Answerability Verification in SQuAD

Stanford CS224N Default Project (Graded)

Thatcher Freeman
Department of Computer Science
Stanford University
tfr@stanford.edu

Albert Zuo
Department of Computer Science
Stanford University
azuo@stanford.edu

Abstract

Question answering has become one of the most important problems in modern NLP research. In this project, we explore the use of pre-trained language models and the use of a separate module for answerability verification in building a retrospective reader model for the SQuAD 2.0 dataset. While we find that our model does not improve on the standard EM and F1 metrics, we note that the design of our model allows for advancements in design and interpretability.

1 Key Information

- Mentor: Haoshen Hong
- External Collaborators: None
- Sharing project: No

2 Introduction

In recent years, the problem of question answering has been at the forefront of modern NLP research. The problem is as follows: given a question and a context passage, find a sequence of words in the passage that answers the question (if the answer is contained in the passage). This task acts as a benchmark by which we can measure how well a model *understands* a piece of text. Success in natural language understanding can lead to success in many other areas of NLP; for many NLP tasks, building a model that understands the relevant text is a crucial first step toward its ultimate success.

The Stanford Question Answering Dataset (SQuAD) is the standard dataset on which question-answer performance is measured. A look at the SQuAD 2.0 leaderboard quickly shows that in the past year, machines have surpassed human performance at this task. Among the many different technical innovations behind this achievement, we decided to explore two specific ones in this project. One is the use of pre-trained language embeddings, and the other is the use of a separate verification module.

Nearly all of the top scoring models on the SQuAD leaderboards have made use of (a variant of) the pre-trained language embedding model, BERT. BERT [1] is a transformer trained on a large corpus to encode word meanings and sentence relations, and has been fine-tuned with minor modifications to achieve state-of-the-art performance on a variety of tasks, SQuAD included.

Outside of pre-trained language embeddings, we also look at higher-level, model architecture modifications. The biggest change between the SQuAD 1.1 dataset and the SQuAD 2.0 dataset is that some questions *have no answer*. To allow the model to be more confident in its own answer, Hu et al. [2] propose the addition of a separate verification module alongside standard span prediction model architectures. Since the publication of their paper, many of the state-of-the-art models now include their own take on this verification module. One of our main goals in this project was to distill out these differences and better understand how effective a verification module is at its core.

3 Related Work

3.1 Pre-Trained Language Embeddings

The use of pre-trained language embeddings comes from the idea that natural language can be represented in space as a high-dimensional vector. Some embedding models such as Word2Vec ([3]) and GloVe ([4]) represent each word as a vector or "word embedding". In 2018, [1] showed the power of contextual language embeddings. The inspiration for their model BERT was two-fold: first, they noticed that textual representations should be based on the context they occur in. Second, while other language embedding models took into account the context, they only took into account left context; BERT takes both directions of context into account. Since then, usage of variants of BERT has become ubiquitous in cutting-edge models.

BERT is a multi-layer bidirectional transformer encoder pre-trained using the Cloze task, otherwise known as masked language modeling (MLM). In MLM, given a set of input text (taken from a variety of NLP tasks, not solely question-answering), the model initially masks a subset of the input, then tries to predict the portion that was masked. In fine-tuning BERT for a specific task, the parameters are tuned using only data from that task. The power of BERT is in its capability of using one architecture to achieve state-of-the-art performance on multiple NLP tasks, further cementing the idea that natural language understanding is fundamental to NLP tasks beyond question-answering.

3.2 Answerability Verification Modules

Prior to the model proposed in [2], the standard method for incorporating a no-answer prediction into a model is to prepend a special "CLS" token to the passage. If the model decides that a question has no answer, it will report that the start and end tokens of the answer are both this "CLS" token. Some more recent models add an extra signal by applying normalization on the span prediction and answerability scores to get a confidence score, which reports how confident the model is in its span prediction answer versus outputting no answer ([5, 6, 7]).

[2] deviated from such models by running the span prediction (answer) through a second module that double-checks whether the predicted answer makes sense. They test three different types of modules: a sequential module, an interactive module, and a hybrid of the two. The former two modules differ in how they process the input. The sequential module concatenates the context and the question with the predicted answer, and runs that through a multi-layer transformer decoder. The latter module passes the context and question through a bidirectional LSTM, and the predicted answer through a separate bidirectional LSTM, and merges these two outputs.

