AnswerQuest: A System for Generating Question-Answer Items from Multi-Paragraph Documents

Melissa Roemmele[†] **Deep Sidhpura**[◇] **Steve DeNeefe**[†] **Ling Tsou**[†]
SDL Research, Los Angeles, CA, USA
[†]{mroemmele,sdeneefe,ltsou}@sdl.com
[◇]deepsidhpura777@gmail.com

Abstract

One strategy for facilitating reading comprehension is to present information in a question-and-answer format. We demo a system that integrates the tasks of question answering (QA) and question generation (QG) in order to produce Q&A items that convey the content of multi-paragraph documents. We report some experiments for QA and QG that yield improvements on both tasks, and assess how they interact to produce a list of Q&A items for a text. The demo is accessible at qna.sdl.com.

1 Introduction

Automated reading comprehension is one of the current frontiers in AI and NLP research, evidenced by the frequently changing state-of-the-art among competing approaches on standard benchmark tasks (e.g. Wang et al., 2018). These systems aim to reach the standard of human performance, but they also have the potential to further enhance human reading comprehension. For instance, many demonstrations of reading comprehension involve eliciting answers to questions about a text. Meanwhile, educational research and conventional writing advice indicate that structuring information in a question-and-answer format can aid comprehension (Knight, 2010; Raphael, 1982). Accordingly, systems that present content in this format by automatically generating and answering relevant questions may help users better understand the content.

The two NLP tasks essential to this objective, question answering (QA) and question generation (QG), have received a lot of attention in recent years. Recent work has started to explore the intersection of QA and QG for the purpose of enhancing performance on one or both tasks (Sachan and Xing, 2018; Song et al., 2018; Tang et al., 2018; Yuan et al., 2017). Among application interfaces that demo these tasks, most have focused on either one or the other (Kaisser, 2008; Kumar et al., 2019). Krishna and Iyyer (2019) presented a system that integrated these tasks to simulate a pedagogical approach to human reading comprehension. In our work, we demo an end-to-end system that applies QA and QG to multi-paragraph documents for the purpose of user content understanding. The system generates a catalog of Q&A items that convey a document's content. This paper first presents some focused contributions to the individual tasks of QA and QG. In particular, we examine the challenging task of QA applied to multi-paragraph documents and show the impact of incorporating a pre-trained text encoding model into an existing approach. Additionally, we report a new set of results for QA that assesses generalizability between datasets that are typically evaluated separately. For QG, we demonstrate the benefit of data augmentation by seeding a model with automatically generated questions, which produces more fluent and answerable questions beyond a model that observes only the original human-authored data. In combining the two tasks into a single pipeline, we show that the information given by the generated Q&A items is relevant to the information humans target when formulating questions about a text.

The demo is implemented as a web application in which users can automatically generate Q&A pairs for any text they provide. The web application is available at qna.sdl.com, and our code is at github.com/roemmele/answerquest.

2 Question Answering

2.1 Model Overview

Our demo implements extractive QA, where answers to questions are extracted directly from some given reference text. State-of-the-art systems have utilized a classification objective to predict indices

[◇]Current affiliation: eBay Inc., San Jose, CA, USA

of answer spans in the text. This approach has achieved success when the reference text is limited to a single paragraph (Devlin et al., 2019). However, QA for multi-paragraph documents has proven to be more difficult. Our system addresses this challenging document-level QA task by adapting an existing method to additionally leverage a pre-trained text encoding model.

Existing work on document-level QA has proposed a pipelined approach that first applies a retrieval module to select the most relevant paragraphs from the document, and then a reader module for extracting answers from the retrieved paragraphs (Chen et al., 2017; Yang et al., 2019). During training, each of the retrieved paragraphs and the corresponding questions are observed independently. To predict answers, the model scores candidate answer spans within each of the paragraphs, ultimately predicting the one with the highest score across all paragraphs. One problem is that the candidate answer scores across paragraphs might not be comparable, since each paragraph-question pair is treated as an independent training instance. To address this issue, Clark and Gardner (2018) suggested a shared-normalization approach (which we refer to here as BIDAF SHARED-NORM) where paragraph-question pairs are still processed independently, but answer probability scores are globally normalized across the document. In their work, they selected the top-k most relevant paragraphs for a given question using a TF-IDF heuristic. They then encoded the question and these paragraphs into a neural architecture consisting of GRU layers and a Bi-Directional Attention Flow (BiDAF) mechanism (Seo et al., 2017). On top of this model is a linear layer that predicts the start and end token indices of the answer within a paragraph, using an adapted softmax function with normalization across all top-k paragraphs for the question.

