CS 546 Final Report - Question Answering over Knowledge Base using Complex Questions

Xiyu Wang, Rishabh Vaish, Liyao Yang, Shubham Goel
University of Illinois, Urbana-Champaign
{xiyuw2, rvaish2, liyaoy2, sgoel11}@illinois.edu

Abstract

Although recent research work has seen improvement in answering simple questions, few researches have studied handling complex questions that require much more inference, common sense or multi-hop reasoning. Also, for question answering over knowledge base, one major assumption is that the knowledge base usually contains all the necessary information, which is not likely to happen in real world. Therefore, our project will delve into the question answering task over knowledge base on complex questions with multiple information sources. We use state-of-the-art GraftNet model and conduct experiment on Complex WebQ dataset under three settings: (1) knowledge base only; (2) text only; and (3) combined sources of both text and knowledge base, and compare the performance with simple question datasets such as WebQuestionsSP and Wikimovies. We have seen far more significant decrease in the Hits@1 score of text only setting than other two settings. We also have observed that when questions are complex, adding text as another information source does not help improve machine's comprehension. We further test fact-dropout on Complex WebQ, and conclude that it can also function as regularization for complex questions.

1 Introduction

Question answering (QA) is the task of finding answers from external information sources to a natural language question (NLQ). Those sources could be documents, or structured knowledge base. In the past, many researches are focused on developing pipeline methods consisting of several machine learning models (such as segmentation, entity recognition, disambiguation, relation classification, and etc.), however, errors from one step could be propagated into following steps. Recently, the emphasis has been shifted towards developing end-to-end deep neural networks, leaving the model to make all the relevant decisions and inference.

QA over knowledge base (KB) has received growing attention among all QA tasks due to the popularity of using structured knowledge representation in downstream applications, where each piece of information is stored in a triple with two entities and one relation. However, to conduct QA over KB still remains a challenge. This is because the KB representation is fundamentally different from natural language documents, which makes it difficult to map NLQs to the corresponding vocabulary elements. Also, the conceptual granularity of NLQs does not often coincide with that the data schema used in KB. Different data schema used in different datasets can provide different ways of answering the same NLQ, which adds to the complexity of evaluating the answers.

Apart from that, because of varying *coverage* of KB and varying *difficulty* of extracting from KB, models performing well on one KB dataset do not always generalize beyond the training documents or to some types of reasoning it has never been exposed to. As has been pointed out in [1], KB suffers from low coverage due to incompleteness, but is not difficult to extract answers. To compensate for this matter, [1] suggests that text can be used along with KB for better QA performance. Therefore, our project will also focus on combined information sources, ie. KB and text.

However, different from paper [1] which focuses on experimenting with simple questions each containing single fact, we will focus on complex questions which requires much more inference, common sense or multi-hop reasonings. This is because for QA tasks in general, although current methods can easily outperform human comprehension on simple questions, there are few researches exploring complex questions and have achieved comparable results.

2 Background

Methods for QA over KB can be divided into three categories: (1) template-based methods, (2) semantic parsing based methods, and (3) deep neural networks [2].

Template-based methods aims at translating NLQs into SPARQL queries by using traditional NLP processes such as POS tagging, Context-Free Grammar, as well as both domain-dependent (ie. produced/VBD, films/NNS) and domain-independent vocabulary (ie. who, the most). However, since template-based methods require much engineering, it cannot scale up and is often applied in domain-specific QA tasks.

Semantic parsing based methods [3, 4] are usually utilizing a parser trained on large-scale dataset to extract entities and relations from NLQs, in order to map them to their logic forms and conduct information retrieval over KB. However, the difficulty lies in constructing a lexicon for the alignment from natural language phrases to predicates with features in KB, and accurately bridging the mapped predicates in the NLQs.

Deep learning based methods [5–7] usually create entity linking for the main topic entities in the NLQs, then identify and rank the core inference chain by comparing the similarity between the question and candidate answers. Many of these works make use of Memory Networks [8] or graph CNN [9–11]. However, subgraphs of both questions and answers needs to be mapped to their embedding space to be compared [12, 13], which usually requires large computational resources for further inference. Therefore, deep learning-based methods can only handle simple questions with single facts.

