LinkSO: A Dataset for Learning to Retrieve Similar Question Answer Pairs on Software Development Forums*

Xueqing Liu
UIUC
USA
xliu93@illinois.edu

Chi Wang
Microsoft Research
USA
chiw@microsoft.com

Yue Leng
UIUC
USA
yueleng2@illinois.edu

ChengXiang Zhai
UIUC
USA
czhai@illinois.edu

ABSTRACT

We present LinkSO, a dataset for learning to rank similar questions on Stack Overflow. Stack Overflow contains a massive amount of crowd-sourced question links of high quality, which provides a great opportunity for evaluating retrieval algorithms for community-based question answer (cQA) archives and for *learning to* retrieve similar questions. However, due to the existence of missing links, one question is whether question links can be readily used as the relevance judgment for evaluation. We study this question by measuring the closeness between question links and the relevance judgment, and we find their agreement rates range from 80% to 88%. We conduct an empirical study that evaluates existing retrieval models' performance on LinkSO. While existing work focuses on non-learning approaches, our preliminary exploration that assembles simple learning models shows great potential for further improving the retrieval performance with machine learning.

CCS CONCEPTS

• **Information systems → Learning to rank**;

KEYWORDS

Information retrieval; Community-based question answering

ACM Reference Format:
Xueqing Liu, Chi Wang, Yue Leng, and ChengXiang Zhai. 2018. LinkSO: A Dataset for Learning to Retrieve Similar Question Answer Pairs on Software Development Forums. In *Proceedings of the 4th ACM SIGSOFT International Workshop on NLP for Software Engineering (NL4SE '18), November 4, 2018, Lake Buena Vista, FL, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3283812.3283815

1 INTRODUCTION

In recent years, community-based question answering (cQA) forums (e.g., Stack Overflow, Quora, Yahoo! Answers) attract large communities of users who regularly search, browse and post on the forums. Stack Overflow has more than 9 million users that mostly consist of software developers. Over the past decade, the Stack Overflow community has generated a large amount of question and answer archives of high quality. Besides helping the question askers, another long-lasting value of Stack Overflow is to help future users find answers by supporting the browsing of existing questions and answer archives [3].

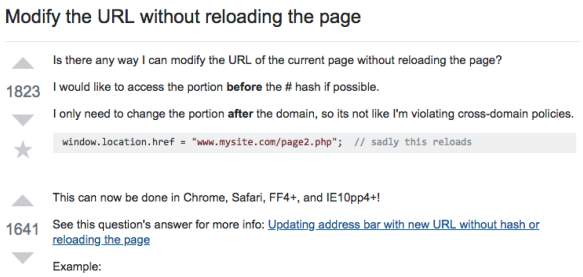

Figure 1: An example of Stack Overflow question link

To support user browsing, cQA forums provide recommender systems that display similar questions to the current one, e.g., "*Related*" questions on Stack Overflow. A major challenge in retrieving similar questions is bridging the knowledge gap. That is, question askers often miss keywords due to their lack of knowledge, and such missing keywords increases the mismatch between question pairs [12]. In addition to the knowledge gap in the general domain, questions on Stack Overflow bring in further challenges in understanding the programming concepts and their relations. For instance, the following two questions share almost the same word token set, but they convey very different meanings: (1) "*How to check whether Java plugins are installed or not in a browser?*" (2) "*How to check if Java is installed on system (*not* in browser)?*".

Existing work has studied the problem of bridging the knowledge gap for cQA retrieval [10, 12, 16]. However, the majority of previous work relies on heuristic-based approaches only, i.e., they do not leverage any machine learning approach (in addition, they focus on general domain data instead of the software development domain). Without learning as the feedback process, it is challenging for the ranking algorithm to find questions based on their semantic relatedness (such as the Java browser examples above).

To the best of our knowledge, little work exists on building datasets for learning to retrieve cQA questions. On the other hand, Stack Overflow "*Linked*" questions could potentially be used for learning and evaluating ranking algorithms. Linked questions are question pairs that are manually linked by community users. An example of such question link is shown in Figure 1, where the question answerer provides a link to another Stack Overflow question to help the question asker find more information. Three major types

*We thank Luyu Gao for his help in exploring MatchZoo. This work is supported in part by National Science Foundation grant numbers CNS-1801652, CNS-1408944, CNS-1513939.

