Solving the SQuAD Problem with Computation Efficiency and Interpretability

Yu Junzhao [1] **Yu Wenxi** [1] **Lu Guo** [1] **Han Qing** [1] **Li Wenqian** [1]

Abstract

Stanford Question Answering Dataset (SQuAD) is a reading comprehension dataset being applied to build prediction models that can accurately predict answers to different types of questions. Our research experiments with different approaches for cutting down the model runtime, and focuses on the trade-off between computation efficiency and prediction accuracy. In addition, model interpretability of deep learning models are also explored.

1. Introduction

Question answering (QA), is a challenging and interesting domain in natural language processing (NLP), with extensive application value in reading comprehension, AI chatbot, etc. It generally refers to machine completing the task of answering a set of questions given a certain text, fueled by the creation of many large-scale datasets (Hewlett et al., 2016; Gaikwad et al., 2015; Trischler et al., 2016). Among these influential reading comprehension datasets, Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016), released by Rajpurkar et al., is one of the most famous ones, which consists of 100K question-answer pairs each with a given context paragraph and it soon becomes a standard test for the reading comprehension task with public leaderboard available. In 2018, the team further released SQuAD 2.0, which combines existing SQuAD data with over 50,000 unanswerable questions. SQuAD 2.0 posts a much harder requirement on model development, which means that to do well on the dataset, systems must not only answer questions when possible, but also determine when no answer is supported by the paragraph and abstain from answering.

Recent years, a number of studies have been spurred to solve the QA problem based on the SQuAD2.0 dataset (Rajpurkar et al., 2018). In general, there are two metrics to measure the performance of a model on the SQuAD task, the one is F1 score, which is the harmonic mean of precision and recall on the prediction result; the other is Exact Match (EM), which is a binary measure (true/false) of whether the system output exactly matches the ground truth answer. There are already various of models achieving high EM score and F1 score on the SQuAD leaderboard (lea, 2020) and the best performance up to date has achieved an EM score of more than 90%.

Besides the research of prediction accuracy, there are another two important aspects deserved to focus on for SQuAD task. The one is about computation efficiency since training NLP model on SQuAD dataset will cost lots of time. For example, BERT, a classical model handling the SQuAD1.0 task, takes as long as 17 hours to train 3 epochs on single GPU with 8GB memory with EM score being 69.84 (Zhu, 2019). We can infer that for a more complex task SQuAD2.0, it would take much more time to train. Therefore, it drives us to think about how can we reduce the training time without sacrificing too much EM score, in other words, improve the computation efficiency. Another aspect we need to consider is model interpretability. Deep learning algorithms often suffer from low interpretability. For this NLP problem, opening the black box can not only helps us understand how the deep learning models work and compare different models, but also understand the token importance in input text and do the error analysis.

Therefore, in this study, based on the SQuAD2.0 dataset, we want to explore the trade-off between computation efficiency and accuracy, and then increase the interpretability of solutions. For the former one, we will start from two classical baseline model on SQuAD problem, BERT (Devlin et al., 2018) and BIDAF (Seo et al., 2016), to calculate and compare the training time as well as the EM score, from the perspective of different dataset size, different model size and model ensemble, to analyze the which model can handle the problem best under different situation. For the latter one, we will analyze from different level of to interpret the model, to understand how does the model work from the perspective of global level, layer level and token level. Overall, the main goal of our study is to not only obtain an efficient deep learning model for SQuAD task from different specific

[*] Equal contribution [1] National University of Singapore. Correspondence to: Yu Junzhao <junzhao.yu@u.nus.edu>, Yu Wenxi <wenxi.yu@u.nus.edu>, Lu Guo <luguo@u.nus.edu>, Han Qing <qing.h@u.nus.edu>, Li Wenqian <wenqian@u.nus.edu>.

Codes: https://github.com/zack66/Deep-Learning-Solutions-for-Question-Answering-with-Computation-Efficiency-and-Interpretability situation, but also explore the model by understanding its operation mechanism and making it more interpretable, and contributes to existing research.

The rest of the paper is organized as following: Section 2 introduces the dataset and our exploration; Section 3 introduces the our experiment, including related methodology and results; In Section 4 we explore model interpretability from different level. Final in Section 5 we discuss our limitation and future diretion.

2. Data Pre-processing and Exploration

2.1. Data Source

The dataset we worked with is the open-sourced Stanford Question Answering Dataset 2.0 which is complied by Rajpurkar et al. via the following four steps:

(1) **Wikipedia passage curation.** 536 passages were drawn from top 10k English Wikipedia passages based on Wikipedia's internal PageRanks, in a random yet balanced manner, covering a spectrum of topics, and further broken down into 23,215 paragraphs. The passages were randomly split into training, development, and test sets with 80:10:10 ratio.

(2) **Answerable questions and answers crowdsourcing.** For each paragraph, 5 questions were generated and answered by crowdworkers via Daemo platform (Gaikwad et al., 2015). Crowdworkders were reminded to rephrase questions instead of extracting words from articles.

(3) **Additional answers crowdsourcing.** To minimize inaccurate answers, 2 additional answers were sourced for each question asked from other crowdworkers for development and test datasets. 2.6% of questions were marked as unanswerable by at least 1 additional crowdworker.

