WikiFlash: Generating Flashcards from Wikipedia Articles

Yuang Cheng[1], Yue Ding[1], Sebastien Foucher[2], Damián Pascual[2], Oliver Richter[2], Martin Volk[1], and Roger Wattenhofer[2] ⋆

[1] University of Zurich, Switzerland
{yuang.cheng, yue.ding}@uzh.ch, volk@cl.uzh.ch
[2] ETH Zurich, Switzerland
{sfoucher, dpascual, orichter, wattenhofer}@ethz.ch

Abstract. Flashcards, or any sort of question-answer pairs, are a fundamental tool in education. However, the creation of question-answer pairs is a tedious job which often defers independent learners from properly studying a topic. We seek to provide a tool to automatically generate flashcards from Wikipedia articles to make independent education more attractive to a broader audience. We investigate different state-of-the-art natural language processing models and propose a pipeline to generate flashcards with different levels of detail from any given article. We evaluate the proposed pipeline based on its computing time and the number of generated and filtered questions, given the proposed filtering method. In a user study, we find that the generated flashcards are evaluated as helpful. Further, users evaluated the quality of human created flashcards that are available open source as comparable to or only slightly better than the automatically generated cards. [1]

Keywords: Question-Answer Extraction · Personalized Education · Natural Language Processing

1 Introduction

The recent development of artificial intelligence make available a new set of tools that can be exploited to advance the field of personalized education. In the last years, we have seen how, thanks to new deep learning methods, machines have attained super-human performance in a large number of language-related tasks [33]. These methods can accelerate the development of personalized education by automatically generating instructional material. Generating instructional materials manually is a costly task that requires instructors to select and cure large amounts of information. With a growing internet, an ever-increasing (and overwhelming) amount of information and data is available. However, it is challenging for a person to learn in a systematic manner from this information. To

⋆ Authors in alphabetical order
[1] Our application is available at: flashcard.ethz.ch

improve human learning, it is necessary to structure the information into instructional materials that select the most relevant points and guide learning. Automatically generating these materials can widely accelerate human learning while giving each person the freedom to learn any arbitrary topic of her interest.

A well-known and effective format for instructional materials are flashcards [30]. Flashcards are small cards (physical or virtual) with a question written on the front face and the answer to that question written on the back face. Flashcards stimulate learning by hiding the answer that the student is trying to learn. A big advantage of flashcards is that they are topic-independent, i.e., flashcards can be used to learn anything: languages, history, mathematics... Nevertheless, a large number of flashcards is necessary to cover a given topic or subtopic, and preparing good flashcards requires good summarization skills, all of which makes the process of manually producing flashcards challenging and time consuming.

In this work, we present a system for automatically generating flashcards about any arbitrary topic. We leverage recent advances in language processing, in particular transformer-based models [32], to extract questions and answers from input text. We implement our system as a web application that takes as input the title of a Wikipedia article and outputs flashcards for that article. We evaluate the application, profiling generation time and the number of flashcards produced. Furthermore, we run a user study to assess the quality of our automatically generated cards in comparison to human-created cards. The results show that the quality of our automatically generated cards is similar to the quality of cards generated by humans. Our system has the flexibility of generating instructional materials (in the form of flashcards) for any topic a student may be interested in, beyond standard curricula. We build our system as a web application that serves as both, a proof-of-concept of how current technologies allow automatic generation of materials for learning, as well as a first step towards a completely functional tool to enhance learning anywhere and about anything.

2 Related Work

Automatic question generation for educational purposes is a growing research area with many works focusing on assessment and template based question generation [13]. In a recent trend, data driven approaches that use neural networks became more prominent in many natural language processing tasks, including question generation [21]. These data driven approaches might struggle to extract questions that require several steps of reasoning as in the LearningQ dataset [6]. However, for flashcard generation, simple factoid questions are often preferred. We therefore focus on models that perform well on the Wikipedia based SQuADv1 dataset [24], which was originally developed for question answering models but can be re-purposed for context based question generation.

On this dataset, transformer based approaches for question generation [12, 5, 19, 7, 3] are currently preforming best in terms of n-gram similarity metrics such as ROUGE-L [17]. This is likely due to the fact that these models benefit from large scale unsupervised pretraining. Our implementation is based on the publicly available code of [22], which follows ideas from [5, 19] and [1] and achieves results not far behind the state-of-the-art [3].