To look into the ability of answering complex questions, [14] develops a complex question dataset based on a simple question dataset WebQuestionsSP [15] by using conjunction and composition techniques. The same paper then proposes a QA algorithm called SplitQA which aims to decompose those complex questions to form simple questions that are easier to answer. Although the SplitQA uses an iterative approach to retrieve answers for multi-hop questions, it has not been tested on KB settings. Also whether the decomposition process is general enough to apply to other dataset is still not known.

Current QA over KB tasks often assume that all the answers are contained in one single KB. However, in real world applications, answers could exist across different KBs or sometimes KB is not complete at all. GraftNet [1] is proposed to explore the case where there is a need to reason over combinations of text and KBs. It extracts answers from question-specific subgraph formed from both text and KB. However, it does not treat nodes from text and nodes from KB in the same manner as previous graph representation learning does. It also introduces the use of a directed propagation method to restrict the graph embeddings from following paths linked to the questions. It is competitive with the state-of-the-art in either KB or text settings alone. Details about GraftNet and its architecture will be discussed in the Section 4.

Similar to GraftNet, PullNet [16] is also focused on the same setting, but improves by relying on only a small amount of retrieval operations at first, but learns how to construct the subgraph gradually through iterative expanding it with newly retrieved information. However, this iterative process is subject to memory limitations and cannot be applied to a large corpus with long passages. Current systems which use a 'retrieve and read' pipeline [17–19], or use a 'phrase-indexed' method [20] cannot be directly applied in PullNet to solve this problem, or cannot even be applied to both KB and text for QA.

Therefore, our project aims to explore the setting where multi-hop reasoning are required, and neither text or KB alone is enough to answer all the questions. We will use complex questions, compare the setting where the model is asked to answer from either text or KB alone, and where the model can obtain information from hybrid sources, and evaluate the performance under all these settings.

3 Task

3.1 Problem statement

As has been discussed in the previous section, our project will experiment with a complex dataset, and compare the performance of QA over this complex dataset with other simple datasets such as WebQuestionsSP and Wikimovies.

To be more specific, we will:

- choose a state-of-the-art QA over KB neural network from the literature discussed in Section 2;
- reproduce its experiment on simple datasets such as WebQuestionsSP and Wikimovies as our baseline results;
- process the complex dataset to form question and answer subgraphs;
- use the same neural network to experiment in three different setting on the complex dataset: (1) KB only; (2) text only, and (3) combination of text and KB;
- further test other techniques used in QA over KB on the complex dataset, such as fact-dropout.

We will compare these three test results with baseline results, to see how the current state-of-the-art model perform on complex dataset. By doing so we can draw our conclusions and point out future improvements.

3.2 Dataset description

For an end-to-end approach, the model should learn to locate the topic entity, and follow the exact logic reasoning steps along the KB leading to the answer itself. However, for multi-hop QA settings, there are few datasets to provide enough annotations with respect to whether the model learns the inference correctly.

After reading and researching, we choose the ComplexWebQuestions version 1.1 (Complex WebQ) and do not focus on the inference logic. Complex WebQ contains 34,689 question-answer pairs each containing: A complex question, an answer (including alias), an average of 366.8 snippets per question, and a SPARQL query. Due to a large amount of questions-answer pairs, the computation still can be very expensive. To prevent memory overload, we randomly choose 20% of the original Complex WebQ dataset to do all the following experiment.

The Complex WebQ is built on the WebQuestionsSP. The WebQuestionsSP contains 5,810 question-answer pairs. Each contains a simple question, an answer, and a SPARQL query. Complex NLQs are generated with composition, conjunction, superlative, and comparative functions based on SPARQL queries from the WebQuestionsSP, using Amazon Mechanical Turk.

The Table 1 shows comparison between questions in the WebQuestionsSP and questions in the Complex WebQ by different functions.

4 Model

As has been discussed in the background section, there are many neural networks to choose from, such as Memory Network, SplitQA, GRAFT-Net, PullNet, and the latest SPARQA [21]. Out of those powerful models, we found that using GRAFT-Net is more practical. Other models either do not provide open source code or are based on very computational expensive models such as BERT or REINFORCE. We are particularly interested in the early fusion version of LSTM ensembles in GRAFT-Net which shows better performance compared to the late fusion version. In their work, GRAFT-Net was only tested on Wikimovies and WebQuestionsSP dataset, and have not been extended to complex questions.