Another document-level QA system, RE^{3}QA (Hu et al., 2019), incorporated the text encoding model BERT (Devlin et al., 2019). BERT has been successfully used for numerous reading comprehension tasks. In contrast to BIDAF SHARED-NORM, RE^{3}QA combined paragraph retrieval and answer prediction into a single end-to-end training process, applying BERT to both steps. Because it obtained favorable results relative to the BIDAF SHARED-NORM approach, we were curious to assess the isolated impact of BERT specifically on the answer prediction component of the pipeline. Therefore we adapted Clark and Gardner's shared-normalization approach by replacing their GRU BiDAF encoder with the BERT-BASE-UNCASED encoder. Wang et al. (2019) used a similar approach for open-domain QA, where answers are mined from the entirety of Wikipedia. We instead evaluate QA with reference to a single document, for which the impact of BERT on the shared-normalization approach has not yet been documented.

We refer to our model here as BERT SHARED-NORM. To rank paragraph relevance to a question, we rely on TF-IDF similarity. During training, we retrieved the top k=4 paragraphs. The BERT SHARED-NORM model consists of the BERT-BASE-UNCASED pre-trained model, which encodes the paragraph and question in the same manner as Devlin et al.'s paragraph-level QA model. The rest of our model is the same as BIDAF SHARED-NORM: the softmax output layer predicts the start and end answer tokens and the same shared-normalization objective function is applied during training. The model can predict that a question is 'unanswerable' by observing an index of 0 for the end token. During inference, the highest-scoring answer span across paragraphs is predicted as the answer. See Appendix A.1 for more details.

2.2 Dataset

Our QA experiments utilized the SQUAD (Rajpurkar et al., 2016) and NEWSQA (Trischler et al., 2017) datasets. SQUAD is derived from Wikipedia articles, while NEWSQA consists of CNN news articles. Both datasets were developed through crowdsourcing tasks where participants authored questions and identified their answers, resulting in text-question-answer items where each answer is a span within the text. There are two versions of SQUAD. SQUAD-1.1 contains 87,599 train and 10,570 test items. SQUAD-2.0 contains an additional 42,720 train and 1,303 test items (a total of 130,319 and 11,873, respectively), distinguished from SQUAD-1.1 by including questions that do not have answers in the text. NEWSQA contains 107,674 and 5,988 train and test items, respectively. As with SQUAD-2.0, some of these questions are unanswerable.[1]

[1]The SQUAD test items we use are actually the items from their 'dev' (development) set: rajpurkar.github.io/SQuAD-explorer. Their official test set is withheld. The other published systems we compare against also report evaluations on this dev set, so for simplicity we refer to it here as the test set. Similarly, we use the dev NEWSQA items as our held-out test set: github.com/Maluuba/newsqa.

SQuAD questions pertain to a single paragraph. Paragraphs are grouped by document and can be concatenated for document-level QA. There are on average 43 paragraphs per document. Paragraph boundaries are not explicit in the NewsQA texts, so we treated each text as a multi-paragraph document by splitting it into chunks of 300 tokens, resulting in 2.55 average paragraphs per document.

2.3 Evaluation

2.3.1 Comparison with other Systems

We first evaluated our BERT Shared-Norm model on SQuAD-1.1 for comparison with the BiDAF Shared-Norm and RE^{3}QA results reported for this dataset. We used the official SQuAD evaluation scripts provided by the website. For direct comparison with BiDAF Shared-Norm, we replicated their setting of k=15 for paragraph retrieval. Table 1 shows the results in terms of the exact match (EM) and F1 accuracy of answers. We improve upon the result for BiDAF Shared-Norm, demonstrating the beneficial impact of incorporating BERT into this approach. The BERT-based RE^{3}QA still outperforms our model, suggesting that its other components outside the BERT encoding for answer prediction additionally contribute to its success.

Model	EM	F1
BiDAF Shared-Norm	64.08	72.37
RE^{3}QA	77.90	84.81
BERT Shared-Norm	72.85	80.58

Table 1: QA results on SQuAD-1.1

2.3.2 Generalizability across Datasets

Our demo accepts any arbitrary text supplied by a user, and we ultimately aim to produce informative Q&A items for varying content domains. State-of-the-art QA systems have matched human-level performance on individual datasets like SQuAD, but it is unclear how much this performance generalizes across different datasets. As a narrow assessment of this issue, we examined the generalizability between SQuAD and NewsQA by alternatively training and evaluating BERT Shared-Norm on different combinations of these datasets.

Table 2 shows the results of this experiment. We trained three different BERT Shared-Norm models on separate datasets: SQuAD-2.0, NewsQA, and SQuAD-2.0 + NewsQA combined (which we term MegaQA). We then evaluated each of these models on the SQuAD-2.0 and NewsQA test sets. Note that the experiments in Section 2.3.1 were evaluated on SQuAD-1.1 for comparison with the other approaches. Here, we only evaluate on SQuAD-2.0, which involves additionally predicting when a question does not have an answer span in the document. For these evaluations, consistent with training, we retrieved the top k=4 paragraphs from each document for answer prediction.

