A Simple Yet Strong Pipeline for HotpotQA

Dirk Groeneveld[†] **Tushar Khot**[†] **Mausam**[‡] **Ashish Sabharwal**[†]

[†] Allen Institute for AI, Seattle, WA, U.S.A.
dirkg,tushark,ashishs@allenai.org
[‡] Indian Institute of Technology, Delhi, India
mausam@cse.iitd.ac.in

Abstract

State-of-the-art models for multi-hop question answering typically augment large-scale language models like BERT with additional, intuitively useful capabilities such as named entity recognition, graph-based reasoning, and question decomposition. However, does their strong performance on popular multi-hop datasets really justify this added design complexity? Our results suggest that the answer may be no, because even our simple pipeline based on BERT, named QUARK, performs surprisingly well. Specifically, on HotpotQA, QUARK outperforms these models on both question answering and support identification (and achieves performance very close to a RoBERTa model). Our pipeline has three steps: 1) use BERT to identify potentially relevant sentences *independently* of each other; 2) feed the set of selected sentences as context into a standard BERT span prediction model to choose an answer; and 3) use the sentence selection model, now with the chosen answer, to produce supporting sentences. The strong performance of QUARK resurfaces the importance of carefully exploring simple model designs before using popular benchmarks to justify the value of complex techniques.

1 Introduction

Textual Multi-hop Question Answering (QA) is the task of answering questions by combining information from multiple sentences or documents. This is a challenging reasoning task that requires QA systems to identify relevant pieces of information in the given text and learn to compose them to answer a question. To enable progress in this area, many datasets (Welbl et al., 2018; Talmor and Berant, 2018; Yang et al., 2018; Khot et al., 2020) and models (Min et al., 2019b; Xiao et al., 2019; Tu et al., 2020) with varying complexities have been proposed over the past few years. Our work focuses on HotpotQA (Yang et al., 2018), which contains 105,257 multi-hop questions derived from two Wikipedia paragraphs, where the correct answer is a span in these paragraphs or yes/no.

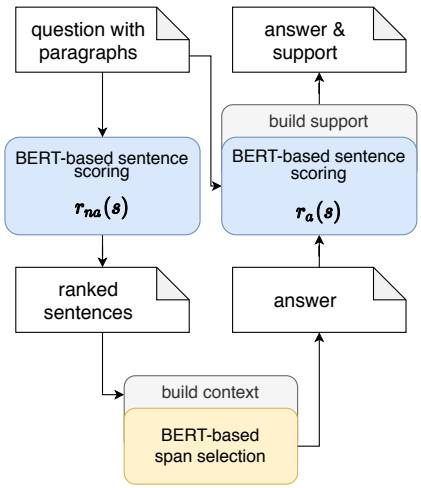

Figure 1: Overview of the QUARK model, with a question and context paragraphs as input. In both blue boxes, sentences are scored independently from one another. $r_{na}(s)$ and $r_a(s)$ use the same model architecture with different weights.

Due to the multi-hop nature of this dataset, it is natural to assume that the relevance of a sentence for a question would depend on the other sentences considered to be relevant. E.g., the relevance of "Obama was born in Hawaii." to the question "Where was the 44$^{\text{th}}$ President of USA born?" depends on the other relevant sentence: "Obama was the 44$^{\text{th}}$ President of US." As a result, many approaches designed for this task focus on *jointly* identifying the relevant sentences (or paragraphs) via mechanisms such as cross-document attention, graph networks, and entity linking.

Our results question this basic assumption. We show that a simple model, QUARK (see Fig. 1), that first identifies relevant sentences from each

Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, pages 8839–8845,
November 16–20, 2020.

paragraph *independent* of other paragraphs, is surprisingly powerful on this task: By using only the context of the corresponding paragraph, QUARK can recover all gold supporting sentences within the top-5 sentences. For QA, it uses a standard BERT (Devlin et al., 2019) span prediction model (similar to current published models) on the output of this module. Additionally, QUARK exploits the inherent similarity between the relevant sentence identification task and the task of generating an explanation given the answer from a QA module: it uses the same architecture for both tasks.

We show that this independent sentence scoring model results in a simple QA pipeline that outperforms other BERT models in both 'distractor' and 'fullwiki' settings of HotpotQA. In the distractor setting (10 paragraphs, including two gold, provided as context), QUARK achieves joint scores (answer and support prediction) within 0.75% of the current state of the art. Even in the fullwiki setting (all 5M Wikipedia paragraphs as context), by combining our sentence selection approach with a commonly used paragraph selection approach (Nie et al., 2019),[1] we outperform all previously published BERT models. In both settings, the only models scoring higher use RoBERTa (Liu et al., 2019), a more robustly trained language model that is known to outperform BERT across various tasks.

