Question Answering using Web Lists

Anoop R Katti, Kai Hui, Adria de Gispert, Hagen Fuerstenau
Amazon Alexa AI
{akatti,kaihuibj,agispert,fuersth}@amazon.com

ABSTRACT

There are many natural questions that are best answered with a list. We address the problem of answering such questions using lists that occur on the Web, i.e. List Question Answering (ListQA). The diverse formats of lists on the Web makes this task challenging. We describe state-of-the-art methods for list extraction and ranking, that also consider the text surrounding the lists as context. Due to the lack of realistic public datasets for ListQA, we present three novel datasets that together are realistic, reproducible and test out-of-domain generalization. We benchmark the above steps on these datasets, with and without context. On the hardest setting (realistic and out-of-domain), we achieve an end-to-end Precision@1 of 51.28% and HITs@5 of 79.38%, effectively demonstrating the difficulty of the task and quantifying the immediate opportunity for improvement. We highlight some future directions through error analysis and release the datasets for further research.

CCS CONCEPTS

• **Computing methodologies** → **Natural language processing**;

KEYWORDS

question answering, semi-structured, list extraction, list ranking

ACM Reference Format:
Anoop R Katti, Kai Hui, Adria de Gispert, Hagen Fuerstenau. 2021. Question Answering using Web Lists. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3459637.3482163

1 INTRODUCTION

A sizeable portion of the web queries includes natural questions that are best answered with a list (eg. How to connect Bluetooth headphones, What are the skills for a job, list the cast of a movie, etc.). Given such questions, List Question Answering (ListQA) attempts to address them using a list answer found on the web [14, 17, 19].

A list answer is an answer with a countable number of items. On a web page, a list answer can be represented (marked up) in a variety of forms - in simple cases as unordered/ordered lists, in more challenging cases as headings and paragraphs. Fig. 1 shows an example list where each item in the list is marked up as a separate paragraph on the web page. Moreover, list-like structures are also used for formatting a webpage (for example, left/top menu, comments and reviews, related/recommended pages, etc.). These two factors together make the task of finding the correct answer list (among a large number of false positives) very challenging.

One approach to ListQA is given a question, to retrieve the top-K documents from the web, extract lists from those pages, and rank them against the question [19]. While document retrieval is common to standard open-domain question answering (QA), list extraction and ranking are particular to ListQA. Therefore, in our work, we assume that we are given a set of top-k URLs corresponding to the given question. We describe an effective heuristic-based list extraction method to achieve high recall and a list ranking method that is similar to the state-of-the-art works on semi-structured ranking [3, 14]. Moreover, we explore the value of using the context surrounding the list in ranking in addition to the list text itself.

Existing QA public datasets, like Natural Questions (NQ) [9], are less realistic (based only on Wikipedia and with only 1 URL annotated per question). In order to train/benchmark the extraction and ranking methods in realistic settings, we present three ListQA datasets - one built on Natural Questions (simpler scenario) and the other two collected using the top-K URLs from the web via manual crowdsourced annotation (a more realistic scenario). Further, one of the two crowdsourced datasets is set aside entirely for testing out-of-domain generalization. The datasets are released to enable further research on the topic[1].

Finally, using the above datasets, we train the ranking models, with and without context text and benchmark both the extraction and ranking steps. To the best of our knowledge, we are the first to benchmark extraction even though it is a vital step in ListQA. We find that while the extractor can recall 71%-88% of the list answers, it extracts 21-67 lists per URL. This highlights the issue of false positive lists on the Web. On the end-to-end ListQA task, we find that the realistic and especially, out-of-domain settings are much harder than NQ-based settings. Further, we also find that using context helps significantly across the board. On the hardest setting (realistic and out-of-domain), our method achieves an end-to-end Precision@1 of 51.28% and HITs@5 of 79.38%, together demonstrating the difficulty of the task as well as quantifying the immediate opportunity for improvement.

Our contributions are (i) we describe methods based on state-of-the-art techniques for list extraction and ranking, (ii) we create three ListQA datasets that together are *realistic*, *reproducible* and *test out-of-domain generalization*, (iii) we benchmark list extraction as well as ranking, with and without context, (iv) we release the data to help advance the research on the topic.