As a pre-processing step, text summarization can be used to reduce the text from which questions are to be generated. Automatically summarizing text is the focus of a large body of research and a number of datasets exist that are used to benchmark progress [11, 28, 26]. There are two types of summarization: extractive [36], the summary consist of sentences copied from the original text; and abstractive [10], the sentences do not coincide with the original text but the meaning does. Abstractive summarization is both, more natural and harder. Recently proposed models [34, 35, 16, 23] have achieved new state-of-the-art results as measured by ROUGE-L score. Here, we leverage this progress and use abstractive summarization for content selection before question generation.

The general idea of filtering questions in a post-processing step has been explored in different settings [14, 4, 18, 20, 1]. Using a question-answering system to filter questions where the answers do not align, was proposed by [1] to create a synthetic data corpus for pretraining a question-answering model. We use this approach in our system with slight adjustments. Compared to their approach of filtering all questions where answers do not align, we relax the filtering by allowing for questions where the extracted answers and the answers produced by the question-answer model yield a sufficient overlap.

The main contribution of this work is an end-to-end application that allows for flashcard generation based on a Wikipedia article freely chosen by the users. Our work thereby differs from the work of [8] that created a fixed size corpus for scientific investigation. Also, despite the existence of many applications that allow for the design and/or studying of flashcards, we only encountered one working application which allows for automated flashcard creation [9]. This application uses a key-phrase based system for the creation of flashcards in the biological and medical domain. In contrast, our approach does not rely on key phrases and is therefore applicable to a much wider range of topics.

3 Method

Generating meaningful flashcards from an arbitrary piece of text is not a trivial problem. Currently, there does not exist a single model that can alone perform this task. We therefore divide the flashcard generation process into four subtasks that cover more general and well-studied problems that can be individually addressed by state-of-the-art models. In particular, we build a pipeline consisting of four stages: summarization, answer identification, question generation and question answering. Figure 1 shows a depiction of this pipeline.

Summarization By definition, a summary contains the most relevant information for a general understanding of the corresponding text. Thus, generating flashcards from a summary reduces the level of detail in the resulting flashcards, in comparison to using the original text as input. A summarization stage gives the user the freedom of deciding between two levels of detail for the information

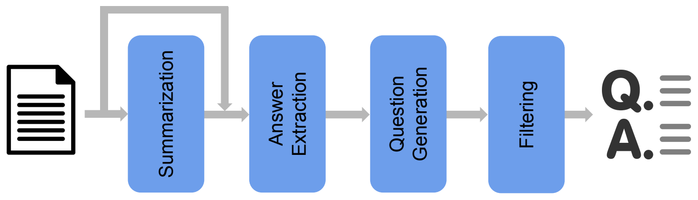


Fig. 1. Pipeline of the flashcard generation system. The summarization step is optional.

contained in the flashcards. If more detailed flashcards are preferred, the summarization step is skipped and the input text is passed directly to the next step of the pipeline. Otherwise, a summary is generated from the input text and fed into the next stage.

Answer extraction After the optional summarization step, we proceed to generate flashcards by identifying potential answers in the text. To this end, we use a model for answer extraction, which receives as input a piece of text and finds words or groups of words that can be answers to questions. These answers, together with the original text, are passed as input to the next stage.

Question generation In this stage we use an answer-aware question generation model to generate answer specific-questions. This way, the output of this stage is the set of question-answer tuples that we need for flashcards. However, the question-answer tuples generated at this point tend to include some questions that either make no sense or are incorrect. Therefore, we include a final step in our pipeline to filter out unusable questions.

Filtering To filter out erroneous questions, we use a model for question answering. For each question-answer tuple we provide this model with the question and the paragraph where the answer can be found. If the answer provided by the question-answering model overlaps enough with the answer from which the question was generated, then the question-answer tuple is accepted, otherwise it is discarded.

4 Implementation

We implement our flashcard generation pipeline as a web application. The interface of our application is simple and intuitive. The main screen displays the existing flashcard decks and their status, e.g., “Generating”, “Complete”, as well as a button to add a new deck. When clicking on this button the user is prompted a screen where they should provide a title of a Wikipedia article they want flashcards from. Given this title, the application suggests a number of actual Wikipedia articles; this way, if the article name is redundant, i.e., there are more than one article with the same name, disambiguation results are suggested. Once the Wikipedia article is selected, the user can define which sections of the article they want flashcards from, and whether summarization should be applied or not. Finally, the user can choose the name of the generated deck, by default the name of the corresponding Wikipedia article is given.