(4) **Unanswerable question crowdsourcing.** 50,000 unanswerable questions on the same articles were sourced, to help train the model to robustly identify whether questions given are answerable. Unanswerable question accounts for roughly 1/3 of the training set, and 1/2 of the development or test set.

2.2. Data Description

We will use the officially released SQuAD 2.0 dataset with modest adjustment. The training set consists of 442 articles, 19,035 contexts and 130,319 questions with average 7 Q&A pairs per context. The development set consists of 16 articles, 646 contexts and 11,873 questions with average 10 Questions per context and 3 answers per question. The detailed information of each variable is represented in the Tabel 1. We will use the development set to evaluate our model.

2.3. Data Preparation

2.3.1. Exploratory data analysis

Before developing and fitting our model , we conducted Exploratory data analysis (EDA) to observe and gain a better understanding of the dataset. There are two labels for the questions in dataset: unanswerable and answerable. By calculation we find that the proportion of unanswerable in training set is 33.4%, which indicates that the training set is imbalanced. answerable question is greater in number than unanswerable question First we explored the unanswerable questions. However, the label distribution in development dataset is balance. In other words, each class accounts for almost 50%.

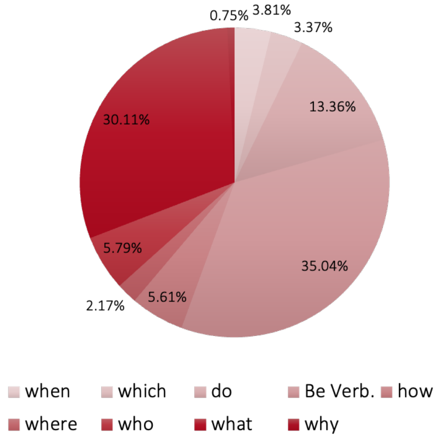


Figure 1. The distribution of 9 categorizations about questions

In addition, we broken down all the questions using certain keywords like where, when, what , etc., into 9 different question types and plot the distribution figure as Figure 1 shows. From the plot we can find that the "Be Verb." type of questions is the largest category, and it consists of 35.04% of all the questions in training set. To be noted that, when categorizing "Be Verb." type of questions, we simply filter by keywords like "is", "are", "was", "were", etc. It is possible that these questions are "if" or "whom" type of questions instead.

2.3.2. Data Pre-processing

Before putting data into our model, we do a series of text mining and feature building.

(1) Text Cleaning The first step is cleaning the data. We first standardize the punctuation and then tokenize the questions and corresponding answers. We generated character list for all tokens and answer span staring and ending index in the context as target y_1 and y_2. In this step, we get a set of tokenized training and dev dataset.

Table 1. Data Description

Variable	Definition	Type
ID	Unique identifier for each question	string
TITLE	Title of the passage	string
CONTEXT	Potentially relevant paragraph of answers	string
ANSWERS/PLAUSIBLE_ANSWERS	All the right answers if the question is answerable, otherwise plausible answers	list
TEXT	Right answer for the question	string
ANSWER_START	Starting position of the answer	list
IS_POSSBILE	Logical variable indicates answerableness	boolen

(2) Text Embedding We used pre-trained vectors GLoVe (Pennington et al., 2014) with 300-dimentional embeddings trained on the CommonCrawl 840B corpus to obtain vector representations for words. Next, toreduce the size of the embedding matrix, we do the trimming step and keep only the words and characters that appear in the training set. Then we generate mapping dictionaries between character and words and indices in the embedding matrices. From these steps, we finally get word/character embedding matrices.

(3) Feature Building We also build additional features for model. Our outputs are answer span starting and end index in the context as target $y1$,$y2$, and we generate word and character index for context and questions as x features. From these steps, we get our target variable as well as features and ready for model building.

3. Model and Computation Efficiency Exploration

In this section, we would explore various computation-efficient approaches to SQuAD. First we would give an overview of traditional approaches we select in the project, and then 3 different experiments would be applied to make a trade-off between model performance and computation efficiency. Due to computation constraint, our exploration of computation-efficient approaches are not run till divergence and all of our experiments are run on 4x RTX 3090 Server.

3.1. Baseline Model

In the project, we choose two classical models on SQuAD task as our baseline: BERT and BiDAF. Their EM and F1 performance serve as benchmarks for our further studies.

3.1.1. BiDAF

Bi-Directional Attention Flow (BiDAF) (Seo et al., 2016) is a hierarchical multi-stage architecture for modelling the representations of the context paragraph at different levels of granularity and is specifically tailored to question answering. It uses bi-directional attention flow to obtain a query-aware context representation.

When the model was introduced initially, it outperformed all previous approaches on the SQuADv1.0 test set leader board at the time of submission with EM score of 69.9 and an F1 score of 78.1.

However, it is unable to capture the multiple meanings of words in different contexts. With researchers constantly testing new architectures, the discipline of QA has experienced breakthroughs with the release of Pretrained contextual Embedding (PCE) based approaches. It‘s performance on SQuAD could no longer satisfactory if we compared it with the current leaderboard with both EM and F1 over 90%. It inspires us to consider more computation efficient methods with higher accuracy.

