Question Answering

Natural Language Processing Project

Francesco Ballerini
francesco.ballerini3@studio.unibo.it

Emmanuele Bollino
emmanuele.bollino@studio.unibo.it

Tommaso Giannuli
tommaso.giannuli@studio.unibo.it

Manuel Mariani
manuel.mariani2@studio.unibo.it

Alma Mater Studiorum Università di Bologna
Second Cycle Degree in Artificial Intelligence
AY 2021–2022

1 Executive Summary

Task Definition. Question Answering is the task of selecting the span of text inside a passage that answers a given reading-comprehension question. Our task is to build a model performing question answering on the SQuAD1.1 dataset, which, as opposed to the 2.0 version, *does not* contain unanswerable questions, namely questions whose answer cannot be extracted anywhere from the corresponding passage.

Proposed Solution. We tackle Question Answering on SQuAD1.1 by fine-tuning knowledge-distilled versions of BERT proposed in the recent literature and comparing their results. Specifically, we test five models [Section 3.3]:

- $\text{BERT}_{\text{TINY}}$, $\text{BERT}_{\text{MINI}}$, $\text{BERT}_{\text{SMALL}}$, and $\text{BERT}_{\text{MEDIUM}}$ (by Google Research): increasingly complex models which are first pre-trained with MLM + NSP objectives and then distilled with $\text{BERT}_{\text{LARGE}}$ as teacher.
- TinyBERT (by Huawei Noah's Ark Lab): another model obtained through distillation—without pre-training—with $\text{BERT}_{\text{BASE}}$ as teacher.

Moreover, we experiment with different perspectives to tackle our task, which can be interpreted as (i) a classification task, (ii) a regression task, or (iii) a combination of the two [Section 4.4]. Guided by our experimental evidence, we decide to adopt the standard classification-task viewpoint, in which there are two classes for each token, one for the start and the other for the end position of the answer inside the passage. The aim of the model is to assign scores indicating how likely it is for a token to be the start/end of the answer.

Methodology. We first compare the results of fine-tuning our models to asses that the largest one—$\text{BERT}_{\text{MEDIUM}}$—is also the best performing [Section 4.2]. Then we test some regularization techniques and other potential improvements [Section 4.3]. Because of our limited computational resources, these tests are performed on $\text{BERT}_{\text{MINI}}$, with the underlying assumption that the conclusions we draw are going to be valid for $\text{BERT}_{\text{MEDIUM}}$ as well.

Results. We finally apply to $\text{BERT}_{\text{MEDIUM}}$ all the modifications that were found beneficial on $\text{BERT}_{\text{MINI}}$. The resulting model reaches an EM score of 0.551 and an F1 score of 0.749 on a portion of the training set used for validation. Results show that our final model overfits less and reaches higher scores than the corresponding baseline, but such improvements are much less significant then when applied to $\text{BERT}_{\text{MINI}}$ [Section 4.5]. This suggests, quite understandably, that a bigger and more powerful model such as $\text{BERT}_{\text{MEDIUM}}$ has less margin for improvement.

Conclusions. Through the use of compressed versions of language representations learned by larger pre-trained BERT models, we manage to get results that, despite being far from the state-of-the-art, provide some insights on this very active research topic. Future extensions could include a more exhaustive search of the hyperparameter space made possible by a higher computational budget than the one at our disposal.

2 Background

SQuAD. Question Answering is the task of selecting the span of text inside a passage that answers a given reading-comprehension question; the passage is usually referred to as *context*. The Stanford Question Answering Dataset (SQuAD) is a popular benchmark dataset for this task, consisting of 100,000+ questions posed by crowdworkers on a set of 500+ Wikipedia articles [4]. We performed our experiments on SQuAD1.1, which, as opposed to the 2.0 version, *does not* contain unanswerable questions, namely questions whose answer cannot be extracted anywhere in the corresponding passage.

