Question Answering on Long Context

Congyun Jin
Center of Data Science
New York University
cj2164@nyu.edu

He Di
Center of Data Science
New York University
dh3171@nyu.edu

Xinli Gu
Center of Data Science
New York University
xg588@nyu.edu

Abstract

Language model pre-training has led to significant performance on question answering purposes but answering questions when the context is the whole document is challenging. For long context, question answering models need a higher inference time and the model performance would drop harshly. The goal of our project is to answer questions in a long context. By matching paragraphs of the SQuAD 2.0 dataset back to Wikipedia articles, we got a long context question answering the dataset. We fine-tuned BERT and RoBERTa on this dataset to solve the question answering problem. On top of that, we reduced the question answering computation costs by introducing a Retriever-Reader framework. We also enable automation by finding out an appropriate confidence threshold.

1 Introduction

Question answering from the real world texts is both an important and challenging problem with many applications. State of the art self-attention based models like BERT and it's variations have shown its power in achieving high performance on question answering benchmark datasets. However, the length of input sequences of the self-attention based model is typically limited to 512 characters, most of the existing QA systems are based on relatively short contexts such as paragraphs, which has limited usage in real life, where people typically want to answer questions from longer contexts. Furthermore, self-attention based models tend to have a n^2 inference time, especially when the context length grows.

Therefore, our objective is that given a real world article and a question, the system should answer the question using a span of the article or recognize that the question is impossible to answer in the given article. We also would like to improve the inference time. Our idea is to first try the Sliding Window method on a basic self-attention based model and then add a selection stage that processes chunks of a text and filters out candidates that are less likely to contain answers and feed only the relevant portions to the previous self-attention based model. Furthermore, once a prediction has been made, we should obtain confidence thresholds for a given accuracy target. The ultimate goal is to answer as many questions as possible, under the accuracy constraints. By applying the thresholding method of prediction, we will be able to create an automation process which can be used in a commercial scenario and reduce manual work. Our results show that by introducing a Retriever-Reader pipeline, we are able give a good answer under reasonable costs and time.

2 Related Work

Early approaches to Question Answering include word2vec and other approaches based on extracting word embeddings. Later, the Transformer, a self-attention based model, was proposed for better capturing the context of each word. Devlin et.al showed a wide variety of natural language processing tasks using BERT, a model based on Bidirectional Encoder Representations using Transformers

Preprint. Under review.

Devlin et al. [2018]. This model can be pre-trained for a domain, and then fine-tuned for specific tasks like Question Answering on datasets like SQuAD v2.0 and achieved very good performance. RoBERTa, a retraining of BERT with improved training methodology, was proposed subsequently Ott et al. [2019]. It removes the Next Sentence Prediction (NSP) task from BERT's pre-training and introduces dynamic masking so that the masked token changes during the training epochs. and use 1000% more data, which achieved 2-20% improvement over BERT.

However, we found that previous works focused on the QA problem where the context is a paragraph, but little research was done on whole-document context. AWS AI Labs proposed a multi-passage BERT model to deal with the long-text problem, which splits articles into passages with the length 100 words by sliding window. This method globally normalized answer scores across all passages of the same question, which help QA models find better answers by utilizing more passages. Wang et al. [2019]. Karpukhin et.al proposed a dense passage retrieval method for open-domain question answering problem but found that the traditional sparse vector space models BM25 outperforms their proposed model in SQuAD dataset Karpukhin et al. [2020]. One potential explanation is that there is a high lexical overlap between passages and questions in SQuAD dataset, which gives BM25 a clear advantage. Our approach was inspired by methods proposed by these two papers.

3 Problem Definition and Algorithm

3.1 Task

Our task is to build a question answering pipeline that can take in a full article and a question and extract the answer for the question with confidence score or return "No Answer" for unanswerable question with confidence score. After getting the raw predictions, we apply a thresholding module, to make automatic decisions based on the confidence. The whole workflow is shown in Figure 1.