2 LIST QUESTION ANSWERING

Let a list L be a set/sequence of items, $I = (h, d)$, where h is the item heading and d is the item description. Let the context of the List,

[1] https://github.com/alexa/wqa_listqa

CIKM '21, November 1–5, 2021, Virtual Event, QLD, Australia

ACM ISBN 978-1-4503-8446-9/21/11.
https://doi.org/10.1145/3459637.3482163

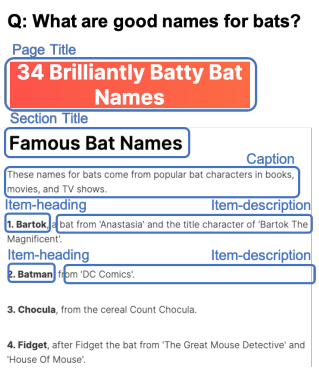

Figure 1: Example list

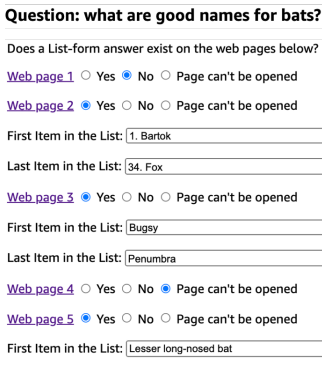

Figure 2: Annotation UI

$C = (p, s, c)$, be a tuple of page title, p, section title, s and caption text, c, that directly precede the List. See Fig 1 for an example.

Given a question, q, and a set of K HTML pages, $\{D_1, D_2, ..., D_K\}$, we wish to extract a set of lists, $\mathbb{L} = \{L_1, L_2, ..., L_N\}$ from the pages and find the list, L^*, that best answers q. We split this into two steps - list extraction and ranking and describe them below.

2.1 List Extraction

A list answer is an answer that has a countable number of items. On a HTML web page, a list answer could be marked up as unordered/ordered lists (<ul>, <ol>) or in more challenging cases, as headings, paragraphs, div's or any other tag (<h1>-<h6>, <p>, , etc.). For example, each item in the example list in Fig 1 is marked up as a separate paragraph on the web page. As per our datasets, only 39%-43% are unordered/ordered lists. Further, list-like structures are also used to format a webpage (left/top menu, comments/reviews, related pages). This makes the task of list extraction (with high recall and precision) very challenging.

Any HTML can be viewed as a DOM Tree. We first select the deepest node containing at least 90% of the textual content as the root of the tree. This selects the node with the main content and eliminates the boiler-plate HTML often seen in web pages. Inspired by the methods discussed in [11, 20], we then visit each node in the DOM Tree, cluster its children by their tags and select a cluster as a list if the cluster has at least 3 nodes and the tag is not a header, footer, script or span. For each item, I, if the beginning of the item is either a HTML heading or is marked in bold (eg. Fig 1), then this becomes the item heading, h, and the rest goes in the description, d. Otherwise, all the text becomes h. Additionally, list context such as page title, section title of the list and the caption text that directly precede the list are also extracted.

2.2 List Ranking

Given a question, q and a set of lists, $\mathbb{L}$, we wish to find L^* that best answers the question. Similar to other answer selection models [5, 16, 18], we define a Transformer-based [15] binary classifier to determine the correctness of the answer relative to the question. Let $P(y = correct \mid q, L)$ be the probability of L being the correct answer to q. The most relevant answer, L^*, is then obtained as

$$L^*(q) = argmax_L P(y = correct \mid q, L), L \in \mathbb{L}$$

The input to the Transformer model is a sequence of subword tokens from the question and the item-headings in the candidate list - $[CLS]q[SEP]h_1; h_2; ...; h_m[SEP]$, where $[CLS]$ and $[SEP]$ are special interaction and separator tokens [3, 4, 14]. The output representation corresponding to the $[CLS]$ token is trained for binary classification using the Cross-entropy loss [5].