While our design uses multiple transformer models (now considered a standard starting point in NLP), our contribution is a simple pipeline without any bells and whistles, such as NER, graph networks, entity linking, etc. The closest effort to QUARK is by Min et al. (2019a), who also propose a simple QA model for HotpotQA, but don't consider the support task and fall several points short of SOTA on the QA task due to reasoning over only one paragraph at a time.

Finally, our ablation study demonstrates that the sentence selection module benefits substantially from using context from the corresponding paragraph. It also shows that running this module a second time, with the chosen answer as input, results in more accurate support identification.

2 Related Work

Most approaches for HotpotQA attempt to capture the interactions between the paragraphs by either relying on cross-attention between documents or sequentially selecting paragraphs based on the previously selected paragraphs.

While Nishida et al. (2019) also use a standard Reading Comprehension (RC) model, they combine it with a special Query Focused Extractor that identifies relevant sentences by updating a RNN state representation in each step, allowing the model to capture dependencies between sentences across time-steps. Xiao et al. (2019) propose a Dynamically Fused Graph Networks (**DFGN**) model that first creates an entity graph from paragraphs, dynamically extracts sub-graphs, and fuses them with paragraph representations. The Select, Answer, Explain (**SAE**) model (Tu et al., 2020) also first selects relevant documents and uses them to produce answers and explanations. However, it relies on a self-attention over *all* document representations to capture potential interactions. Additionally, it relies on a Graph Neural Network (GNN) to answer the questions. Hierarchical Graph Network (**HGN**) model (Fang et al., 2020) builds a hierarchical graph with three levels: entities, sentences and paragraphs to allow for joint reasoning. **DecompRC** (Min et al., 2019b) takes a completely different approach of learning to decompose the question (using additional annotations) and then answer the decomposed questions using a standard single-hop RC system.

Others such as Min et al. (2019a) have noticed that many HotpotQA questions can be answered just based on a single paragraph. However, they did not consider the support identification task (which we show can also be done independently). While they achieve strong (but not quite SOTA) QA performance by *only reasoning over a single paragraph*, we show that interaction is actually valuable for QA! Specifically, by using relevant sentences spread across multiple paragraphs, our simple model outperforms previous models with more complex interactions. We thus view QUARK as a different, stronger baseline for multi-hop QA.

In the fullwiki setting, each question has no associated context and models are expected to select paragraphs from Wikipedia. To be able to scale to such a large corpus, the proposed systems often select the paragraphs independent of each other. A recent retrieval method in this setting is Semantic Retrieval (Nie et al., 2019) where first the paragraphs are selected based on the question, followed by individual sentences from these para-

[1] While their approach selects the paragraphs jointly using the link structure, our sentence selection approach is still independent of the other paragraphs.

graphs. However, unlike our approach, they do not use the paragraph context to select the sentences, missing key context needed to identify relevance.

3 Pipeline Model: QUARK

Our model works in three steps. First, we score individual sentences from an input set of paragraphs D based on their relevance to the question. Second, we feed the highest-scoring sentences to a span prediction model to produce an answer to the question. Third, we score sentences from D a second time to identify the supporting sentences using the answer. These three steps are implemented using the two modules described next in Sections 3.1 and 3.2.

3.1 Sentence Scoring Module

In the distractor setting, HotpotQA provides 10 context paragraphs that have an average length of 41.4 sentences and 1106 tokens. This is too long for standard LM-based span-prediction—most models scale quadratically with the number of tokens, and some are limited to 512 tokens. This motivates selecting a few relevant sentences E to reduce the size of the input to the model without losing important context. In a similar vein, the support identification subtask of HotpotQA also involves selecting a few sentences that best explain the chosen answer. We solve both of these problems with the same transformer-based sentence scoring module, with slight variation in its input.

We score every sentence s from every paragraph $p \in D$ independently by feeding the following sequence to the model: `[CLS] question [SEP] p [SEP] answer [SEP]`. This sequence is the same for every sentence in the paragraph, but the segment ID for the sentence being classified is set to 1 for tokens from the sentence, and to 0 for the rest. Each annotated support sentence forms a positive example and all other sentences from D form the negative examples. Note that our classifier scores each sentence independently and never sees sentences from two paragraphs at the same time. (details in App. A.1)

We train two variants of this model: (1) $r_{na}(s)$ is trained to score sentences given a question but no answer (`answer` is replaced with a `[MASK]` token); and (2) $r_a(s)$ is trained to score sentences given a question and its gold answer. We use $r_{na}(s)$ for relevant sentence selection and $r_a(s)$ for support identification.