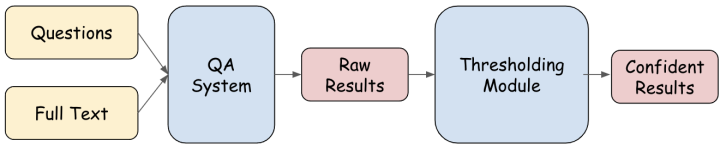

Figure 1: Workflow

3.2 Algorithm

3.2.1 Sliding Window Reader

Like the workflow shown in Figure 2. The whole article is separated into several overlapping passages based on windows size 320 and stride 128. The window size 320 is determined by our maximum input length 384 minus maximum question length 64.The query and every candidate passage will be passed to the reader to extract possible answers and its confidence scores based on start/end position, also the confidence score for impossible to answer. The final result is given by the one with the highest confidence score.

The module which extracts the answer from the passage is called reader. For the reader, we fine-tuned the BERT and RoBERTa model using both the original SQuAD 2.0 dataset and our enriched dataset by introducing more unanswerable cases.

- BERT is a bi-directional transformer pre-training over unlabeled textual data that can be used to fine-tune for our question answering tasks.
- RoBERTa is a retraining of BERT with improved training methodology.

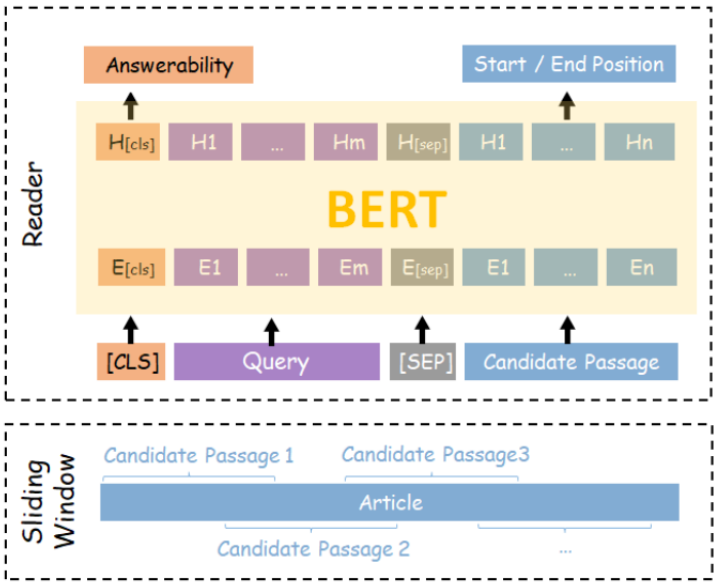

Figure 2: Sliding Window Method

3.2.2 Retriever-Reader

The Reader, same as the one of the Sliding Window method, performs the core task of question answering: extract answer based on the query and candidate passages. The Retriever assists the Reader by reducing the number of passages that the Reader has to process.

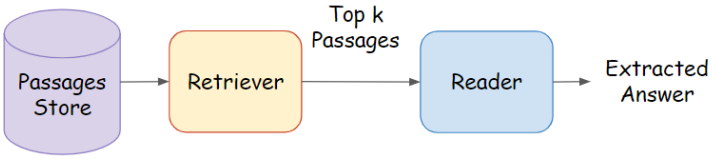

Figure 3: Retriever-Reader Pipeline

The following two retrievers were used in our experiments.

Lexical Retriever looks for literal matches of the query words in passages. And the method we used was BM25. As introduced in Christopher D. Manning [2009], BM25 is a bag-of-words retrieval function that ranks a set of documents based on the query terms appearing in each document, regardless of their proximity within the document. The function is given as follows.

Given a query Q, containing keywords $q_1, q_2, ...q_n$, the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot (1 - b + b \cdot \frac{|D|}{avgdl})}$$

where $f(q_i, D)$ is q_i's term frequency in document D, $|D|$ is the length of the document D in words, and $avgdl$ is the average document length. k_1, b are free parameters.

BM25 can also be considered as a variant of TF-IDF:

- It saturates term frequency(TF) after a set number of occurrences of the given term in the document.
- It also normalises by document length so that short documents are favoured over long documents if they have the same amount of word overlap with the query.

Semantic Retriever (or dense retrieval) encodes the query and passages into vectors and retrieves the top k passages which are most similar with the query in vector space. The encoding method we used is Sentence Transformer, as Figure 4 shows. The particular Sentence Transformer model we chose is sentence-transformers all-MiniLM-L6-v2 from Huggingface. And the similarity is measured by cosine similarity.