Table 1: Comparison between the WebQuestionsSP and the Complex WebQ

Compositionality type of Complex WebQ	Complex WebQ	WebQuestionsSP
Composition	what character did brian austin green play on **the tv program that has the theme song Beverly Hills** 90210	what character did brian austin green play on 90210
Composition	what type of currency is used in **the location where the film "The Proud and Profane" takes place**	what type of currency is used in **puerto rico**
Conjunction	what influenced whitman 's poetry **and that who was the speaker in Third Joint Debate at Jonesboro**	what influenced whitman 's poetry
Conjunction	what movies does alyson stoner play in **and is the same genre as film The True Story of the Three Little Pigs**	what movies does alyson stoner play in
Superlative	what are the places to see in dubai **and the number of building floors is smallest**	what are the places to see in dubai
Superlative	what movies does alyson stoner play in **and the film netflix_id is largest**	what movies does alyson stoner play in
Comparative	who did the cleveland browns draft **and is the football player that has passing attempts statistics greater than 74**	who did the cleveland browns draft
Comparative	what movies does alyson stoner play in **and is the film that was released after 2004-10-19**	what movies does alyson stoner play in

4.1 STAGG

STAGG stands for staged query graph generation. The basic idea proposed in [4] involves a query graph, a sub-graph of the knowledge base and semantic parsing, a search problem that grows the graph through staged state action. The STAGG approach has following steps:

- Link topic entity - Prepare a surface-form lexicon for entities in the knowledge base. Up-to top 10 ranked entities are considered as topic entity
- Identify the core-inferential chain - A possible path is explored and it is represented in a K-dimensional space. The cosine similarity between the answer and the end node is calculated which is followed by using softmax to get probability density
- Augment constraints - One or more constraints are added to the event (eg - character(y, Meg Griffin)) or to the answer entity (eg - gender = female). Only a subset of constraint nodes are considered

A learning reward function is used to judge whether a guery graph is the correct semantic parsing. The learning reward function is a log-linear model with pairwise ranking objective.

4.2 GraftNet

The problem GraftNet tries to solve is combining the structured data i.e knowledge graph and unstructured data i.e relevant text to answer question. This can be done in broadly two ways - Late fusion and early fusion. The late fusion technique is an ensemble model, wherein a model is trained on each knowledge source separately and the predictions are merged. The second method of early fusion works by combining all the knowledge sources into a single data structure, and training a model to make inferences from this data structure. [1] found that early fusion performs better than late fusion. In order to make a single data structure to combine all the knowledge, a bipartite graph is proposed with knowledge base relations as the entities joined with the entities found in text through edges. So a subject entity node is connected to a relation entity node which is in turn connected to another subject entity node. In order to combine the text source in this graph, a entity linking algorithm is run which connects the subject entity nodes to their references in the text.

After making a combined data structure, the model is build using embedding propagation on graphs methodology. The series of steps performed once a question is given is as follows:

- Search for the subject entities mentioned in the question
- Propagate through the knowledge base side for relational entity links, followed by propagating through the text side
- Next step is to propagate through the neighbors of these relational entities and text reference. This embedding propagation mechanism will continue
- Eventually, we will have the updated embedding for all these entity nodes. The model uses these updated embedding to classify one of the entities as the answer.

In order to initialise the embedding for the text note, a LSTM is used to read the text node and the output of the LSTM is used as embedding. The length of the embedding of text node is same as the text itself because there could be multiple entities mentioned in the same text node and we want to differentiate that.

4.3 Architecture

There are two fundamental differences in GraftNet from previous graph-based classification tasks, namely the presence of heterogenous nodes (from both the KB, representing symbolic objects, and from the textual documents), and secondly the conditioning of the representation of nodes on the input natural language question.

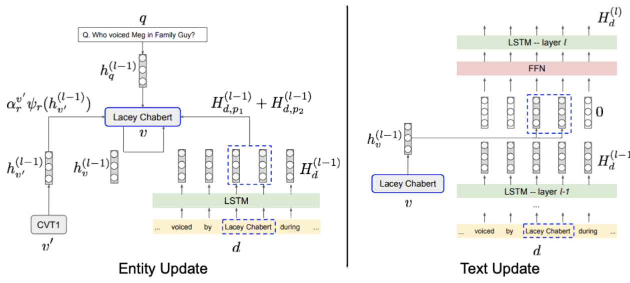

Figure 1: Illustration of the heterogeneous update rules for entities **(left)** and text documents **(right)** (image from [1])

The first issue is tackled using a modified update rule for entity node, which consists of three parts. The first part comes from itself, the same as previous iteration. The second part comes from knowledge base and the third part comes from the text side. For updating the text node, the previous iteration text node embedding is combined with its entity embedding and then an LSTM is used to propagate information to other text. The output of this LSTM will be the updated embedding for text node in next iteration.