3.1.2. BERT

Bidirectional Encoder Representations from Transformers (BERT) (Devlin et al., 2018) model is designed to pre-train deep bidirectional representation from unlabeled text by jointly conditioning on both left and right context in all layers. It‘s contextual embedding approach could overcome the weakness of a one-size-fits-all word embedding in models. On SQuAD v1.1, BERT surpassed the previous score and got better performance than OpenAI GPT and ELMo. It is considered as one of the best deep learning models for natural language processing .

However, it is compute-intensive at inference time and hard to bring into production.

In the following part, we would work with a new dataset SQuAD 2.0 and tried 3 different experiments based on BERT model to explore the computation efficiency.

3.2. Experiment 1 – Downsized Training Data

3.2.1. Methodology

In the experiment 1, we consider to change the dataset to do the training. Here we make a hypothesis that, if using the smaller dataset will have slightly worse performance but much less time consuming to train, we would consider this method make computation efficiency. In this experiment, we use BERT$_{base}$ model to do the training and the input datasets are resampled as 5%, 10% and 100% (unchanged), respectively.

When sampling the training data randomly, the number of contexts per article is mostly maintained at the same distribution as original dataset. We developed 3 different BERT models named BERT$_{05}$,BERT$_{10}$ and BERT$_{100}$, which are corresponding to 5%, 10% and 100% of training set.

3.2.2. Evaluation

For each model, we trained only 2 epochs due to computation limitation and the result is displayed in Table 2. The F1 scores of BERT$_{05}$ and BERT$_{10}$ are 53.43 and 53.83, respectively. It seems that the BERT$_{05}$ is more computation efficient because BERT$_{05}$ only requires 20 minutes whereas BERT$_{10}$ requires 1 hour. However , the F1 score of BERT$_{100}$ with 4 hours to train is 76.62, which has much better performance than models with small dataset. The reason for the similar performance between BERT$_{05}$ and BERT$_{10}$ is that both models suffer from underfitting due to the very limited training data. Therefore, the Exact Matching score and F1 score will have no significant increase with 10 $\%$ training set. Here we drew a training curve plot of the model with different training set and discovered the performance would maintain around 53 points at the beginning, and then the score would decrease slightly with more training set, and then performance has a great improvement with more training data. Therefore, we would choose the whole dataset and then explore the ensemble architecture in the following part.

3.3. Experiment 2 – Different Model Sizes

3.3.1. Methodology

We tried out variations of BERT to study the impact of model size on prediction accuracy in the context of computation efficiency. Among the many BERT model architectures available, we focused on three of them: BERT$_{mini}$, BERT$_{base}$ and BERT$_{large}$. These three model architectures are different in the number of encoder layers, the number of hidden layers, the number of attention heads, and the number of parameters. The mini model has 4 layers with 256 hidden layers. The base model has 12 encoder layers stacked on top of each other, while the large model has 24 layers of encoders. In addition, base has 768 hidden layers and large has 1024 hidden layers. What‘s more , the base model has a total of 12 attention heads and 110 million parameters whereas the large model has 16 attention heads with 340 million parameters. The increase in parameters from base to large played a role in considerably improving the model's natural language understanding but it is also considerably more expensive to train.

3.3.2. Evaluation

In our experiment, we trained 2 epochs based on BERT$_{base}$ model, and due to computation constraints we just trained 1 epoch based on BERT$_{large}$ model. The large model needs over 12 hours per epoch whereas the base model just need 2 hours per epoch. The result can be found in Table 2. From the result we can find that the F1 score of BERT$_{mini}$ model, BERT$_{base}$ model and that of large model are 60.28, 76.62 and 83.13, respectively. Combined with our literature reviews, BERT$_{large}$ model should give a better result after $2^n d$ epoch thus we could believe EM and F1 scores larger than the reported performance. Even though the base model has relatively worse performance than the large one, the base model has increased around 30 points than the baseline BiDAF model, and there are only 6.5 points below the large performance, which is 3x larger and requires a lot of horsepower. Therefore, considering the both model performance and computation efficiency, the BERT$_{base}$ model seems to better match to our expectations.

3.4. Experiment 3 – BERT+BiDAF Ensemble

3.4.1. Methodology

Given that training BERT takes considerable amount of time and resources, an ensemble method of "BERT Embeddings + BiDAF Architecture" is experimented with 2 variations. Instead of directly taking in context and question from the beginning, the BiDAF models take in BERT embeddings or certain hidden states in subsequent layers.

a. Before Highway Network We took the combined question and context , and converted them to BERT embeddings before passing them on to subsequent layers of the BiDAF model. Input embeddings for a given batch are generated during the train step, before the forward pass of the BiDAF model . Consequently, we investigated how readily useful the BERT embeddings are for generic models.

b. Before Bi-LSTM Network It is the first model replacing BiDAF 's embedding layers and attention flow layer with BERT. For example, it used BiDAF 's modeling layer and output layer on top of BERT.