[8] iterates on the model in [2] by separating their architecture into a "sketchy reading module" and an "intensive reading module." Instead of having the first module produce a candidate answer, the sketchy reading module guesses the answerability of the question. The intensive reading module then creates a candidate answer span, and combines the output from the sketchy reading module to give a final answer.

4 Approach

After generating a baseline score with the provided BiDAF code, we decided to augment this baseline model with a pre-trained version of BERT.

4.1 Pre-processing

This change required a complete overhaul of the pre-processing stage of the pipeline, necessitating that the input contexts and questions are tokenized by BERT's wordpiece tokenizer instead of at simple word boundaries. Additionally, if the output hidden states of our model were at a wordpiece granularity, then we would need to transform the predicted answer start and end token positions back into normal strings after inference.

Modifying the setup script to use wordpieces was nontrivial. For questions that are answerable, the SQuAD dataset includes the index of the character in the context string at which the answer starts (the Answer Start), and the substring of the context which comprises the answer (the Answer

Text). Many of the answers in the SQuAD dataset start or end mid-word, which meant that simply tokenizing the Answer Text would result in tokens that were not a subsequence of the tokenized context. Additionally, given the Answer Start, we needed to convert this context-character index into a context-token index. The library BERT wordpiece tokenizer did not provide any such mapping. Sometimes the tokenizer would drop unicode characters, and other times it would replace a unicode character with a latin equivalent (é $\rightarrow$ e), so applying the tokenizer and converting back to a string could result in different strings, potentially with different lengths and start indices for each word.

We address the first problem by extending the Answer Text left and right to include the entirety of its start and end words. We then tokenize the expanded Answer Text and search within the tokenized context to find all occurrences of the answer's tokens within the context. Short answers may appear in the context multiple times, so it would be important to train our model with the correct instance of the answer. Using the Answer Start divided by the length of the original context string as a guide, we choose the occurrence of the answer that is most similarly positioned within the tokenized context, and we derive our Answer Start and End token indices from this answer occurrence.

The input to BERT for most of the following models was a single series sequence of n tokens, of the form: `[CLS]`, Tokenized Context, `[SEP]`, Tokenized Question, `[SEP]`. The BERT + BiDAF model is the sole exception, which received the tokenized context and tokenized question as separate inputs.

4.2 BERT + BiDAF

For our initial tests, we replaced the embeddings layer of the BiDAF model with a frozen, pre-trained BERT Base model, still using the Adadelta optimizer with which we trained the baseline model. We found that without fine-tuning BERT, this model did not quickly converge to a useful system, with the loss continuously decreasing, but the EM and F1 scores also decreasing over time, well below the baseline's scores even after 18 epochs. In consideration of time, we decided to abandon this model in favor of simpler, fine-tuned BERT models.

4.3 Span Prediction with Bert

In coming up with a span prediction module, we decided to replace our BERT + BiDAF model with a fine-tuned BERT model. Since the purpose of this module is mainly to output a span prediction (and not necessarily to decide the answerability of the question), we tried fine-tuning the module based on both the SQuAD 1.1 task (no unanswerable questions) and the SQuAD 2.0 task (some unanswerable questions).

We borrowed our fine-tuning procedure and model architecture from [1]. Suppose h is the hidden size of our BERT model. We simply define Start and End vectors of size $h \times 1$ which we dot product against each of the n outputs of BERT, and we apply softmax to these outputs to get a probability distribution for the start and end positions, with the first token representing 'No Answer'. In our model, we used these two linear layers on top of a pre-trained BERT Base model, and we fine-tuned for the SQuAD 1.1 task for 2 epochs and the SQuAD 2.0 task for 4 epochs, choosing the best found model for different metrics when needed. Initially, we tried training both models with 0, 6, 9, and all 12 transformer encoder layers frozen, but found that we had the best results when fine-tuning across all layers (0 frozen layers), so we focused on this variant.

In our results table below, the models fine-tuned on SQuAD 1.1 and SQuAD 2.0 are denoted as "SQuAD v1.1/v2.0 Span Predictor".

4.4 Answer Verification with BERT

The next change we made was to add a verifier module that focuses on the answerability of a question. This idea is based on the work in [2], running the model's span prediction (answer) through a second module that double-checks whether the predicted answer makes sense. We wish to explore the conceptual idea behind [2] that the model should take a second read through the given context. While they use this second pass to verify that their predicted answer/no-answer prediction is correct, we think it is more natural to have this pass verify whether there is an answer to the question in the first place. Instead of mixing the span prediction and the no-answer prediction into a single module, we wish to separate out these two, then combine the output of these two modules into a single prediction. We tried three different verifier architectures.

