Unsupervised multiple-choice question generation for out-of-domain Q&A fine-tuning

Guillaume Le Berre[1,2], **Christophe Cerisara**[1], **Philippe Langlais**[2], **Guy Lapalme**[2]
[1] University of Lorraine, CNRS, LORIA, France
[2] RALI/DIRO, University of Montreal, Canada
{leberreg, felipe, lapalme}@iro.umontreal.ca, cerisara@loria.fr

Abstract

Pre-trained models have shown very good performances on a number of question answering benchmarks especially when fine-tuned on multiple question answering datasets at once. In this work, we propose an approach for generating a fine-tuning dataset thanks to a rule-based algorithm that generates questions and answers from unannotated sentences. We show that the state-of-the-art model UnifiedQA can greatly benefit from such a system on a multiple-choice benchmark about physics, biology and chemistry it has never been trained on. We further show that improved performances may be obtained by selecting the most challenging distractors (wrong answers), with a dedicated ranker based on a pretrained RoBERTa model.

1 Introduction

In the past years, deep learning models have greatly improved their performances on a large range of question answering tasks, especially using pretrained models such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019) and T5 (Raffel et al., 2020). More recently, these models have shown even better performances when fine-tuned on multiple question answering datasets at once. Such a model is UnifiedQA (Khashabi et al., 2020), which, starting from a T5 model, is trained on a large number of question answering datasets including multiple choices, yes/no, extractive and abstractive question answering. UnifiedQA is, at the time of writing, state-of-the-art on a large number of question answering datasets including multiple-choice datasets like OpenBookQA (Mihaylov et al., 2018) or ARC (Clark et al., 2018). However, even if UnifiedQA achieves good results on previously unseen datasets, it often fails to achieve optimal performances on these datasets until it is further fine-tuned on dedicated human annotated data. This tendency is increased when the target dataset deals with questions about a very specific domain.

One solution to this problem would be to fine-tune or retrain these models with additionnal human annotated data. However, this is expensive both in time and resources. Instead, a lot of work has been done lately on automatically generating training data for fine-tuning or even training completely unsupervised models for question answering. One commonly used dataset for unsupervised question answering is the extractive dataset SQUAD (Rajpurkar et al., 2016). Lewis et al. (2019) proposed a question generation method for SQUAD using an unsupervised neural based translation method. Fabbri et al. (2020) and Li et al. (2020) further gave improved unsupervised performances on SQUAD and showed that simple rule-based question generation could be as effective as the previously mentioned neural method. These approches are rarely applied to multiple-choice questions answering in part due to the difficulty of selecting distractors. A few research papers however proposed distractor selection methods for multiple-choice questions using either supervised approaches (Sakaguchi et al., 2013; Liang et al., 2018) or general purpose knowledge bases (Ren and Q. Zhu, 2021).

In this paper, we propose an unsupervised process to generate questions, answers and associated distractors in order to fine-tune and improve the performance of the state-of-the-art model UnifiedQA on unseen domains. This method, being unsupervised, needs no additional annotated domain specific data requiring only a set of unannotated sentences of the domain of interest from which the questions are created. Contrarily to most of the aforementioned works, our aim is not to train a new completely unsupervised model but rather to incorporate new information into an existing state-of-the-art model and thus to take advantage of the question-answering knowledge already learned.

We conduct our experiments on the SciQ dataset (Welbl et al., 2017). SciQ contains multiple-

Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics
Volume 2: Short Papers, pages 732 - 738
May 22-27, 2022 ©2022 Association for Computational Linguistics

Question:
What type of organism is commonly used in preparation of foods such as cheese and yogurt?
(A) mesophilic organisms (B) protozoa
(C) gymnosperms (D) viruses

Support text:
Mesophiles grow best in moderate temperature, typically between 25°C and 40°C (77°F and 104°F). Mesophiles are often found living in or on the bodies of humans or other animals. The optimal growth temperature of many pathogenic mesophiles is 37°C (98°F), the normal human body temperature. Mesophilic organisms have important uses in food preparation, including cheese, yogurt, beer and wine.