BERT. BERT (Bidirectional Encoder Representations from Transformers) [11] is an architecture proposed by Google which consists of a bidirectional Transformer [8]—that is, a Transformer encoder—and has been pre-trained on large unlabeled corpora for the tasks of Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). It can be fine-tuned on a downstream task simply by (i) initializing the model with the parameters from pre-training, (ii) adding a task-specific output layer, and (iii) training the resulting architecture end-to-end. The rationale behind the choice of BERT is that, differently from other recent Transformer-based approaches—most notably, OpenAI's GPT [10]—it is *bidirectional*, which has been shown to be beneficial when applying fine-tuning based approaches to token-level tasks such as question answering, where it is crucial to incorporate context from both directions [11].

Distillation. Knowledge Distillation is the process of transferring information from a highly-parameterized and accurate *teacher* model to a more compact and thus less expressive *student*. For classification tasks, distillation exposes the student to soft labels, namely the class probabilities produced by the teacher $p_c = \text{softmax}(z_c/T)$, where p_c is the output probability for class c, z_c the logit for class c, and T is a constant called *temperature* that controls the smoothness of the output distribution. The softness of the labels enables better generalization than the gold *hard labels* [3, 12]. The reason we were drawn to models pre-trained trough distillation with BERT as teacher is that such an approach allowed us to enjoy the language representations learned by BERT—or, at least, a compressed version of them—while being able to perform fine-tuning with our limited computational resources.

3 System Description

As anticipated in Section 2, our architectures are all pre-trained smaller versions of BERT which were proposed in the recent literature and are publicly available. Such models are supposed to be fine-tuned on a downstream task, which, in our case, is question answering on SQuAD1.1.

3.1 Task Definition

We modeled question answering as a sequence classification task by adding a linear layer on top of our distilled BERT architectures. For each input token, the linear layer—which, from now on, we will refer to as *question-answering head* (*QA head* for short)—outputs two scores (logits), one for the "start-token class" and the other for the "end-token class". The predicted answer is then computed as the span of text inside the context that goes from the argmax of the start-token scores

to the argmax of the end-token scores[1]. For the fine-tuning step, the loss is computed as the average of the cross-entropy of the start-token scores and the end-token ones [14].

3.2 Data Pre-processing

The SQuAD1.1 dataset is partitioned into training, development and test set [4]. Both the training set and the development set are available to the public; however, one of our problem requirements was to rely on no sources of data other than the training set. We therefore extracted ~10% of the training set to use as validation set, with some additional care to make sure that the same question topic—as denoted by the `title` field in the data—was not getting split between the two subsets. From now on, when talking about *training* and *validation set*, we will refer to this 90–10 split of the original SQuAD1.1 training set.

Once the text has been converted to lowercase and accents have been stripped, the input samples—one for each question—go through the standard BERT tokenization pipeline, namely WordPiece embeddings [5] and a question–context encoding of shape

`[CLS]` question tokens `[SEP]` context tokens `[SEP]` `[PAD]` ... `[PAD]`

where `[CLS]` (classification token), `[SEP]`, and `[PAD]` are special reserved tokens [11].

3.3 Models

The architectures we experimented with can be divided into two categories:

1. $\text{BERT}_{\text{TINY}}$, $\text{BERT}_{\text{MINI}}$, $\text{BERT}_{\text{SMALL}}$, and $\text{BERT}_{\text{MEDIUM}}$ [12, 15] are increasingly complex BERT-based models which are first pre-trained with MLM + NSP objectives and then distilled with $\text{BERT}_{\text{LARGE}}$ [11] as teacher. Pre-training and distillation are shown to have a compound effect even when sequentially applied on the same data [12].

2. TinyBERT [13, 16] is instead distilled with $\text{BERT}_{\text{BASE}}$ as teacher. While no pre-training precedes the distillation process in this case, the authors propose to fine-tune the model through a second distillation, where the teacher is now a fine-tuned $\text{BERT}_{\text{BASE}}$ [13]. Instead, we adopted the standard fine-tuning procedure described in Section 2 for two reasons: (i) we did not have the computational resources to fine-tune $\text{BERT}_{\text{BASE}}$ and (ii) we wanted to provide a fair comparison between TinyBERT and the models in the previous category.

An overview on the described models is shown in Table 1.

4 Experimental Setup and Results

We performed our experiments on a laptop with GPU NVIDIA GeForce RTX 3070 (8GB). When citing a model from Section 3.3, we will implicitly refer to the corresponding architecture with the addition of the question-answering head described in Section 3.1.