3.2 Question Answering Module

To find answers to questions, we use Wolf et al. (2019)'s implementation of Devlin et al. (2019)'s span prediction model. To achieve our best score, we use their *BERT-Large-Cased* model with whole-word masking and SQuAD (Rajpurkar et al., 2016) fine-tuning.[2] We fine-tune this model on the HotpotQA dataset with input QA context E from $r_{na}(s)$. Since BERT models have a hard limit of 512 word-pieces, we use $r_{na}(s)$ to select the most relevant sentences that can fit within this limit, as described next. (See Appendix A.2 for training details.)

To accomplish this, we compute the score $r_{na}(s)$ for each sentence in the input D. Then we add sentences in decreasing order of their scores to the QA context E, until we have filled no more than 508 word-pieces (incl. question word-pieces). For every new paragraph considered, we also add its first sentence, and the title of the article (enclosed in `<t></t>`). This ensures that our span-prediction model has the right co-referential information from each paragraph. We arrange these paragraphs in the order of their highest-scoring sentence, so the most relevant sentences come earlier – a signal that could be exploited by our model. The final four tokens are a separator, plus the words `yes`, `no`, and `noans`. This allows the model to answer yes/no comparison questions, or give no answer at all.

3.3 Bringing it Together

Given a question along with 10 distractor paragraphs D, we use the $r_{na}(s)$ variant of our sentence scoring module to score each sentence s in D, again without looking at other paragraphs. In the second step, the selected sentences are fed as context E into the QA module (as described in Section 3.2) to choose an answer. In the final step, to find sentences supporting the chosen answer, we use $r_a(s)$ to score each sentence in D, this time with the chosen answer as part of the input.

We define the score $R_a(S)$ of a set of sentences $S \subset D$ to be the sum of the individual sentence scores; that is, $R_a(S) = \sum_{s \in S} r_a(s)$.[3] In HotpotQA, supporting sentences always come from exactly two paragraphs. We compute this score for all possible S satisfying this constraint and take the highest scoring set of sentences as our support.

[2] While we use the model fine-tuned on SQuAD, ablations show that this only adds 0.2% to the final score.

[3] Note that $r_a(s)$ is the logit score and can be negative, so adding a sentence may not always improve this score.

QA Model	Ans EM	Ans F1	Sup EM	Sup F1	Joint EM	Joint F1
Single-paragraph (Min et al., 2019a)	–	67.08	–	–	–	–
QFE (Nishida et al., 2019)	53.70	68.70	58.80	84.70	35.40	60.60
DFGN (Xiao et al., 2019)	55.66	69.34	53.10	82.24	33.68	59.86
SAE (Tu et al., 2020)	61.32	74.81	58.06	85.27	39.89	66.45
HGN (Fang et al., 2020)	–	79.69	–	**87.38**	–	71.45
QUARK (Ours)	**67.75**	**81.21**	**60.72**	86.97	**44.35**	**72.26**
SAE (RoBERTa) (Tu et al., 2020)	67.70	80.75	**63.30**	87.38	**46.81**	72.75
HGN (RoBERTa) (Fang et al., 2020)	–	81.00	–	**87.93**	–	**73.01**

Table 1: HotpotQA's distractor setting, Dev set. The bottom two models use larger language models than QUARK.

QA Model	Ans EM	Ans F1	Sup EM	Sup F1	EM	F1
QFE (Nishida et al., 2019)	28.66	38.06	14.20	44.35	8.69	23.10
SR-MRS (Nie et al., 2019)	45.32	57.34	38.67	70.83	25.14	47.60
QUARK + SR-MRS (Ours)	**55.50**	**67.51**	**45.64**	**72.95**	**32.89**	**56.23**
HGN + SR-MRS (Fang et al., 2020)	**56.71**	**69.16**	**49.97**	**76.39**	**35.36**	**59.86**

Table 2: HotpotQA's fullwiki setting, Test set. The bottom-most model uses a larger language model than QUARK.