The second issue, is solved using two ways, by attention over relations, and by personalized propagation. While computing for the update for entity nodes, the attention weight is computed using the question and relation embedding, allowing for propagation along edges more relevant to the question. Personalized propagation is achieved using a PageRank inspired technique [22]. The objective of the technique is to resolve the issue of multi-hop reasoning, which is required for answering several questions. The propagation starts at the seed entities in the question, and using the previous attention weights and a newly introduced PageRank score (measuring the total weight of path from the seed to the current node), the weights near the relevant nodes to the question get a higher weight. A hyperparameter is used to tweak the level of hops we need to propagate to.

5 Experiments

Our first experiment consists of three parts:

- reproduce the baseline results on WebQuestionsSP and Wikimovies dataset;

- preprocess Complex WebQ dataset to create question subgraphs by linking topic entities, forming inferential chain, and conducting information retrieval;
- test the processed Complex WebQ dataset for the three settings mentioned in the Section 3.

5.1 Preprocessing of Complex WebQ

For this part, although the idea is borrowed from [1, 4, 23, 24], we still need to write our own code to preprocess Complex WebQ because it has never been preprocessed thus does not have open source code, and that it has a different schema from other previous datasets such as WebQuestionsSP and Wikimovies.

5.1.1 Linking topic entity

To conduct experiment on the Complex WebQ dataset, we first need to go through its own query graph generation process by creating a single-node graph that corresponds to the topic entity in the NLQ. Similar to [4], we also use a surface-form lexicon that lists all possible mentions of an entity, and pair the possible mentions with their possible entities. We then form a surface-form lexicon by building a statistical model with considering up to 10 highly ranked entities as top entities.

5.1.2 Relation embedding

For preprocessing, we use Freebase which contains 46 million topics and 2.6 billion facts [4]. For simplicity, we use topic entities with their corresponding freebase ID, and use the relations described in freebase. There are 589 relations mentioned in Complex WebQ, vs. only 9 relations metioned in Wikimovies.

We also use GloVe embedding [25] with a dimension of 300 for the keywords mentioned in the relations to form relation embeddings by considering the position of the keywords, ie. considering whether the keyword is the head entity, tail entity or the predicate. This relation embedding matrix is created to facilitate generating question embeddings in the following step.

5.1.3 Question embedding

We first remove the stopwords and wh-words to extract the keywords from the NLQs. We then use the GloVe embedding [25] with a dimension of 300. Embedding for each question is the average of the GloVe embedding vector of keywords.

5.1.4 Retrieval from KB

To retrieve entities from KB, we use the the results from entity linking as our seed entities. We then follow the methods of Personalized PageRank (PPR) described in [23] to identify potential answers. We read all triples from the relevant freebase fact files and create a sparse adjacency matrix between the entities. We use cosine similarity as edge weights, and calculate PPR score for a given seed and adjacency matrix. After running PPR, we only consider the top 50 entities by PPR score, along with edges among them to add to the question subgraph. The results of retrieval from KB is a set of entities mapped to their freebase ID along with their PPR score.

5.1.5 Retrieval from text

We use the snippets provided in the Complex WebQ as corpus and conduct a sentence-level text retrieval. We use Lucene [24] index with sentences, and retrieve the top ranking 20 document indices. The results of retrieval from corpus is a set of document indices.

5.1.6 Postprocessing

The results of retrieval from text and retrieval from KB are combined together to form a fully connected question subgraph. The entity based recall of such retrieval is 0.296 with considering the top 50 entities. The average length of passages for each question is 56.

Other necessary processing includes building a vocabulary list, relation list, and entity list, which are too trivial to discuss.

5.2 Tricks for training

We use three layers of LSTM, with a dropout rate of 0.3, pagebank interpolation coefficient of 0.8 to control the information propagation. We also randomly dropout the linear layer with a probability of 0.2. For hybrid sources, we are mainly interested in the performance of early fusion.