Train Data	Test Data			
	SQuAD-2.0		NewsQA	
	EM	F1	EM	F1
SQuAD-2.0	71.37	74.65	40.88	48.67
NewsQA	45.85	49.88	52.68	61.26
MegaQA	70.29	73.55	53.85	62.46

Table 2: Generalizability of BERT Shared-Norm across datasets

The results reveal a generalizability problem, where the model trained on SQuAD-2.0 fails to perform as well on NewsQA and vice-versa, presumably due to their domain difference (Wikipedia vs. Newswire). However, combining the datasets with the MegaQA model generalizes well to both. Related to this, Talmor and Berant (2019) found combining multiple datasets from different domains to be advantageous for BERT-based reading comprehension models. Based on these results, the BERT Shared-Norm MegaQA model is currently integrated into our demo.

3 Question Generation

3.1 Model Overview

We follow the same paradigm of much recent work on QG, which has applied encoder-decoder (i.e. sequence-to-sequence) models to text-question pairs (Du et al., 2017; Duan et al., 2017; Scialom et al., 2019; Song et al., 2018; Zhao et al., 2018). Similar to Scialom et al., we utilize the Transformer architecture for the encoder and decoder layers of the model, and enhance the decoder with a copy mechanism. The encoder input is a single sentence and the decoder output is a question, where the input sentence contains the answer to the question. Following the standard procedure for sequence-to-sequence model training, we used the cross-entropy

of the output question tokens as the loss function. When generating questions, we use a beam size of 5. See Appendix A.2 for further details.

3.2 Dataset

We trained and evaluated the model on SQuAD and NewsQA concatenated, the same datasets used for the QA experiments. Our QG model aims to produce questions whose answers are contained in their corresponding input texts, so we only included SQuAD-1.1 items and answerable NewsQA items (this excluded 32,764 NewsQA items from the train and test sets). For each paragraph-question-answer item, we sentence-segmented the paragraph, isolated the sentence with the answer span, and inserted special tokens into the sentence (<ANSWER> and </ANSWER>) designating the start and end of the span. These answer-annotated sentences were the model inputs and the aligned questions were the target outputs. We applied Byte-Pair-Encoding (BPE) tokenization (Sennrich et al., 2016) to the inputs and targets (see Appendix A.2). We used the same train-test dataset splits as the QA experiments, allocating a small subset of training items to a validation set for hyperparameter tuning. Overall the train, validation, and test sets consisted of 160,876, 3,281, and 14,910 sentence-question pairs, respectively.

3.3 Data Augmentation Experiments

We examined three different versions of the model described in 3.1, differentiated by their training inputs. The purpose of this experiment was to assess using the output of a rule-based QG system as a means of augmenting the training data. We specifically evaluated the three configurations below:

Standard: In this model, no data augmentation was applied. We trained the model directly on the SQuAD/NewsQA items described in 3.2.

RuleMimic: This model observed only the automatically generated augmentation data, without the original data. The source of the augmentation data was the QG system by Heilman and Smith (2010)[2]. This system applies linguistic rule-based transformations (i.e. clause simplification, verb decomposition, subject-auxiliary inversion, and wh-movement) to convert a sentence into a question answered by the sentence, then scores the fluency of the question using a statistical model. Du et al. (2017) found favorable results for a neural sequence-to-sequence approach relative to this rule-based system, but we were curious about its use as a strategy for augmenting our training data. We anticipated that a neural model could learn to 'mimic' the system's generic transformation rules by observing its inputs and outputs. Thus, we applied the system to the raw paragraphs in the SQuAD/NewsQA training set, which resulted in 1,531,233 questions, each aligned with a sentence. We then followed the same steps described in 3.2 to tokenize the sentence and mark the answer span. The training set for this model consisted only of these automatically generated questions (1,500,610 train items with 30,623 used for validation), with no human-authored questions.

Augmented: This model observed both the original data seen by the Standard model and the augmentation data seen by the RuleMimic model, via a two-stage fine-tuning process. After training the RuleMimic model, we used its parameters to initialize another model, then fine-tuned this new model on the Standard model dataset. The hypothesis behind this approach is that it can simulate linguistic rules underlying question formulation, while also capturing the more abstractive features of human questions that are harder to derive using deterministic syntactic and lexical transformations.

3.4 Evaluation

Many QG systems are evaluated using BLEU or similar metrics that reward overall token overlap between generated and human-authored questions. However, Nema and Khapra (2018) argue that these metrics are ill-suited for QG. In particular, comparatively fluent questions with the same answer could have few tokens in common. Moreover, certain tokens within a question have far more impact than others on its perceived quality. They encourage alternative metrics that focus instead on the 'answerability' of questions. Guided by this, we conducted both automated and human ratings-based evaluations in order to assess the answerability of our QG output. Because our demo performs extractive QA, our evaluations focus on whether questions are answerable relative to the input text from which the question is generated.