At first blush, the number of possible sentence subsets S to consider grows exponentially with the total number of sentences, making it impossible to evaluate them all. Fortunately, it is part of the task specification that the sentences in S always come from exactly two different paragraphs. This makes the problem exponential in the number of sentences in a single paragraph, not in the total number of sentences across paragraphs. The number of sets to consider is $O(p^2 \times 2^{2s})$, where p is the number of paragraphs and s is the number of sentences per paragraph. In practice, this results in a median of 12000 sets evaluated per question. While evaluating this number of sets is feasible, we also use another trick to reduce the sets: For every paragraph, we only consider sentences with a positive score, plus the top two sentences with a negative score.

In the fullwiki setting, we use the paragraphs SR-MRS (Nie et al., 2019) as a starting point. SR-MRS assigns a score to individual paragraphs, so we determined a score cut-off of -7.0 for optimal performance on the dev set. The rest of the experimental setup is identical to the distractor setting.

4 Experiments

We evaluate on both the distractor and fullwiki settings of HotpotQA with the following goal: *Can a simple pipeline model outperform more complex approaches?* We present the EM (Exact Match) and F1 scores on (1) answer selection, (2) support selection, and (3) Joint score.

Table 1 shows that on the distractor setting, QUARK outperforms previous models based on BERT. Moreover, we are within 1 point of models that use RoBERTa embeddings—a stronger language model that has shown big improvements in previous HotpotQA models. QUARK also performs better than the recent single-paragraph approach for the QA subtask (Min et al., 2019a) by 14 points F1. While most of this gain comes from using a larger language model, QUARK scores 2 points higher even with a language model of the same size.

We observe a similar trend in the fullwiki setting (Table 2). While we rely on retrieval from SR-MRS (Nie et al., 2019) for our initial paragraphs, we outperform the original work.[4] Even when we use the same language model as SR-MRS, BERT-Base, we achieve a joint F1 score of 51.8 on the Dev set compared to their joint F1 score of 49.2.

We attribute this improvement to two factors: our sentence selection capitalizing on the sentence's paragraph context leading to better support selection, and a better span selection model leading to improved QA.

4.1 Ablation

To evaluate the impact of context on our sentence selection model in isolation, we look at the number of sentences that score at least as high as the lowest-scoring annotated support sentence. In other words, this is the number of sentences we must send to the QA model to ensure all annotated support is included. Table 3 shows that providing the model

[4]Recent retrieval approaches outperform SR-MRS (Asai et al., 2020). Evaluating our system on their retrieval is a potential direction of future work.

	top-*n*	Sup F1	Ans F1
B-Base w/o context	10	74.45	78.59
B-Base w/ context	6	83.15	80.92
+ B-Large ($r_{na}(s)$)	**5**	85.35	**81.21**
w/ answers ($r_a(s)$)	5	**86.97**	–
Oracle	3	–	–

Table 3: Ablation study on sentence selection in the distractor setting. top-*n* is the number of sentences required to cover the annotated support sentences in 90% of the questions.

with the context from the paragraph gives a substantial boost on this metric, bringing it down from 10 to only 6 when using BERT-Base (an oracle would need 3 sentences). It further shows that this boost carries over to the downstream tasks of span selection and choosing support sentences (improving it by 9 points to 83%). Finally, the table shows the value of running the sentence selection model a second time: with BERT-Large, $r_a(s)$ outperforms $r_{na}(s)$ by 1.62% on the Support F1 metric.

5 Conclusion

Our work shows that on the HotpotQA tasks, a simple pipeline model can do as well as or better than more complex solutions, such as graph networks, cross-document attention, or NER. Powerful pre-trained models allow us to score sentences one at a time, without looking at other paragraphs. By operating jointly over these sentences chosen from multiple paragraphs, we arrive at answers and supporting sentences on par with state-of-the-art approaches. This result shows that supporting sentence identification in HotpotQA is itself not a multi-hop problem, and suggests focusing on other multi-hop datasets to demonstrate the value of more complex retrieval techniques.

Acknowledgments

Mausam is supported by grants from Google, Bloomberg, and 1MG, Jai Gupta Chair Fellowship, and a Visvesvaraya faculty award by the Govt. of India. Computations on beaker.org were supported in part by credits from Google cloud.

References

Akari Asai, Kazuma Hashimoto, Hannaneh Hajishirzi, Richard Socher, and Caiming Xiong. 2020. Learning to retrieve reasoning paths over wikipedia graph for question answering. In *ICLR*.