Regarding, the implementation of each stage of our system, we use the following models to build the pipeline:

Summarization We use DistilBART for summarization [29, 16], pre-trained on the CNN/DailyMail summarization dataset [11]. The maximum input length of this model is 1024 tokens, which is less than a long Wikipedia article. To circumvent this issue, our summaries are generated paragraph-wise.

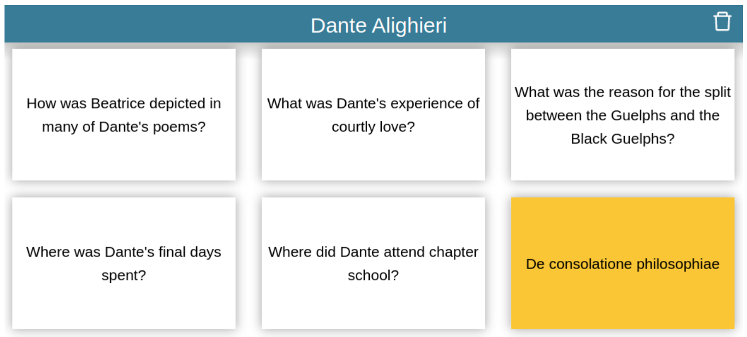
Answer extraction We use T5 fine-tuned on the SQuADv1 dataset for answer extraction [22, 23]. At inference time, for each paragraph we highlight one sentence at a time and feed it together with the rest of the paragraph to the model. The model extracts answers from the highlighted sentence leveraging the additional context information contained in the rest of the paragraph. To stay within the admitted input size of the model, we clip the paragraphs to 512 tokens.

Question generation Here we use T5 fine-tuned on the SQuADv1 dataset for answer-aware question generation [22, 23]. For each extracted answer, we append the corresponding paragraph as context and feed it to the model. Again, to not exceed the maximum input size we clip the input to a length of 512 tokens.

Filtering For filtering we use DistilBERT fine-tuned using a second step of knowledge distillation on the SQuADv1 dataset for question answering [31, 27]. Similar to the previous steps, we feed the model at inference time with each of the generated questions together with their corresponding paragraphs. We calculate an overlap score between the answer obtained in the answer extraction step and the answer produced by this question-answering model. The overlap score we calculate here is the ratio of identical bigrams over the total number of bigrams. Questions with an overlap score below 0.75 are discarded. Duplicates and questions whose answer is the title of the article are also discarded.

For each of the stages of our system, many different models exist in the literature. We selected each specific model based on their fitness to the task (i.e., models that are trained on Wikipedia based data-sets) as well as their availability as open source implementation.

Once a deck is generated, the user can interact with the flahscards in two different ways: 1) in grid view, 2) in study mode. In grid view, all the cards of the deck are displayed in a grid, showing the question faces. When the mouse is placed over the question, the card flips showing the answer to that question. In study mode, one question is presented at a time and after clicking the answer is revealed. In this mode the user can edit the card as well as give feedback



How was Beatrice depicted in many of Dante's poems?

What was Dante's experience of courtly love?

What was the reason for the split between the Guelphs and the Black Guelphs?

Where was Dante's final days spent?

Where did Dante attend chapter school?

De consolatione philosophiae

Fig. 2. Grid view for the topic *Dante Alighieri*. The white squares contain the questions and the yellow square is the answer to *"What Boethius work did Dante read?"*.

about it, choosing from four options: *"The answer does not fit the question"*, *"The answer is wrong"*, *"The answer is trivial"* and *"Other"* (where the user can input their own text). Finally, after the answer is shown, the user is asked to label the difficulty of the card choosing from five possibilities, from *"Too easy"* to *"Too hard"* or alternatively, as *"Not interested"*. These feature may be used in future work for designing algorithms that decide the optimal card ordering for human learning. Figure 2 shows an example of generated cards displayed in grid view.

To make our cards usable beyond our web application, we provide the option of exporting the generated cards as text file that can be imported into Anki. Anki is a popular framework for flashcard-based learning with a large community of users that share their own flashcard decks as well as a number of commercial applications for smart-phones and web to help learning. This way, our generated flashcards are compatible with existing commercial applications and the users can choose the learning platform they prefer.