3.4.1 Automated Evaluation

Some work has utilized automated QA as a scoring metric for QG systems, based on the rationale that a QA system's ability to predict correct answers to generated questions indicate how well the ques-

[2]Code at cs.cmu.edu/~ark/mheilman/questions

tions are formulated to elicit these answers (Duan et al., 2017; Zhang and Bansal, 2019). Following this idea, we generated questions for sentence inputs in the SQUAD/NEWSQA test set. As with the training inputs, these inputs were derived by annotating the answer span of the corresponding human-authored question for the paragraph, and isolating the sentence containing that span. We then provided each generated question and corresponding paragraph to the BERT SHARED-NORM MEGAQA model described in Section 2. The results for each QG model in terms of answer F1 accuracy are shown in Table 3, compared alongside the result for human-authored questions.

As shown, the questions generated by the RULEMIMIC model are much better at eliciting the designated answers than the STANDARD model questions, indicating that observing the rule-generated questions alone is impactful. Additionally, the AUGMENTED model generates more answerable questions than the RULEMIMIC model, showing the usefulness of combining rule-generated questions with human-authored questions as a data augmentation strategy.

Model	F1
STANDARD	0.354
RULEMIMIC	0.503
AUGMENTED	0.551
HUMAN	0.718

Table 3: Accuracy of QA system on QG output

3.4.2 Human Evaluation

Model	Rating	Answer Present
STANDARD	2.813	0.225
RULEMIMIC	2.934	0.381
AUGMENTED	3.140	0.399
HUMAN	3.776	0.793

Table 4: Human assessment of QG output

We also elicited human judgments for a subset of the same generated questions. Participants were recruited from an internal team of linguists as well as Amazon Mechanical Turk (AMT). We selected questions corresponding to 175 inputs. Table 5 in the appendix shows examples of these items. Participants read the input sentence in its paragraph context, then observed all four questions associated with the input (one generated by each of the three models plus the corresponding human-authored question). The presentation order of the questions for a given paragraph was randomized. Participants rated the fluency and answerability of questions on a scale of 1-4 based on the following statements:

1: *Question is completely ungrammatical. It's impossible to know what this question is asking.*

2: *Question is mostly grammatical, but it doesn't fully make sense. It's not clear what this question is asking.*

3: *Question is strangely worded, vague, or contains errors. However, I can make a guess about what the question is asking.*

4: *Question is clearly worded. I understand what this question is asking.*

If the participant indicated that the question was answerable by rating it a 3 or 4, they were then asked if the answer to the question was contained in the paragraph. If they indicated 'yes', they were asked to verify this by selecting all text spans in the paragraph that qualified as correct answers to the question. Based on this, we scored a question as having an 'answer present' if it was marked as being answerable and if at least one of the participant-selected answer spans was the same one the question was conditioned upon when generated (signifying that the question actually elicited the answer the model observed in the input). 41 participants assessed a total of 1,560 paragraph-question items, with each item being rated by at least two participants (see Appendix A.3 for inter-rater reliability statistics). We averaged the scores for the same questions across participants. Table 4 shows the mean ratings and answer presence for each set of generated questions including the HUMAN questions. In terms of ratings, the results follow the same pattern as the automated evaluation: the RULEMIMIC questions are rated higher than the STANDARD questions, and the AUGMENTED questions are rated higher than the RULEMIMIC questions. All sets of generated questions are rated much lower than the HUMAN questions. The models are ordered the same in terms of answer presence, though the difference between the RULEMIMIC and AUGMENTED models is slight. Overall these results again show the benefit of augmenting the training data with automatically generated questions. Accordingly, our demo currently runs the AUGMENTED model.

4 Generating Q&A Pairs

We combined our best-performing QG and QA models into a system that takes a text as input and returns a list of Q&A pairs. Our web demo illustrates this system (see Appendix A.4 for details).

For our evaluations in Section 3, the QG models observed annotated answer spans upon which the generated questions were conditioned. However, these annotations are obviously not available by default for any arbitrary text. Consequently, after splitting the text into sentences, we automatically identify syntactic chunks and named entities as candidate answers to questions (see Appendix A.5 for details). For each candidate answer in a sentence, we produce an input consisting of that sentence annotated with that span. We also include sentences with no answer annotations as inputs, since they are not formally required by the model. We provide all inputs for a given sentence to the AUGMENTED QG model to get a list of questions that can be passed to the QA component. Note that even though some of the questions are already associated with annotated answers, we still apply QA as an additional means of verifying their answerability, and defer to the QA-predicted answer. To prepare the QA inputs, for each sentence-question item, we extend the sentence to include the sentences immediately preceding and following it, so each question becomes aligned with a 3-sentence passage. This enables the QA system to possibly retrieve additional context beyond the sentence that it may deem as part of the answer span. We provide these passage-question pairs to the BERT SHARED-NORM MEGAQA model, then retain output items for which answers are found. We reduce the redundancy of items by filtering those with duplicate questions or answers, as well as items where the question and answer concatenated share 60% or more of the same tokens. In these cases, we only retain the item with the highest QG probability.