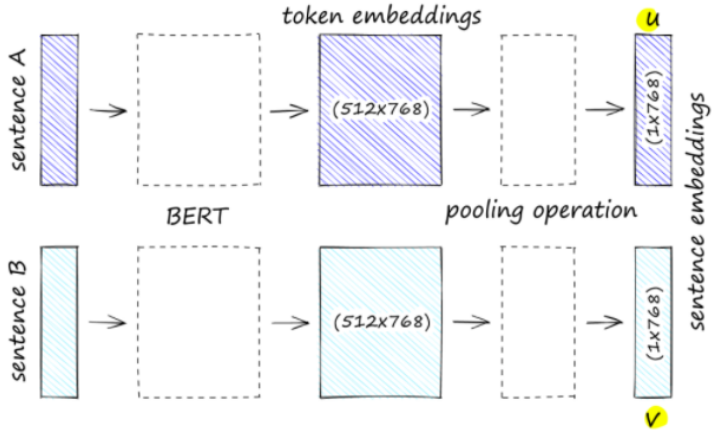

Figure 4: Sentence Transformer Example

4 Experimental Evaluation

4.1 Data

We built the dataset based on SQuAD 2.0 (The Stanford Question Answering Dataset), which is a popular benchmark dataset for past question answering works. The original dataset consists of questions posed by crowdworkers on a set of Wikipedia articles, where the answer to every question is a segment of text, or span, from the corresponding reading passage, or the question might be unanswerable. In this work, we matched the Wikipedia full articles for the specific questions in order to create a very long context for QA systems. Figure 5 shows an example of the matched dataset5. In this case, we can compare the result with the "performance ceiling" given by the original dataset SQuAD 2.0.

```
QUESTION1: 'Who was the duke in the battle of Hastings?'
ANSWER1: 'William the Conqueror'  Question ID: '56dddf4066d3e219004dad5f'

QUESTION2:'What type of major impact did the Norman dynasty have on modern Europe?
ANSWER2: 'Not Answerable'         Question ID: '5ad3a266604f3c001a3fea27'

FULL TEXT:
The Normans (Norman: Nourmands; French: Normands; Latin: Normanni) ……
……
The Norman dynasty had a major political …… Norman adventurers founded the Kingdom of
Sicily under Roger II after conquering southern Italy on the Saracens and Byzantines, and
an expedition on behalf of their duke, William the Conqueror, led to the Norman conquest
of England at the Battle of Hastings in 1066 …… where their prince Bohemond I founded the
Principality of Antioch in the Levant, to Scotland and Wales in Great Britain, to
Ireland, and to the coasts of north Africa and the Canary Islands.'
……
Under the Norman abbot Robert de Grantmesnil, several monks of Saint-Evroul fled to
southern Italy, where they were patronised by Robert Guiscard and established a Latin
monastery at Sant\'Eufemia. There they continued the tradition of singing.
```

Figure 5: A training example from the SQuAD dataset, consisting of a question, context paragraph, and answer span (in green). In this project, we match the context with the full text in Wikipedia for training and prediction

Since Wikipedia articles keep changing, in order to best match the original context in SQuAD 2.0 dataset, we first matched the full article and then located the paragraph associated with the original context, and substituted the paragraph with the context in SQuAD 2.0 dataset. Data overview is shown in Table 1.

	Train	**Validation**	**Test**
Total Articles	394	29	32
Answerable Questions	78,703	4,589	5,406
Unanswerable Questions	38,630	2,901	5,506

Table 1: Dataset Overview

4.2 Methodology

4.2.1 Retriever

The full Wikipedia articles are separated into passages for every 3 sentences. Then BM25 or Sentence Transformer is used to retrieve proper passages for the question. To compare the retriever's performance, we used the following steps to get the retriever's matching score for a single question + answer pair.

