Who is Ernie? Just ask Bert!

Barthold Albrecht (bholdia)
Yanzhuo Wang (yzw)
Xiaofang Zhu (zhuxf)

Abstract

Question answering, like many other prominent tasks of NLP, has recently experienced a significant progress on established performance measures through the introduction of pre-trained language representation models. At the same time, the concept of multitask learning which tries to integrate different fields of NLP has gained considerable momentum. In this paper we combine both lines of research and show that adding an auxiliary tasks to a BERT-based question answering system can improve the performance for a single given task. We discuss the implications of our findings for this specific task as well as for multitask learning approaches in general.

1 Introduction

Question answering is one of the most relevant NLP tasks for assessing the reading comprehension ability of an algorithm, and the Stanford Question Answering Dataset (SQuAD) has become the standard benchmark for this assessment: Given a paragraph and a question concerning that paragraph, the request is to predict the correct answer to this question, which is a span of text directly from the paragraph. Since its launch in June 2016 (Rajpurkar et al. (2016)[1]) the perfomance on the SQuAD 1.1 leaderboard has seen steady improvements in the F1 score up to the human performance level of 91.221 and this level was finally surpassed in October 2018 by the pretrained language model BERT (Devlin et al. (2018)[2]). However, no variant of BERT has yet managed to do so for SQuAD 2.0 which also contains questions that can not be answered (F1 human level performance of 89.452). This suggests that these models have not yet learned sufficiently well to distinguish between certain question types and that a "one task fits it all"-approach might not be optimal. In the following we show that the addition of a classification task for the answer type (answerable or not) indeed enhances the performance of a BERT-based model with a single downstream task of answer prediction.

2 Related Work

As pointed out in Ruder (2017)[3], charging a model with multiple related tasks boosts performance under certain circumstances. This effect may be attributed to the fact that multitasking forces the internal representations to focus attention on parts of the input that might otherwise be ignored; alternatively, the multiple tasks can be viewed as a regularization mechanism (quite obviously so if the tasks are reflected in a combined loss function) which prevents the model from overfitting and increases its generalization capabilities. Recent multitask learning approaches to NLP have been proposed e.g. by Hashimoto et al. (2017)[4]and McCann et al. (2018)[5]. However, while these papers present models that try to simultaneously solve several NLP tasks sufficiently well at the same time, our approach considers the addition of tasks that are of auxiliary nature only with regard to the main task. Examples where such an introduction of auxiliary tasks is beneficial can be found in Yu et al. (2016)[6] for sentiment classification or Cheng et al. (2015)[7] for name error detection. In the context of question answering the prime candidate for an auxiliary task, as mentioned above, is question type classification (answerable / not answerable).

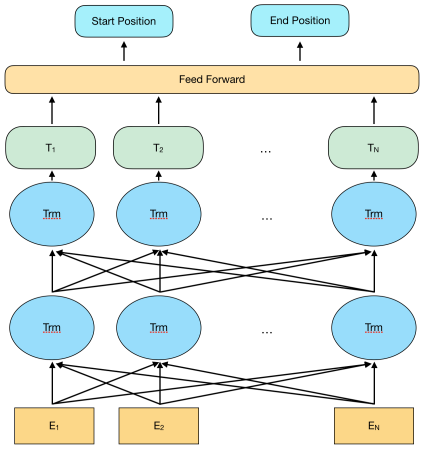

Figure 1: Model architecture for BERT with a question answering downstream task

Previous work in that direction includes e.g. Liu et al. (2018)[8] who add the output of a one-layer binary classifier with equal weight to the loss function and Hu et al. (2018)[9] who train a classifier to verify plausible answers to impossible questions. In both cases the performance on SQuAD 2.0 could be increased (by F1 score points of 0.5 and 1.5 respectively). However, as pointed out by Hu et al. (2018)[9] even with the addition of the classifier to their model more than 20 percent of impossible answers still get misclassified leading them to conclude that the main performance bottleneck lies in no-answer detection instead of answer extraction.