Figure 1: Example of a question in SciQ. The answer in bold is the correct one.

choice questions (4 choices) featuring subjects centered around physics, biology and chemistry. An example of question can be found in Figure 1. We focus on the SciQ dataset because it has not yet been used for training UnifiedQA and it requires precise scientific knowledge. Furthermore, our experiments reveal that the direct application of UnifiedQA on the SciQ benchmark leads to a much lower performance than when fine-tuning it on the SciQ training set (see Section 4). Our objective in this work is to solve this gap between UnifiedQA and UnifiedQA fine-tuned on supervised data with the unsupervised question generation approach described in Section 2. We additionally test our method on two commonly used multiple choice question answering datasets: CommonsenseQA (Talmor et al., 2019) and QASC (Khot et al., 2020). These datasets contain questions with similar domains to SciQ even though the questions are slightly less specific. Furthermore, neither of them has been used during the initial training of UnifiedQA.

2 Question Generation Method

We propose a method for generating multiple-choice questions in order to fine-tune and improve UnifiedQA. This process is based on 3 steps. First, a set of sentences is being selected (Section 2.1) from which a generic question generation system is applied (Section 2.2). Then a number of distractors are added to each question (Section 2.3).

Dataset	Sentences	Questions
SciQ data	53 270	77 873
SciQ data (train only)	45 526	66 552
Wikipedia data	45 327	62 848

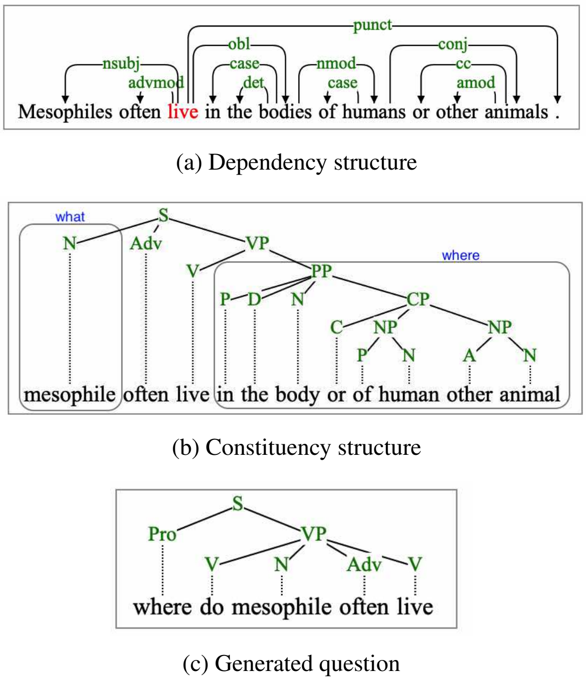
Table 1: Number of sentences selected for each of the datasets considered as well as the number of questions automatically generated from these sentences.

2.1 Sentence Selection

Our question generation method uses a set of unannotated sentences from which the questions will be generated. We compare three selection methods.

First, we consider a scenario where the application developer does not manually collect any sentence, but simply gives the name (or topic) of the target domain. In our case, the topics are “Physics”, “Biology” and “Chemistry” since these are the main domains in SciQ. A simple information retrieval strategy is then applied to automatically mine sentences from Wikipedia. We first compute a list of Wikipedia categories by recursively visiting all subcategories starting from the target topic names. The maximum recursion number is limited to 4. We then extract the summary (head paragraph of each Wikipedia article) for each of the articles matching the previously extracted categories and subcategories. We only keep articles with more than 800 average visitors per day for the last ten days (on April 27, 2021), resulting in 12 656 pages.

The two other selection methods extract sentences from SciQ itself and therefore are not entirely unsupervised but rather simulate a situation where we have access to unannotated texts that precisely describe the domains of interest such as a school book for example. The SciQ dataset includes a support paragraph for each question (see Figure 1). Pooled together, these support paragraphs provide us with a large dataset of texts about the domains of interest. We gather the paragraphs corresponding to all questions and split them into sentences to produce a large set of sentences that are no longer associated with any particular question but cover all the topics found in the questions. We compare two different setups. In the first one, we include all the sentences extracted from the train, validation and test sets thus simulating a perfect selection of sentences that cover all the knowledge expressed in the questions. Still, we only use the support paragraphs and not the annotated questions themselves. As compared to the classical


punct
obl
nsubj
case
nmod
conj
advmod
det
case
cc
amod
Mesophiles often live in the bodies of humans or other animals .
what
S
N
Adv
VP
V
PP
where
P
D
N
CP
C
NP
NP
P
N
A
N
mesophile often live in the body or of human other animal
S
Pro
VP
V
N
Adv
V
where do mesophile often live

(a) Dependency structure

(b) Constituency structure

(c) Generated question

Figure 2: Question generation process for a sentence similar to the one used to produce the question in Figure 3. A dependency parse (a) produced by Stanza is transformed into a constituency structure (b) in which two subtrees can be identified as the answers to two questions: *What* and *Where*. (c) shows the transformed constituency tree for the *Where* question.

supervised paradigm, this setting removes all annotation costs for the application developer, but it still requires to gather sentences that are deemed useful for the test set of interest. We then compare this setup with another one, where only the sentences from the train set are included. This scenario arguably meets more practical needs since it would suffice to gather sentences close to the domain of interest. The number of sentences for each dataset is presented in Table 1.