[1] A more refined sampling strategy will be discussed in Section 4.3

Model		#Params	L	H
$\text{BERT}_{\text{TINY}}$	[12, 15]	4.4M	2	128
$\text{BERT}_{\text{MINI}}$	[12, 15]	11.3M	4	256
TinyBERT	[13, 16]	14.5M	4	312
$\text{BERT}_{\text{SMALL}}$	[12, 15]	29.1M	4	512
$\text{BERT}_{\text{MEDIUM}}$	[12, 15]	41.7M	8	512
$\text{BERT}_{\text{BASE}}$	[11]	110.0M	12	768
$\text{BERT}_{\text{LARGE}}$	[11]	340.0M	24	1024

Table 1: Student models we fine-tuned on SQuAD1.1. $\text{BERT}_{\text{BASE}}$ and $\text{BERT}_{\text{LARGE}}$ are provided for comparison. L is the number of Transformer layers and H is the hidden embedding size.

4.1 Metrics

In line with previous work on SQuAD [4, 11, 13], we evaluate our models with the Exact Match (EM) and F1 metrics, which are defined as follows: given a SQuAD sample (i.e. a question), let g be the ground-truth answer and p the answer predicted by the model; EM and F1 of that sample are then computed as[2]

$$\text{EM}(g,p) = \begin{cases} 1 & \text{if } g = p \\ 0 & \text{otherwise} \end{cases}$$

$$\text{P}(g,p) = \frac{\#\text{ words shared between } g \text{ and } p}{\#\text{ words in } p} \qquad \text{(Precision)}$$

$$\text{R}(g,p) = \frac{\#\text{ words shared between } g \text{ and } p}{\#\text{ words in } g} \qquad \text{(Recall)}$$

$$\text{F1}(g,p) = \frac{2 \cdot \text{P}(g,p) \cdot \text{R}(g,p)}{\text{P}(g,p) + \text{R}(g,p)}$$

whereas EM and F1 of a set of samples are computed as the mean of the EMs and F1s of the samples in the set. The rationale behind these metrics is that, while EM simply measures whether ground-truth answer and prediction are identical, F1 provides a more nuanced measure of similarity between the two, where such similarity is defined in terms of shared words, independently of their position within the answer.

4.2 Comparison by Model Size

As a first step in our experimentation, we compared the results of fine-tuning the models introduced in Section 3.3 with some of the hyperparameters suggested in [11], namely:

- 5 epochs of training with checkpointing based on the F1 score on the validation set.
- Batch size 16.

[2]Punctuation, articles and extra whitespace are removed before computing the metrics, as in the SQuAD2.0 evaluation script.

Model	#Params	Val F1	Epoch (1–5)
$\text{BERT}_{\text{TINY}}$	4.4M	0.392	4
$\text{BERT}_{\text{MINI}}$	11.3M	0.628	3
TinyBERT	14.5M	0.615	3
$\text{BERT}_{\text{SMALL}}$	29.1M	0.687	2
$\text{BERT}_{\text{MEDIUM}}$	41.7M	0.745	2

Table 2: Best validation F1 scores of baseline models during the 5 epochs of training.