Modeling Layer. The input to the modeling layer is the final hidden states of BERT. They are passed to two layers of bidirectional LSTM with hidden size h. Thus we obtain output $\boldsymbol{M} \in \mathbb{R}^{N\times 2h}$. This layer should further capture the interaction of the input query-passage sequence.

Output Layer. The output will produce probability distributions of start and end positions for the answer span. To get the distribution of start position, we compute:

$$\boldsymbol{P}_{start} = softmax(\boldsymbol{W}_{start}\ [\boldsymbol{G}; \boldsymbol{M}])$$

To get the distribution of end position, the output of modeling layer, $\boldsymbol{M}$, is first fed into a one-layer bidirectional LSTM and obtain $\boldsymbol{M}^{'} \in \mathbb{R}^{N \times 2h}$, then we compute:

$$\boldsymbol{P}_{end} = softmax(\boldsymbol{W}_{end}\ [\boldsymbol{G}; \boldsymbol{M}^{'}])$$

Here $\boldsymbol{W}_{start}$, $\boldsymbol{W}_{end} \in \mathbb{R}^{1 \times (H+2h)}$ are linear layers with learnable parameters.

3.4.2. Evaluation

Both ensemble models gain extra EM and F1 from baseline. With the limit of 2 epochs, “Variation 2 Before Bi-LSTM Network” achieves 2% higher EM and 2% higher F1 compared to “Variation 1 Before Highway Network”. However, both of them are much lower than the F1 score of BERT$_{base}$ model.

Table 2. Model Comparison

Models	Epochs	EM(%)	F1(%)
Dataset Size			
Baseline (BERT$_{100}$)	2	73.31	76.62
BERT$_{05}$	2	53.43	53.21
BERT$_{10}$	2	53.83	53.68
Model Size			
BERT$_{mini}$	2	60.28	63.16
BERT$_{bese}$	2	73.31	76.62
BERT$_{large}$	1	80.84	83.13
Embedding Ensemble			
Baseline (BiDAF Emperical)	2	49.14	47.84
BERT+BiDAF$_{100}$ (Before Highway)	2	51.55	51.54
BERT+BiDAF$_{100}$ (Before Bi-LSTM)	2	53.54	53.71

3.5. Results and Discussion

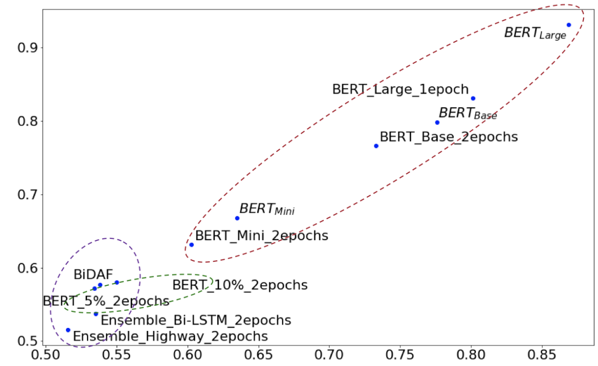


Figure 2. EM and F1 of all models in a two-dimensional space. Overall , BERT fits the SQuAD context better than BiDAF. Given that BiDAF suffers from significant underfitting in early epochs, it is not a good model choice when computation efficiency is constrained.

We evaluated our proposed models on the dev set and the results are summarized in Table 2, and also plotted EM and F1 of all models in a two-dimensional space and observed three interesting clusters of model performance in Figure 2.

The top-performing cluster consists of three baseline models with their 1 or 2 epoch versions, with clear separations across BERT$_{large}$, BERT$_{base}$, and BERT$_{mini}$. It reveals that model size matters most. Given that there are no significant gaps between the fully fine-tuned models and the 2-epoch versions, we learn that 2 epochs can already achieve relatively satisfactory results so there is no need to run BERT till divergence.

The green cluster consists of BERT$_{base}$ model with 5% / 10% training set. The cluster locates far away from the BERT$_{base}$ baseline, so we conclude that training data size matters a lot. When resource is limited, cutting down epoch is a better compromise than cutting down training data size.

The purple cluster is the BiDAF-related cluster. It consists of one BiDAF baseline and two ”BERT+BiDAF ensemble” models. Their performance are generally bad.

In total, we have 9 models that exceed the baseline BiDAF model in terms of F1 or EM. Among all these models, BERT$_{large}$ model with the whole dataset has the best F1 score 83.13 and EM 80.14 but low computation efficiency. BERT$_{base}$ model with the whole dataset has the second best F1 score 76.62 and EM 73.31 as well as considering both performance and computation efficiency. We could also discover that encoder-decoder architectures added on top of BERT is not generally effective.

4. Interpretability Exploration

NLP challenges such as Question Answering usually require large complex neural network models, and this poses challenges to model transparency and interpretability, setting higher bar for model troubleshooting and business sense sanity check. To enhance model interpretability, 3 approaches from different angles were explored: performance across-question type, layer-wise transformation, and token attribution.

4.1. Performance Comparison Across Question Types

To conduct error analysis and model performance comparison, we further computed within-question-type EM & F1, and compared between-question-type EM & F1 for BERT large, BERT base and BERT base model with only 10% data in Figure 3 and Figure 4.