- Use the retriever to extract top-10 passages from the article in order of matching the question.
- From top-1 to top-10 passages, formulate 10 candidate passages set: {1^{st} passage}, {1^{st} passage, 2^{nd} passages}, {1^{st} passage, 2^{nd} passages, 3^{rd} passages}, ...
- Find the sentences where the right answer located. And mark the sentences as "Right-answer Sentences".
- For each candidate passages set k in step 2, calculate the highest matching score between each candidate passage in the set and "Right-answer Sentences" from step 3. Record this score as the retriever's matching score at top k for this question+answer pair. k is from 1 to 10.

After getting the matching score at top-k for every question + answer pair, we calculate the percent of 100 matching score as its retrieved accuracy at top-k. We will compare the accuracy to choose which method is better and which k is proper to keep the reader efficient because k controls the number of passages fed into the reader.

4.2.2 Reader

For the Sliding Window method, we first fine-tuned BERT and RoBERTa on the original SQuAD 2.0 dataset. Then, we also fine-tuned on an enriched SQuAD 2.0 dataset which can be seen as a form of data augmentation. Although the test set has balanced answerable and unanswerable questions, the reader will face more unanswerable cases when the article is separated into many passages. Therefore, we enriched our training dataset by introducing more unanswerable cases. Those unanswerable cases were made by questions in answerable questions and random selected context in the same article.

For the Retriever-Reader method, the chosen retriever and best performance reader will be used to get its performance.

We used Exact Match and F1 as the metric to evaluate those methods.

- **Exact Match(EM)** For each question+answer pair, if the characters of the model's prediction exactly match the characters of the true answer, the exact match will be 1, otherwise 0. This is a strict all-or-nothing metric.
- **F1 Score** The F1 score is the harmonic mean of precision and recall. Precision is the ratio of the number of shared words to the total number of words in the prediction, and recall is the ratio of the number of shared words to the total number of words in the ground truth.

4.2.3 Thresholding

When pushing the model into production, it's crucial to be able to make automatic decisions with high confidence. Our work allows us to answer questions automatically on long documents with high confidence, which results in fast quality answers for the users while lowering costs of human labeling for the company. We used the validation set to select the best threshold that can cover as many questions as possible while maintaining the 90% or 80% Exact Match score. For "HasAnswer " questions (answerable questions), we used the softmaxed score for the "Start" token logit and "End" token logit for thresholding metric. For "NoAnswer " questions (unanswerable questions), we used the NoAnswer Gap which is the difference in NoAnswer logit and HasAnswer logit for thresholding.

4.3 Results

4.3.1 Retriever Results

Figure 6 shows the matching score between the retrieved top-k passages and the right answer context.

- BM25 is better than Sentence Transformer. Because from top-1 to top-10, BM25 always has higher accuracy than Sentence Transformer. Therefore, BM25 was chosen as our retriever to do Retriever-Reader experiments.
- The performance increase came from higher top-k has a diminishing effect. Hence BM25 for top 5 (BM25@5) was chosen as the passage retriever.

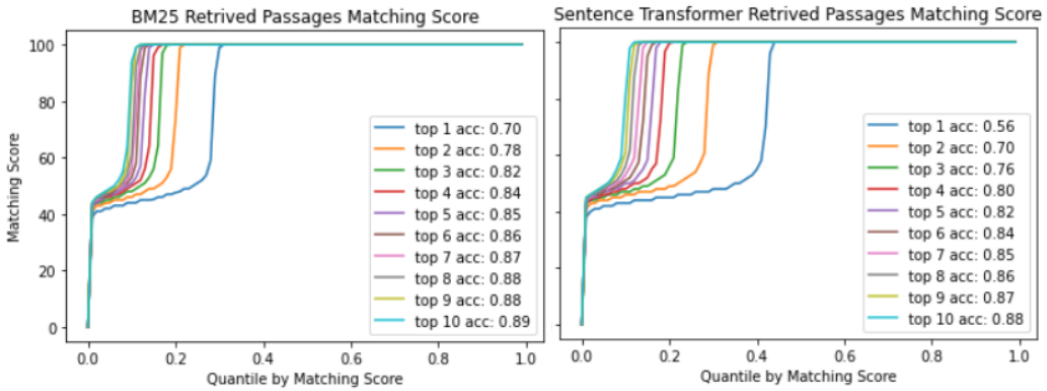

Figure 6: BM25 vs Sentence Transformer. The matching score is the highest fuzzy partial score between retrieved k passages and the real answer sentence. The accuracy is calculated by the percentage of question + answer paris which has 100 matching score. For all question + answer paris, we plot the figure by sorting their matching score from low to high. The x-axis is matching score quantiles, the y-axis is matching score.