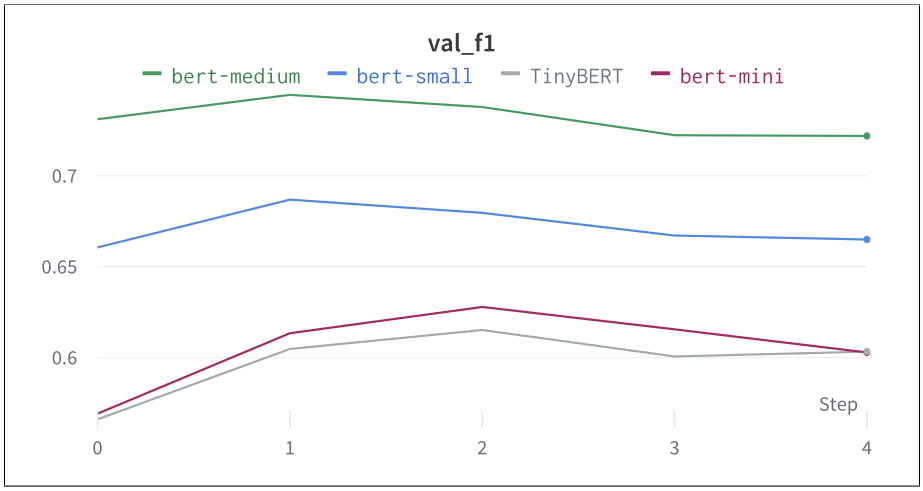


Figure 1: Validation F1 scores of baseline models during the 5 epochs of training. $\text{BERT}_{\text{TINY}}$ is not shown to preserve a sensible scale.

- Adam optimizer [6] with learning rate 3e−5 and its other parameters set to their default values.

We will refer to this models as *baselines*.

Results are shown in Table 2 and Figure 1. Unsurprisingly, F1 scores increase with model size, and bigger models overfit sooner than smaller ones. The only exception is TinyBERT, which performs slightly worse than the smaller $\text{BERT}_{\text{MINI}}$: this is probably due to the fact that, as already discussed in Section 3.3, (i) it is the only model among the ones we tested that was not pre-trained before being distilled and (ii) we did not apply distillation in the fine-tuning step, as suggested by the authors.

4.3 Improving a Model

Once established that increasing model size is beneficial for the task at hand and that there are signs of overfitting for bigger models, we tested some regularization techniques and other potential improvements. Ideally, we would have liked to perform such tests on $\text{BERT}_{\text{MEDIUM}}$, but, because of our limited computational resources, we applied them to $\text{BERT}_{\text{MINI}}$ instead, with the underlying

Model	Modification	Val F1
$\text{BERT}_{\text{MINI}}$-v1	Full-Freeze	0.090 ✗
$\text{BERT}_{\text{MINI}}$-v2	Emb-Freeze	0.639 ✓
$\text{BERT}_{\text{MINI}}$-v3	Weight Decay	0.641 ✓
$\text{BERT}_{\text{MINI}}$-v4	Dropout	0.637 ✗
$\text{BERT}_{\text{MINI}}$-v5	LR Scheduling	0.652 ✓
$\text{BERT}_{\text{MINI}}$-v6	Start > End check	0.671 ✓
$\text{BERT}_{\text{MINI}}$ (baseline)	None	0.628

Table 3: Validation F1 scores of the $\text{BERT}_{\text{MINI}}$ baseline with incremental modifications, as described in Section 4.3. The ✓ and ✗ symbols denote whether or not the corresponding model gave a higher F1 score than the previous successful (i.e. marked with ✓) version.

assumption that the conclusions we were going to draw would have been valid for $\text{BERT}_{\text{MEDIUM}}$ as well. The potential improvements were heuristically ordered by us and applied incrementally: at each step, if the tested technique improved the validation F1, then it was passed on to the next step, where the following technique was applied on top of the previous one(s). The resulting list of tested models is the following, where ✓ and ✗ denote whether or not the corresponding technique was found beneficial—as shown in Table 3—and therefore added to the model:

1. $\text{BERT}_{\text{MINI}}$-v1: $\text{BERT}_{\text{MINI}}$ baseline, but with the whole BERT backbone being frozen; in other words, only the QA head is trained. ✗

2. $\text{BERT}_{\text{MINI}}$-v2: $\text{BERT}_{\text{MINI}}$ baseline, but with the BERT embedding layers being frozen; in other words, only the BERT encoding portion and the QA head are trained. ✓

3. $\text{BERT}_{\text{MINI}}$-v3: $\text{BERT}_{\text{MINI}}$-v2, but with weight decay set to 0.1 and AdamW as optimizer, as it decouples weight decay and learning rate [9]. ✓

4. $\text{BERT}_{\text{MINI}}$-v4: $\text{BERT}_{\text{MINI}}$-v3, but with dropout [2] applied to the QA head with $p = 0.1$. ✗

5. $\text{BERT}_{\text{MINI}}$-v5: $\text{BERT}_{\text{MINI}}$-v3, but with a cyclic triangular learning-rate scheduling [7] with range $[3\text{e}{-5}, 6\text{e}{-5}]$ and a cycle for each epoch. ✓