3 Approach

3.1 Baseline

For a considerable time recurrent neural networks have been the most widely used and most succesful model class for question answering and other NLP tasks. However, a structural limitation of these models in an encoder-decoder setting is that the encoder compresses the whole input sequence into one fixed length vector which serves as the context for the decoder while generating its output. This potential problem was overcome with the introduction of attention mechanisms (Bahdanau et al. (2015)[10], Luong et al. (2015)[11]) which allow to adaptively capture the information from all hidden states of the encoder for the calculation of the context vector. In particular the BiDAF-attention model (Seo et al. (2017)[12]), where attention flows bidirectionally (from query to context and from context to query) and the resulting vectors for each sequence step are passed as query-aware representations of context words to a LSTM modelling layer, had been able to achieve state-of-the-art performance at the time of submission (with F1 score of 77,3 (single model) and 81,1 (ensemble) as of 6 Dec 2016 on SQuAD 1.1).

We use the BiDAF model, as provided to us with the starter code, for the baseline.

3.2 BERT

With the introduction of the transformer by Vaswani et al. (2017)[13] the attention mechanism changed its role from being a layer within a RNN-based architecture to replacing the RNN altogether in both the encoder and decoder. While this step was initially motivated by the goal to achieve a similar performance as RNNs with much greater computational efficiency, the transformer recently became the building block for a new way of modelling representations that enables significant gains in performance on a broad range of tasks. Pre-trained models like ELMo (Peters et al., 2017)[14], OpenAI GPT (Radford et al., 2018)[15] and BERT (Devlin et al. 2018)[2] which make the embedding of a word sensitive to the context in which the word is used, can serve as powerful entry points for different down-stream purposes.

As illustrated in Figure 1, BERT provides a relatively straight-forward platform for fine-tuning to a given downstream-task like question answering. In their original paper, Devlin et al. (2018)[2]

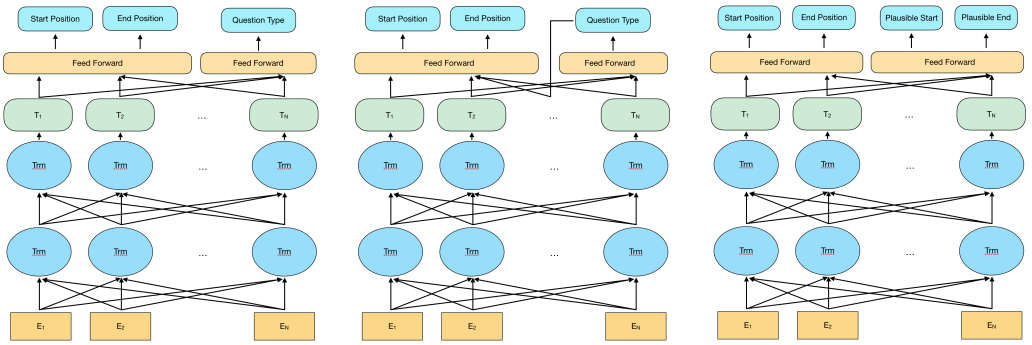

Figure 2: Model architectures from left to right for a) BERT for question answering with question type classifier b) Model a) with additional classification feature c) BERT for question answering with plausible answer prediction

demonstrate this by leveraging on their threefold embedding of the input sequences: each token of a question and its paragraph is represented as the sum of its word embedding, its positional embedding and its segment embedding where the latter indicates whether a token belongs to the question or to the paragraph. This way both the question and the paragraph can be fed into the system as one combined sequence. These representations are then encoded by several subsequent transformer blocks that yield a final representation for every token of the combined input sequence. Fine-tuning this "standard" use of BERT to the task of predicting an answer span within the paragraph is realized by learning two additional vectors: a start vector S and an end vector E with $S, E \in \mathbb{R}^H$ and H denoting the hidden size. These additional learned parameters are then used to compute the probability for any token, represented by it's final hidden state T_i, to be the start or the end word of the answer span. Concretely, these probability distributions over all tokens in the paragraph are obtained by applying the softmax function to the dot product of ST_i and ET_i:

$$P_{i,X} = \frac{e^{XT_i}}{\sum_j e^{XT_j}}, \quad X \in \{S, E\}, \ \forall \ \ T_i \in \text{paragraph} \tag{1}$$

With these relatively small adjustments, BERT can be charged with the task of question answering and achieve excellent results. For the large version of their model trained on an augmented dataset Devlin et al. (2018)[2] report scores on the SQuAD 2.0 of 85.1 and 91.8 for EM and F1 respectively.

