International Journal of Computer Science and Applications,
©Technomathematics Research Foundation
Vol. 14, No. 2, pp. 31 – 46, 2017

INFORMATION RETRIEVAL FOR QUESTION ANSWERING SYSTEM USING KNOWLEDGE BASED QUERY RECONSTRUCTION BY ADAPTED LESK AND LATENT SEMANTIC ANALYSIS

SARAVANAKUMAR KANDASAMY, ASWANI KUMAR CHERUKURI

School of Information Technology and Engineering,

VIT University,

Vellore, Tamilnadu, India
ksaravanakumar@vit.ac.in, cherukuri@acm.org

The purpose of accessing search engines by most of the users is for answers rather than set of documents. Identifying set of relevant documents for a question is the first part of any question answering systems. The accuracy of a question answering system is highly depending on the quality of the set of documents that have been chosen for further investigation. In this paper, a method is proposed to improve the information retrieval part of the open domain question answering system. The authors found the list of documents that may be relevant to the given question and rank them by using a) alternate queries to increase the quality of the context represented in the query without compromising the semantic meaning, b) search engine results and c) application of the concept of Latent Semantic Analysis (LSA). The result shows that the top ranked documents for a given factoid question contain solid answers.

Keywords: Information retrieval; Question answering system; Query disambiguation; Natural language processing; Query expansion; LSA.

1. Introduction

Due to the invention of World Wide Web and Internet, we have access to an enormous amount of data that are available worldwide. The available information is growing in a faster rate every day. This growth rate causes one to ask the question “How easily one could access the data?” We have *Search Engines* as the very simple answer to deal with this issue. Search engines are handling the user queries to some extent. They are able to fetch all the documents available in the Internet to your screen in mill-seconds. Most of the times those documents are ranked appropriate to the search query raised by the user. But the size of the available data always increases the number of documents as search results. Now, the major problems are “Are we getting all the websites that are very much related to our query?”, “Are the results are optimally ranked?”, “How do we get the answer we are looking for?” etc. If these questions are not properly answered, then there arise various problems for any user who use search engines. Some of them are as follows; a) one may not get proper words to frame his/her query, b) one may not be good in English to frame the query, c) the suggested results may not contain the result which is expected by the user, d) the fatigue of user who could not get the relevant results, e) the expected results may need to be searched from a suggested document which is very large,

f) the time available to anyone for searching an answer, etc. These problems may be answered by a system that consider query related words, alternate query statements and extraction of relevant portions of the documents for the users. Question Answering system (hereafter referred as QA system) is such a system that tries to ease the work of any user. Any system of the said nature should be able to retrieve all the top ranked documents that are optimally related to the given user question and then extracts the relevant answers from them.

The processing of a QA system may broadly have three stages, i.e., *question analysis* where the user question is parsed for understanding purposes, question is classified and finally questions are reformulated; *document analysis* where the candidate documents are extracted and possible answers are identified; and *answer analysis* where the candidate answers are extracted and ranked optimally [Dwivedi and Singh, (2013)] [Hazrina *et al.*, (2017)]. A QA system basically combines techniques from various domains such as artificial intelligence, natural language processing, statistical analysis, pattern matching, information retrieval, and information extraction. Most of the recent works on QA would suggest the integration of few or all of the above said approaches to build enhanced systems that can deal with scarcity of these approaches. All the existing QA systems may be classified into one of the many categories based on the approaches they used to extract answers; Linguistic based approaches, Statistical based approached, and Pattern matching based approached [Dwivedi and Singh, (2013)].

We would say that a QA system would be a good one to answer any questions only when it is capable of identifying all relevant documents that might contain the required answers, able to filter out the best among them, and able to extract the right portion of the document. Hence, they need to analyze a huge amount of documents. World Wide Web is one of the major sources of all relevant documents. The popular systems like START [Katz, (1997)] built by MIT Artificial Intelligence Laboratory, and AskMSR [Brill *et al.*, (2002)] are few examples of non-restricted domain question answering systems that used web as a source. Few of the question answering system researches like [Mishra *et al.*, (2010)] [Mollá and Vicedo, (2007)] on restricted domain have also used web as a source of documents.