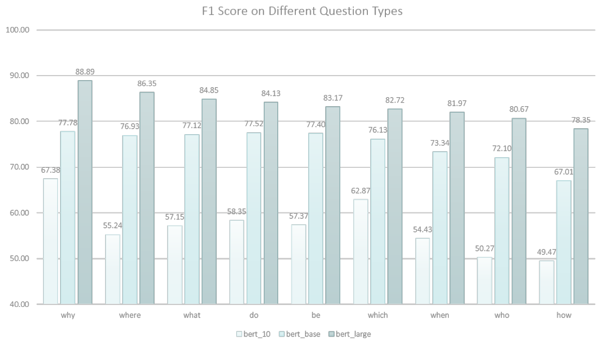


Figure 3. F1 Score on Different Question Type

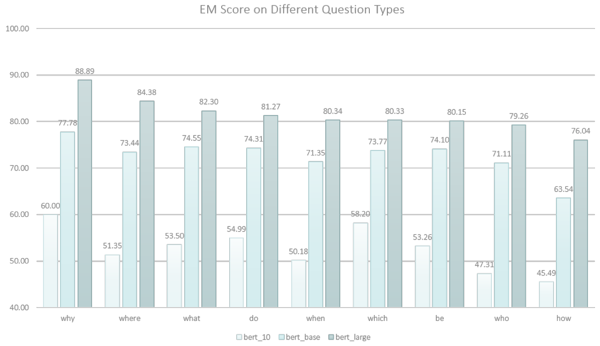


Figure 4. EM Score on Different Question Type

The comparison revealed that all models were able to answer ”why” questions with higher confidence than other question types. These 2 graphs can serve as references for end-users of our SQuAD solutions to make model choices when the distribution of question types is informed in a prediction task.

4.2. Layer-wise Transformation Representation Analysis

For transformer-based models, one common method to read into the layers is to inspect and draw analysis from attention weights. However, some researchers disputed over this way of interpretation as the correlation between attention values and model output may not be well justified(Jain & Wallace, 2019).

Thereafter, we learned from van Aken et al. to study the transformation done by transformer-based models via a qualitative layer-wise representation analysis (van Aken et al., 2019), which can be summarized in below key steps:

(1) **Hidden-State Pulling** To look into how one specific question-context pair is being transformed by BERT transformer through different layers and ultimately used to derive the answer token, a specific question-context pair was picked for the visualization and fed into the pre-trained BERT model, from which the hidden layers outputs were collected.

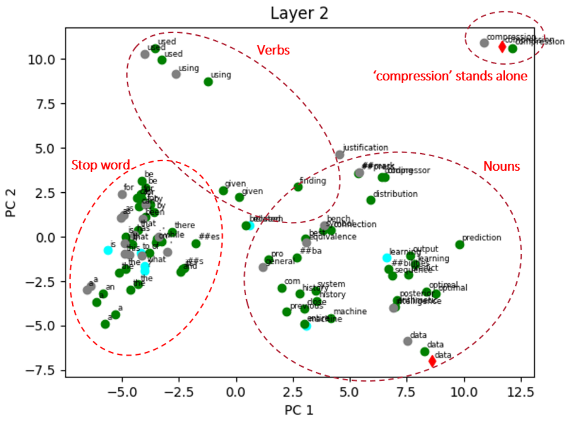


Figure 5. Phase 1: Semantic Clustering. Within first 2-3 transformation layers, word matching and semantic clustering was observed, where stop words, verbs, nouns were segregated. ”compression” and ”data” were not yet associated as a phrase.

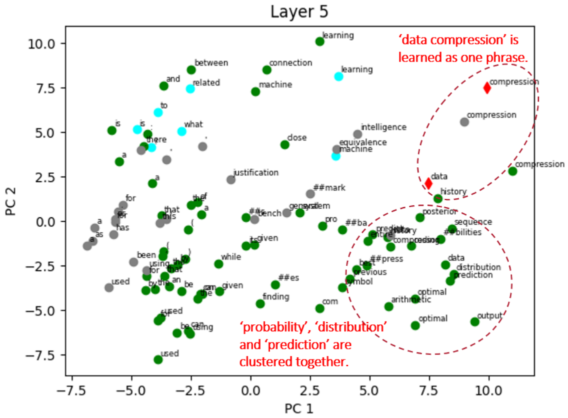


Figure 6. Phase 2: Task-Specific Entity Matching. More task-specific topical information was learned and similar entities were connected such as ”distribution” and ”probability”, and ”data compression” was learned as a phrase.

(2) **Dimensionality Reduction** The hidden layer outputs from BERT were in high-dimensional feature vector space, and PCA and t-SNE was done to re-fit each layer’s output into 2-dimensional space for ease of token visualization.

(3) **Clustering** To validate whether the newly generated 2D vector space could represent the high-dimensional feature distribution, K-means clustering was done for tokens in each layer with k being set as the count of observed clusters within each layer.