6. $\text{BERT}_{\text{MINI}}$-v6: $\text{BERT}_{\text{MINI}}$-v5, but with a more refined sampling strategy than the naive one described in Section 3.1. Indeed, by simply computing the start-token position (`stp`) and end-token position (`etp`) as the argmax of the start-token scores and end-token scores, respectively, it might happen that `stp` > `etp`, resulting in an empty span of text predicted as the answer. The solution we implemented is: if `stp` > `etp`, (i) order candidate `stp`s by decreasing score and do the same for candidate `etp`s; (ii) take the first 20 candidate `stp`s and the first 20 candidate `etp`s; (iii) compute the sum of the scores of each (`stp`, `etp`) couple; (iv) take the couple with the highest sum. ✓

Among all the applied techniques, Table 3 shows that the greatest improvements were given by $\text{BERT}_{\text{MINI}}$ vs. $\text{BERT}_{\text{MINI}}$-v2 and $\text{BERT}_{\text{MINI}}$-v5 vs. $\text{BERT}_{\text{MINI}}$-v6. These comparisons are shown in more detail in Figure 2.

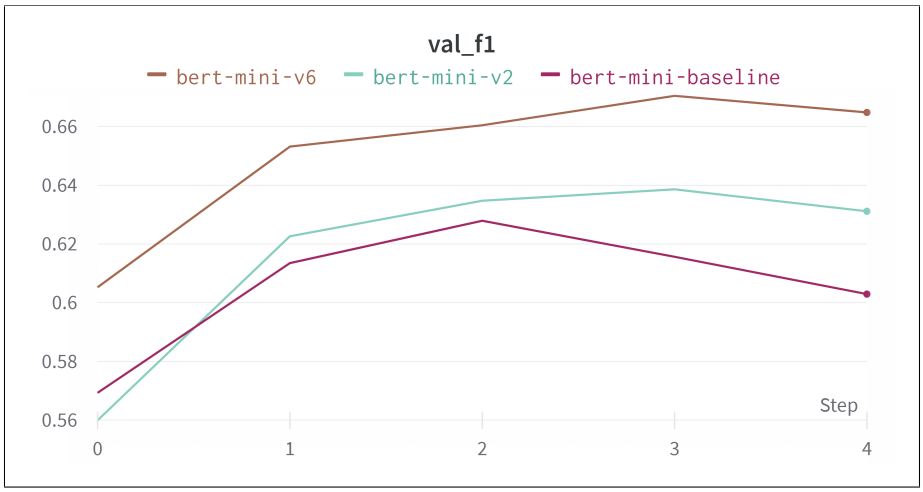


Figure 2: Validation F1 scores of $\text{BERT}_{\text{MINI}}$ (baseline), $\text{BERT}_{\text{MINI}}$-v2 (Emb-Freeze), and $\text{BERT}_{\text{MINI}}$-v6 (Start > End check) during the 5 epochs of training.

4.4 Alternative Task Interpretations

Our task is addressed as a classification problem in which the final output is a couple of scores for each token position indicating whether it can be considered as the start/end token or not, as discussed in Section 3.1. Indeed, the dimension of the output is $(\#\text{tokens}, 2)$. However, despite our classes being intrinsically ordinal, the current models do not take this observation into account. We therefore tried to change the perspective of the problem by injecting the notion of an ordering between token indexes into our models. We explored two different viewpoints:

Regression. Our task can be seen as a regression problem with two outputs: start-token index and end-token index. In order to do so, we add a *regression head* to the BERT backbone, namely (i) a linear layer with one output for each token index, followed by (ii) a linear layer with two outputs (and ReLu activation function). For training purposes, targets were standardized. We observed experimentally that the application of this technique results in a complete degradation of the performance of the classification-based baseline, as shown in Figure 3. This might be due to the sudden dimensionality reduction caused by the last linear layer; however, introducing intermediate linear layers was too memory-heavy for our GPU.