Like currently all top ranking entries on the SQuAD leaderboard we incorporate BERT and its pre-trained weights as a central building block for our model and a first step to improve on the baseline. To this aim we use an op-for-op reimplementation in PyTorch from huggingface (https://github.com/huggingface/pytorch-pretrained-BERT).

3.3 Multitask setting

As mentioned above, our multitask approach focuses on question type classification as an auxiliary task. To this aim we augment the training data for the BERT model with labels (0 / 1) for all questions (possible / impossible). As illustrated in Figure 2 a) these labels are then used to train a binary classifier which takes as input the same representations from BERT as the answer span predictor. For this purpose the second and third dimension of the BERT output sequences are concatenated and subsequently fed into a two layer Feed Forward network with ReLU activation. The layers have hidden sizes of (maximum sequence length * hidden size BERT $\times$ 64) and (64 $\times$ 1) respectively. The output of the network is passed through the sigmoid function and the resulting logits are used to compute the binary cross entropy. This individual loss for the classification task is finally summed up with the respective cross entropy losses for start / end prediction respectively. We choose a scaling factor of 7 for the classification loss to ensure that all three losses have about the similar size at the start of training.

We use this BERT multitask model to asses the effect of the auxiliary task as well as to explore alternative designs or variants of it. Among those variants we consider two in our following discussion: BERT with additional feature classification and BERT with additional plausible answer prediction. In the first one (Figure 2 b)) we try to make use of the high accuracy of the question type predictor and feed it's output as an additional feature to the answer span predictor. In the second model (Figure 2 c)) we introduce a second answer span predictor which is trained to return start / end positions of "plausible" answers given in the training data. In this way, we hope the model will have a deeper understanding of the difference between answerable and unanswerable questions.

4 Experiments

4.1 Data

The dataset we used is the SQuAD 2.0 dataset tailored for CS224N. The train, dev and test splits are pre-defined. More specifically, the train dataset has 129914 examples; the dev dataset has 6078 examples; and the test dataset has 5921 examples.

4.2 Evaluation method and experimental details

All models in our project are evaluated via F1 and EM scores that are averaged across the whole dataset. In addition, we also record the intermediate losses for every training step as a way to monitor training progress. This help us better understand whether the model performs properly. Confusion matrix on unanswerable classification is computed for each experiment to further benchmark the mutitask classification performance.

Table 1: Experiment Configuration Parameters

Models	**Details**
BiDAF	Default
BERT_base	batch_size=6, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=384, max_query_len=64
BERT_base + binary_classifier	batch_size=6, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=384, max_query_len=64
	batch_size=10, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=500, max_query_len=96
BERT_large + binary_classifier	batch_size=4, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=384, max_query_len=64
BERT_base + binary_classifier + classification feature	batch_size=6, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=384, max_query_len=64
BERT_base + plausible_answer	batch_size=12, epoch=2, learning_rate=$3e^{-5}$, max_seq_len=384, max_query_len=64

Experiment details are summaries in Table 1. More specifically, we start by adding additional modules to BERT_base, then move to BERT_large with the best performed module and proceed to fine-tuning hyper-parameters such as epochs, batch_size and max_seq_len. BERT_base model has 12-layer, 768-hidden, 12-heads and 110M parameters. BERT_large modelhas 24-layer, 1024-hidden, 16-heads and 340M parameters.

