SEMO: Searching Majority Opinions on Movies using SNS QA Threads

Jukyoung Lee
Korea University
Seoul, Korea
rudolph0724@korea.ac.kr

Yonghwa Choi
Korea University
Seoul, Korea
yonghwachoi@korea.ac.kr

Suhkyung Kim
Korea University
Seoul, Korea
suhkyungkim@korea.ac.kr

Seongsoon Kim
Korea University
Seoul, Korea
seongkim@korea.ac.kr

Jaewoo Kang
Korea University
Seoul, Korea
kangj@korea.ac.kr

ABSTRACT

Many people seek majority opinions by searching for question-answers that are uploaded by others or uploading their own questions on social media sites. However, people have to read through a large number of documents returned by search services to find the majority opinions. Moreover, even when users upload questions on social media sites, they cannot immediately obtain answers. To address these problems, we present Searching Majority Opinions System (SEMO), a novel majority opinion-based search system that uses QA threads uploaded on SNS and cQA websites. SEMO returns entities based on majority opinions for opinion-finding queries in real time. We also tackled a data sparsity problem using a novel query component expansion approach. To prove SEMO's usefulness in finding majority opinions, we implemented a prototype of SEMO for the movie domain. We believe that our method can cause a paradigm shift in opinion-finding query search and help people make decisions. SEMO is available at http://semo.korea.ac.kr/

Keywords

Majority Opinions Search, Social Question Answer, Entity Search

1. INTRODUCTION

People are often curious about the opinions of others, when making decisions. Nowadays, many are heavily dependent on commercial search engines for this reason [3]. When utilizing web search engines, users scan through a list of documents provided by a search engine and find an answer from one of them. This process is quite effective for "fact-finding" queries, such as "What is the capital of Canada?" or "Who was the president of the U.S. in 1975?" since the users are likely to obtain answers after reading only a small number of documents.

However, this process of current search engines fails to deliver satisfactory results for opinion-finding queries such as "What is a good thriller movie?" or "What is the best Christopher Nolan movie?" Different from fact-finding queries, opinion-finding queries do not have a correct answer. To obtain relevant answers to opinion-finding questions, we have to find majority opinions. However, due to time constraints, users tend to derive answers from a limited number of documents. Therefore, the answers may not represent the opinions of the majority.

One possible way to find majority opinions is to use community Question Answering (cQA) sites or social networking service (SNS) sites. There exists a large number of question and answers written by users on cQA sites such as Yahoo! Answers[1] and Reddit[2]. Moreover, most of the cQA sites provide search services, so users can explore questions that are related to their interests. In addition, social networking service sites such as Facebook and Twitter are utilized as a means of obtaining information. Morris et al. pointed out that 10% of SNS users have posted questions on SNS sites, and more than 40% of them were questions asking for opinions or recommendations [4].

To find the opinions of the majority, users may utilize search services provided by cQA or SNS sites. However, like commercial search engines, these sites return a large number of documents as a result and leave users with the burden of reading and analyzing voluminous documents. Additionally, these sites may not have answers to users' questions. Many of the questions posted on cQA or SNS sites are answered late or remain unanswered.

To address these problems and help SNS and cQA sites find the opinions of the majority, we suggest the following two methods. First, after processing and analyzing data, we return entities as a result, rather than returning numerous text documents. We expect that this will save users time and thus help them make decisions since they do not have to read countless documents. Second, we expand an input query to subqueries and aggregate the result of the subqueries. We suppose that this method can handle the cases where questions that matched an input query do not exist.

WWW 2016 Companion, April 11–15, 2016, Montréal, Québec, Canada.
ACM 978-1-4503-4144-8/16/04.
http://dx.doi.org/10.1145/2872518.2890553.