4.1 Human Evaluation

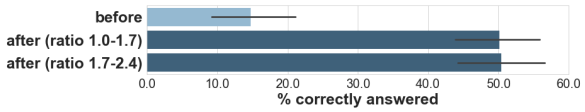

Figure 1: Human accuracy on target questions before and after observing generated Q&A pairs

We used our system to generate Q&A pairs for ten texts from the SQUAD test set. Appendix Table 6 shows an example of a generated Q&A list for one text. We conducted an evaluation of the informativeness of these pairs with 38 AMT participants. In the first stage of the evaluation, participants were shown only the title of one text (e.g. "Tesla") and the human-authored SQUAD questions (no answers) corresponding to the text. Without referencing any material, they were asked to answer these target questions or respond with "X" for questions they couldn't answer. Because no generated Q&A pairs were shown to participants during this stage, the accuracy of their answers indicated their prior mental knowledge of the information in the text. In the second stage, the generated Q&A pairs for the same text were revealed to them and they answered the same target questions again. Participants never observed the original text itself. The logic of this design is that the more questions people could correctly answer in the second stage relative to the first, the more informative the generated Q&A list could be deemed. The ratio of generated Q&A pairs to target questions per text varied from 1 to 2.4 (e.g. ratio = 2 for a text with 30 generated pairs and 15 target questions). Figure 1 shows the percentage of target questions participants answered correctly before and after observing the Q&A list, grouped by ratio. The overall difference in these conditions (14.74% vs. 50.26%) shows that the generated items were partially informative for answering the target questions, signifying that the system does highlight some of the same content people ask questions about. However, accuracy did not markedly improve as participants saw more items (50.18% for the lower ratio vs. 50.38% for the higher ratio), suggesting that the information coverage of the items could be improved. See Appendix A.6 for more details about this evaluation.

5 Conclusion and Future Work

In this paper, we present a system that automatically produces Q&A pairs for multi-paragraph documents. We report some novel experiments for QA and QG that motivate techniques for improving these tasks. We show that combining these components can produce informative Q&A items. Our future work will focus on more advanced modeling of information structure in documents. For example, the ideal design of Q&A items varies by domain (e.g. news stories vs. financial reports vs. opinion editorials), and items should target the content readers find most substantial in each domain.

References

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada. Association for Computational Linguistics.

Christopher Clark and Matt Gardner. 2018. Simple and effective multi-paragraph reading comprehension. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 845–855, Melbourne, Australia. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Nan Duan, Duyu Tang, Peng Chen, and Ming Zhou. 2017. Question generation for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 866–874, Copenhagen, Denmark. Association for Computational Linguistics.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Minghao Hu, Yuxing Peng, Zhen Huang, and Dongsheng Li. 2019. Retrieve, read, rerank: Towards end-to-end multi-document reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2285–2295, Florence, Italy. Association for Computational Linguistics.

Michael Kaisser. 2008. The QuALiM question answering demo: Supplementing answers with paragraphs drawn from Wikipedia. In *Proceedings of the ACL-08: HLT Demo Session*, pages 32–35, Columbus, Ohio. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Chris Knight. 2010. Question & answer article template.

Kalpesh Krishna and Mohit Iyyer. 2019. Generating question-answer hierarchies. In *Association for Computational Linguistics*.

Vishwajeet Kumar, Sivaanandh Muneeswaran, Ganesh Ramakrishnan, and Yuan-Fang Li. 2019. Paraqg: A system for generating questions and answers from paragraphs. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, pages 175–180.

Preksha Nema and Mitesh M Khapra. 2018. Towards a better metric for evaluating question generation systems. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3950–3959.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Taffy E Raphael. 1982. Improving question-answering performance through instruction. *Reading education report; no. 32*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Mrinmaya Sachan and Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 629–640, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Scialom, Benjamin Piwowarski, and Jacopo Staiano. 2019. Self-attention architectures for answer-agnostic neural question generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6027–6032.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*.

Linfeng Song, Zhiguo Wang, Wael Hamza, Yue Zhang, and Daniel Gildea. 2018. Leveraging context information for natural question generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 569–574, New Orleans, Louisiana. Association for Computational Linguistics.

Alon Talmor and Jonathan Berant. 2019. MultiQA: An empirical investigation of generalization and transfer in reading comprehension. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4911–4921, Florence, Italy. Association for Computational Linguistics.

Duyu Tang, Nan Duan, Zhao Yan, Zhirui Zhang, Yibo Sun, Shujie Liu, Yuanhua Lv, and Ming Zhou. 2018. Learning to collaborate for question answering and asking. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1564–1574, New Orleans, Louisiana. Association for Computational Linguistics.

Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. 2017. NewsQA: A machine comprehension dataset. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 191–200, Vancouver, Canada. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. 2019. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5881–5885.

Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. 2019. End-to-end open-domain question answering with BERTserini. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 72–77, Minneapolis, Minnesota. Association for Computational Linguistics.

Xingdi Yuan, Tong Wang, Caglar Gulcehre, Alessandro Sordoni, Philip Bachman, Saizheng Zhang, Sandeep Subramanian, and Adam Trischler. 2017. Machine comprehension by text-to-text neural question generation. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, pages 15–25, Vancouver, Canada. Association for Computational Linguistics.