NL4SE '18, November 4, 2018, Lake Buena Vista, FL, USA

ACM ISBN 978-1-4503-6055-5/18/11…$15.00
https://doi.org/10.1145/3283812.3283815

of question links on Stack Overflow are: (1) *answer link*. A link in the answer (e.g., Figure 1) provides external knowledge that helps answering the question; (2) *question link*. A link in the question is often used to clarify the question, e.g., by pointing out the similarity and difference with another question; (3) *duplicated questions* [2].

As of August 2018, there are more than 4.6 million question links on Stack Overflow. Meanwhile, our observation shows that such links seem to indicate a high degree of semantic-relatedness. Namely, two semantically-related questions can be linked even though their tokens are different. However, does such observation imply that question links can be used as the gold standard for evaluating question relevance? Due to the scale of Stack Overflow, there is no guarantee that every pair of relevant questions are linked together. As a result, there can exist a substantial amount of *missing links* which may lower the trustworthiness of evaluation using question links. The question we want to study is, can we still use question links for evaluation, albeit the existence of missing links? To answer this question, we collect a dataset as described in Section 2. We compare the agreement rates between question links and manually annotated relevance judgment (Section 3). The results show that question links preserve the orders of the relevance judgment with a probability of 80% to 88%.

The goal of the LinkSO dataset is for future work to propose new models (such as neural network models) to improve cQA retrieval in the SE domain. The dataset may also help with designing retrieval models in the general domain. LinkSO consists of three datasets corresponding to three popular programming languages (Python, Java, JavaScript), 690K question pairs and 26K linked question pairs (i.e., positive examples). We conduct a preliminary study on LinkSO by comparing the performance of learning to rank approaches with non-learning approaches. The results show that learning-based approaches slightly outperform state-of-the-art non-learning approach (Section 4). Because the non-learning approach is designed specifically for cQA retrieval whereas the learning approach is for a more general task, such result may imply that learning-based approaches have the potential to further improve the retrieval results with a specialized model for the task, e.g., by modeling the interaction between different fields in the candidate question. The dataset can be found on the LinkSO website [1].

2 PROBLEM FORMULATION/DATA PREPARATION

In this section, we formally define the retrieval problem studied in this paper, and the process for preparing the dataset.

Problem formulation. Given a (query) question q_1 from Stack Overflow, we study the problem of retrieving the top-K similar questions q_2's from all SO questions. Following the common setting in previous work [10, 12, 16], we consider four data fields for measuring the similarity between q_1 and q_2: the title of q_1, and all the three fields of q_2 (i.e., q_2's title, body, and answer(s)). The relevance judgment between q_1 and q_2 is whether either one of them has a link to the other. Therefore, if a question q_1 is not linked, we remove such q_1 from the dataset.

Data preparation. We extract the LinkSO dataset from Stack Overflow's data dump in April 2018 by leveraging the following four-step process:

Table 1: Statistics of LinkSO dataset

	#link	#q_1	train	dev	test
Python	7,406	6,410	4,910	500	1000
Java	9,743	8,448	6,948	500	1000
JavaScript	9,444	8,069	6,569	500	1000

Step 1: data cleaning. We perform conventional data cleaning steps such as the removal of non-ascii characters, email addresses, URLs, and code blocks. When a question has more than two answers, we keep only the top-2 voted answers, use their concatenation as the answer field, and discard the other answers.

Step 2: pre-processing. We remove English stop words from all the data. All words are stemmed using the Porter stemmer. The vocabulary size of LinkSO is 55K.

Step 3: preparing tag-based datasets. The original size of the Stack Overflow dataset is large (15 million questions). To improve the efficiency of retrieval, we prepare three smaller datasets for experiments, based on three popular question tags (JavaScript, Python, and Java).

Step 4: caching candidate similar questions. After Step 3, each dataset contains approximately 1 million questions, which is still too large for performing efficient retrieval on a large number of queries. For efficiency, we cache a small number of (30) candidate questions q_2's for each query question q_1 and use the re-ranking results on this smaller dataset for evaluation. The caching ranks q_2 by the TF-IDF score between q_1 and q_2's titles. If no relevant q_2 is found among top-30, we discard the query q_1.

After step 1-4, we obtain three datasets (split into train/devx/test folds) containing 26,593 query questions in total (Table 2).

3 QUALITATIVE STUDY

To what extent can we trust the evaluation results on LinkSO? The data preparation process for LinkSO (Section 2) assumes the relevance judgment is approximated by whether or not the link exists. How close are links to the relevance?