[1]https://answers.yahoo.com/
[2]https://www.reddit.com/

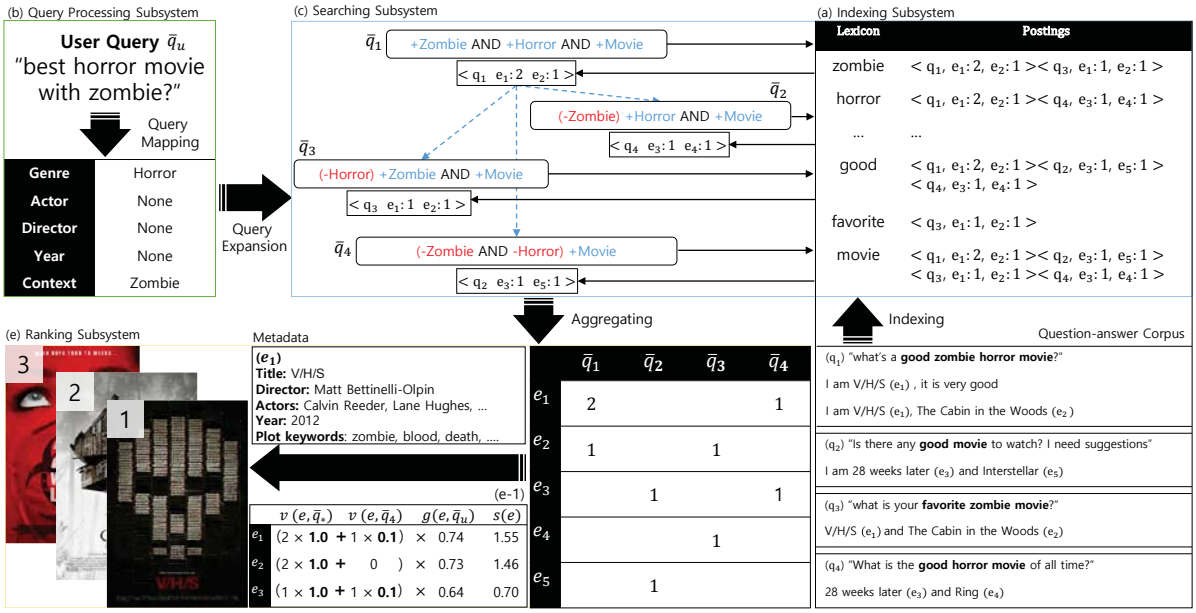

Figure 1: SEMO overview

To apply the above methods, we utilize the indexing structure that was introduced in [1] for returning processed and analyzed results in real time, and we extract and use components from user queries in the query processing step for expanding the queries.

The main contributions of our demo are summarized as follows:

- Our proposed novel system, SEMO, uses question-answers that are uploaded on social media sites to answer opinion-finding queries. SEMO returns a ranked list of movies that is based on majority opinions for opinion-finding queries in real time.
- We address a data sparsity problem using query expansion and a filtering method. SEMO resolves input queries that do not match any questions in the question-answer corpus.

2. SYSTEM OVERVIEW

As a prototype of SEMO, we created a vertical search engine in the movie domain. For our prototype, we constructed a question-answer corpus and movie metadata by collecting data from social media and IMDb[3]. The detailed explanation about data collection is provided in Section 3. Figure 1 illustrates an overview of SEMO. We highlight and explain in detail the following major subsystems: indexing, query processing, searching and ranking.

1. Indexing Subsystem: For real-time query processing, we apply the index scheme suggested by [1] to index our question-answer corpus. The index maps each word that appears in questions to each word's location in question-answer threads (a). Moreover, SEMO's index includes a list of movies that appear in answers of respective questions. For instance, let us assume the two movies *V/H/S* (e_1) and *The Cabin in the Woods* (e_2) are given as answers to the question, "What's a good zombie horror movie?" (q_1). In this case, the index includes additional information <q_1,e_1:2,e_2:1>, where the numbers indicate the count of each movie extracted from the answers of (q_1). Extracting movies is done by matching answers to movies in the dictionary built in advance. The dictionary of movies is built from the titles of movies in our metadata.

