Replication of AnswerBot: Automated Generation of Answer Summary to Developers' Technical Questions

Pratik Patel†
Department of CS
University of Manitoba
Winnipeg, MB, Canada
patelpk3@myumanitoba.ca

Swetul Patel†
Department of CS.
University of Manitoba.
Winnipeg, MB, Canada.
patels15@myumanitoba.ca

INTRODUCTION

The prevalence of questions and answers on domain specific Q&A sites like Stack Overflow constitutes a core knowledge asset for the software engineering domain. Typically, developers formulate their questions as a query to some search engine like google, and the search engine returns a list of relevant posts that may contain answers. Then, developers' need to read the returned posts and digest the information in them to find the answers to their questions. This type of information seeking is rendered difficult by the sheer amount of questions and answers available on the Q&A site. In this work, we aim to develop an automated technique for generating answer summaries to developers' technical questions, instead of merely returning answer posts containing answers. The goal is to give developers relevant information regarding their query automatically, removing the need for the developer to manually search and read through the results from search engines or Q&A sites and reduce the overall effort for the developer.

RELATED WORK

Many text summarizations approaches have been applied in different software engineering tasks, aiming to reduce the developers' effort to read an immense quantity of information. Rastkar et al. proposed an extractive approach for automatic bug report summarization [1]. Their approach selects a subset of bug report comments by training a binary classifier to determine whether a comment should be selected or not. Andrea et al. proposed an approach *SURF* to produce a summary for user reviews [2]. SURF classifies each user review sentence to one of the user-intention categories, and groups together sentences covering the same topic. It then uses a sentence selection and scoring mechanism to generate the user-review summary. Different from the above studies, we focus on answer summarization on online Q&A sites which requires a different set of features. Additionally, we formulate our problem as a ranking problem instead of a classification one.

A number of studies have also proposed methods to identify relevant or high-quality posts in question and answering sites. Gottipati et al. proposed a semantic search engine framework to process posts in discussion threads to recover relevant answers to a user query [3]. They classify posts in the discussion threads into 7 categories (e.g., questions, answers, clarifying questions, junk, etc.) and use the category labels to filter less useful information to improve the search experience. Yao Et al. proposed an approach to detect high-quality posts in community question answering sites [4]. Different from the above studies, we not only identify relevant posts but also create summaries of these posts.

Popular search engines like Google can provide direct answers for some types of queries. For example, Google can extract a paragraph from the best answer of the Stackoverflow question "what are the differences between hashtable and HashMap?" as the answer for a query like "differences between hashtable and HashMap". However, Google does this only for

specific kinds of 5W+1H (who, what, where, when, why, how) questions, and it does not generate direct answers to all 5W+1H questions, e.g., "What are the differences between quicksort and bubble sort?". More importantly, Google extracts only a paragraph from one answer of one question, while our approach is multi-answer-posts summarization.

DATA COLLECTION

We will collect Java questions (i.e., questions tagged with Java) and their corresponding answers from Stack Overflow Data Dump of March 2016. These questions must have at least one answer. To ensure the quality of the question repository, we require that at least one of the answers of the selected questions is the accepted answer or has at least one vote. When collecting questions, we avoid duplicate questions of the already-selected questions, because duplicate questions discuss the same question in different ways and can be answered by the same answer. We use these Java questions and their answers as a repository for answering Java-related technical questions.

We randomly select another 100 questions and use the titles of these questions as query questions. We ensure that our question repository does not contain these 100 query questions and their duplicate questions. The randomly selected 100 query questions cover a diversity of aspects of Java programming. For example, some of them are related to language features, such as **multi-threading (e.g. How does volatile actually work?) and I/O (e.g., Can Buffered Reader read bytes?),** while others are related to many third-party libraries, such as TEST-Assertions **(e.g., Testing API which returns multiple values with JUnit)** and REST **(e.g., Is there an equivalent to ASP.NET WEB API in JAVA world?).** Some of the query questions are easy to answer **(e.g., How to convert String into Deformation java?),** while others are difficult **(e.g., How does volatile work?).** The diversity of these 100 questions can improve the generality of our study and reduce the bias that our approach might be only effective for a specific type of questions. The participants of the study are going to be students from the department of computer science who will score the results from the answer-bot as well as the two baselines. The participants are unaware which results they are looking at.

RESEARCH QUESTIONS

We conduct three user studies to answer the following three research questions, respectively:

RQ1: **How effective is our approach in generating answer summaries with relevance, useful and diverse information for developers' technical questions?**

Motivation. In the current information retrieval practice, developers retrieve relevant questions by entering their technical questions to a search engine. Then they have to manually browse the answer of the returned relevant questions to find the needed information. In contrast, our approach can automatically generate an answer summary of the key points in the answers of the returned relevant questions. We would like to investigate the overall performance of our approach in terms of the relevance, usefulness and diversity of the generated summary, compared with manual information seeking.

Approach. We will compare our approach against two baselines built based on Google search engine and Stack Search engine, respectively. We add "site:stackoverflow.com" to the query of Google search engine so that it searches only posts on Stack Overflow. For a query question, we use the first ranked Stack Overflow question returned by a search engine as the most relevant question. We assume that a developer would read the best answer (i.e., the accepted answer) or the answer with the highest vote if there is no accepted answer of the relevant question. We refer to the collected best or highest-voted answer of the two baseline approaches as the *Baseline_Google*

answer summary and the *Baseline_SO* answer summary, respectively.