GraftNet is prone to overfit. To avoid this situation, we add early stopping, and use fact dropout with a probability of 0.1, which means that we randomly drop edges from the graph during training. This can also prevent the model from being too dependent on KB than documents.

5.3 Results on WebQuestionsSP

For this dataset, we use a 10 maximum query words, 50 maximum document words, entity embedding size of 50, word embedding size of 100, and graph embedding size of 100. We use a batchsize of 10, and train for 100 epochs with a learning rate of 0.001. We use Hits@1 score to evaluate the result. Hits@1 represents the probability of the real next utterance ranking the highest according to the model. We have obtained Hits@1 score of 0.25 for text only setting, 0.67 for KB only setting, and 0.68 for the early fusion of hybrid setting, which are organized in Table 2.

Table 2: Results for the three settings of WebQuestionsSP.

Setting	**Hits@1 %**
Text only	25.494 %
KB only	67.099 %
Hybrid sources with early fusion	68.333 %

5.4 Results on Wikimovies

For this dataset, we use 10 maximum query words, 40 maximum document words, entity embedding size of 80, word embedding size of 80, and graph embedding size of 80. We use a batchsize of 80, and train for 80 epochs with a learning rate of 0.001. We have obtained Hits@1 score of 0.86 for text only setting, 0.97 for KB only setting, and 0.97 for the early fusion of hybrid setting, which are organized in Table 3.

Table 3: Results for the three settings of Wikimovies.

Setting	**Hits@1 %**
Text only	86.358 %
KB only	96.861 %
Hybrid sources with early fusion	97.032 %

Similar as WebQuestionsSP, QA under text only setting is the most difficult among the three. And the hybrid setting only improves the performance slightly.

5.5 Results on Complex WebQ

For this dataset, we use 15 maximum query words, 50 maximum document words, entity embedding size of 50, word embedding size of 100, and graph embedding size of 100. We use a batchsize of 10, and train for 50 epochs with a learning rate of 0.0005. We have obtained Hits@1 score of 0.14 for text only setting, 0.52 for KB only setting, and 0.51 for the early fusion of hybrid setting, which are organized in Table 4.
Compared with WebQuestionsSP, Hits@1 score has decreased for 45.65%, 22.50%, and 25.78%. The most significant drop happens under text only setting. It could be that when questions are complex, the model does not know which document is the most relevant, since text information is not structured and too noisy, even though the expected answers are simple entities.

From the above results, even the early fusion of hybrid setting does not produce higher Hits@1 score than then KB only setting. It could be that we need to further fine-tune hyperparameters to

Table 4: Results for the three settings of Complex WebQ.

Setting	**Hits@1 %**
Text only	13.857 %
KB only	52.000 %
Hybrid sources with early fusion	50.714 %

obtain better performance. Or it could be when the questions are too complex, information in text is scattered and unstructured. Adding text as another information source is nothing but adding more noise in the data, especially considering the significant drop in the Hits@1 score under text only setting.

6 Experiments: Further Analysis

6.1 Fact dropout

To improve the robustness of the model, [1] uses random fact dropout as regularization, to reduce the dependence of the model on KB. This is because extracting answers from KB is usually much easier than from text. We further test the impact of this technique on complex questions.

We keep the other parameters unchanged, but set fact-dropout rate ranging from 0.1 to 0.9 under hybrid setting, and plot the performance of the model in the following Figure 2. Similar to the trend described in [1], Hits@1 score reaches the highest when fact-dropout rate is around 0.1 to 0.3, then continuously drops until the fact-dropout rate reaches 0.9. Therefore, fact dropout can also prevent over dependency thus function as a good regularization technique for complex dataset.

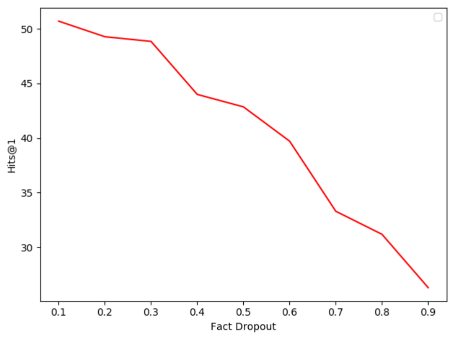

Figure 2: Impact of fact-dropout rate on Hits@1

6.2 Future improvement

Although questions in Complex WebQ are far more challenging, answers to these NLQs on the other hand, are usually simple KB entities. Further exploration could focus more on designing and experimenting on complex answers with relations, logic, correference or even combination of these.

