are not part of the data. We tried to extract the locations in the context using the supporting facts, but this did not work properly.

	Evaluation dataset									
	SQuAD 1.1		SQuAD 2.0				CoQA			
Question count	Answerable		Answerable		Unanswerable		Answerable		Unanswerable	
	10514		5882		5895		7743		65	
	F1	EM	F1	EM	F1	EM	F1	EM	F1	EM
Model trained on SQuAD 1.1	86.3	78.9	86.1	78.3	4.3	4.3	33.5	13.6	12.3	12.3
Model trained on SQuAD 2.0	77.3	70.8	75.6	68.9	73.2	73.2	24.9	10.0	47.7	47.7
Model trained on CoQA	44.1	11.6	43.4	10.8	4.6	4.6	48.6	13.0	12.3	12.3

Table 5: Performance results of the model split out into answerable and unanswerable

Finally, we decided to drop HotPotQA from the test and continued with 3 datasets.

5.2 CoQA

One thing that might have influenced the results of the model trained on the CoQA dataset, is the fact that we trimmed our evaluation datasets such that the input for the models could not be larger than 512. This means that the `story` plus `question` once tokenized can not be longer than 512 tokens otherwise it is removed from the evaluation dataset. For the two SQuAD datasets this is most likely a very small number of Samples, but CoQA was designed with conversational questions in mind, which lead to larger stories and questions. While this was not tested it is highly likely that a higher relative number of Samples was removed from the CoQA evaluations sets in comparison to the other datasets. This means that the model trained on CoQA will have received Samples with more than 512 tokens during training and then not during evaluation. If this number of removed Samples is significant, the results for the CoQA evaluation would not be fair. This can of course be fixed by using another evaluation method specifically for CoQA, or by not trimming the datasets.

5.3 Hyperparameters

As mentioned before, due to the limited time and available resources, some hyperparameters had to be altered, in comparison to the settings used in the BERT paper, in order to allow us to perform the experiments.

Due to the low GPU computing power of our machines, we decided to fine-tune our models in Google Colab Pro. There were still some limitations to both the available amount of GPU computing power as well as the maximal time that can be spend in one session before a time-out occurs, but there was still a high enough increase in computing power to perform the fine-tuning. In order to do this though, some hyperparameters needed to be lowered or changed.

First of all, the `epochs` were set to 2, while most fine-tuning in the BERT paper uses 3 or more `epochs`. By decreasing the amount of epochs, the time it took to fine-tune a model was decreased significantly, allowing us to perform all desired experiments in time.

Secondly, the `batch size` was set to 8, while BERT uses a `batch size` of 32 or more for most of their fine-tuning. By decreasing the `batch size`, both the amount of time it takes to finish the fine-tuning process, as well as the amount of ram required to run it, decreases significantly. Unfortunately, a smaller `batch size` can lead to a decrease in model performance compared to a larger `batch size`, but since our resources were limited, we had no choice but to choose a smaller `batch size`.

The last parameter that was changed is the `learning rate`. The `learning rate` used to fine-tune the models in this paper is set to $1e^{-5}$ as opposed to $5e^{-5}$ used in BERT. By decreasing the value of this parameter, the performance of the fine-tuned model should be better than one fine-tuned with a higher value. We found that decreasing this parameter still fitted within the GPU computing power and did not increase the time it took to fine-tune by much, which is why we opted for higher performance by choosing a lower `learning rate`.

5.4 Future work

A possible continuation of this research could be an investigation of the cross-dataset performance of a model trained on a union of several datasets. Relative performance drop on specific evaluation sets can then be compared with the improved generality of the model if any such improvement occurs.

Another way this research could be expanded upon is by performing the same experiments as described in this paper, but with the hyperparameters set to the same values used by BERT. This would of course mean that some extra computing power is needed, but it would also make the results easier to compare to the results in the BERT paper.

6 CONCLUSION

Cross-dataset performance of question answering models is an important component of creating applicable question answering systems, as it can not simply be assumed that the question types or formatting of real world situation will be comparable to the dataset used for training. Testing models on different datasets, can simulate these differences in real world questions.