4.4.1 BERT-CLS Verifier

The first model we attempted was simply a linear layer connected only to the hidden state corresponding to the `[CLS]` token, with a single output that is sent through the sigmoid function to get a probability distribution. We trained this verifier model for 3 epochs and used the version we found with the best answerability (AvNA) score.

4.4.2 BERT-FC Verifier

Inspired by the "sketchy reading" module in [8], we begin by flattening BERT Base's n output vectors into a $n \times h$ vector. This is passed into a Linear layer with an output dimension of 1. The output of the linear layer is passed through a sigmoid function to get the probability that the question is answerable. We trained for 4 epochs, but we strangely found that the AvNA score started decreasing shortly after epoch 2, so we terminated training and reported the best AvNA score found.

4.4.3 BERT-SP Verifier

We observed that our BERT span predictor that was trained on SQuAD 2.0 had a higher AvNA than the briefly fine-tuned BERT-CLS and BERT-FC verifiers we tried. As a result, to emulate the high performance of the verifier reported in [8], we decided to adapt one of our span predictor modules to verification. We ran inference on the SQuAD 2.0 span predictor model we previously trained in Section 4.3, and used its choice of "Answer" or "No Answer" as the output of the verifier.

4.5 Retro-Reader

After training our verification and span prediction modules separately, we then unified our verification and span prediction modules into a single model. The architecture we pursued was similar to [8]: the verifier generates a probability p_a that the question is answerable, and the span predictor generates a probability distribution of where the answer's start and end span occur, S and E. With a SQuAD 1.1 span predictor, we simply set a final answerability score equal to p_a. With the SQuAD 2.0 span predictor, the final answerability score is computed by the following expression:

$$\begin{gathered}
\textit{No answer score} = \lambda_1(S_0 + E_0) + \lambda_2(1 - p_a) \\
\textit{Has answer score} = \max(S_i + E_j) \;\; \textit{with} \;\; 1 \leq i, j < n \\
\textit{Answerability score} = \textit{sigmoid}(\textit{Has answer score} - \textit{No answer score})
\end{gathered}$$

where λ_1, λ_2 are parameters that are either optimized for or selected in advance, and S_0 and E_0 represent the probability associated with the `[CLS]` token. During prediction, the model outputs "No Answer" if the answerability score is less than 0.5, and returns the best answer given by $S_1 ... S_n$ and $E_1 ... E_n$ if the answerability score exceeds 0.5.

We tried a few different retro-readers. The first two models we attempted had the answerability score solely determined by the verifier. In these models, the span predictor was the SQuAD 1.1 span predictor from 4.3, and the verifier was the BERT-CLS or the BERT-SP verifier. Because the span predictor, verifier, and λ_1, λ_2 were all frozen, we could quickly test on this model without any form of training. The results of these models are reported in Table 1 as "(SPv1.1, BERT-CLS) RR" and "(SPv1.1, BERT-sp) RR", respectively.

In our third and fourth retro-reader models, we use the SQuAD 2.0 span predictor from 4.3. We combine this with the BERT-CLS and BERT-FC verifiers for these two models. Because this span predictor can also choose "No Answer", we instead let an AdamW optimizer choose the values of λ_1 and λ_2. The results for these two retro-reader models are reported as "(SPv2.0 BERT-CLS) RR" and "(SPv2.0 BERT-FC) RR" respectively.

5 Experiments

5.1 Data and Evaluation Methods

We use the SQuAD 1.1 and the SQuAD 2.0 training datasets. SQuAD 1.1 contains roughly 100,000 question-answer pairs on over 500 Wikipedia articles. In addition to the question-answer pairs,

SQuAD 2.0 contains over 50,000 unanswerable questions written adversarially to look similar to answerable questions. For each model, we measured the AvNA score, as well as the F1 and EM scores (if applicable). In addition to AvNA, we report the number of False Positive and False Negative results in the Dev set (roughly 6000 question-answer pairs), to indicate the number of times that the model incorrectly predicted a question was answerable or not answerable, respectively.