5 Evaluation

In this section we evaluate objective parameters of our flashcard generation pipeline, such as compute time or number of questions generated. Conversely, in Section 6 we evaluate the subjective quality of the generated cards through a user study. We divide our objective evaluation in two parts: 1) summarization step and 2) question generation and filtering step.

5.1 Summarization

Since we do not have reference summaries of the pieces of text that we are aiming to summarize, we cannot rely on the ROUGE score, which is the most common metric for summary quality. Instead, we calculate two values, similarity and error rate, that do not require a reference summary. The similarity score gives us a

Table 1. Comparison of T5, BART and DistilBART summarization.

Model	Similarity	Error Rate
T5	0.912	0.129
BART	0.947	0.057
DistilBART	0.937	0.052

notion of how faithful the summary is to the original text, while the error rate quantifies the linguistic correctness of the summary.

To calculate the similarity score we use Sentence-BERT [25] to compute an embedding of each sentence in the original text and in the summary. Then, we calculate a context vector for the original text by adding up all of its sentence embeddings. We do the same for the summarized text. This results in two context vectors, one representing the original text and one representing the summary. Our similarity score is the cosine similarity of these two vectors. The error rate is the percentage of erroneous tokens. To calculate it, we determine the number of wrong tokens using *LanguageTool* [15] and divide this number by the total number of tokens. If a sentence has no end-of-sentence token, it is considered incomplete and an error is added to the count.

To determine which model to use in the summarization step, we compare three state-of-the-art models: T5, BART, and DistilBART (the distilled version of BART). In Table 1 we compare the models in terms of similarity and error rate scores over the introduction of 256 Wikipedia articles. These articles were randomly selected based on the requirement that their introductions have more than 200 tokens. BART presents the highest similarity score and DistilBART the lowest error rate. This result is in line with the fact that BART obtains higher ROUGE score than T5 in summarization benchmarks such as CNN/DailyMail [23, 16].

Since we are implementing our system as a web application, we need to consider computation time: to improve user experience we are interested in reducing as much as possible the time needed for the system to generate the cards. Using the same set of 256 Wikipedia articles we calculate the average time it takes for BART and DistilBART to summarize the introductions, when running on a 24GB Nvidia Titan RTX GPU. While BART needs on average 6.1 seconds per article, DistilBART requires only 3.7 seconds, i.e., DistilBART is 1.64 faster. While the absolute difference in computation time might seem small, we note that the computation time scales linearly with the article length, as articles are fed one paragraph at a time. We therefore choose DistilBART for the summarization step of our system, as the total speed up is significant and obtains a similarity score and an error rate comparable to BART.

5.2 Question Generation and Filtering

We study the performance of the question generation and filtering stage of our pipeline in terms of computing time and questions generated. We use 1024 randomly selected articles from Wikipedia with more than 200 tokens and analyse

Table 2. Average number of questions generated and kept after filtering.

		Time	Number of Questions	Questions after Filter
Original	Per section	14.3 s	10.4	8.7
	Per article	240.5 s	178.4	148.2
Summary	Per section	9.3 s	8.6	7.2
	Per article	151.2 s	144.0	120.5

the number of questions generated. In Table 2, we report the average number of flashcards generated and the average number of flashcards kept after the filtering stage.

We see that even after applying our filtering step the number of questions kept, i.e., questions that meet a minimal quality requirement, is relatively large. In particular, generating 148.2 questions on average for a Wikipedia article implies that a student can access a significant amount of information from the cards. Furthermore, from the results we see that summarization helps in reducing the number of questions that are discarded.

From the results presented in this section, we cannot assess the quality and usefulness of the generated cards, since this is a feature that depends on human perception. However, we can visually examine some examples of flashcards to have a notion of what kind of question-answer pairs our model generates. Table 3 shows the first four question-answer tuples generated for the article *Animal Farm* (novel by George Orwell) for the summary of the introduction. From the examples we see that generally, the generated cards are grammatically correct and contain meaningful information. However, to evaluate flashcard quality in a more rigorous manner, in the next section we conduct a user study.

Table 3. Flashcards from the Wikipedia article on *Animal Farm* by George Orwell.