The missing links. The above question is equivalent to two sub-questions: (1) if q_1 and q_2 are linked, are they relevant? (2) if q_2 is relevant to q_1, must there be a link between them? An immediate answer to question (2) is no. Theoretically speaking, to guarantee that every relevant pair is linked, every pair of questions (15 million choose $2 \approx 10^{14}$) must be manually judged, which is intractable even with crowdsourcing. In practice, we can also find a significant amount of unlinked yet relevant question pairs, e.g., question "*How to revert a Git repository to a previous commit*" and "*How do I go back to previous Git Commit?*". Other factors may further lower the recall of linked relevant questions, such as users' awareness of the question link feature.

Agreement test. Knowing that there may exist a substantial amount of missing links, can we still use the question links for relevance judgment? In other words, to what extent do links preserve the correct orders of relevance judgment? To answer this question, we propose to run the following test on the LinkSO dataset: Given a query question q_1, between question q_2 and q_3 where q_2 is linked to q_1 and q_3 is not, how likely is q_2 more relevant to q_1 than q_3?

Relevance judgment through manual annotation. We obtain the relevance judgment for q_2 and q_3 through manual annotation.

More specifically, for each dataset, we randomly sample 50 (q_1, q_2, q_3) triples such that q_2 is linked to q_1 and q_3 is not. We display the triples for manual annotation (all the three fields are displayed), with q_1 displayed first, q_2 and q_3 displayed next, where the order of q_2 and q_3 is randomly shuffled so the annotators cannot observe which one is linked. We ask the annotators to select the more relevant question between q_2 and q_3. The annotators are three of the authors. We avoid leveraging crowdsourcing due to the domain knowledge required in the annotation. The three authors have an average of 10 years of programming experience and 5 years of using Stack Overflow. On average, the annotators spend 5.6 minutes annotating each triple.

Result analysis: link/relevance agreement rate. We use the voted result among the three annotators as the relevance judgment. In Figure 2 (right) we plot the agreement rates between the relevance judgment and the question links. We can observe that the average agreement rates are 80% (Python), 82% (Java) and 88% (JavaScript). As a result, the relevance judgment mostly agrees with the question links. Among the three tags, JavaScript has the highest agreement rate. In addition to the overall agreement rate (overall), we plot the average agreement rates under two specific cases. In the first case (denoted by preserve in Figure 2), the linked question is textually more similar than the unlinked question; whereas in the second case (denoted by reverse in Figure 2), the unlinked question is textually more similar. By observing both the left and right plot in Figure 2, we can see that the annotators generally more easily reach an agreement in the preserve case.

Result analysis: annotator agreement rate. In Figure 2 (left), we plot the agreement rates among the three annotators, which is the proportion of questions where all the three authors select the same order. The overall agreement rates range from 32% (Java) to 70% (JavaScript). Notice the agreement rate if all annotators randomly select their orders is 25%. JavaScript still shows the highest agreement rate, which may be because JS questions contain more front-end terms which make the questions easier to understand.

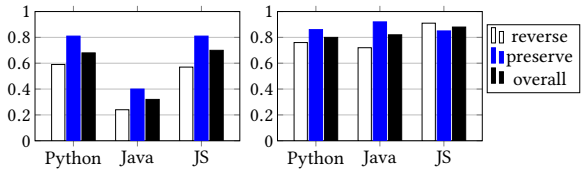

Figure 2: Left: agreement rates between the three annotators, right: agreement rates between the voted human judgment and the question link.

In summary, Figure 2 indicates that question links preserve the ranking orders of the ground truth relevance judgment with high probabilities (80% to 88%).

Discussion on the choice of the top-K value. The agreement rate of a dataset is associated with the top-K value, i.e., the number of cached top-ranked candidate questions for re-ranking (Step-4 in Section 2). The top-K of LinkSO is set to 30. However, if top-K is large, it can make it difficult to obtain a high agreement rate. The reason is that, if the linked question q_2 is ranked (based on title TF-IDF) at the K-th position, while q_3 is at the first position, it is difficult to tell that q_2 is more relevant than q_3 because the latter looks much more similar to q_1.

4 EMPIRICAL STUDY ON EXISTING RANKING ALGORITHMS

We conduct a preliminary empirical study on the performance of six ranking algorithms on the LinkSO dataset. The goal of our study is to answer the following question: to what extent does machine learning help with retrieving similar questions on Stack Overflow?