When list context is used, the input to the model are the tokens from the question, context and list items headings (similar to [3]) - $[CLS]q[SEP]c; s; p[SEP]h_1; h_2; ...; h_m[SEP]$. The context and the list items share the same segment-id.

3 DATASETS

In this section, we describe the three ListQA datasets that we use in this work. The format of the ListQA datasets is *<Question, URL, List-answer/No-answer>*.

3.1 NQWikiList

NQ dataset [9] consists of 300K *<Question, URL, Answer>* tuples, where the answers are in the forms of paragraphs, tables or lists, known as long answer and a short answer span within the long answer if such a span exists. Some questions are also not answerable based on the page. From these, we select the questions that have a list as a long answer and do not have a short answer span, a total of 2594 questions. We name this dataset as *NQWikiList*.

NQ has answers sourced from a single website, i.e Wikipedia. Further, only 1 URL is annotated per question. Lastly, NQ only captures unordered/ordered/data/ lists (<ul>, <ol>, <dl>). These factors make NQ less realistic.

3.2 NQWebList and GQWebList

For a realistic setting, for each question, we collect 10 relevant URLs from the web and annotate the URLs for list-type answers. This is carried out on Mechanical Turk in two separate steps.

Annotation Task: For the purposes of annotation, we define a list-form answer as *an answer with a countable number of items*. Given a question, the annotators are asked if they understand the question. If yes, for each URL (that can be opened), they are asked to annotate the list if it exists. Ideally, we would like to annotate all items in the list. However, since lists can be indefinitely long, we request the annotators to annotate only the first and the last items of the list. The annotation interface is shown in Fig 2.

Quality Control: Finding a list answer from a URL involves understanding the subject of the question, the content of the page and judging whether one of the lists appearing on a page answers the question. The wide range of the question and the web page topics makes the task fairly complex. In order to control the quality of the annotations, we opt for workers on Mechanical Turk with *Masters* qualification. Further, we ask the annotators to take a quiz of 8 *<Question, URL>* pairs and only the master workers that correctly answer 6 or more pairs are invited to work on the task. The median time for each question (10 URLs per question) is 10 minutes, and the annotators are paid at the rate of $9.6 per hour ($1.6 per question).

Quality Inspection: We randomly sample 50 *<Quesion, URL>* pairs with a list answer and 50 pairs with no list answer and inspect them. We find that the annotation accuracy is 86%. Further, we find that the precision of the 50 list answers is 84% and the recall is 87.5%.

Table 1: Summary of different variants of the ListQA datasets

Dataset	Question-Answer Source	Annotations	♯Questions	♯URLs	♯ Websites	♯ Positive lists	♯ Negative lists
NQWikiList-Train / Eval	NQ-Wikipedia	NQ	2386 / 208	2386 / 208	1 / 1	2469 / 212	173888 / 13776
NQWebList-Train / Eval	NQ-Web	MTurk	854 / 204	7788 / 1915	2594 / 831	2823 / 615	423064 / 74693
GQWebList-Eval	GooAQ-Web	MTurk	195	1826	1104	910	29130

Crowdsourced Datasets: We collect the above annotations for 1058 list-type questions from NQ. We name this dataset as *NQWebList*. Beyond NQ, GooAQ [8] is another recent dataset including 360K questions with collection-type answers collected from Google search engine page. However, the dataset does not include the source URLs of the answers. Therefore, we randomly sample 195 questions from these 360k questions and annotate the top-10 URLs as described above. We name this dataset *GQWebList*. While NQWikiList and NQWebList are used for training and evaluation, GQWebList is used solely for evaluating out-of-domain generalization. Table 1 shows more statistics for the datasets.

Table 2: List extraction results

*-Eval	NQWikiList	NQWebList	GQWebList
Recall	87.5%	70.7%	72.5%
#lists-per-URL	67.3	48.3	20.9

4 EXPERIMENTS

4.1 List Extractor

Here, we measure the effectiveness of the list extractor in terms of extracting the list answers for the QA task.

Metric: An extracted list is said to match a ground-truth list if all the annotated items in the ground-truth list are matched to unique items in the extracted list in the correct order (item matching is performed using string overlap). Equipped with this, we compute the recall of the list answers (presented in Table 2).