For each query question, we provide the participants the answer summary generated by *Baseline_Google*, *Baseline_SO* and our approach, respectively. The participants do not know which answer summary is generated by which approach. They Are asked to score the three answer summaries from three aspects, i.e., relevance, usefulness and diversity. The generated summary involves diverse aspects of information. **The score is a 5-point scale, with 1 being "Highly Irrelevant/Useless/Identical" and 5 being "Highly Relevant/Useful/Diverse"**

RQ2: How effective is our approach's relevant question retrieval component?

Motivation. An answer summary is generated from the answers of some questions relevant to a query question. If the retrieved questions are not relevant to the query question, it is unlikely to generate a high-quality answer summary for the query question. Our question retrieval approach combines word embeddings with traditional IDF metrics. We would like to investigate the effectiveness of our method, compared with the traditional TF-IDF based methods and other word-document-embedding based methods.

Approach. We build three baseline approaches: TF-IDF based [5], word-embedding based document retrieval [6], and document-to-vector (Dov2Vec) based document retrieval [7]. For each query question, we collect the top-10 ranked questions retrieved by our question retrieval approach or one of the baseline approaches. We ask the participants to identify the relevant questions in the top-10 ranked questions. The participants do not know which approach generates which list of top-10 ranked questions. We use the following metrics in comparison of different approaches.

Top-K Accuracy: Given a query questions, if at least one of the top-k ranked questions is relevant, we consider the retrieval to be successful, and set the value $Success(q, top-k)$ to 1. Otherwise, we consider the retrieval to be unsuccessful, and set the value to $Success(q, top-k)$ to 0. Given a set of queries, denoted as qs, its top-k accuracy Top@k is computed as: $\boldsymbol{Top@k(qs) = \Sigma_{q \epsilon qs} Success(q, top-k)/|qs|}$. The higher the top-k accuracy score is, the better a relevant question retrieval approach ranks the first relevant question. In this paper, we set k = 1, 5 and 10.

Mean Reciprocal Rank: Given a query question, its reciprocal rank is the multiplicative inverse of the rank of the first relevant question in a ranked list of questions. Mean Reciprocal Rank (MRR) is the average of the reciprocal ranks of all queries in a set of queries:

$$\boldsymbol{MRR(qs) = \frac{1}{|qs|} \sum_{q \epsilon qs} \frac{1}{Rank(q)}}.$$

We denote *qs* as the set of queries. *Rank(q)* refers to the position of the first relevant question in the ranked list of questions returned by a relevant question retrieval approach for a query. The higher the MRR is, the higher the first relevant questions are ranked for a set of queries.

RQ3: How effective is our approach's answer paragraph selection component?

Motivation. To select the most relevant and salient answer paragraphs for summarizing, our approach considers three types of features of answer paragraphs, i.e., **query-related, user-oriented and paragraph content features**. We would like to investigate the impact of different types of features on the results of answer paragraphs selection.

Approach. We remove one type of features at a time from the full feature set and reserve the other two types. Thus, **three baseline approaches are built,** i.e., **without (w/o) query-related features, w/o user-oriented features, and w/o paragraph content features.** We let each approach output a ranked list of 10 answer paragraphs with the highest overall score. We take the union of the 40 answer paragraphs selected by our approach (with all features) and the three

baselines. Participants are asked to judge whether the selected paragraphs contain salient information relevant to the query question. They don't know which answer paragraph is selected by which approach. We use *Top@k* accuracy and MRR in the top 10 ranked candidate answer paragraphs to evaluate the performance of answer paragraph selection with and without certain types of features.

REFERENCES

[1] Rastkar, G. C. Murphy, and G. Murray, "Automatic summarization of bug reports,"IEEE Transactions on Software Engineering, vol. 40,no. 4, pp. 366–380, 2014.

[2] Di Sorbo, S. Panichella, C. V. Alexandru, J. Shimagaki, C. A.Visaggio, G. Canfora, and H. C. Gall, "What would users change in myapp? summarizing app reviews for recommending software changes," in Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering. ACM, 2016, pp. 499–510.

[3] S. Gottipati, D. Lo, and J. Jiang, "Finding relevant answers in software forums," inProceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering.IEEE Computer Society, 2011, pp. 323–332.

[4] Y. Yao, H. Tong, T. Xie, L. Akoglu, F. Xu, and J. Lu, "Detecting High-quality posts in community question answering sites, "information Sciences, vol. 302, pp. 70–82, 2015.

[5] S. Haiduc, G. Bavota, A. Marcus, R. Oliveto, A. De Lucia, and T. Men-zies, "Automatic query reformulations for text retrieval in software engineering," in Software Engineering (ICSE), 2013 35th International Conference on. IEEE, 2013, pp. 842–851.

[6] X. Yang, D. Lo, X. Xia, L. Bao, and J. Sun, "Combining Word Embedding with Information Retrieval to Recommend Similar Bug Reports," in Software Reliability Engineering (ISSRE), 2016 IEEE 27thInternational Symposium on. IEEE, 2016, pp. 127–137.

[7] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in Proceedings of the 31st International Conference on Machine Learning (ICML-14), 2014, pp. 1188–1196