2.2 Questions Generation

The generation of questions from a sentence relies on the jsRealB text realizer (Lapalme, 2021) which generates an affirmative sentence from a constituent structure. It can also be parameterized to generate variations of the original sentence such as its negation, its passive form and different types of questions such as *who*, *what*, *when*, etc. The constituency structure of a sentence is most often created by a user or by a program from data. In this work, it is instead built from a Universal Dependency (UD) structure using a technique developed for *SR'19* (Lapalme, 2019). The UD structure of a sentence is the result of a dependency parse with Stanza (Qi et al., 2020). We thus have a pipeline composed of a neural dependency parser, followed by a program to create a constituency structure used as input for a text realizer, both in JavaScript. Used without modification, this would create a *complex* echo program for the original affirmative sentence, but by changing parameters, its output can vary.

Question:
What often is found living in or on the bodies of humans or other animals?
Right answer: mesophile

Random distractors:
(A) the most magnetic material in nature
(B) this energy
(C) climate

Refined distractors:
(A) carbohydrates
(B) small cell fragments called platelet
(C) echinoderm

Figure 3: Example of a synthetic question generated from the second sentence of the support paragraph in Figure 1 with a set of random distractors and with the set of refined ones.

In order to create questions from a single constituency structure, jsRealB uses the *classical* grammar transformations: for a *who* question, it removes the subject (i.e. the first noun phrase before the verb phrase), for a *what* question, it removes the subject or the direct object (i.e. the first noun phrase within the verb phrase); for other types of questions (*when*,*where*) it removes the first prepositional phrase within the verb phrase. Depending on the preposition, the question will be a *when* or a *where*. Note that the *removed* part becomes the answer to the question.

In order to determine which questions are appropriate for a given sentence, we examine the dependency structure of the original sentence and check if it contains the required part to be removed before parameterizing the realization. The generated questions are then filtered to remove any question for which the answer is composed of a single stopword. Table 1 shows the number of questions generated for each dataset. An example of a synthetic question is shown in Figure 3.

2.3 Distractors Selection

Since SciQ is a multiple-choice dataset, we must add distractors to each question we generate, to match the format of SciQ. A simple solution to this problem is to select random distractors among answers to other similar questions generated from the dataset of sentences we gathered. Obviously, selecting random distractors may lead to a fine-tuning dataset that is too easy to solve. Therefore, we propose another strategy that selects hard distractors for each question. To do so, starting from our synthetic dataset with random distractors, we fine-tune RoBERTa (Liu et al., 2019) using the standard method of training for multiple choices question answering. Each pair question/choice is fed to RoBERTa and the embedding corresponding to the first token ("[CLS]") is given to a linear layer to produce a single scalar score for each choice. The scores corresponding to every choice for a given question are then compared to each other by a softmax and a cross-entropy loss. With this method, RoBERTa is trained to score a possible answer for a given question, based on whether or not it is a credible answer to that question. For each question, we then randomly select a number of candidate distractors from the answers to other questions and we use our trained RoBERTa to score each of these candidates. The 3 candidates with the highest scores (and thus the most credible answers) are selected. The idea is that during this first training, RoBERTa will learn a large amount of simplistic logic. For example, because of the initial random selection of distractors, it is highly unlikely that even one of the distractors will be close enough to the question's semantic field. Furthermore, a lot distractors have an incorrect grammar (eg: a distractor might be plural when the question expects a singular). Therefore, in this initial training, RoBERTa might learn to isolate the answer with a corresponding semantic field or the one with correct grammar. The re-selection then minimizes the amount of trivial distractors and models trained on this new refined dataset will have to focus on deeper and more meaningful relations between the questions and the answers. The process is better shown in Figure 4, and an example of refined distractors can be found in Figure 3.