In our $\text{BERT}_{\text{BASE}}$ based architecture, every model fine-tuned using a specific dataset, outperformed all other models on the corresponding evaluation set, by a large margin, often more than 10 points. This means that the given BERT architecture does not generalize well for question answering tasks, if only trained on a single one

of the tested datasets.
Real world applications that require support for many different ways of phrasing questions would therefore require to be fine-tuned using a dataset where question formatting is less homogeneous, or would require a different model, which is intrinsically better generalizable, such that cross-dataset performance would improve.

7 RELATED WORK

The research of this report was designed and based on earlier work on the BERT general language model. This section will discuss the current state of the art considering BERT and the use of the datasets the BERT model was trained and evaluated on in this research.

7.1 BERT

This research adopted the $\mathbf{BERT_{BASE}}$ implementation used in the work of [Devlin et al. 2019], with its foundations found in the work of [Vaswani et al. 2017]. Devlin et al. (2019) already fine-tuned their $\mathbf{BERT_{BASE}}$ model on the SQuAD 1.1 dataset to test its performance on Question Answering (QA) tasks. Evaluation on the Dev set of SQuAD 1.1 resulted in an F1 score of 88.5, meaning it outperformed the top leaderboard systems as of Dec 10th, 2018.
One study replicated the work of Devlin et al. (2019), aiming to assess the influence of hyperparameters and the size of training data. [Liu et al. 2019] The result was a Robustly Optimized BERT approach (RoBERTa), where the BERT model was pre-trained over more data and for longer. It was fine-tuned and evaluated on both the SQuAD 1.1 and SQuAD 2.0 datasets and resulted in approximately state-of-the-art results.
Another follow-up study on the BERT model as proposed by Devlin et al. (2019) introduces two approaches for parameter cutback, supposed to lead to decreased memory consumption and boosted training speed. [Lan et al. 2019] The result, being A Lite BERT (ALBERT) resulted in comparable performances after fine-tuning and evaluating on SQuAD 1.1 as well as on SQuAD 2.0.
None of the aforementioned papers tested a BERT model for generality, that is, fine-tuning on dataset x and testing it on distinctive datasets. This research, therefore, used the SQuAD 1.1, SQuAD 2.0, and CoQA datasets for fine-tuning and cross-evaluation in its intent to test BERT models for generality.

7.2 SQuAD 1.1

As aforementioned, the first dataset used to fine-tune our BERT model was the Stanford Question Answering Dataset (SQuAD 1.1) [Rajpurkar et al. 2016]. This dataset consists of 100,000+ QA pairs of which the answer types can be divided into ten categories such as dates, persons, or locations. Rajpurkar et al. (2016) constructed a logistic regression model and tested its performance on their SQuAD 1.1 dataset, resulting in an F1 score of 51.0%, being less than the human F1 of 86.8% that was found (See appendix B).

7.3 SQuAD 2.0

Following the SQuAD 1.1 dataset for QA, this report also fine-tuned and evaluated the BERT model on the SQuAD 2.0 dataset. SQuAD 2.0 was constructed to address fragility in earlier datasets, primarily being their focus on answerable questions only [Rajpurkar et al. 2018]. Aside from existing SQuAD data, it contains 50,000+ unanswerable questions which are rather similar to answerable questions. Hence, for good performance on SQuAD 2.0, a model must be able to identify unanswerable questions aside from correctly answering answerable questions. For comparison, Rajpurkar et al. (2018) present Exact Match (EM) and F1 scores of different systems on both SQuAD v1.0 and SQuAD 2.0. The presented systems, being BNA, DocQA, and DocQA + ELMo, all generated an F1 score of about 20 points lower on SQuAD 2.0 than on SQuAD 1.1, indicating that there is still room for improvement in general language models for QA.