To search and identify a right set of documents that would probably contain the required answers, the next important thing is to represent the question in a much clear way. This can be achieved by query expansion [Oh and Jung, (2015)] [Karisani *et al.*, (2016)] [Colace *et al.*, (2015)] [Sharma *et al.*, (2015)] [Hahm *et al.*, (2014)]. That is, expanding a given query into a more meaningful one and an understandable one by using synonyms, stemming and spelling corrections. [Colace *et al.*, (2015)] have used their query expansion technique to improve the retrieval system's accuracy. [Hahm *et al.*, (2014)] proposed a personalized query expansion technique in the view of disambiguating engineering terminologies and increasing the accuracy of personalized information retrieval of an engineer. Word Sense Disambiguation (WSD) plays a vital role in query expansion process. A semantic similarity method called wpath is proposed in [Zhu and

Iglesias, (2017)] to find the similarities between concepts. They used both the path length (to differentiate the concepts) and the information content (to find the commons) as the measurement components. Pedersen *et al.* proved that Gloss vector based similarity measure performed well for similarity relatedness measurement dependent word sense disambiguation [Patwardhan and Pedersen, (2006)] to find correct senses of words in a sentence during question expansion.

Over the years of research in the field of Question and Answering, researchers found lot of natural language processing techniques to extract right answers from a textual document. The following factors affect the use of these techniques; 1) the need for extensive application of these techniques on the document set to find the answers, 2) the number of available documents that would possibly contain the relevant answers, and 3) the quality of the set of documents that we have for extracting answers. If we are able to somehow reduce the number of available documents to a minimal set of quality documents much relevant to the question, then we are able to apply the existing natural language processing techniques to extract the quality answers. Searching for answers in a small set of documents would possibly increase the effectiveness of the system [Yen *et al.*, (2013)] and if those small set of data are quality data then the accuracy of the answers would well be increased [Bilotti and Nyberg, (2008)]. Bilotti and Nyberg, (2008) proved the essentials in improving the quality of information retrieval. They found that the process not only narrowed down the large set of documents into a smaller sub-set but also increased the quality of the answers that would be extracted from them [Bilotti and Nyberg, (2008)]. A proximity based approach is proposed in Monz, (2004) to assess the proximity between terms and the proximity between question terms and documents to retrieve relevant documents.

At present, some questions are directly answered by search engines. For an instance, Google can answer almost all questions that search for a person or a thing. For example, the question “Who is the chairperson of JSW steel company?” can be answered directly. If not, at least the first few URLs suggested by the search engine will definitely contain the answer. But few ‘Who’ questions are not answered directly by the search engines. Let us consider the same question with the slight modification, “Who is the chairperson of the largest steel company in India?” The search engines are not able to answer this question directly. Hence, they suggest set of relevant ranked documents. Such queries highly demand very good understanding about the requirement specified. In this work, the authors have proposed a method that understands the question, rewrite complete set of alternate questions that are equivalent, searches for relevant documents and finally compiling the outcomes and rank the results in open domain question answering system.

2. Related Work

The inception of the research in the area of question answering systems dated back to early 1960s. Initially, the idea of answering a question using computers has started with

databases as the data sources. We can divide the evolution of question answering system into 1) QA with structured data sources, for example, structured data from databases are identified using dictionary definitions (BASEBALL [Green Jr *et al.*, (1961)], LUNAR [Woods, (1973)], PRECISE [Popescu *et al.*, (2003)], MASQUE [Androutsopoulos *et al.*, (1993)]), and 2) QA with unstructured data sources. Furthermore, QA with unstructured data sources can be further divided into 1) documents in data collection as data sources and 2) World Wide Web as data sources (Omnibase [Katz *et al.*, (2002)], AnswerBus [Zheng, (2002)], Webclopedia [Hovy *et al.*, (2000)]). Some of the researchers further focused on the size of the target data sources. Hence, the idea of restricted domain and open domain has initiated. BASEBALL [Green *et al.*, (1961)] and LUNAR are some of the works on closed domain and Omnibase [Katz *et al.*, (2002)], AnswerBus [Zheng, (2002)], Webclopedia [Hovy *et al.*, (2000)], AskMSR+ [Tsai *et al.*, (2015)] and QuASE [Sun *et al.*, (2015)] are some of the open domain question answering systems. TREC Evaluation campaign [Voorhees, (2001)] is one of the major initiator of open domain question answering with unstructured data sources. In TREC Question Answering track, they investigated on providing answers to factoid, list and definition questions by increasing the complexity level in each track.