Results: We find that the extractor can recall 70.7%-87.5% of the list answers (depending on the dataset). We note that the recall on NQWikiList is higher than the other datasets. This is because in NQWikiList, list answers are always represented in HTML as unordered/ordered/data lists (<ul>, <ol>, <dl>), making the extraction task easier. We further observe that it extracts 20.9-67.3 lists per web page, highlighting the issue of the false-positive lists. This issue is exaggerated in NQWikiList as Wikipedia pages often have much more content than a standard web page.

4.2 List Ranking

Data for ranking: The training/evaluation data for the ranker is tuples of <*Question, List, Label*>. Given a question and a URL, we use the extractor described in section 2.1 to extract all the lists from the URL. The lists are then matched with the annotation as described in section 4.1. The lists that match the annotation are labeled positive and other lists from the URL are labeled negative. Table 1 shows the number of positive and negatives for training and evaluation. 100 questions from the training set are used for validation and the rest are used for training.

Metric: We use Precision@1, HITs@5 to measure the effectiveness of the ranker. The gap between Precision@1 and HITs@5 highlights the opportunity for improvement in the ranker. Further, the ranker uses only those Lists that the extractor could extract. This means, in addition to measuring the effectives of the ranker, these numbers also capture the end-to-end ListQA performance.

Answer selection models: As mentioned in Section 2.2, the list ranking problem is formulated in the form of answer selection. Intuitively, the models that have been trained for similar tasks might be directly used to select lists as the answers. Specifically, based on the investigations in [5], we fine-tune a pre-trained language model (in our case, ELECTRA) to transfer it to the task of answer selection. We experiment with ASNQ [5] (a dataset derived from NQ [9]) and MS Marco [12]. While ASNQ transfers to a sentence selection task, MS Marco transfers to a passage ranking task. Similar to TANDA [5], we use a maximum sequence length of 128 tokens. On ASNQ, we pretrain with a learning rate of $3e-5$ and a batch size of 1536 for 40k steps. On ASNQ dev dataset, we obtain a Precision@1 of 84% and HITs@5 of 97.2%. On MSMarco, we pretrain with a learning rate of $3e-5$ and a batch size of 256 for 60k steps. On MS Marco dev, we obtain a MRR@10 of 0.3767.

The results on ListQA evaluation sets have been summarised in Table 3, where the two models are grouped under *LR-AS2*, denoted as ASNQ and MS Marco, respectively. Under a zero-shot setting, we find that these models perform poorly.

Fine-tuning on ListQA: As a next step, we use each of the above transferred models and adapt them to the domain of ListQA by fine-tuning them on NQWikiList-Train and NQWebList-Train. For all fine-tuning experiments, we use a learning rate of $3e-5$ and a batch size of 512 with early stopping and present the results in Table 3, where the models are grouped under *LF-FT*.

We wish to highlight three observations: (i) Adaption to the list domain significantly improves the performance on the evaluation datasets. (ii) Realistic setting with top-10 URLs from the Web is much harder than the simpler setting with top-1 URL from Wikipedia as gathered by the performance gap between NQWikiList-Eval and the other two ListQA evaluation sets. (iii) A model trained on NQWebList-Train shows much better generalization capability over a model trained on NQWikiList-Train (as measured by their performances on GQWebList-Eval)

Using Context: Here, we explore the usage of the context, cumulatively adding one field at a time. The token lengths are limited to 20, 10, 10 tokens for the caption, section title and page title respectively. The results are presented in Table 3 and are grouped under *LF-FT+Context*. We find that using context text helps significantly irrespective of the training/evaluation data. On the hardest setting, GQWebList, which is realistic as well out-of-domain, we reach a Precision@1 of 51.28% and HITs@5 of 79.38%, gaining an absolute improvement of 31.79% over the zero-shot answer selection models.