5.2 Experimental details

BiDAF, our baseline, was trained with the Adadelta optimizer with a 0.5 learning rate, in batch sizes of 64 across 30 epochs. BERT+BiDAF was trained for 18 epochs using the same optimizer and parameters as our baseline BiDAF, with the BERT portion frozen. The BERT Span predictors and BERT verifiers were trained using the AdamW optimizer with a learning rate of 3e-5, in batch sizes of 8, for 2 to 4 epochs. The retro-readers with the SQuAD 1.1 span predictors were not trained, assigning the entire answerability prediction to the verifier's output, and the Retroreader with the Squad 2.0 span predictor and the BERT-CLS verifier was trained with AdamW with a learning rate of 1e-3 for two epochs and a batch size of 32. All other AdamW optimizer parameters are left at their defaults (weight decay: 0.01, ϵ: 1e-8, β_1: 0.9, β_2: 0.999).

5.3 Results

For each of the models described above, Table 1 contains the EM, F1, AvNA, False Positive (FP), and False Negative (FN) scores.

	SQuAD v2.0					SQuAD v1.1		
Model	EM	F1	AvNA	FP	FN	EM	F1	AvNA
BiDAF Baseline	57.50	60.94	67.94	1409	499			
SQuAD v1.1 Span Predictor	35.75	40.31	47.84	3121	0	74.71	84.24	100.00
SQuAD v2.0 Span Predictor	68.13	72.18	78.26	957	344			
BERT-CLS Verifier	-	-	69.32	1478	358			
BERT-FC Verifier	-	-	67.23	1686	275			
BERT-SP Verifier	-	-	78.26	957	344			
(SPv1.1, BERT-CLS) RR	58.82	62.86	69.32	1478	358			
(SPv1.1, BERT-SP) RR	68.43	72.38	78.26	957	344			
(SPv2.0, BERT-CLS) RR	67.01	71.22	77.66	1031	306			
(SPv2.0, BERT-FC) RR	66.78	71.08	77.54	1054	290			

Table 1: **Dev Set Results**: Retro-readers are denoted with (Span Predictor, Verifier) for clarity.

We submitted the (SPv1.1, BERT-SP) retro-reader to the Test leaderboard and got F1 and EM scores of 71.643 and 67.168, respectively.

Overall, we found that our EM and F1 scores were not as high as results found with BERT-Base in previous work. We suspect that this is a results of several contributing factors. First, because we wanted to build our project iteratively over the provided BiDAF code, we may have had some errors in our handling of the tokenization process. Generally, transformer architectures like BERT will operate by consuming a window of 384 tokens, that slides over the input with a stride length of 128. In the time given, it was not obvious how exactly to implement this properly, so our models are trained only on the examples that fit within $n = 512$ tokens, and the samples from the dev and test dataset that exceed 512 tokens in length were truncated.

In [1], the authors placed the question before the context in the input to BERT for the SQuAD task. In our model, we placed the context first in the input string so that we could re-use prediction logic between the BERT + BiDAF model and the BERT models. Placing the context tokens at the start of the BERT input/output allowed for indexes into the input representation to also be used as indexes into the tokenized context, without having to offset by the length of the question. While our initial tests did not seem to show a difference in convergence between placing the question or the context first in the input, it is conceivable that this may have affected our overall scores.

Our BERT-CLS and BERT-FC verifiers underperformed our expectations in their resulting AvNA scores. Possibly, we did not give these verifiers sufficient time to train, as it seemed that the BERT-FC

model should have reached more useful levels of performance as in [8]. However, in a similar timeframe, the SQuAD v2.0 span predictor was able to reach a significantly higher AvNA. We first considered that the BERT-FC verifier had a much larger number of parameters to learn than the span predictor, but we discounted this theory as the BERT-CLS verifier had similar results. Therefore, we hypothesize that this was a result of the verifier not having access to the position of the answer for answerable questions, so it therefore had difficulty learning the relationship between the input question and context tokens and how they affected answerability.

The (SQv2.0, BERT-CLS) retro-reader underperformed relative to SQuAD v2.0 span predictor on its own. With the two models being mostly identical in terms of span prediction, this comes from the reduction in AvNA, due to the retro-reader not enforcing that $i \leq j$ in the computation for the *Has answer score*, whereas this was enforced for the Span Predictor and BERT-SP models. We did observe, however, that the (SQv2.0, BERT-CLS) retro-reader *did* have a lower False Negative rate than either of its components. This observation is what led us to attempt the (SPv2.0, BERT-FC) retro-reader, observing that the BERT-FC verifier had the lowest False Negative rate of all our models. We found that the (SPv2.0, BERT-FC) retro-reader had a good balance of the low FN rate from the BERT-FC verifier, with a reduced FP rate closer to the SPv2.0's FP rate than to the BERT-FC false positive rate.