Shiyue Zhang and Mohit Bansal. 2019. Addressing semantic drift in question generation for semi-supervised question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509.

Yao Zhao, Xiaochuan Ni, Yuanyuan Ding, and Qifa Ke. 2018. Paragraph-level neural question generation with maxout pointer and gated self-attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3901–3910, Brussels, Belgium. Association for Computational Linguistics.

A Appendices

A.1 QA Model Details

The TF-IDF method for ranking paragraph relevance to the question specifically uses the BM-25 ranker[3] (Robertson et al., 2009). We implemented the QA model in PyTorch using the HuggingFace Transformers library[4]. As described in Section 2.1, we use the pre-trained BERT-BASE-UNCASED model, which has 12 layers, 768 nodes per layer, 12 heads per layer, and 110M parameters overall. The maximum sequence length for BERT-BASE-UNCASED is 384 tokens (including both paragraph and question tokens combined), so we truncated paragraphs when this total was exceeded. The output layer consists of a 384 x 2 matrix whose dimensions correspond to token indices for the start and end of the answer span. We trained the model in parallel across 4 Nvidia Tesla V100 GPUs with a paragraph-question batch size of 48 with gradient accumulation at step 1 (12 paragraph-question pairs per GPU, which was the maximum size a single V100 GPU could accommodate). Following Devlin et al.'s BERT-based fine-tuning procedure for paragraph-level QA, the model was trained for 3 epochs and a learning rate of 3e-5 using Adam optimization.

A.2 QG Model Details

We used OpenNMT-py[5] (Klein et al., 2017) for implementation of the QG model. For BPE tokenization, we use the OpenAI GPT-2 tokenizer implemented by the HuggingFace transformers library cited above. The vocabulary included all tokens observed in the training data. The Transformer encoder and decoder each consist of 4 layers with 2048 nodes and 8 heads each. We include position encodings on the token embeddings and a copy attention layer in the decoder. We used a training batch size of 4096 tokens, normalizing gradients over tokens and computing gradients based on 4 batches. We trained for a maximum of 100,000 steps and validated every 200 steps, with early stopping after one round of no improvement in validation loss. We applied the other hyperparameter settings recommended for training transformer sequence-to-sequence models on the OpenNMT-py website[6]. This included Adam optimization with $\beta_1 = 0.998$, gradient re-normalization for norms exceeding 0, Glorot uniform parameter initialization, 0.1 dropout probability, noam decay, 8000 warmup steps for decay, learning rate = 2, and label smoothing $\epsilon = 0$.

A.3 QG Evaluation Details

The sentence inputs for the evaluated questions were randomly sampled after filtering for those inside paragraphs longer than 500 characters, to ensure participants could efficiently complete the evaluation. AMT workers were paid $7 for their participation in this evaluation, with the expected time commitment of about 35 minutes.

The Cohen's kappa inter-rater agreement on the fluency/answerability ratings of 1-4 was 0.422, indicating moderate agreement. The kappa for answer presence in the paragraph was 0.465, also indicating moderate agreement.

A.4 System Implementation Details

The system UI is implemented using React JS with Bootstrap CSS for styling. Figure 2 shows a screenshot of the interface. The QA and QG functionalities run as web services implemented using Flask.

As an additional feature of the UI, users have the option to obtain answers to their own custom questions. They supply the question via a text box. The QA system receives the entire document text as input along with the question. We enforce paragraph boundaries by splitting the document into non-overlapping paragraphs of 300 tokens, and then apply the BERT SHARED-NORM MEGAQA model with top k=4 for paragraph retrieval[7]. If the model predicts the answer is not in the text, the user sees a message indicating this.

A.5 Candidate Answers for QG

We use the spaCy[8] library to extract all named entities and noun chunks. Additionally, we extract all dependency parse subtrees whose head is labeled as one of the following: clausal complement (xcomp), attribute (attr), prepositional modified (prep), object (obj), indirect object (iobj), flat multiword expression (flat), fixed multiword expression (fixed), clausal subject (csubj), clausal

[3] pypi.org/project/rank-bm25

[4] huggingface.co/transformers

[5] github.com/OpenNMT/OpenNMT-py

[6] opennmt.net/OpenNMT-py/FAQ.html#how-do-i-use-the-transformer-model

[7] Section 2.3.1 reported the result for k=15, but k=4 performs only slightly lower (71.21 EM and 78.89 F1 vs. 72.85 and 80.58, respectively) with significantly higher efficiency.

[8] spacy.io

complement (ccomp), adjectival clause (acl), and conjunct (conj). All extracted chunks are annotated as answer spans.

A.6 Q&A List Evaluation

We truncated each of the ten SQuAD documents to its first three paragraphs. There were on average 334.5 tokens per truncated document. For each document we selected all SQuAD questions corresponding to the first three paragraphs as the list of target questions participants were prompted to answer. There were on average 16.2 target questions per truncated document. We provided the truncated document to the system to generate a list of Q&A items. As described in Section 4.1, the ratio of generated Q&A items per target questions varied from 1 to 2.4 with an average of 1.66, resulting in an overall average of 26.3 generated items per document.