4.3 Results

Table 2: Experiment Results

	EM	F1
BiDAF	57.491	61.097
BERT_base	73.017	76.086
BERT_base + binary_classifier	73.593	76.538
BERT_large + binary_classifier	**74.038**	**77.412**
BERT_base + binary_classifier + classification feature	71.422	74.833
BERT_base + plausible_answer	69.809	72.794

The table above shows the EM and F1 scores in the PCE leaderboard.

For BiDAF model we achieve scores for EM of 57.419 and F1 of 61.097 respectively. These results are below the ones of the "BiDAF-No-Answer (single model)" (59.174 and 62.093 respectively) submission to the SQuAD 2.0 leader board mainly because unlike in the original BiDAF model our starter code doesn't use character-level embeddings in addition to word-level embeddings.
The BERT_base model has significant improvements over the BiDAF on both EM and F1 scores, which is expected as BERT is a much powerful language model.
The BERT_base + binary_classifier achieves a slightly better score than plain BERT_base model. This is expected because the additional classification task acts like a regularization mechanism that prevents the model from over-fitting. Moreover, since the model emphasises even more on punishing incorrect prediction of impossible questions, the model addresses one of the weak point of BERT model, thus has a improved score.
The BERT_large + binary_classifier scores the highest among all the models. This is expected due to that fact that BERT_large has more than twice of the parameters than BERT_basic, implying much greater learning potentials. In fact we expected the score improvement to be even greater. One potential explanation for the limited gain could be that the BERT_large model was not trained long enough to reach its full potential.
The BERT_large + binary_classifier + classification feature variant performed slightly worse than the same model without added feature. We interpret this result this way that adding this feature to the representations from BERT did harm rather than help in predicting the correct answer span.
The result of BERT_base + plausible_answer is much lower than expected.We presume that training an additional task for prediction plausible answers can help the model develop a more robust structure to distinguish between answerable and unanswerable questions. The low score indicates some overlooked conflicts in designing the model.

5 Analysis

As we are trying to focus on improving the learning's ability to predict on the answer type task. Analysis on the performance about how well our classification works is proceeding along with our experiments.

The tables 3 to 6 compare the confusion matrices of predictions on unanswerable questions for the baseline model, the BERT model and the 2 of our multitask models respectively.

Table 3: BiDAF on the Unanswerable

Pred \ Truth	False*	True*
False	2912	1490
True	472	1678

Table 4: BERT_base on the Unanswerable

Pred \ Truth	False	True
False	2541	867
True	369	2301

Table 5: BERT_base + binary_classifier on the Unanswerable

Pred \ Truth	False	True
False	2312	617
True	598	2551

Table 6: BERT_large + binary_classifier on the Unanswerable

Pred \ Truth	False	True
False	2544	763
True	366	2405

Table 7: BERT_base + binary_classifier + classification_feature on the Unanswerable

Pred \ Truth	False	True
False	2364	698
True	546	2470

Table 8: BERT_base + plausible_answer on the Unanswerable

Pred \ Truth	False	True
False	2555	698
True	355	2094

**In the 6 tables above, 'True' represents the unanswerable attribute and 'False' the answerable attribute.*

The bar graph below presents the precision, recall and F1 on the question type prediction. We can see that adding a classification task or configure the Bert model to large can both improve performance on the whole. Particularly for the BERT_large model plus a binary classifier gets the best F1 outcome. However, in the experiment of when we were trying to take plausible answers in consideration, the prediction under-performed.