To date, the majority of existing work on cQA retrieval has not leveraged machine learning [12, 16], potentially because there exist few publicly available datasets with a large amount of relevance judgment. However, semantic matching techniques for *question answering* has been well explored [7, 8, 13], i.e., learning to rank answers based on the question. We can leverage such techniques to help with our task. The question in semantic matching is replaced by the title of the query question q_1, whereas the answer in semantic matching is replaced by the title, body, and answer of the candidate question q_2 respectively.

To compare learning approaches with non-learning approaches, we first evaluate the three non-learning approaches below:

TF-IDF. The TF-IDF approach considers the cosine similarity of the TF-IDF vectors, and the output score is the linear interpolation of scores in each field.

BM25. Similar to TF-IDF, the output score is the linear interpolation of the BM25 score in each field.

TransLM [12]. The machine translation-based language model is the state-of-the-art retrieval model for cQA archives. The model is based on the KL-divergence retrieval framework, where the document language model is smoothed by the machine translation language model from the answer to the question field, plus the basic Jelinek-Mercer smoothing.

Then, we evaluate the three approaches below based on semantic matching:

DSSM [8]. The input feature of DSSM is the original word tokens in the three fields, therefore it requires to train a model with a large number of parameters. Existing work that studies DSSM or similar networks usually leverage billions of search engine click logs for the experiments [8]. The output score is the linear interpolation of the DSSM scores in the three fields (same for DRMM and aNMM).

DRMM [7]. The input feature of DRMM is a matrix, where the numbers of rows and columns are both bounded by constants (typically 10 to 20), therefore the model contains only a few hundred parameters, which are orders of magnitude smaller than that of DSSM.

aNMM [13]. The input feature of the attention-based neural matching model is the same as DRMM, therefore it also requires only a small number of parameters. Different from DRMM, aNMM leverages the attention mechanism over the question words.

Implementation details. We run the six models and evaluate their performance on the testing data of LinkSO. The translation language model in TransLM [12] is trained on the iBM-1 model from GIZA++ [6], and the training takes 8 hours on a machine with 48 processors. For training DSSM, DRMM and aNMM, we leverage a semantic-matching toolkit named MatchZoo [5]. The hyperparameters of DSSM, DRMM and aNMM follow the default configurations in MatchZoo, except that the dropout rates for DSSM are set to 0.95. The weights for the linear interpolation are empirically set to 0.5 (title), 0.25 (body), and 0.25 (answer) in all the six models.

Table 2: Empirical study comparing the performance between non-learning vs learning-based approaches.

	Python			Java			JavaScript		
	R	@5	@10	R	@5	@10	R	@5	@10
TF-IDF	.299	.301	.360	.285	.282	.352	.305	.315	.378
BM25	.313	.320	.384	.311	.321	.382	.329	.344	.412
TransLM	.468	.502	.553	.455	.487	.544	.483	.528	.573
DSSM	.430	.461	.519	.416	.443	.500	.424	.461	.519
DRMM	.478	.509	.564	.465	.506	.555	**.500**	.546	.595
aNMM	**.481**	**.514**	**.570**	**.472**	**.507**	**.559**	.499	**.548**	**.597**

Result analysis. In Table 2 we display the results of our empirical study on the six retrieval models. The evaluation metrics we use are the mean reciprocal rank (denoted by R), NDCG@5 (denoted by @5), and NDCG@10 (denoted by @10). From Table 2 we can make the following observations. First, the best-performing learning-based approach (aNMM) is slightly better than the state-of-the-art non-learning approach (TransLM). We run statistical significance tests between aNMM and TransLM, but the T-test results are not significant. Second, among all the non-learning approaches, the translation language model outperforms BM25 by nearly 50%. We further conduct an ablation study on TransLM, which shows that the translation language model itself contributes to only 1% of the improvement, while most of the improvement comes from the KL-divergence framework and the Jelinek-Mercer smoothing. Third, among all the learning approaches, both aNMM and DRMM significantly outperform DSSM. This result may be explained by the contrast of the parameter space between the three approaches. Indeed, it could be challenging to train DSSM (with tens of thousands of parameters) with LinkSO. On the other hand, the results of aNMM and DRMM show the potential of improving the retrieval results with learning to rank approaches.

5 RELATED WORK

Community-based question and answer retrieval. A large body of existing work studies the retrieval of community-based question and answer archives. Most of such work focuses on improving the retrieval performance by bridging the knowledge gap between the question and answer field. Xue et al. [12] first propose to leverage a translation-based model to bridge such gap. Later work studies leveraging topic modeling [9], external knowledge-base [16], and mining user intent [11]. In Section 4, we skip evaluating the later work, because they are either not applicable [11] or the models are incremental compared with Xue et al. [12].