2.1. *Question analysis*

User questions have to be analyzed carefully for various reasons. Careful analysis of a user question in natural language can lead to identify various features like question type, expected answer type, and the focus of the question. The question analysis may require one or all of the question analysis methods like deep or shallow application of NLP techniques, statistical techniques, and semantic methods.

2.2. *Web as the knowledge source*

The amount of information available in World Wide Web has attracted many researchers to concentrate on looking for answers in the WWW. Many of the proposed researches in the field of Information Retrieval and Questions Answering have used web as their knowledge sources. Some of the existing QA systems such as START [Katz *et al.*, (2002)], AnswerBus [Zheng, (2002)], Webclopedia [Hovy *et al.*, (2000)], Context-aware Geographic QA [Mishra *et al.*, (2010)], and Aranea [Lin and Katz, (2003)] have acquired the knowledge from the web. Using Google for searching web documents would reduce much time on identifying target set of documents. For utilizing the web as the knowledge source, many researchers have used Google or Bing to extract web documents for further processing [Khodadi and Abadeh, (2016)] [Heie *et al.*, (2012)].

2.3. *Query expansion*

The ability of a short sentence to project itself in information search ends up in ambiguous or ineffective search results. The user question should be first understood by the search tool for various reasons like domain identification, ambiguity elimination,

relationship among the terms that are present in the query etc [Hazrina *et al.*, (2017)] [Zhao *et al.*, (2016)]. The queries that are not able to present the above stated information may not end up in good search result. A query that lacks enough information is one of the major causes of ambiguity [Hahm *et al.*, (2014)]. The lexical database WordNet has been used successfully in many works for query expansion and similarity finding [He *et al.*, (2016)]. Use of domain specific terminology based query expansion would be appropriate in many cases, for example, in closed domain question answering [Oh and Jung, (2015)].

2.4. *Latent semantic analysis (LSA)*

In the domain of information retrieval, the documents that are to be examined are considered as a bag-of-words. Each word in a document is considered as an independent entity. In IR, to retrieve related documents, we need to consider the relationship among the documents and among the words present. Latent Semantic Analysis is one such concept that is used in document classification. The ability of LSA is to find the hidden relationship among the documents. Performance of LSI is extensively studied and found best over vector space models [Aswani Kumar and Srinivas, (2009)]. The classification of less popular web pages of sparse nature is successfully done using LSA in [Wang *et al.*, (2015)]. The application of LSA has revealed that more keywords are close to certain keywords using higher order co-occurrence finding. LSA is used for automatic assessment of student answers of a particular subject with ideal answer set as training set in [Thomas *et al.*, (2015)] and for the evaluation of short answers for open ended questions in [Dos Santos and Favero, (2015)]. Latent semantic indexing (LSI) uses the singular values generated by singular value decomposition (SVD) process which factorize the original term-by-document matrix. Though the process is efficient, it may be impossible to use SVD in case of large document collection. Eigen value based LSI for information retrieval is proposed in [Aswani Kumar and Srinivas, (2006)] to overcome the problem of SVD.

3. Proposed System Architecture

The proposed architecture shown in Fig 1 has two major components;

1. Query processing – At this level, we process the question to understand and find set of best alternate questions that would best describe the user's question optimally.