Table 3: ListQA Results (average of 5 runs)

Model	NQWikiList-Eval		NQWebList-Eval		GQWebList-Eval	
	P@1	HITs@5	P@1	HITs@5	P@1	HITs@5
LR-AS2: Answer Selection or Passage Ranking without Fine-tuning on ListQA						
ASNQ [5]	26.44%	67.79%	10.78%	30.39%	19.49%	51.79%
MS Marco [13]	19.71%	41.83%	2.45%	15.20%	17.95%	45.64%
LR-FT: Fine-tuned Model on ListQA dataset without using Context						
ASNQ -> NQWikiList-Train	**78.27%**	**95.77%**	35.00%	60.29%	22.77%	58.87%
ASNQ -> NQWebList-Train	57.40%	82.98%	40.59%	60.98%	43.90%	74.67%
MS Marco -> NQWikiList-Train	75.00%	93.46%	34.71%	60.98%	26.97%	62.67%
MS Marco -> NQWebList-Train	57.50%	82.21%	43.33%	62.45%	48.51%	76.31%
LR-FT+Context: Fine-tuned Model on ListQA dataset with Context (using NQWebList-Train only)						
ASNQ -> NQWebList-Train + (Caption)	62.60%	87.12%	47.16%	65.39%	47.38%	79.59%
ASNQ -> NQWebList-Train + (Caption, Section title)	67.69%	88.17%	**50.69%**	**68.73%**	47.18%	**80.21%**
ASNQ -> NQWebList-Train + (Caption, Section title, Page title)	69.90%	90.00%	49.12%	68.63%	48.82%	79.79%
MS Marco -> NQWebList-Train + (Caption)	60.00%	84.52%	44.51%	66.47%	49.74%	78.97%
MS Marco -> NQWebList-Train + (Caption, Section title)	65.58%	87.60%	47.35%	68.14%	**51.28%**	79.38%
MS Marco -> NQWebList-Train + (Caption, Section title, Page title)	62.40%	87.02%	49.90%	68.63%	50.67%	79.08%

4.3 Error Analysis

The gap between Precision@1 and HITs@5 highlights the opportunity for immediate improvement. In this section, we analyze this gap and discuss some frequent errors.

Too Many False Positive Lists: While the recall of the extractor is sufficiently high, we see that the extractor extracts too many lists. Transformer-based rankers are capable of selecting the correct list in spite of all the noise (as seen from the results in Table 3). Yet, we find that false positive lists are being ranked to the top. This could perhaps be addressed by a sophiscated list filtering step before ranking the lists.

Context-awareness: We explore one way to incorporate the context text while ranking the lists (similar to [3]). However, we find that many correct lists with good contexts are not being ranked to the top. More advanced models should be explored to incorporate context in ranking. We also find that many positive lists do not have the proper context extracted. More work could go into better extracting the context text.

Extractor recall errors: In spite of the high recall, a non-negligible portion of lists are still not extracted. These mainly constitute lists with less than 3 items or of items which are clustered into subgroups (i.e. a hierarchical list instead of a flat list). Novel techniques should be explore for extracting such lists.

5 RELATED WORK

Dataset: Natural Questions (NQ) dataset [9] has triggered a lot of work on unstructured [7] and semi-structured [6, 14] question answering. While NQ is a valuable dataset, we find the setting of NQ less realistic as all URLs come from a single website (Wikipedia) and only 1 URL is annotated per question. Moreover, all lists in NQ are marked up with the HTML tags for unordered/ordered/data lists (<ul>, <ol>, <dl>), whereas in general, they could be expressed with any tag. In GrassLM [19], semi-structured answers are sourced from the top-K pages from the web (although there is no discussion on their tags or the extraction step). However, unfortunately, they do not release their data. In contrast, our ListQA datasets are realistic, containing list answers from top-10 URLs from the Web. Neither our annotation nor our extraction are dependent on some specific tag mark-ups. Further, we reserve one of the datasets entirely for testing out-of-domain generalization. Lastly, we publish our datasets for reproducibility.

Extraction: While there has been a plenty of work on table extraction [1, 2, 10, 19], lists have not attracted the same amount of attention. In reality, though, the number of pages on the web that present information as lists could be comparable to, if not higher than, the pages with Tables [8, 9]. Our extractor is based on [11, 20]. The main difference, however, is that while they perform tag-path clustering [11] on *all* nodes in the HTML DOM Tree, we perform clustering only on the siblings (as described in Sec. 2.1), leading to the items in a list all coming from the same parent.