4.3.2 Reader Results

Both the results of the Sliding Window method and the Retriever-Reader method are shown in table 2.

Paragraph Results The first two rows of the results are based on the original SQuAD 2.0 dataset, which can be viewed as the "performance ceiling" of the reader. We can compare them with performances in article context.

Sliding Window + BERT: It is our baseline model. We found it has a 20 percent Exact Match gap from the ceiling results. And it performs worse on unanswerable cases.

Sliding Window + BERT + Enriched Data Set: After fine-tuning the BERT model on our enriched data set, its ability to distinguish answerability gets enhanced.

Sliding Window + Roberta: It performs much better than the BERT model. And it just has a 7 Exact Match gap from the ceiling results.

	Model			Total		Has Ans.		No Ans.	
Context	Method	Reader	Data	EM	F1	EM	F1	EM	F1
Paragraph	NA	BERT	NA	72.87	76.20	72.09	78.81	73.64	73.64
Paragraph	NA	RoBERTa	NA	78.81	83.24	75.40	84.30	82.14	82.14
Article	Sliding Window	BERT	NA	52.28	55.93	64.19	71.56	40.59	40.59
Article	Sliding Window	BERT	Enriched	65.96	69.35	62.30	69.15	69.56	69.56
Article	Sliding Window	RoBERTa	NA	71.07	74.69	64.50	71.85	77.48	77.48
Article	BM25@5	RoBERTa	NA	69.93	72.82	65.21	71.03	74.57	74.57

Table 2: Overall Results

Retriever BM25@5 + Reader Roberta: One problem of the Sliding Window method is the heavy computation. Retriever-Reader could alleviate it by filtering the passages. And its performance is close to the Sliding Window method, which makes it more practical.

4.3.3 Thresholding Results

We chose 90% and 80% EM thresholds, shown in Figure 7, as for some use cases a 80% Exact Match might be enough, in which case we can increase the automation and lower costs of human review.

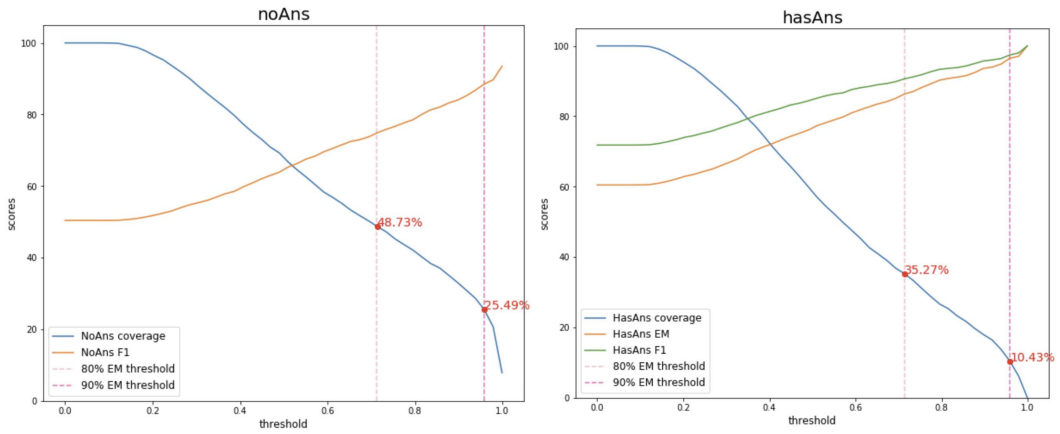

Figure 7: Thresholding Result

From the plots we can see that for NoAns examples, 80% EM threshold is 0.72, which covers 48.73% of the samples and 90% covers 25.49% of samples. For HasAns examples, 80% EM threshold covers 35.27% of the examples and 90% threshold covers 10.32% examples.