Question	Answer
Who did George Orwell write a letter to about Animal Farm?	Yvonne Davet
Who do the farm animals rebel against?	Human farmer
When was Animal Farm written?	Between November 1943 and February 1944
What are two other variations of the title of Animal Farm?	A Satire and A Contemporary Satire

6 User study

Given the strong perceptual component of flashcards, the best way of evaluating the quality of automatically generated cards is with a user study. In this study, we are interested in determining three aspects: usefulness for learning, linguistic

Table 4. Questions in the user study

Question	Scale
1) *Is this card helpful for people who are studying this topic?*	$0-3$
2) *The text on the card makes sense to me.*	$0-3$
3) *Is the answer to this question correct?*	$0-1$

comprehensibility and content correctness. In our user study, we ask about this three aspects and define a four-point scale for helpfulness and comprehensibility (strongly disagree, disagree, agree and strongly agree), and a binary scale for perceived content correctness ("I think the answer is incorrect", "I think the answer is correct/I do not know"). Table 4 displays the detail of the questions asked in the study. During the study, the user is shown one card at a time and has to answer the three questions before the next card is displayed.

The study consisted of 50 cards, from which 25 are generated by our automatic flashcard generation system, and the other 25 are created by humans. We obtain the human-created cards from flashcard decks that are freely available online in Anki format [2]. At the beginning of the study, the user can choose between two topics, History and Geography; this gives the user the possibility of deciding the topic she is most familiar with, or interested in. We chose History and Geography as representative topics since they consist of factual knowledge, which is often studied with flashcards. The human-created cards for History were taken from decks with titles: "Christianity" and "French Revolution", while our cards were generated from the Wikipedia article "French Revolution" and the history section of the article "Germany". For Geography, the topics were "India", "Physical Geography" and "General Geography" for the human-created cards; and our cards were generated from the article "Atmosphere", and the geography section of the articles "India" and "China". For each category, we randomly chose 25 cards from the generated cards and mixed them with 25 randomly chosen cards from the human-created decks. The origin of the flashcards, i.e., whether they are automatically generated or human created, was not revealed to the participants. 50 participants from Amazon Mechanical Turks took part in the study. Data from participants which completed the study in less than $1,000$ seconds was discarded. From the remaining, 21 participants selected the category history and 27 geography. Figure 3 show the results of our user study. The maximum score for helpfulness and comprehensibility is 3 and minimum is 0; and for correctness maximum is 1 and minimum is 0.

The results show that in the case of geography there is no statistically meaningful difference between human-created and our cards for either of the three aspects. For history, the difference for helpfulness and comprehensibility is statistically significant ($p < 0.01$), with human cards being marginally better than our cards. Neither category revealed a statistically significant difference in perceived correctness. Upon further investigation we found that the difference in the history category is mainly due to three automatically generated flashcards

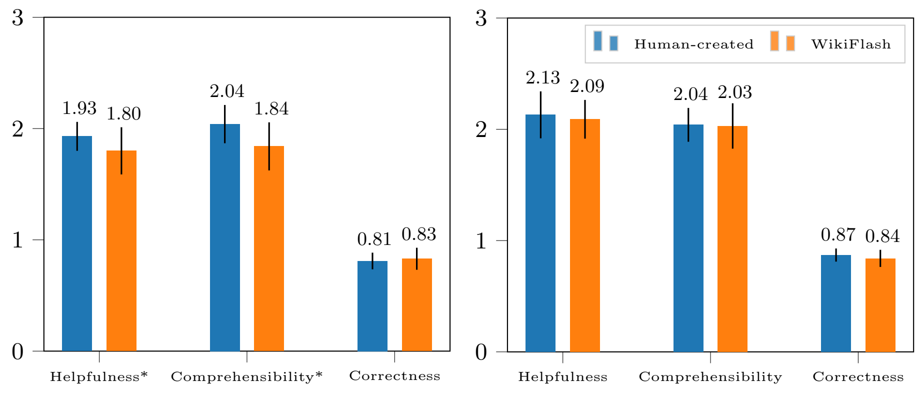


Fig. 3. Results of the user study. **Left:** Category history. Statistically significant differences ($p < 0.01$) are marked with an asterisk. **Right:** Category geography. The differences are not statistically significant.

which are too ambiguous. We intend to improve our generation and filtering procedure in future work based on this insight.

Overall, this study demonstrates that the quality of our automatically generated cards is close to the quality of cards created by humans. This result validates our system and evidences its potential for enhancing personalized learning.