Peter Clark, Oren Etzioni, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, Niket Tandon, Sumithra Bhakthavatsalam, Dirk Groeneveld, Michal Guerquin, and Michael Schmitz. 2019. From 'F' to 'A' on the N.Y. Regents science exams: An overview of the Aristo project. *ArXiv*, abs/1909.01958.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jing jing Liu. 2020. Hierarchical graph network for multi-hop question answering. In *EMNLP*.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. QASC: A dataset for question answering via sentence composition. In *AAAI*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*, abs/1907.11692.

Sewon Min, Eric Wallace, Sameer Singh, Matt Gardner, Hannaneh Hajishirzi, and Luke S. Zettlemoyer. 2019a. Compositional questions do not necessitate multi-hop reasoning. In *ACL*.

Sewon Min, Victor Zhong, Luke S. Zettlemoyer, and Hannaneh Hajishirzi. 2019b. Multi-hop reading comprehension through question decomposition and rescoring. In *ACL*.

Yixin Nie, Songhe Wang, and Mohit Bansal. 2019. Revealing the importance of semantic retrieval for machine reading at scale. In *EMNLP-IJCNLP*.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *ACL*.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *EMNLP*.

Alon Talmor and Jonathan Berant. 2018. The web as a knowledge-base for answering complex questions. In *NAACL-HLT*.

Ming Tu, Kevin Huang, Guangtao Wang, Jui-Ting Huang, Xiaodong He, and Bufang Zhou. 2020. Select, answer and explain: Interpretable multi-hop reading comprehension over multiple documents. In *AAAI*.

Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. 2018. Constructing datasets for multi-hop reading comprehension across documents. *TACL*, 6:287–302.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, R'emi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Yunxuan Xiao, Yanru Qu, Lin Qiu, Hao Zhou, Lei Li, Weinan Zhang, and Yong Yu. 2019. Dynamically fused graph network for multi-hop reasoning. In *ACL*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*.

A Appendix

A.1 Training the sentence scoring model

Both $r_{na}(s)$ and $r_a(s)$ are trained the same way. We use the 90447 questions from the HotpotQA training set, shuffle them, and train for three epochs. We construct positive and negative examples by choosing the two paragraphs containing the annotated support sentences, plus two more randomly chosen paragraphs. All sentences from the chosen paragraphs become instances for the model. If an instance contains more than 512 tokens, we truncate the paragraph until the question, special tokens, and paragraph tokens fit within the 512 token limit.

During training, we follow the fine-tuning advice from (Devlin et al., 2019), with two exceptions. We ramp up the learning rate from 0 to 10^{-5} over the first 10% of the batches, and then linearly decrease it again to 0.

Except where stated otherwise, we use *BERT-Large-Cased* model trained with whole-word masking as a starting point. Here, whole word masking refers to a BERT variant that masks entire words instead of word pieces during pre-training.

To avoid biasing the training towards questions with many context sentences, we create batches at the question level. Three questions make up one batch, regardless of how many sentences they contain. We cap the batch size at 5625 tokens for practical purposes. If a batch exceeds this size, we drop sentences at random until the batch is small enough. As is standard for BERT classifiers, we use a cross-entropy loss with two classes, one for positive examples, and one for negative examples.

Training one of these models takes about 55 hours on a single NVidia Quadro RTX 8000. Inference on the same model gets up to 1.1 questions per second on the same GPU. Since our model does not differ in architecture from the one described in Devlin et al. (2019), it has the same 340M parameters. Most of our experiments centered around the format of the input data instead of hyperparameters. We only varied the number of epochs from 1 to 4, choosing 3 as the one giving the best performance on the development set.

A.2 Training the span prediction model

We train the BERT span prediction model on the output paragraphs from $r_{na}(s)$. We use a batch size of 16 questions and maximum sequence length of 512 word-pieces. We use the same optimizer settings as the sentence selection model with an additional weight decay of 0.01. The model is trained for a fixed number of epochs (set to 3) and the final model is used for evaluation.

Under the hood, this model consists of two classifiers that run at the same time. One finds the first token of potential spans, and one finds the last token of potential spans. Each classifier uses a cross entropy loss. The final loss is the average loss of the two classifiers. We train one model on the output from our best $r_{na}(s)$ selection model and use it in all our experiments (and ablations).

Once again, this model differs from BERT only in the input and training data, and thus has the same 340M parameters. Our choice of hyperparameters is derived from the work in Clark et al. (2019). We did no further hyperparameter searches. Experiments with three different random seeds show a variation of QA F1 score of $\pm 0.2\%$. Training takes about 10 hours on a NVidia Quadro RTX 8000. During inference, the span prediction model can process 25 questions per second.