2. Query Processing Subsystem: In this step, we convert a natural language query to a predefined structure. A predefined structure represents a natural language query with combinations of components such as director, genre, and context. For example, the query "best horror movie with zombie?" is converted to the predefined structure as shown in (b). We made a dictionary for each component. Also, our system maps natural language queries to predefined structures by dictionary matching. We label the words that are not in the dictionaries as "context." For example, the directors or actors who do not exist in our dictionary or the sub-genres such as "vampire" or "zombie" are categorized as "context." As shown in (b), the keyword "zombie" is labeled as "context."

3. Searching Subsystem: SEMO searches the question-answer corpus to select movie candidates to be listed on a final system result. In this searching subsystem, as a search query, we use components which were extracted from user query $\bar{q_u}$ in the query processing subsystem. However, the number of questions that contain all the combinations of certain components may be very small, or some questions may not even exist. To tackle this sparsity problem, we generate an expanded query by combining existing components. Expanded queries $E(\bar{q_u})$ are expressed using Boolean operators such as "AND," "OR," "+" (for required) and "-" (for prohibited). Expanded queries include a base query such as $\bar{q_4}$ which consists of only the word "movie." Expanded queries are shown in (c).

4. Ranking Subsystem: In the ranking subsystem step, we aggregate and rank the movie candidates extracted during the searching process. Some movie candidates may not

[3]http://www.imdb.com/

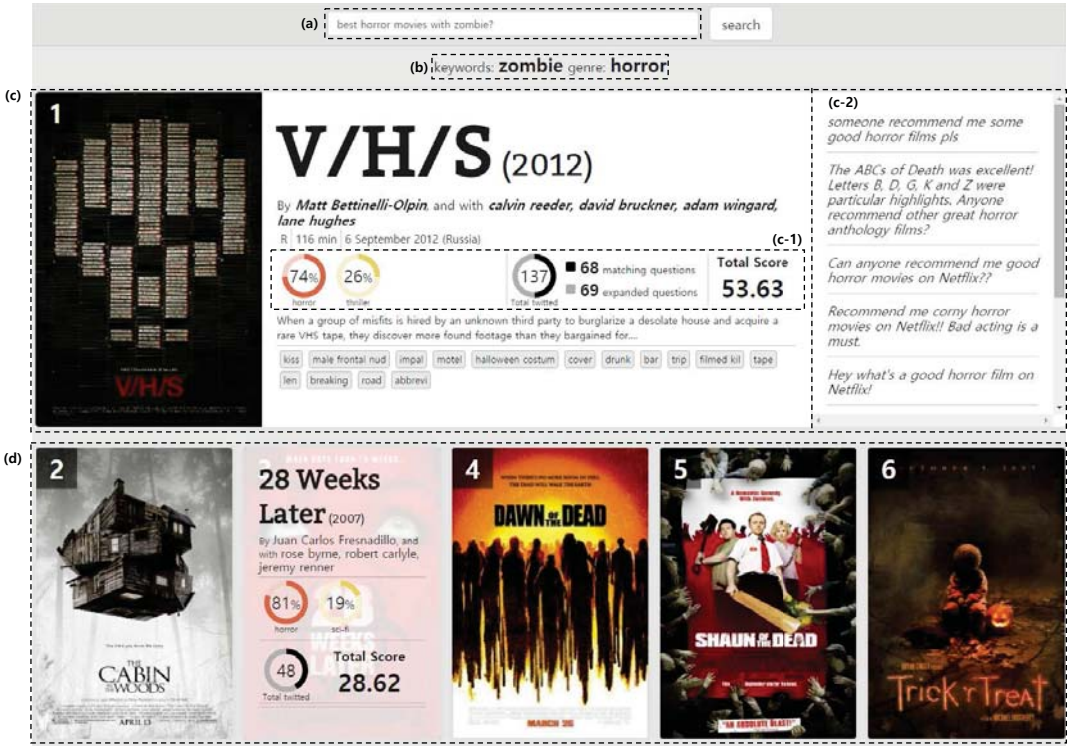

Figure 2: SEMO Movies - search result view.

correspond to a user's input query. For example, the movie set retrieved from $\bar{q_2}$ falls under the horror genre, but zombies do not appear in some of the movies. To address this problem, we filter movies if their attributes do not match the analyzed components. For instance, *Ring* (e_4) is filtered in this process because its "plot summary keyword" attribute does not match previously analyzed components, and *Interstellar* (e_5) is filtered since its "genre" does not match. If the components extracted from the query are labeled as "context," we compare these "context" components with every attribute of all the movies in the candidate movie set.