(4) **Transformation Interpretation** After tokens in each layer were represented in 2D plots, plots were compared to

Table 3. Sample for Layer-Wise Transformation Visualization

Question	Context	Answer
”What is Machine learning related to?””	”There is a close connection between machine learning and data compression: a system that predicts the posterior probabilities of a sequence given its entire history can be used for optimal data compression(by using arithmetic coding on the output distribution) while an optimal compressor can be used for prediction (by finding the symbol that compresses best, given the previous history). This equivalence has been used as a justification for using data compression as a benchmark for ’general intelligence.’”	Ground Truth: ”data compression” Prediction: ”data compression”
”What is one example of an instance that the quantitative answer to the traveling salesman problem fails to answer?”	”To further highlight the difference between a problem and an instance, consider the following instance of the decision version of the traveling salesman problem: Is there a route of at most 2000 kilometres passing through all of Germany’s 15 largest cities? The quantitative answer to this particular problem instance is of little use for solving other instances of the problem, such as asking for a round trip through all sites in Milan whose total length is at most 10km. For this reason, complexity theory addresses computational problems and not particular problem instances.”	Ground Truth: ”asking for a round trip through all sites” Prediction: ”a round trip through all sites in Milan whose total length is at most 10 km”

detect the pattern and impact of transformation achieved by BERT, which were then summarized into phases. Upon reviewing plotted hidden layers for multiple question-context sample pairs, 4 commonly observed phases of transformation were identified: token/topical clustering, entity matching, supporting fact matching, and answer extraction. These phases are illustrated in details in Figure 5 to Figure 8, using the ”data compression” example in Table 3.

It was also found that in some cases, such as the ”travelling postman” example, the phase 2 or phase 3 stages were not as clear cut and obvious as other examples, because tokens from different clusters were observed to remain nearby in the 2D vector space.

4.3. Token Attribution via Integrated Gradients

To further study how specific tokens contribute to the final answer predictions, we looked into a method called integrated gradient (Sundararajan et al., 2017). Integrated Gradient computes the gradient of the model’s prediction output to its input features and requires no modification to the original deep neural network. By adopting this mechanism, we successfully discovered word importance for right samples and analysed reasons for wrong predictions.

To explore the relative token importance, we visualized Attribution Score calculated by Integrated Gradients for each token. A rightly predicted sample as well as a false one are shown in Figure correspondingly. Obviously, the starting and ending tokens are correctly found in the first example since both of them are highlighted with deep green, representing the highest attribution scores. As for the second instance, the word ”asking” is incorrectly predicted as the staring word, and a potential reason would be that it is considered relevant to the question in BERT’s contextual embeddings. The predicted answer span fails to stride over ”whose”, the signal word of attributive clause. These observations reveal the potential pitfalls of the BERT model.

5. Conclusion and Future Work

5.1. Conclusion

In this report, we explored the trade-off between prediction accuracy and computation efficiency focusing on SQuADv2.0 dataset, and also enhanced the model interpretability from three different levels to make the SQuAD problem-solving process more transparent and understandable.

For the modelling part, we worked with BERT and BiDAF and created variations of them from the perspectives of training data size, model size, and ensemble, with the goal of balancing computation efficiency and prediction accuracy. Overall, we found that BERT is generally more suitable for the SQuAD problem compared to BiDAF, and BiDAF suffers from significant underfitting in early epochs. By comparing the model performance of our experimental models and their corresponding baseline, we also ranked the 3 variable factors. Training data size is the most important one. Then model size, followed by ensemble. By comparison, the BERT_{Base} model can be considered as the best

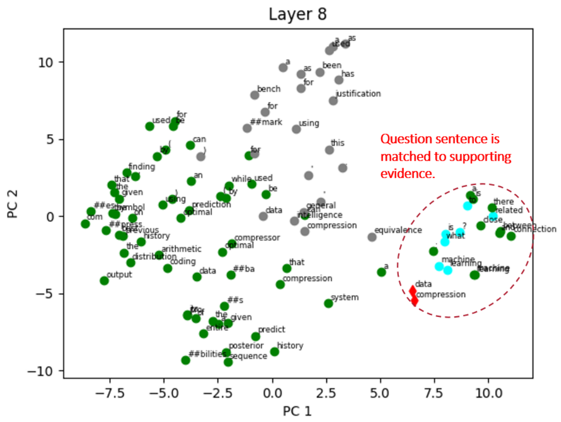


Figure 7. Phase 3: Question and Supporting Fact Matching. Tokens in question sentence were put together to identify the relevant supporting fact tokens in nearby vector space, which were composited to give the full impression of supporting fact sentence.

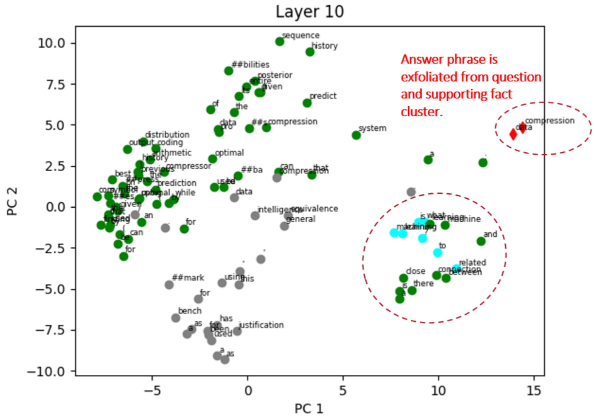


Figure 8. Phase 4: Answer Extraction. Most promising answer token candidate was identified and exfoliated from supporting fact sentence. Most other less critical token clusters were dissolved together and distanced from correct answer.

model if users want to save training time and achieve good accuracy simultaneously.