Ordinal Targets. We also tried to assign to each class a bitmap representation in which the number of different bits between classes (i.e. token indices) increases as their distance does [1]. The network structure is the same as in the standard classification architecture, with the addition of a sigmoid activation function after the last linear layer. The loss criterion can be either the binary cross-entropy or the mean squared error between each bit of the bitmap representation of the class. To retrieve the token index as outcome, we count the number of consecutive 1-bits from the beginning of the prediction. Similarly to what we already observed for the regression strategy, this alternative approach was not found to be beneficial, as shown in Figure 3. This might be due to its intrinsic sensitivity to noise and the large number of classes.

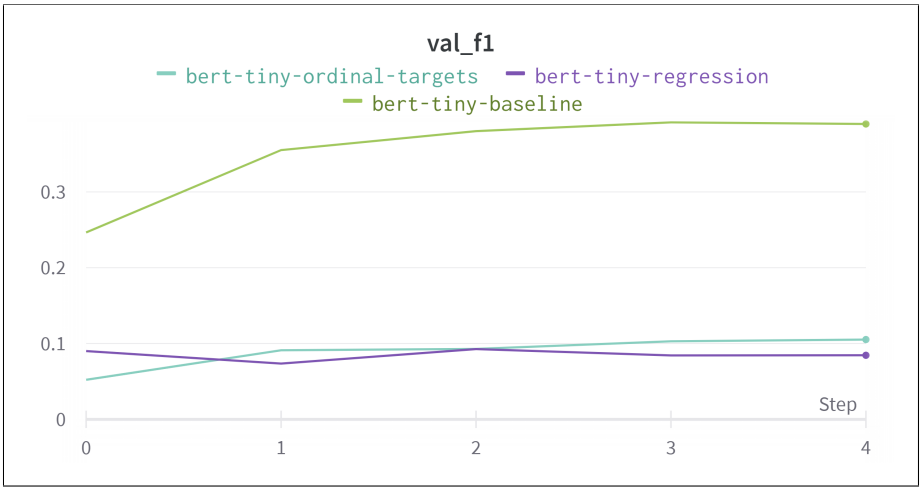


Figure 3: Validation F1 scores when applying regression and ordinal targets compared to $\text{BERT}_{\text{TINY}}$ (classification-based baseline).

Because of the poor experimental results, we decided to stick to the standard formulation of the problem based on classification and discard these alternative viewpoints.

4.5 Final Model

Thanks to the collective experimental evidence of Sections 4.2, 4.3, and 4.4, our final model is a $\text{BERT}_{\text{MEDIUM}}$ with the following features:

- 5 epochs of training (fine-tuning) with checkpointing based on the validation F1 score.
- The embedding portion of the BERT backbone is not updated during training.
- Batch size 16.
- AdamW optimizer with weight decay 0.1.
- Cyclic triangular learning-rate scheduling with range $[3\text{e}{-5}, 6\text{e}{-5}]$ and a cycle for each epoch.
- A sampling strategy that ensures that the predicted end-index follows the predicted start-index.

We call this model $\text{BERT}_{\text{MEDIUM}}$-final. Although Figure 4 shows that it overfits less and reaches higher scores than the corresponding baseline, Table 4 reveals that such improvements are far less significant than what we gained by going from $\text{BERT}_{\text{MINI}}$ to $\text{BERT}_{\text{MINI}}$-v6 in Section 4.3. This seems to suggest, quite understandably, that a bigger and more powerful model such as $\text{BERT}_{\text{MEDIUM}}$ has less margin for improvement.

Model	Val EM	Val F1
$\text{BERT}_{\text{MEDIUM}}$ (baseline)	0.546	0.745
$\text{BERT}_{\text{MEDIUM}}$-final	0.551	0.749
%Diff	0.9%	0.5%

(a) $\text{BERT}_{\text{MEDIUM}}$ vs. $\text{BERT}_{\text{MEDIUM}}$-final.

Model	Val EM	Val F1
$\text{BERT}_{\text{MINI}}$ (baseline)	0.435	0.628
$\text{BERT}_{\text{MINI}}$-v6	0.462	0.671
%Diff	6.0%	6.6%

(b) $\text{BERT}_{\text{MINI}}$ vs. $\text{BERT}_{\text{MINI}}$-v6.