7 Discussion

In this work, we have presented a system for flashcard generation from raw text. Our system builds on recent advances in natural language processing, in particular on summarization, answer extraction, question generation and question answering. We thereby base our work on recent ideas on combining different models for question-answer generation and filtering. We have implemented our system as a web application that generates flashcards from Wikipedia articles with four different levels of detail. Our user study shows that the quality of the cards generated by our application is comparable, or only slightly worse, than human-created flashcards. Our work makes available a valuable tool for personalized education. By speeding up and automatizing flashcard generation, we give students the flexibility to decide which topics to learn, beyond standard curricula. Moreover, our work can be extended and combined with existing curricula by mapping course concepts to Wikipedia pages. A usage of knowledge graphs can also be envisioned to link a user to adjacent topics for an automatically generated curriculum. We will explore these ideas in future work. We believe that in the near future tools and applications such as the one presented here will play a major role in enhancing autonomous and personalized learning. Although our application is already functional, there is still a lot of room for improvement and we plan to develop it further in order to improve computing efficiency and user experience.

References

1. Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic QA corpora generation with roundtrip consistency. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, 2019.
2. AnkiWeb. Shared decks - ankiweb, 2020.
3. Hangbo Bao, Li Dong, Furu Wei, Wenhui Wang, Nan Yang, Xiaodong Liu, Yu Wang, Songhao Piao, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unilmv2: Pseudo-masked language models for unified language model pre-training. *CoRR*, abs/2002.12804, 2020.
4. Miroslav Blšták and Viera Rozinajová. Automatic question generation based on analysis of sentence structure. In *Text, Speech, and Dialogue*, pages 223–230, Cham, 2016. Springer International Publishing.
5. Ying-Hong Chan and Yao-Chung Fan. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering, MRQA@EMNLP*, 2019.
6. Guanliang Chen, Jie Yang, Claudia Hauff, and Geert-Jan Houben. Learningq: A large-scale dataset for educational question generation. In *Proceedings of the Twelfth International Conference on Web and Social Media, ICWSM*, 2018.
7. Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Annual Conference on Neural Information Processing Systems, NeurIPS*, 2019.
8. Xinya Du and Claire Cardie. Harvesting paragraph-level question-answer pairs from Wikipedia. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 2018.
9. The Examiners. theexaminers, 2020.
10. Som Gupta and SK Gupta. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65, 2019.
11. Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, 2015.
12. Kettip Kriangchaivech and Artit Wangperawong. Question generation by transformers, 2019.
13. Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30(1):121–204, 2020.
14. C. Kwankajornkiet, A. Suchato, and P. Punyabukkana. Automatic multiple-choice question generation from thai text. In *2016 13th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 1–6, 2016.
15. LanguageTool. Languagetool, 2020.
16. Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL*, 2020.
17. Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
18. M. Liu, V. Rus, and L. Liu. Automatic chinese factual question generation. *IEEE Transactions on Learning Technologies*, 10(2):194–204, 2017.

19. Luis Enrico Lopez, Diane Kathryn Cruz, Jan Christian Blaise Cruz, and Charibeth Cheng. Transformer-based end-to-end question generation. *CoRR*, abs/2005.01107, 2020.
20. Nobal Bikram Niraula and Vasile Rus. Judging the quality of automatically generated gap-fill question using active learning. In *Proceedings of the Tenth Workshop on Innovative Use of NLP for Building Educational Applications*, 2015.
21. Liangming Pan, Wenqiang Lei, Tat-Seng Chua, and Min-Yen Kan. Recent advances in neural question generation. *arXiv preprint arXiv:1905.08949*, 2019.
22. Suraj Patil. Question generation using transformers, 2020.
23. Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.
24. Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*.
25. Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. 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*, 2019.
26. Alexander M. Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*, 2015.
27. Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020.
28. Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1073–1083, 2017.
29. Sam Shleifer. Distilbart model, 2020.
30. Will Thalheimer. The learning benefits of questions. *Work Learning Research*, 2003.
31. HuggingFace Transformers. Question answering using distilbert, 2020.
32. Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.
33. Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In *Advances in Neural Information Processing Systems*, 2019.
34. Yu Yan, Weizhen Qi, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. *CoRR*, abs/2001.04063, 2020.
35. Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. *CoRR*, abs/1912.08777, 2019.
36. Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL*, 2019.