7 Conclusion

Our project explores QA over KB using GraftNet on Complex WebQ dataset. We test under KB only, text only and hybrid sources settings, and compare with WebQuestionsSP and Wikimovies. Significant decrease in text only setting for complex questions is observed. Using hybrid sources does not help improve machine's comprehension for complex questions. Using fact-dropout as regularization also works for complex dataset.

Reference

[1] H. Sun, B. Dhingra, M. Zaheer, K. Mazaitis, R. Salakhutdinov, and W. W. Cohen, "Open domain question answering using early fusion of knowledge bases and text," *arXiv preprint arXiv:1809.00782*, 2018.

[2] N. Chakraborty, D. Lukovnikov, G. Maheshwari, P. Trivedi, J. Lehmann, and A. Fischer, "Introduction to neural network based approaches for question answering over knowledge graphs," *arXiv preprint arXiv:1907.09361*, 2019.

[3] X. Yao and B. Van Durme, "Information extraction over structured data: Question answering with freebase," in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2014, pp. 956–966.

[4] S. W.-t. Yih, M.-W. Chang, X. He, and J. Gao, "Semantic parsing via staged query graph generation: Question answering with knowledge base," 2015.

[5] D. Lukovnikov, A. Fischer, J. Lehmann, and S. Auer, "Neural network-based question answering over knowledge graphs on word and character level," in *Proceedings of the 26th international conference on World Wide Web*, 2017, pp. 1211–1220.

[6] J. Bao, N. Duan, Z. Yan, M. Zhou, and T. Zhao, "Constraint-based question answering with knowledge graph," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2503–2514.

[7] Y. Zhang, H. Dai, Z. Kozareva, A. J. Smola, and L. Song, "Variational reasoning for question answering with knowledge graph," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[8] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[9] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[10] Y. Li, D. Tarlow, M. Brockschmidt, and R. Zemel, "Gated graph sequence neural networks," *arXiv preprint arXiv:1511.05493*, 2015.

[11] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*, Springer, 2018, pp. 593–607.

[12] Q. Wang, Z. Mao, B. Wang, and L. Guo, "Knowledge graph embedding: A survey of approaches and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 12, pp. 2724–2743, 2017.

[13] X. Huang, J. Zhang, D. Li, and P. Li, "Knowledge graph embedding based question answering," in *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, 2019, pp. 105–113.

[14] A. Talmor and J. Berant, "The web as a knowledge-base for answering complex questions," *arXiv preprint arXiv:1803.06643*, 2018.

[15] W.-t. Yih, M. Richardson, C. Meek, M.-W. Chang, and J. Suh, "The value of semantic parse labeling for knowledge base question answering," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2016, pp. 201–206.

[16] H. Sun, T. Bedrax-Weiss, and W. W. Cohen, "Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text," *arXiv preprint arXiv:1904.09537*, 2019.

[17] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[18] B. Dhingra, K. Mazaitis, and W. W. Cohen, "Quasar: Datasets for question answering by search and reading," *arXiv preprint arXiv:1707.03904*, 2017.

[19] M. Joshi, E. Choi, D. S. Weld, and L. Zettlemoyer, "Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension," *arXiv preprint arXiv:1705.03551*, 2017.

[20] M. Seo, T. Kwiatkowski, A. P. Parikh, A. Farhadi, and H. Hajishirzi, "Phrase-indexed question answering: A new challenge for scalable document comprehension," *arXiv preprint arXiv:1804.07726*, 2018.

[21] Y. Sun, L. Zhang, G. Cheng, and Y. Qu, "Sparqa: Skeleton-based semantic parsing for complex questions over knowledge bases," *arXiv preprint arXiv:2003.13956*, 2020.

[22] T. H. Haveliwala, “Topic-sensitive pagerank,” *Proceedings of the 11th international conference on World Wide Web*, 2002.

[23] T. H. Haveliwala, “Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search,” *IEEE transactions on knowledge and data engineering*, vol. 15, no. 4, pp. 784–796, 2003.

[24] M. Michael, H. Erik, and G. Otis, “Lucene in action: Covers apache lucene 3.0,” *Greenwich, CT: Manning Publications Co*, 2010.

[25] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.