After modelling, we also applied interpretability tools at global, layer, and token level to open the black-box of deep learning models. This part mainly helps us to understand different models' strengths and shortcomings across different question types, couple the functionalities of BERT transformation layers and certain key classification tasks in Question Answering, and conduct error analysis and word importance analysis based on token attribution to understand the "how" and "why" of BERT making correct and wrong predictions.

5.2. Future Work

In the study, we also acknowledged the significance of some exploration directions, but was not able to implement due to computation power and time constraints. Here, we list them down as future work.

5.2.1. Computation Efficiency

(1) Data preprocessing When we construct small sample training dataset from original dataset for experiment, we can further take class imbalance into consideration since the number of answerable questions is much larger than unanswerable questions'.

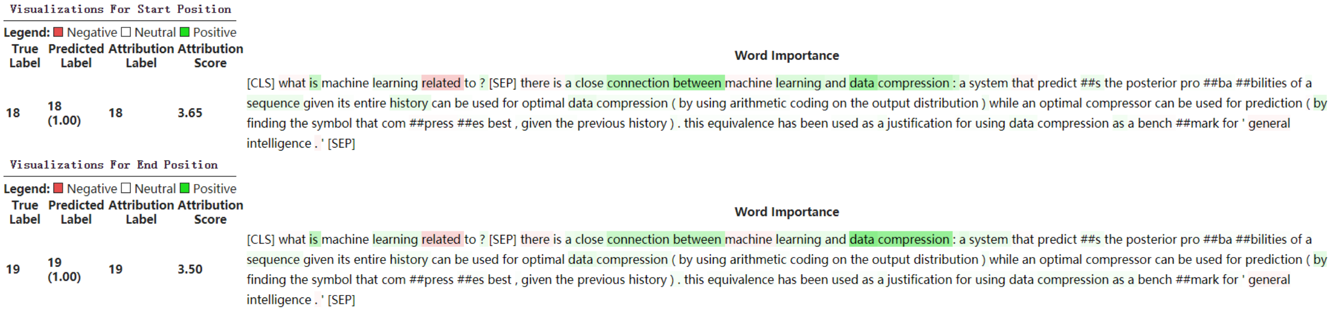
(2) Hyper-parameter Tuning Deep learning models often have complex architecture so adjusting the position of layers or deleting unnecessary layers may increase the speed of computation and save time at fine-tune stage. For example, we can add dropout layer or add regulation to prevent model overfitting and increase the speed of computation.

(3) Model architecture Although in this research we mainly focus on increasing computation efficiency of fine-tune stage, there are also some strategies to save time on pre-train stage and deserve to have a try. There are already some state-of-art new model architecture can be used to increace the speed of pre-train or even on fine-tune. For instance, DistilBERT (Sanh et al., 2019) is a smaller general-purpose language representation model, which can then be fine-tuned with good performances on a wide range of tasks like its larger counterparts. What's more, ALBERT (Lan et al., 2019), using two parameter-reduction techniques, can also increase the training speed of BERT. Another example is EarlyBERT (Chen et al., 2020), which identifies structured winning tickets in the early stage of BERT training and increase the computation speed on both pre-train and fine-tune stage. In addition, we can also consider to use smarter answer spans, using the highest probability combination of starting and end indices instead of simply taking the argmaxes of both, to improve F1 score.

(4) Different embedding ensemble methods In our research, we mainly focus on ensembling models on embedding level. We can also try other embedding experiments to combine different layers of BERT hidden states. Moreover, it's also a good idea to construct simple classifiers to extract features from embedding. Also, we can try to ensemble model by integrating identical model predictions from varied training stages

5.2.2. Interpretability

There are also some more directions for model interpretability. For example, we can compare model performance across different answer lengths. What's more, We can also analyze the query-to-context attention flow of BiDAF through visu-



Visualizations For Start Position

Legend: 🟥 Negative □ Neutral 🟩 Positive

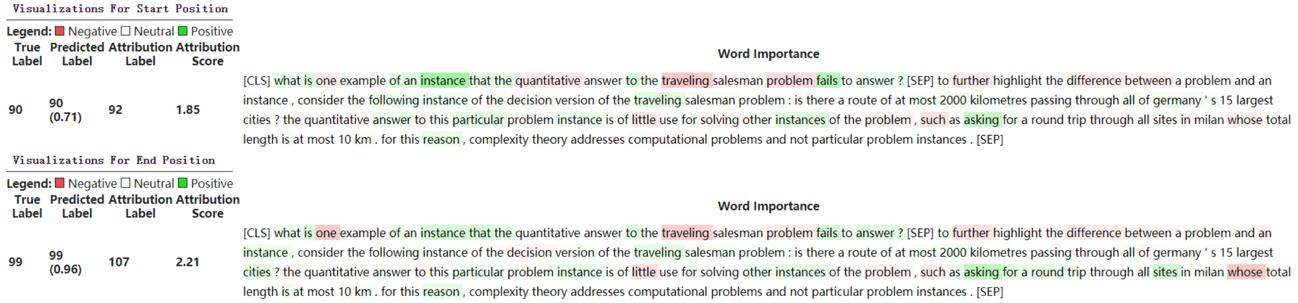
True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
18	18 (1.00)	18	3.65	[CLS] what is machine learning related to ? [SEP] there is a close connection between machine learning and data compression : a system that predict ##s the posterior pro ##ba ##bilities of a sequence given its entire history can be used for optimal data compression (by using arithmetic coding on the output distribution) while an optimal compressor can be used for prediction (by finding the symbol that com ##press ##es best , given the previous history) . this equivalence has been used as a justification for using data compression as a bench ##mark for ' general intelligence . ' [SEP]