Regardless, our goal in this project was not to show state-of-the-art results, but to demonstrate that combining separate, specialized models can match or exceed the performance of a monolithic, general purpose model. By keeping our tokenization and training procedure largely constant between the different models, the resulting scores from each model are comparable to the others presented within this paper.

6 Analysis

6.1 SQuAD 1.1 Span Predictor:

This model performed fairly well on the SQuAD 1.1 task, considering that it was fine tuned for only 2 epochs on the smaller SQuAD 1.1 dataset, half as long as the SQuAD 2.0 span predictor. We found that the two models made similar span predictions, and were usually similar to or matched the reference answers, when answerable. Occasionally, there were hiccups, as in the following example:

1. **Context:** The time required by a deterministic Turing machine M on input x is the total number of state transitions, or steps, the machine makes before it halts and outputs the answer ("yes" or "no").
   Question: The time required to output an answer on a deterministic Turing machine is expressed as what?
   Gold Answer: the total number of state transitions, or steps
   Our Model: "yes" or "no"
   It seems that the model had learned that parenthesized expressions often contain clarifications or alternate definitions for previous phrases, like an appositive. In this case however, the phrase "yes" or "no" serves to describe the word "answer", not the entire prior sentence, and the model failed to identify that relationship.

6.2 SQuAD 2.0 Span Predictor:

We found that the core issue with the SQuAD 2.0 span predictor was that it had difficulty determining whether a question was answerable or not. In most cases, when a question was answerable and the span predictor guessed an answer, the answer was very similar to the reference answer. However, for non-answerable questions, the model was prone to choosing a span of text that would be in the format of a correct answer, or answer a similar question. We observe a couple examples of this below:

1. **Context:** Gatherings of thousands of people on the banks of the Vistula on Midsummer's Night for a festival called Wianki (Polish for Wreaths) have become a tradition and a yearly event in the programme of cultural events in Warsaw. The festival traces its roots to a peaceful pagan ritual where maidens would float their wreaths of herbs on the water to predict when they would be married, and to whom.

Question: What will maidens be able to predict by floating their programmes down the Vistula?

Gold Answer: No Answer

Our Model: when they would be married

In this example, the model incorrectly concluded that the question was answerable. With this tricky question, it is only unanswerable from the passage because the question mentions floating "programmes" rather than "wreaths of herbs". Adversarial questions created in this way—a correct, answerable question where some descriptor or object is replaced by something nonsensical or irrelevant—are difficult for the model. As the MLM stage of BERT training is intended to make the model robust to missing or substituted words, BERT may be prone to substituting a different meaning of a word in the question in scenarios where something does not make sense.

2. **Context:** The FSO Car Factory was established in 1951. A number of vehicles have been assembled there over the decades, including the Warszawa, Syrena, Fiat 125p (under license from Fiat, later renamed FSO 125p when the license expired) and the Polonez. The last two models listed were also sent abroad and assembled in a number of other countries, including Egypt and Colombia.

   Question: What car is licensed by the FSO Car Factory and built in Egypt?

   Gold Answers: Polonez, 125p

   Our Model: No Answer

   In order to determine the existence of an answer, the model would have to understand that the vehicles mentioned in the second sentence were manufactured in that specific car factory, and that "the last two models listed" in the third sentence was a reference to the Fiat 125p and the Polonez, which were separated by a long paranthesized expression. This long chain of relationships was probably difficult for the model to understand, and therefore it came to the incorrect conclusion.

6.3 BERT-CLS Verifier

The verifier used in our final retro-reader model, the BERT-CLS verifier, seemed to correctly decide answerability on easier questions, but made mistakes on trickier ones. The following two examples show some of the types of question formats that tripped it up.

1. **Context:** Torque is the rotation equivalent of force in the same way that angle is the rotational equivalent for position, angular velocity for velocity, and angular momentum for momentum. As a consequence of Newton's First Law of Motion, there exists rotational inertia that ensures that all bodies maintain their angular momentum unless acted upon by an unbalanced torque. Likewise, Newton's Second Law of Motion can be used to derive an analogous equation for the instantaneous angular acceleration of the rigid body:

   Question: What is the rotational equivalent of velocity?