Table 4: Comparison of validation metrics of our final model and the baseline it originated from, where $\%\text{Diff}(x, y) = \frac{|x-y|}{(x+y)/2} \cdot 100$. A comparison between $\text{BERT}_{\text{MINI}}$ and $\text{BERT}_{\text{MINI}}$-v6 is shown for reference.

5 Error Analysis

When inspecting the output of $\text{BERT}_{\text{MEDIUM}}$-final on the validation set, two phenomena stood out among those causing a prediction to get suboptimal EM and F1 scores:

1. It should be noted that the actual SQuAD1.1 development set, which we were not allowed to use, contains multiple ground-truth answers for some questions [17], whereas the training set does not—we verified it ourselves. The need for multiple ground truths at evaluation time becomes apparent when inspecting the examples in Table 5a, where the model outputs a correct answer which however gets penalized just because it extracts slightly more or slightly less text from the context.

2. The data contains ambiguities that cause some predictions to get a lower score even though the model provided sensible answers, as shown in Table 5b.

6 Conclusions and Future Work

We tackled the Natural Language Processing task of Question Answering on the SQuAD1.1 dataset by fine-tuning knowledge-distilled versions of BERT proposed in the recent literature and comparing their results. Our best model—$\text{BERT}_{\text{MEDIUM}}$-final—reaches an EM score of 0.551 and an F1 score of 0.749 on our validation set.

We followed an incremental and rigorous experimental procedure, within the constraints imposed by our computational budget; however, many different choices could have been made and other routes could have been explored. By investing more resources—both in terms of human time and computational power—a more exhaustive search of the hyperparameter space may shed light on those choices that were mainly guided by intuition or general guidelines provided by previous work on the topic.

As many other current research efforts, our work points in the direction of a more extensive democratization of the power of big pre-trained language models, by exploring what can be achieved by compressing their knowledge and transferring it to more compact architectures, in the hope for a future where a wider audience can contribute to significant advances in the Deep Learning field.

References

[1] Jianlin Cheng, Zheng Wang, and Pollastri Gianluca. 2008. A neural network approach to ordinal regression.

[2] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors.

[3] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the Knowledge in a Neural Network.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text.

[5] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation.

[6] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization.

[7] Leslie N. Smith. 2017. Cyclical Learning Rates for Training Neural Networks.

[8] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need.

[9] Ilya Loshchilov and Frank Hutter. 2019. Decoupled Weight Decay Regularization.

[10] Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

[12] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-Read Students Learn Better: On the Importance of Pre-training Compact Models.

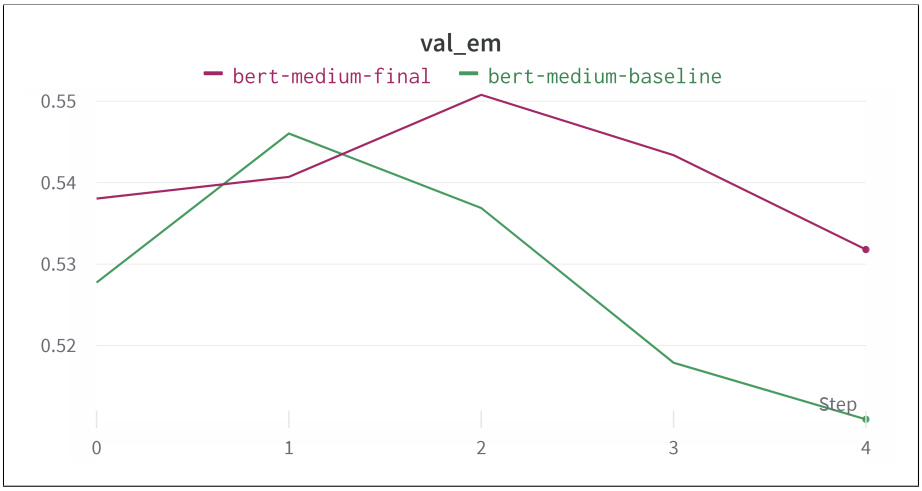
[13] Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for Natural Language Understanding.

[14] Hugging Face implementation of `BertForQuestionAnswering` (Retrieved: 02/2022).