Datasets for community-based question answering. There exist multiple datasets for community question answering, including question and answer pairs from Yahoo! Answers and Baidu Zhidao. Nevertheless, these datasets do not contain the relevance judgment between question pairs. To the best of our knowledge, our dataset is the first large-scale dataset for learning to rank similar questions in the software development domain.

Question retrieval on software development forums. As an important step toward assisting software development, question retrieval has also been studied in particular for the software engineering domain. For example, researchers study automatically detecting duplicate question pairs so that the new question asker can quickly browse existing answers to the same question [2, 14, 15]. In addition, Chen et al. [4] study cross-lingual question retrieval to assist non-native speakers more easily retrieve relevant questions. We do not evaluate these approaches in Section 4 because our problem is different from duplication detection.

6 FUTURE WORK

Future work includes designing models to better capture the semantic-relatedness between question pairs. One potential direction is how to train the model to automatically learn the most critical information for deciding the question similarity. In our qualitative study (Section 3), we observe that the difference between question pairs can usually be captured by a few keywords. For example, the critical keyword for distinguishing "*How to check whether Java plugins are installed or not in a browser?*" and "*How to check if Java is installed on system (*not* in browser)?*" is *system*. Can we train the model to recognize such keywords? Another potential direction is studying the interaction between different fields, e.g., we observe that for a significant proportion of question q_1's, adding q_2's body and answer does not improve the performance. Can we improve the overall ranking performance by assigning different field weights for different q_1's?

7 CONCLUSION

In this paper, we propose LinkSO, a dataset for learning to retrieve similar questions on community question answering forums in the software engineering domain. Our qualitative study shows that the agreement rates between question links and the relevance judgment range from 80% to 88%. Our further empirical study reveals the potential for learning-based approaches to outperform state-of-the-art non-learning approaches.

REFERENCES

[1] Accessed: 2018-08-26. LinkSO website. https://sites.google.com/view/linkso.
[2] M. Ahasanuzzaman, M. Asaduzzaman, C. K. Roy, and K. A. Schneider. 2016. Mining duplicate questions in Stack Overflow. In *MSR*.
[3] A. Anderson, D. Huttenlocher, J. Kleinberg, and J. Leskovec. 2012. Discovering value from community activity on focused question answering sites: a case study of Stack Overflow. In *KDD*.
[4] G. Chen, C. Chen, Z. Xing, and B. Xu. 2016. Learning a dual-language vector space for domain-specific cross-lingual question retrieval. In *ASE*.
[5] Y. Fan, L. Pang, J. Hou, J. Guo, Y. Lan, and X. Cheng. 2017. MatchZoo: a toolkit for deep text matching. *CoRR abs/1707.07270* (2017).
[6] J. O. Franz and N. Hermann. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics* (2003).
[7] J. Guo, Y. Fan, Q. Ai, and B. W. Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *CIKM*.
[8] P. S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *CIKM*.
[9] Z. Ji, F. Xu, B. Wang, and B. He. 2012. Question-answer topic model for question retrieval in community question answering. In *CIKM*.
[10] X. Qiu and X. Huang. 2015. Convolutional Neural Tensor Network Architecture for Community-Based Question Answering. In *IJCAI*.
[11] H. Wu, W. Wu, M. Zhou, E. Chen, L. Duan, and H. Y. Shum. 2014. Improving search relevance for short queries in community question answering. In *WSDM*.
[12] X. Xue, J. Jeon, and B. W. Croft. 2008. Retrieval models for question and answer archives. In *SIGIR*.
[13] L. Yang, Q. Ai, J. Guo, and B. W. Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *CIKM*.
[14] E. W. Zhang, Q. Z. Sheng, J. H. Lau, and E. Abebe. 2017. Detecting duplicate posts in programming QA communities via latent semantics and association rules. In *WWW*.
[15] Y. Zhang, D. Lo, X. Xia, and J. L. Sun. 2015. Multi-factor duplicate question detection in Stack Overflow. *Journal of Computer Science and Technology* (2015).
[16] G. Zhou, Y. Liu, F. Liu, D. Zeng, and J. Zhao. 2013. Improving Question Retrieval in Community Question Answering Using World Knowledge. In *IJCAI*.