The number of scored candidate distractors is an hyper-parameter. A small number of candidates may result in a situation where none of the candidates are credible enough, while a large number requires more computation time, since the score of each candidate for every question needs to be computed, and has a higher risk of proposing multiple valid answers. In our experiments, we use a number of 64 candidates in order to limit computation time.

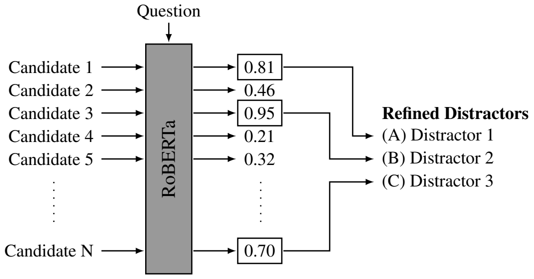


Figure 4: Description of the distractor refining method. RoBERTa scores each candidate distractor with regard to the question and the best 3 are selected to become the new refined distractors.

3 Training and Implementation Details

To refine distractors, we use the "Large" version of RoBERTa and all models are trained for 4 epochs and a learning rate of 1×10^{-5}. These hyper-parameters are chosen based on previous experiments with RoBERTa on other multiple-choice datasets. The final UnifiedQA fine-tuning is done using the same multiple choices question answering setup as the one used in the original UnifiedQA paper (Khashabi et al., 2020). We use the "Large" version of UnifiedQA and all the models are trained for 4 epochs using Adafactor and a learning rate of 1×10^{-5}. The learning rate is loosely tuned to get the best performance on the validation set during the supervised training of UnifiedQA. We use the Hugging Face pytorch-transformers (Wolf et al., 2020) library for model implementation. Experiments presented in this paper were carried out using the Grid'5000 testbed (Balouek et al., 2013), supported by a scientific interest group hosted by Inria and including CNRS, RENATER and several Universities as well as other organizations (see https://www.grid5000.fr).

4 Results

Accuracy results in Table 2 have a 95% Wald confidence interval of ±2.8%. The first row of Table 2 presents the accuracy results of a vanilla UnifiedQA large model on SciQ. The second line shows the accuracy when UnifiedQA is fine-tuned over the full training corpus. Our objective is thus to get as close as possible to this accuracy score using only unsupervised methods. The results using Wikipedia are the only ones that are unsupervised and therefore are the ones directly comparable to UnifiedQA with no fine-tuning or other unsupervised methods. The other results serve to illustrate what could be obtain with a tighter selection of sentences.

Model	**Dev**	**Test**
UnifiedQA (no fine-tuning)	64.6	63.4
UnifiedQA (supervised)	78.7	78.7
Unsupervised - Random distractors		
SciQ data	71.3	70.8
SciQ data (train only)	70.9	70.1
Wikipedia data	68.3	67.5
Unsupervised - Refined distractors		
SciQ data	75.4	74.2
SciQ data (train only)	73.1	72.4
Wikipedia data	70.6	69.4

Table 2: Accuracy on SciQ by UnifiedQA fine-tuned on our synthetic datasets. "SciQ data" refers to the questions generated using the support paragraphs in SciQ while "Wikipedia data" refers to questions generated using sentences harvested from Wikipedia. All scores are averaged over 3 independent runs (including the complete question generation process and the final UnifiedQA fine-tuning).

Fine-tuning UnifiedQA on synthetic questions with random distractors improves the results as compared to the baseline and, as expected, the closer the unlabeled sentences are to the topics of the questions, the better is the accuracy. Hence, generating questions from only the train set of SciQ gives performances that are comparable but slightly lower to the ones obtained from the combined train, dev and test set of SciQ. Finally, questions selected from Wikipedia also improve the results, despite being loosely related to the target test corpus. Our distractor selection method further boosts the accuracy results in all setups. This suggests that a careful selection of distractors is important, and that the hard selection criterion used here seems adequate in our context.

The results for CommonsenseQA and QASC using the same selection of sentences from Wikipedia are reported in table 3. Overall, we obtain similar results to SciQ with a large improvement of performances when generating questions and a further boost with refined distractors. However compared to SciQ, the improvement brought by the distractor refining process is less significant. This could be partly explained by the fact that the distractors in the original QASC and CommonsenseQA datasets are overall easier and therefore it is less advantageous for a model to be trained on harder questions.