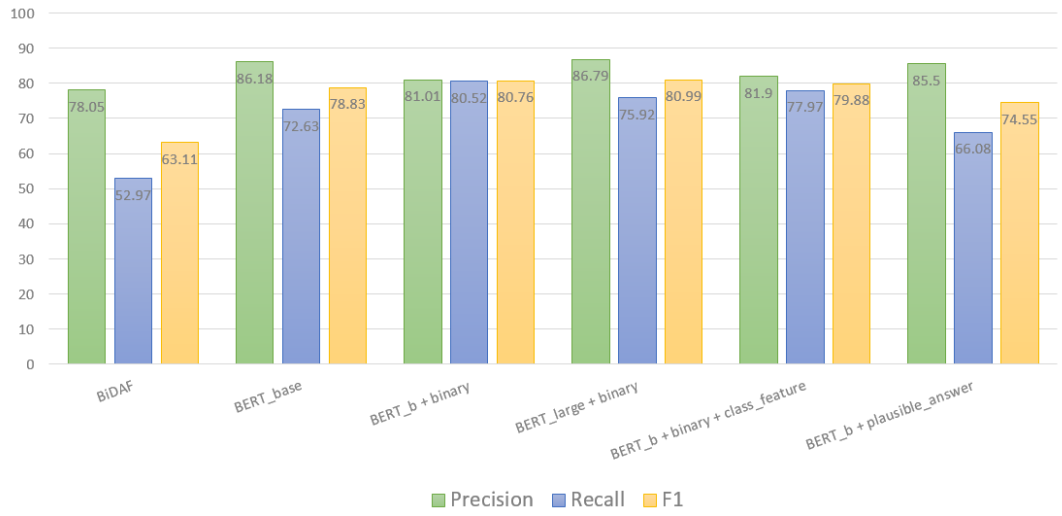

Figure 3: Precision, Recall and F1 on Unanswerable Prediction

While it sounds "plausible" for the model to learn plausible answers as a separate task, the result is short from expectation. One possible explanation is that predicting the plausible answer span somewhat contradicts predicting the correct answer span as the model is ask to predict both incorrect and correct answer at the same time. Additional mechanisms should be added to prevent the model from mixing the two tasks. Another legit explanation is that maybe learning from plausible answers is orthogonal to the main learning goal, so it does not contribute to the overall performance; instead the task just consumes learning power.

Another interesting finding is that increasing maximum sequence length and maximum query length in the model configuration has a negative effect on the score. We increased the max_query_len and max_seq_len for BERT_base + binary_classification model and both EM and F1 score (71.273, 74.386 respectively) decreased. Intuitively, a model that receives only partial question should under-perform compared to a model receiving the entire question. Based on the analysis, the average question length is 60 with the maximum question length being over 180. Our best performing model sets the max_query_len to be 64, which is barely over the average question length. One conjecture is that most of the questions have important information towards the beginning of the sentence, thus even with partial questions, the context is mostly preserved. On the other hand, some questions might be lengthy and wordy, which makes a negative impact on model's understanding of question language models.

6 Conclusion

Our approach on multitask learning shows that the addition of a classification task for the answer type enhances the performance of a BERT-based model for question answering. Starting from there we see potential for further progress in finetuning the classification task as well as making use of it's output. Along the line of our model variants described above we could envisage model architectures that will further leverage the potential of combining BERT with multitask learning.

References

[1] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

[2] 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.

[3] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[4] Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. A joint many-task model: Growing a neural network for multiple nlp tasks. *arXiv preprint arXiv:1611.01587*, 2016.

[5] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. The natural language decathlon: Multitask learning as question answering. *arXiv preprint arXiv:1806.08730*, 2018.

[6] Jianfei Yu and Jing Jiang. Learning sentence embeddings with auxiliary tasks for cross-domain sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 236–246, 2016.

[7] Hao Cheng, Hao Fang, and Mari Ostendorf. Open-domain name error detection using a multi-task rnn. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 737–746, 2015.

[8] Xiaodong Liu, Wei Li, Yuwei Fang, Aerin Kim, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for squad 2.0. *arXiv preprint arXiv:1809.09194*, 2018.

[9] Minghao Hu, Yuxing Peng, Zhen Huang, Nan Yang, Ming Zhou, et al. Read+ verify: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1808.05759*, 2018.

[10] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio. End-to-end attention-based large vocabulary speech recognition. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4945–4949. IEEE, 2016.

[11] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[12] Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

[14] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[15] Alec Radford, Karthik Narasimhan, Time Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. Technical report, Technical report, OpenAI, 2018.