Ranking: Similar to other semi-structured ranking methods [3, 14], we explore the simple but effective method of linearising the lists and ranking them using Transformers [15]. We further employ **T**ransfer and **A**dapt strategy from TANDA [5], where we first transfer a pre-trained language model to the task of answer/passage selection and then adapt to the domain of lists.

6 CONCLUSION

We address the problem of List Question Answering. We describe methods for list extraction and ranking. We present three novel datasets - one built from Wikipedia-dominated Natural Question, the other two built from the top-K URLs from the web using crowdsourced manual annotations. Together, these datasets are *realistic*, *reproducible* and *test out-of-domain generalization*. We benchmark the extraction and ranking methods, including the usage of the context. We highlight the difficulty of the task, quantify the opportunity for immediate improvement and conduct error analysis to identify the future directions.

REFERENCES

[1] Chandra Sekhar Bhagavatula, Thanapon Noraset, and Doug Downey. 2015. Tabel: Entity linking in web tables. In *International Semantic Web Conference*. Springer, 425–441.

[2] Michael J Cafarella, Alon Halevy, Daisy Zhe Wang, Eugene Wu, and Yang Zhang. 2008. Webtables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment* 1, 1 (2008), 538–549.

[3] Zhiyu Chen, Mohamed Trabelsi, Jeff Heflin, Yinan Xu, and Brian D Davison. 2020. Table search using a deep contextualized language model. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 589–598.

[4] 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)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[5] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. 2020. TANDA: Transfer and Adapt Pre-Trained Transformer Models for Answer Sentence Selection. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (Apr 2020), 7780–7788. https://doi.org/10.1609/aaai.v34i05.6282

[6] Jonathan Herzig, Thomas Müller, Syrine Krichene, and Julian Martin Eisenschlos. 2021. Open Domain Question Answering over Tables via Dense Retrieval. *arXiv preprint arXiv:2103.12011* (2021).

[7] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[8] Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. 2021. GooAQ: Open Question Answering with Diverse Answer Types. *arXiv preprint* (2021).

[9] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics* 7 (2019), 453–466.

[10] Oliver Lehmberg, Dominique Ritze, Robert Meusel, and Christian Bizer. 2016. A large public corpus of web tables containing time and context metadata. In *Proceedings of the 25th International Conference Companion on World Wide Web*. 75–76.

[11] Gengxin Miao, Junichi Tatemura, Wang-Pin Hsiung, Arsany Sawires, and Louise E Moser. 2009. Extracting data records from the web using tag path clustering. In *Proceedings of the 18th international conference on World wide web*. 981–990.

[12] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. In *CoCo@ NIPS*.

[13] Rodrigo Nogueira and Kyunghyun Cho. 2019. Passage Re-ranking with BERT. *arXiv preprint arXiv:1901.04085* (2019).

[14] Barlas Oguz, Xilun Chen, Vladimir Karpukhin, Stan Peshterliev, Dmytro Okhonko, Michael Schlichtkrull, Sonal Gupta, Yashar Mehdad, and Scott Yih. 2020. Unified Open-Domain Question Answering with Structured and Unstructured Knowledge. *arXiv preprint arXiv:2012.14610* (2020).

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762* (2017).

[16] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the Jeopardy model? A quasi-synchronous grammar for QA. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*. 22–32.

[17] Hui Yang and Tat-Seng Chua. 2004. Web-based list question answering. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*. 1277–1283.

[18] Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 287–296.

[19] Xingyao Zhang, Linjun Shou, Jian Pei, Ming Gong, Lijie Wen, and Daxin Jiang. 2020. A Graph Representation of Semi-structured Data for Web Question Answering. *arXiv preprint arXiv:2010.06801* (2020).

[20] Zhixian Zhang, Kenny Q Zhu, Haixun Wang, and Hongsong Li. 2013. Automatic extraction of top-k lists from the web. In *2013 IEEE 29th International Conference on Data Engineering (ICDE)*. IEEE, 1057–1068.