Model	**CQA**	**QASC**
UnifiedQA (no fine-tuning)	60.9	44.5
UnifiedQA (supervised)	74.3	61.0
Wikipedia data (Random)	64.9	57.2
Wikipedia data (Refined)	65.1	59.4

Table 3: Accuracy results obtained on the dev set of CommonsenseQA and QASC when fine-tuning UnifiedQA using data from Wikipedia.

5 Conclusion

In this work, we proposed a multiple-choice question generation method that can be used to fine-tune the state-of-the-art UnifiedQA model and improve its performance on an unseen and out of domain dataset. Our contributions are:

- We have shown that simple unsupervised methods could be used to finetune existing multipurpose question answering models (in our case UnifiedQA) to new datasets or domains.
- We propose a novel distractor refining method able to select harder distractors for a given generated question and show its superiority compared to a random selection.

Future work includes comparing our method to other question generation methods (including supervised methods: Liu et al. (2020), Puri et al. (2020)) in order to assess the effect of both the generation method and the questions quality on the final performances of our models. Also, we will further compare different variations of our question generation and distractor refining methods in order to more thoroughly understand the effect of hyper-parameters such as the number of candidate distractors.

References

Daniel Balouek, Alexandra Carpen Amarie, Ghislain Charrier, Frédéric Desprez, Emmanuel Jeannot, Emmanuel Jeanvoine, Adrien Lèbre, David Margery, Nicolas Niclausse, Lucas Nussbaum, Olivier Richard, Christian Pérez, Flavien Quesnel, Cyril Rohr, and Luc Sarzyniec. 2013. Adding virtualization capabilities to the Grid'5000 testbed. In Ivan I. Ivanov, Marten van Sinderen, Frank Leymann, and Tony

Shan, editors, *Cloud Computing and Services Science*, volume 367 of *Communications in Computer and Information Science*, pages 3–20. Springer International Publishing.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*, pages 4171–4186. Association for Computational Linguistics.

Alexander Fabbri, Patrick Ng, Zhiguo Wang, Ramesh Nallapati, and Bing Xiang. 2020. Template-based question generation from retrieved sentences for improved unsupervised question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4508–4513, Online. Association for Computational Linguistics.

Daniel Khashabi, Sewon Min, Tushar Khot, Ashish Sabharwal, Oyvind Tafjord, Peter Clark, and Hannaneh Hajishirzi. 2020. UNIFIEDQA: Crossing format boundaries with a single QA system. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1896–1907, Online. Association for Computational Linguistics.

Tushar Khot, Peter Clark, Michal Guerquin, Peter Jansen, and Ashish Sabharwal. 2020. Qasc: A dataset for question answering via sentence composition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8082–8090.

Guy Lapalme. 2019. Realizing Universal Dependencies structures using a symbolic approach. In *The Second Multilingual Surface Realisation Shared Task (SR'19): Overview and Evaluation Results. In Proceedings of the 2nd Workshop on Multilingual Surface Realisation (MSR), (EMNLP-2019)*, page 8 pages, Hong-Kong. ACL.

Guy Lapalme. 2021. The jsRealB text realizer: Organization and use cases. (arXiv:2012.15425).

Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, Florence, Italy. Association for Computational Linguistics.

Zhongli Li, Wenhui Wang, Li Dong, Furu Wei, and Ke Xu. 2020. Harvesting and refining question-answer pairs for unsupervised QA. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6719–6728, Online. Association for Computational Linguistics.

Chen Liang, Xiao Yang, Neisarg Dave, Drew Wham, Bart Pursel, and C. Lee Giles. 2018. Distractor generation for multiple choice questions using learning to rank. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

Bang Liu, Haojie Wei, Di Niu, Haolan Chen, and Yancheng He. 2020. Asking questions the human way: Scalable question-answer generation from text corpus. In *Proceedings of The Web Conference 2020*, pages 2032–2043.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Raul Puri, Ryan Spring, Mohammad Shoeybi, Mostofa Patwary, and Bryan Catanzaro. 2020. Training question answering models from synthetic data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5811–5826, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

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.

Siyu Ren and Kenny Q. Zhu. 2021. Knowledge-driven distractor generation for cloze-style multiple choice questions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(5):4339–4347.

Keisuke Sakaguchi, Yuki Arase, and Mamoru Komachi. 2013. Discriminative approach to fill-in-the-blank quiz generation for language learners. In *Proceedings of the 51st Annual Meeting of the Association* *for Computational Linguistics (Volume 2: Short Papers)*, pages 238–242, Sofia, Bulgaria. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

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