Visualizations For End Position

Legend: 🟥 Negative □ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
19	19 (1.00)	19	3.50	[CLS] what is machine learning related to ? [SEP] there is a close connection between machine learning and data compression : a system that predict ##s the posterior pro ##ba ##bilities of a sequence given its entire history can be used for optimal data compression (by using arithmetic coding on the output distribution) while an optimal compressor can be used for prediction (by finding the symbol that com ##press ##es best , given the previous history) . this equivalence has been used as a justification for using data compression as a bench ##mark for ' general intelligence . ' [SEP]

Figure 9. Word Importance Example - ”Data Compression”, a confident answer extraction was achieved, with ”data compression” highlighted in deep green.



Visualizations For Start Position

Legend: 🟥 Negative □ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
90	90 (0.71)	92	1.85	[CLS] what is one example of an instance that the quantitative answer to the traveling salesman problem fails to answer ? [SEP] to further highlight the difference between a problem and an instance , consider the following instance of the decision version of the traveling salesman problem : is there a route of at most 2000 kilometres passing through all of germany ' s 15 largest cities ? the quantitative answer to this particular problem instance is of little use for solving other instances of the problem , such as asking for a round trip through all sites in milan whose total length is at most 10 km . for this reason , complexity theory addresses computational problems and not particular problem instances . [SEP]

Visualizations For End Position

Legend: 🟥 Negative □ Neutral 🟩 Positive

True Label	Predicted Label	Attribution Label	Attribution Score	Word Importance
99	99 (0.96)	107	2.21	[CLS] what is one example of an instance that the quantitative answer to the traveling salesman problem fails to answer ? [SEP] to further highlight the difference between a problem and an instance , consider the following instance of the decision version of the traveling salesman problem : is there a route of at most 2000 kilometres passing through all of germany ' s 15 largest cities ? the quantitative answer to this particular problem instance is of little use for solving other instances of the problem , such as asking for a round trip through all sites in milan whose total length is at most 10 km . for this reason , complexity theory addresses computational problems and not particular problem instances . [SEP]

Figure 10. Word Importance Example - ”Travelling Salesman”, answer extraction was not as obvious since incorrect starting word ”instance” or ”fails” were greener than the correct starting word ”asking”.

alization and using PCA to project word representations to understand embeddings (Zehui Wang, 2019).

References

The stanford question answering dataset. Website, 2020. https://rajpurkar.github.io/SQuAD-explorer.

Chen, X., Cheng, Y., Wang, S., Gan, Z., Wang, Z., and Liu, J. Earlybert: Efficient bert training via early-bird lottery tickets. *arXiv preprint arXiv:2101.00063*, 2020.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Gaikwad, S., Morina, D., Nistala, R., Agarwal, M., Cossette, A., Bhanu, R., Savage, S., Narwal, V., Rajpal, K., Regino, J., et al. Daemo: A self-governed crowdsourcing marketplace. In *Adjunct proceedings of the 28th annual ACM symposium on user interface software & technology*, pp. 101–102, 2015.

Hewlett, D., Lacoste, A., Jones, L., Polosukhin, I., Fandrianto, A., Han, J., Kelcey, M., and Berthelot, D. Wikireading: A novel large-scale language understanding task over wikipedia. *arXiv preprint arXiv:1608.03542*, 2016.

Jain, S. and Wallace, B. C. Attention is not explanation. *arXiv preprint arXiv:1902.10186*, 2019.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*, 2019.

Pennington, J., Socher, R., and Manning, C. D. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.

Rajpurkar, P., Jia, R., and Liang, P. Know what you don’t know: Unanswerable questions for squad. *arXiv preprint arXiv:1806.03822*, 2018.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Sundararajan, M., Taly, A., and Yan, Q. Axiomatic attribution for deep networks. In *International Conference on Machine Learning*, pp. 3319–3328. PMLR, 2017.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. Newsqa: A machine comprehension dataset. *arXiv preprint arXiv:1611.09830*, 2016.

van Aken, B., Winter, B., Löser, A., and Gers, F. A. How does bert answer questions? a layer-wise analysis of transformer representations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pp. 1823–1832, 2019.

Zehui Wang, X. Z. Machine comprehension with bidaf and answer point. 2019. URL https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6895380.pdf.

Zhu, J. Squad model comparison. 2019. URL https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1194/reports/default/15708284.pdf.