   Gold Answer: No Answer

   Our Model: Is Answerable

   In this example, it actually seems that the gold answer is incorrect. While a little complicated, the correct answer here does seem to be that the question is answerable, and that the gold answer should be "angular velocity." This was but one of many examples we encountered where the gold answer seemed to be incorrect, and so is understandable that our verifier made a mistake here. However, the verifier's answer is good evidence that the verifier correctly handled the word analogies in this case.

2. **Context: Context:** The FSO Car Factory was established in 1951. A number of vehicles have been assembled there over the decades, including the Warszawa, Syrena, Fiat 125p (under license from Fiat, later renamed FSO 125p when the license expired) and the Polonez. The last two models listed were also sent abroad and assembled in a number of other countries, including Egypt and Colombia.

   Question: What car is licensed by the FSO Car Factory and built in Egypt?

Gold Answers: Polonez, 125p

Our Model: No Answer

This example is particularly difficult for a number of reasons. First, the question asks where the car was built, while the relevant part of the context uses the word "assembled." The context also does not directly refer to the gold answers when it discusses the assembly location, and instead uses the phrase "the last two model listed." In order to put all this together, the verifier would need to connect either "Polonez" or "125p" to the correct portion in the context, which requires even human readers multiple passes through the text to catch this. The separation between the gold answer and the question within the context made this a difficult one for the verifier to get, suggesting that our verifier might not be powerful enough to capture this information yet.

7 Design Analysis and Future Work

As our retro-reader model separates the span prediction and answerability tasks, we found that this made it simple to identify which component was contributing most significantly to a low EM or F1 score. Additionally, when one of the two modules is found to have poor performance, it could be addressed through fine-tuning or by changing the architecture without needing to re-train the other module(s). This implies that a modular design could be beneficial within the Machine Learning space. The technique of aggregating separately trained, frozen models could allow for improvements in workflow, enabling different teams to work on different modules and for greater interpretability of the results. With the subcomponents frozen, the parameters underlying the aggregation process can be quickly fine-tuned as the subcomponents only needs to run inference, a process much faster than training. When inference is expensive, the frozen subcomponent outputs across the entire training dataset could be stored in advance, as these outputs do not change with each epoch of training the aggregation parameters.

As a simple example, when running our (SPv2.0, BERT-CLS) retro-reader model, we noticed that while the AvNA score was not any better than either of its components, it did exhibit a reduced False Negative score. This implied that our model could be improved by splitting the verifier into two verifiers, one that is trained with a loss function that penalizes false negatives, and one that is trained to minimize false positives, and have these two verifiers contribute separately to the *Has Answer* and *No Answer* scores. We did not have the time to test this configuration, but we see the potential in identifying useful statistics and targeting them with dedicated modules and loss functions, and we hope that this design philosophy could be used in the industry.

8 Conclusion

In this project, we explored the use of fine-tuned pre-trained language embeddings alongside the addition of a separate verification module to form our own take on a retrospective reader model. While we did not see improvements in the standard EM and F1 metrics beyond the baseline of our fine-tuned version of BERT, our deep dive into the AvNA score through the false-positive and false-negative rates revealed some conceptually interesting takeaways.

First, this type of analysis on the different components of the model that are responsible for separate metrics in the output can provide interpretability insight into the overall model. If we were to only report or train for EM or F1, we would not be able to understand if a model was performing poorly because it was doing a poor job of identifying where answers were, or if it was doing a poor job of identifying which questions were answerable. With our retro-reader models based on the SQuAD v1.1 span predictor and a verifier module, we have a clean division where the verifier is solely responsible for AvNA score and the span predictor is solely responsible for answer correctness. If this concept were to be applied more generally, the application would be that engineers could train separate, specialized models with distinct target metrics, without having to retrain the entire system. Training separate models for different statistics and loss functions allows for an easier distribution of work in a team, reduced hardware requirements during training, and faster iteration time.

References

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

[2] Minghao Hu, Furu Wei, Yuxing Peng, Zhen Huang, Nan Yang, and Dongsheng Li. Read + verify: Machine reading comprehension with unanswerable questions, 2018.

[3] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[4] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[5] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. Zero-shot relation extraction via reading comprehension, 2017.

[6] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension, 2017.

[7] Souvik Kundu and Hwee Tou Ng. A nil-aware answer extraction framework for question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4243–4252, 2018.

[8] Zhuosheng Zhang, Junjie Yang, and Hai Zhao. Retrospective reader for machine reading comprehension, 2020.