Next, we calculated the score of each movie that is included. Then we ranked movies according to the calculated scores. We also assigned a weighted value to the movie genre in the metadata to reflect each movie in our score. Note that many movies are annotated with multiple movie genres. For example, *Scary Movie 2 (2001)* belongs to the genres of comedy and horror. However, we believe that this movie is a somewhat inappropriate answer to the query "good horror movie" since this movie has many comical characteristics.

To solve this issue, we utilized reviews to assign a weighted value of genre to each movie. We first selected movies that belong to only one genre. Then for each genre, we created one huge review document by merging all the reviews assigned to one genre. Next, we converted each document to tf-idf term vector. Using this process, we generated a genre vector for each genre in our metadata. We also created a movie review vector by merging the reviews of a movie into one document which is also done when creating a genre vector. Last, we calculated the cosine similarity between the genre vector and the movie review vector, and decided its genre weight. We calculated the similarity score using the review vector of *Scary Movie 2* with horror vector and comedy vector, respectively. Using this process, we could assign weighted values of 43% and 57% to the horror genre and comedy genre, respectively. The score of each candidate movie $s(e)$ is calculated as in (1).

$$s(e) = \sum_{\bar{q_i} \in E(\bar{q_u})} v(e, \bar{q_i}) \times w(\bar{q_i}) \times g(e, \bar{q_u}) \quad (1)$$

In Equation (1), $v(e, \bar{q_i})$ refers to the number of votes for e in $\bar{q_i}$, $w(\bar{q_i})$ refers to the weighted value of $\bar{q_i}$ and $g(e, \bar{q_u})$ refers to the weighted value of the genre of e that is relevant to the user query $\bar{q_u}$. In case $\bar{q_u}$ does not include the movie genre, the value of $v(e, \bar{q_i})$ is set to 1. In the SEMO prototype, we assigned $w(\bar{q_i})$ to 1, except for the weighted value of a base query $\bar{q_4}$. We decreased the weighted value of a base query to 0.1, since it is much more general when compared with other queries.

Figure 1 (e-1) shows the scoring process for movies ranked first, second, and third. For example, *V/H/S* obtained one vote for $\bar{q_4}$ and two votes for expanded queries denoted as $\bar{q_*}$ (e.g., $\bar{q_1}, \bar{q_2}, \bar{q_3}$). The final score of *V/H/S* is $(2 \times 1.0 + 1 \times 0.1) \times 0.74 = 1.55$ where 0.74 is the weighted value for the horror genre.

3. DATA COLLECTION

1. Twitter Question & Answer DataSet: We consider tweets that contain questions as "questions" and comments attached to tweets as "answers." To collect public tweets that are created in real time, we use Twitter Stream API[4]. We only collect tweets that are in English and contain

[4]https://dev.twitter.com/streaming/overview/

the words "what," "where," or "who," which is also known as the W3 words, and a question mark. We collected a total of 185 million tweets from July 2015 to November 2015.

To classify tweets that contain questions, we applied rule based methods as was done in [2]. We first generated rules that classify movie-related questions such as "What is the best (...) movie?" and "Can anyone recommend a good (...) movie?" Then we selected only tweets that satisfy these rules. Furthermore, to reduce noise, we eliminated tweets that included hashtags (#) or URLs.

To collect question-answer threads, we gathered conversations from collected question tweets containing questions. To gather conversations, we extracted some information such as ID and user_name from each collected tweet. Next, based on the extracted information, we generated accessible URLs for the mobile Twitter page. Using the generated URLs, we obtained Twitter conversations. We finally collected 52,573 question-answer threads (343 tweets per day on average).