Each of the 38 AMT participants answered the target questions for a single document, so approximately four participants answered each unique list of target questions. They were paid $8 for their participation, with the expected time commitment of around 30 minutes. The instructions emphasized that they should not use any external information to answer the questions other than the reference Q&A list (which was only used when participants answered the questions for the second time). They were told their participation would not be judged based on the number of questions correctly answered. Participants were not informed that the reference Q&A items were automatically generated.

Because all answers were provided as free text and there could be some token variation in correct answers for the same question (e.g. "Parliament of the United Kingdom" vs. "UK Parliament"), we used a fuzzy metric for judging answers as correct. We counted a participant answer as correct if it had at least one token in common with the answer given in the SQuAD dataset. This is a permissive threshold that can yield false positives (e.g. "300 years" vs. "500 years"), but because it was consistently applied across both stages of the evaluation (i.e. before and after observing the Q&A list), we deemed it sufficient for quantifying the relative impact of the generated items in the 'after' condition.

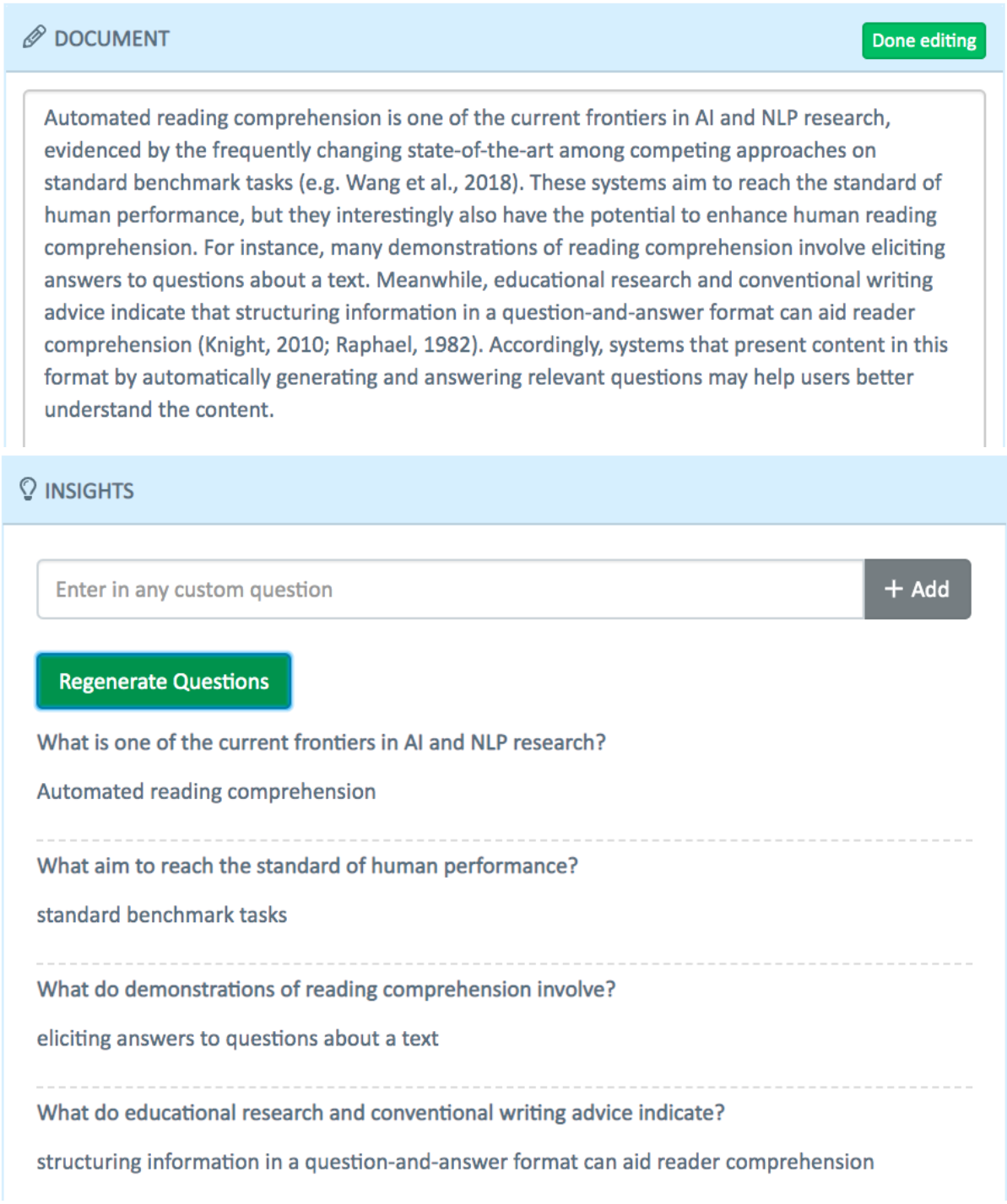

Figure 2: Screenshot of UI with generated Q&A pairs for the displayed document

Input Sentence	Model	Output Question
By far the most famous work of Norman art is the <ANSWER> Bayeux Tapestry </ANSWER>, which is not a tapestry but a work of embroidery.	STANDARD	What is the name of the work of Norman art?
	RULEMIMIC	What is not a tapestry but a work of embroidery?
	AUGMENTED	What is the most famous work of Norman art?
	HUMAN	What is Norman art's most well known piece?
Later in life, Tesla made claims concerning a "teleforce" weapon after studying the <ANSWER> Van de Graaff generator </ANSWER>.	STANDARD	What was the name of the weapon that studying the Teslaforce?
	RULEMIMIC	What did Tesla make claims concerning a "teleforce" weapon after studying?
	AUGMENTED	What did Tesla study?
	HUMAN	What was he studying that gave him the teleforce weapon idea?
The Super Bowl 50 Host Committee has vowed to be "the most giving Super Bowl ever", and will dedicate <ANSWER> 25 percent </ANSWER> of all money it raises for philanthropic causes in the Bay Area.	STANDARD	How much of the Super Bowl raises?
	RULEMIMIC	What will the Super Bowl 50 Host Committee dedicate it raises for philanthropic causes in the Bay Area?
	AUGMENTED	What is the Super Bowl 50 Host Committee?
	HUMAN	How many will the host committee dedicate to local charities?
In 1899, John Jacob Astor IV invested $100,000 for Tesla to further <ANSWER> develop and produce a new lighting system </ANSWER>.	STANDARD	What did Jacob Astor IV do?
	RULEMIMIC	What did John Jacob Astor IV invest $100,000 for in 1899?
	AUGMENTED	Why did Jacob Astor IV invest $100,000?
	HUMAN	What did Astor expect the money be used for?