7.4 CoQA

The third and last dataset that was used to fine-tune and evaluate the performance of our BERT model was CoQA, an abbreviation for Conversational Question Answering. It consists of 127,000 QA pairs, with the questions being conversational and the answers free-form text [Reddy et al. 2019]. Reddy et al. (2019) evaluated several reading comprehension models on CoQA and found the best performing system resulting in an F1 score of 65.4%, which is 23.4 points worse than the human performance (See table 8 in appendix B).

ACKNOWLEDGMENTS

We would like to acknowledge and thank M. Fang for his guidance and advice during the development of this paper and the implementation of the experiments described in it.

REFERENCES

Barla Cambazoglu, Mark Sanderson, Falk Scholer, and Bruce Croft. 2020. A Review of Public Datasets in Question Answering Research. https://dl.acm.org/doi/abs/10.1145/3483382.3483389
Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]
Anthony Gillioz, Jacky Casas, Elena Mugellini, and Omar Abou Khaled. 2020. Overview of the Transformer-based Models for NLP Tasks. In *2020 15th Conference on Computer Science and Information Systems (FedCSIS)*. 179–183. https://doi.org/10.15439/2020F20
Huggingface. 2019. Huggingface/tokenizers: Fast state-of-the-art tokenizers optimized for research and production. https://github.com/huggingface/tokenizers
Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. arXiv:1909.11942 [cs.CL]
Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. arXiv:1907.11692 [cs.CL]
Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know What You Don't Know: Unanswerable Questions for SQuAD. arXiv:1806.03822 [cs.CL]
Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250 [cs.CL]
Siva Reddy, Danqi Chen, and Christopher D. Manning. 2019. CoQA: A Conversational Question Answering Challenge. arXiv:1808.07042 [cs.CL]
Nikita Schneider. 2021a. Dredwardhyde/Bert-Examples: Fine tuning Bert for text classification and question answering using tensorflow and pytorch frameworks. https://github.com/dredwardhyde/bert-examples
Nikita Schneider. 2021b. Fine tuning bert for text classification and question answering using tensorflow framework. https://medium.com/swlh/fine-tuning-bert-for-text-classification-and-question-answering-using-tensorflow-framework-4d09daeb3330
Stanford. 2018. The Stanford Question Answering Dataset. https://rajpurkar.github.io/SQuAD-explorer/
Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. arXiv:1706.03762 [cs.CL]

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-Art Natural Language Processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*. Association for Computational Linguistics, Online, 38–45. https://doi.org/10.18653/v1/2020.emnlp-demos.6

A SOURCE CODE

All source code is publicly available on GitHub.
The used trimmed dev datasets are also included in this repository, as well as all predictions for the corresponding dev sets.
Source code can be found here: https://github.com/JelleBootsma/Cross-Dataset-QA-Performance

B LEADERBOARD RESULTS

System	Test dataset	
	EM	F1
Human Performace	82.3	91.2
#1 Single model - ANNA	90.6	95.7
#2 Single model - LUKE	90.2	95.3
#3 Single model - XLNet	89.8	95.0
#25 BiDAF + Self Attention + ELMo - Single	78.5	85.8

Table 6: The SQuAD 1.1 leaderboard results with human performance

System	Test dataset	
	EM	F1
Human Performace	86.8	89.4
#1 Ensemble model - IE-Net	90.9	93.2
#2 Ensemble model - FPNet	90.8	93.1
#3 Ensemble model - IE-NetV2	90.8	93.1
#78 BERT-base (single-model)	70.8	74.4

Table 7: The SQuAD 2.0 leaderboard results with human performance[Stanford 2018]

System	Test dataset	
	In-domain	Overall
Human Performace	89.4	88.8
#1 Ensemble model - RoBERTa + AT + KD	91.4	90.7
#2 Ensemble model - TR-MT	91.5	90.7
#2 Single model - RoBERTa + AT + KD	90.9	90.4
#3 Ensemble model - TR-MT	91.1	90.2
#40 Vanilla DrQA - Single model	54.5	52.6

Table 8: The CoQA leaderboard results with human performance. In-domain F1 score refers to evaluating the trained model with the dataset which belongs to the question-answering family