2. Reddit DataSet & IMDb: Reddit is a community question-answering website. A user can upload a posting on a topic of his or her interest, also called a "subReddit," and other users can discuss it. To collect threads on Reddit, we use the Python Reddit API Wrapper[5] (PRAW). Utilizing PRAW, users can collect Reddit threads in subReddits that are related to their interests. We collected the following three subReddits:"AskReddit," "movies," and "movie suggestions." Among the data collected, we selected only question Reddit threads based on the classification rules that were applied to Twitter data. Using the rules, we collected 3,513 Reddit threads.

We crawled data from IMDb to construct metadata. The metadata includes information such as movie genre, director, actor, plot summary keywords, and so on.

4. DEMONSTRATION

Figure 2 shows our system's result for the query "Best horror movie with zombie." As shown in (a), a user inputs query keywords in the search box. Our system shows the analyzed results of the input query (b) and the ranked list of movies for the input query (c, d). (c) shows a movie that was ranked first by our system, and (d) displays movies that were ranked second to sixth. The first-ranked movie is provided with the movie poster and other attributes.

We also give additional information which is analyzed by our system (c-1). The first two pie charts show the weighted value of each genre. The third pie chart shows the number of times a movie was mentioned on Twitter. The total tweet count is separated into two parts: the "expanded query" and the "base query." In the last part, (c-1) shows the calculated score from the system. As shown in (c-2), the system provides questions that were mentioned on Twitter.

In (d), our system shows the movies ranked second to sixth. When a user mouses over a movie poster, general information of the movie is shown. When the user clicks on the poster, detailed information is shown, as in (c).

5. RELATED WORK

Many of the recent research studies focus on understanding the characteristics of question-answers uploaded on SNS sites [2, 4]. Several studies have surveyed and analyzed samples of data. In [4], they analyzed survey replies from 624 Microsoft employees. The survey was about motivations and behavioral characteristics of uploading questions on SNS sites. In [2], the authors collected data from Twitter, and analyzed them in a taxonomic manner.

Qaster[6] is a search engine system which is most relevant to our system SEMO. Qaster collects question-answer conversations from Twitter and enables users to input queries to retrieve them. Qaster faces the same problem as that faced by commercial search engines and cQA services. The system returns numerous documents of question-answer conversations as a result, leaving the users to read through the documents.

6. CONCLUSION AND FUTURE WORK

In this work, we presented SEMO, a novel majority opinion-based search system that uses QA threads on SNS and cQA websites. SEMO returns entities based on majority opinions for opinion-finding queries in real time. We also tackled the data sparsity problem using the novel query component expansion approach. To the best of our knowledge, our work is the first majority opinion search system that uses question-answer threads uploaded on social sites.

Although the pilot system shows satisfactory results, there is still much room for improvement. Our future work will focus on developing a more fine-grained language processing module that includes NER query processing.

7. ACKNOWLEDGEMENTS

This work was supported by the National Research Foundation of Korea (NRF-2014R1A2A1A10051238, 2012M3C4-A7033341).

8. REFERENCES

[1] J. Choi, D. Kim, S. Kim, J. Lee, S. Lim, S. Lee, and J. Kang. Consento: a new framework for opinion based entity search and summarization. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 1935–1939. ACM, 2012.

[2] M. Efron and M. Winget. Questions are content: a taxonomy of questions in a microblogging environment. *Proceedings of the American Society for Information Science and Technology*, 47(1):1–10, 2010.

[3] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proceedings of the 19th international conference on World wide web*, pages 431–440. ACM, 2010.

[4] M. R. Morris, J. Teevan, and K. Panovich. What do people ask their social networks, and why?: a survey study of status message q&a behavior. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 1739–1748. ACM, 2010.

[5]https://praw.readthedocs.org/en/stable/

[6]http://www.qaster.com/