Most influential among these was the definition of Turing machines by Alan Turing in <ANSWER> 1936 </ANSWER>, which turned out to be a very robust and flexible simplification of a computer.	STANDARD	When was the definition of the definition of Turing?
	RULEMIMIC	When turned out to be a very robust and flexible simplification of a computer?
	AUGMENTED	When did Alan Turing write machines?
	HUMAN	In what year was the Alan Turing's definitional model of a computing device received?
In addition to the <ANSWER> Vince Lombardi </ANSWER> Trophy that all Super Bowl champions receive, the winner of Super Bowl 50 will also receive a large, 18-karat gold-plated "50".	STANDARD	What is the name of the Super Bowl?
	RULEMIMIC	Who will the winner of Super Bowl 50 also receive a large in addition to the Vince Lombardi Trophy that all Super Bowl champions receive?
	AUGMENTED	Who wrote the Super Bowl 50?
	HUMAN	Who is the trophy given to the Super Bowl champion named for?
In 1874, Tesla evaded being drafted into the Austro-Hungarian Army in Smiljan by running away to <ANSWER> Tomingaj </ANSWER>, near Gračac.	STANDARD	What was the name of Tesla's Army in 1874?
	RULEMIMIC	Who was near Gračac?
	AUGMENTED	Where did Tesla travel to?
	HUMAN	Where did Tesla run to avoid the army draft?

Table 5: Examples of questions produced by each evaluated QG model for the given input sentences

Q: What is separate from the combustion products? A: working fluid	Q: Where was the water supply for driving waterels? A: factories
Q: What is solar power? A: Non-combustion heat sources	Q: What did the mine provide? A: water supply
Q: What is the ideal thermodynamic cycle used for? A: to analyze this process	Q: Where was it employed? A: draining mine workings
Q: What is heated and transforms into steam? A: water	Q: Where was the storage reservoir? A: above the wheel
Q: Why is mechanical work done? A: When expanded through pistons or turbines	Q: What was passed over the wheel? A: Water
Q: What is then condensed and pumped back into the boiler? A: reduced-pressure steam	Q: When was the first railway journey? A: 21 February 1804
Q: Who invented the first commercially true engine? A: Thomas Newcomen	Q: Where was the train? A: along the tramway from the Pen-y-darren ironworks, near Merthyr Tydfil to Abercynon in south Wales
Q: What could generate power? A: atmospheric engine	Q: What was built by Richard Trevithick? A: The first full-scale working railway steam locomotive
Q: Who proposed the piston pump? A: Papin	Q: The design incorporated a number of what? A: important innovations that included using high-pressure steam which reduced the weight of the engine and increased its efficiency
Q: What happened to Newcomen's engine? A: relatively inefficient	Q: What did England become the leading centre for? A: experimentation and development of steam locomotives
Q: What was the engine used for? A: pumping water	Q: Where was the railways colliery? A: north-east England
Q: What was the vacuum worked by? A: condensing steam under a piston within a cylinder	Q: Who visited the Newcastle area in 1804? A: Trevithick
Q: What was the reason for draining waterelswheels? A: providing a reusable water supply	

Table 6: Generated Q&A list for the first three paragraphs of the SQuAD document titled "Steam engine"