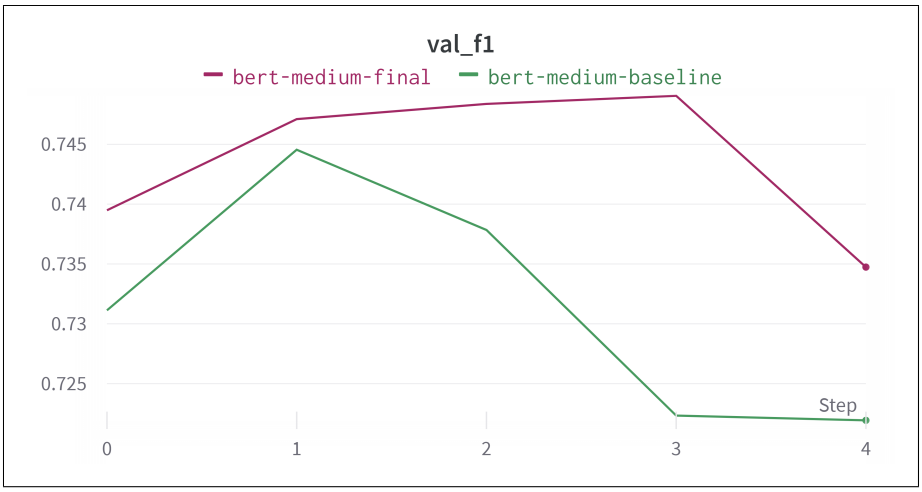
[15] PyTorch pre-trained `bert-tiny`, `bert-mini`, `bert-small`, and `bert-medium` (Retrieved: 02/2022).

[16] PyTorch pre-trained `TinyBERT_General_4L_312D` (Retrieved: 02/2022).

[17] SQuAD1.1 Dev Set web explorer (Retrieved: 02/2022).



(a) Val EM.



(b) Val F1.

Figure 4: Validation EM and F1 scores of $\text{BERT}_{\text{MEDIUM}}$ and $\text{BERT}_{\text{MEDIUM}}$-final during the 5 epochs of training.

Question	Context	Ground truth	Prediction	F1	EM
How long has the domestic dog been selectively bred?	The domestic dog [...] is a domesticated canid which has been selectively bred for millennia for various behaviors [...]	millennia	for millennia	0.667	0
What was undertaken in 2010 to determine where dogs originated from?	[...] extensive genetic studies undertaken during the 2010s indicate that [...]	extensive genetic studies	genetic studies	0.8	0
What 1982 publication listed regular family dogs under wolves?	[...] In 1982, the first edition of Mammal Species of the World listed Canis familiaris under Canis lupus with the comment [...]	Mammal Species of the World	the first edition of Mammal Species of the World	0.727	0
Due to admixture, what species are many Arctic dogs related to?	[...] except several Arctic dog breeds are close to the Taimyr wolf of North Asia due to admixture.	Taimyr wolf of North Asia	Taimyr wolf	0.571	0
What type of nut is poisonous to dogs?	A number of common human foods and household ingestibles are toxic to dogs, including [...] grapes and raisins, macadamia nuts, xylitol, as well as [...]	macadamia	macadamia nuts	0.667	0

(a) The prediction is a contiguous span of text extracted from the context that is semantically equivalent to the ground truth, but not identical to it: the resulting F1 score is therefore < 1, even though the answer is essentially correct.

Question	Context	Ground truth	Prediction	F1	EM
Where does the word dog originate?	[...] The English word dog comes from Middle English dogge, from Old English docga, a "powerful dog breed". [...]	Old English docga	Middle English dogge	0.333	0
What are canine offspring referred as?	[...] A group of offspring is a litter. [...] Offspring are, in general, called pups or puppies, from French poupée, until they are about a year old. [...]	litter	pups or puppies	0	0

(b) The question is ambiguous, meaning that there are multiple disjoint spans of text in the context that provide a reasonable answer. The model is able to find one of them, but it does not coincide with the ground truth.

Table 5: Some (dog-themed) examples of predictions made by $\text{BERT}_{\text{MEDIUM}}$-final on the validation set which do not get maximum scores, although being sensible answers to the corresponding questions.