4.4 Discussion

For the retriever, BM25 has a better performance than Sentence Transformer. We think the reason is that the annotators wrote questions after seeing the passage. As a result, there is a high lexical overlap between passages and questions, which gives BM25 a clear advantage. What's more, the Sentence Transformer is not fine-tuned by our Wikipedia articles. But our dataset only contains hundreds of articles, which is extremely biased. This also discusses previously by Lee et al. [2019]

For the reader, Sliding Window RoBERTa has the best performance and is quite reasonable. Every passage of the article will be passed to the reader, the contextual embedding will be produced in the attention model. Further, the question+passage input will let the model get the task awareness rather than just get individual embedding of the query or the passage. However, even though the performance is good, the computation costs are too high to give an answer instantly. While BM25@5 could help to filter most of the improper passages and only pass 5 passages to the reader, which largely lowers the reader workload. Therefore, the BM25@5 with RoBERTa would be a more efficient method to give a good answer under reasonable costs and time.

For the thresholding part, it's reasonable that the 90% EM threshold has much lower coverage (average coverage 17.96%) than the 80% EM threshold (average coverage 42%). However, in commercial settings, lower coverage means more human review and more cost. Therefore, companies can flexibly choose between different thresholds. For example, if the project requires high level accuracy for automated results, then they may want to choose the 90% EM threshold. Otherwise, in order to cover as many as samples, one may want to choose the threshold for a lower accuracy target.

5 Conclusion

Sliding Window method helps to solve the question answering on long context problems, because the performance of this method is not far from the "performance ceiling". RoBERTa with Sliding Window is the best model from our experiments with a EM score gap of 7.74 and F1 score gap of 8.55 from the "performance ceiling". However, as this method needs all passages of the article as input, the large workload will affect its responding speed.

The Retriever-Reader method could accelerate the whole pipeline and make it more efficient. For a small decrease in performance we can reduce the computational costs largely. BM25@5 + RoBERTa shows great potential; it could filter the whole article to 5 short contexts as the input to the reader.

For model automation, we could cover 42 % questions which a human does not need to review the answer while maintaining the 80 Exact Match score by setting the threshold based on the validation set. With the thresholding module, the model can be pushed into production and conduct automatic selection for confident predictions.

In the future, we plan to explore retrievers since lots of the work has been done on question answering in short context and the reader already has really good performance in short context. Therefore, retrievers could be a bottleneck for our model performance, especially when we want to approach our "performance ceiling". Instead of using sentence transformers to get embedding of query and context individually, we may try something like Cross-Encoder that introduced in Reimers and Gurevych [2019], to get embedding of both query and context simultaneously. We can also fine-tune the transformer to make it suitable for our task.

We will also try to improve the thresholding of our model. The thresholds results were not great which means there might be something more structural to the model itself. How to measure the confidence of the answer accurately is key to calibration. We will investigate the model and find more consistent methods to do the thresholding.

6 Lessons learned

From technical prospective, we learnt how to use Huggingface transformers on our problem. This interface provides thousands of pretrained models to perform different tasks on different modalities such as text, vision, and audio Wolf et al. [2020]. It helped us a lot when choosing the baseline model, which is a crucial skill we wish to carry into our future projects.

As our project is a collaboration with company Hyperscience, we learnt something that matters in real business applications. For example, answering correctly is just one factor. Answering instantly with low computation costs is also important. And when implementing the method in production, we have to consider costs from the user's perspective. In our case, the costs of giving a user a false answer is larger than the costs of telling the user that there may exist an answer but we can't give the answer with high confidence. That is the reason we set the threshold. And its setting should be based on the quantification of the two different business costs and company's reputation concerns.

7 Student contributions

Congyun Jin: implemented the Sliding Window method, wrote the report.

Di He: implemented Retriever-Reader method, wrote the report.

Xinli Gu: implemented the thresholding and automation part, wrote the report.

References

Hinrich Schütze Christopher D. Manning, Prabhakar Raghavan. An introduction to information retrieval. In *An Introduction to Information Retrieval*, pages 232–234, Online, 2009. Association for Computational Linguistics. URL https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf.

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.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://www.aclweb.org/anthology/2020.emnlp-main.550.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. *In Association for Computational Linguistics (ACL), pages 6086–6096*, 2019.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*, 2019.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv:1908.10084*, 2019.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage bert: A globally normalized bert model for open-domain question answering, Oct 2019.

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