2. Document processing – We process the top documents that were produced as a result of alternate queries search using Latent Semantic Analysis (LSA)

3.1. *Query processing*

User query has to go through several steps before constructing alternate queries. They are, POS tagging, Named Entity Recognition, Parsing, Keyword extraction, Finding synsets, and Similarity measurement. Proper understanding and the creation of alternate queries is of prime importance because the user query may not be rich in representing

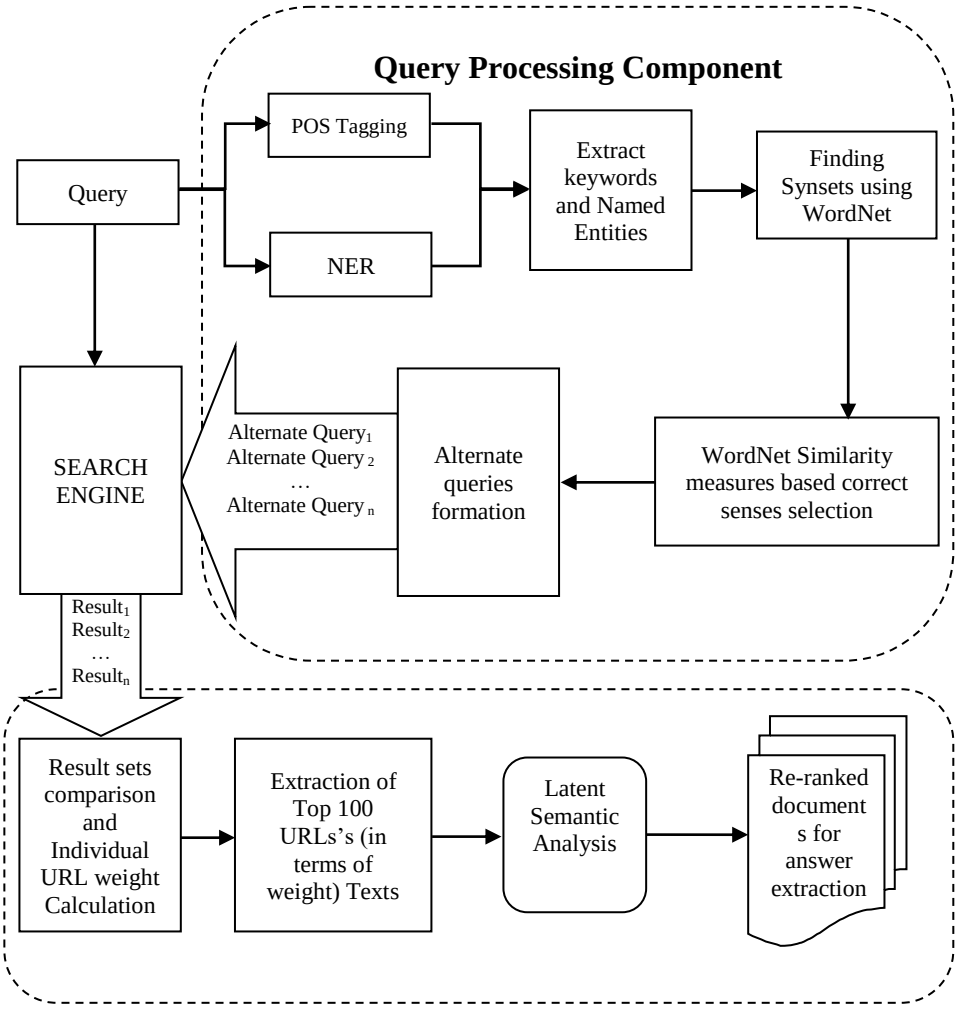

Fig. 1. Proposed architecture.

what is needed or may be inappropriate [Bing *et al.*, (2015)]. Hence, careful analysis is required to reconstruct a given query that is both contextually and semantically similar to the user query.

3.1.1. POS and NER

The role of Part of Speech tagging (POS) and Named Entity Recognition (NER) are very important in identifying the keywords that contribute more in understanding the given query or a sentence. They will help us in selecting good candidate documents for analyzing [Negri, (2004)]. We have used the Stanford Parser [Chen and Manning,

(2014)] to tag the Point-Of-Speeches in the given user query. Named Entity Recognition is an important aspect in our work. Our previous work [Saravanakumar and Cherukuri, (2014)] showed some amount of degradation in recognizing a relevant meaning in some cases, especially on Named entities. We were not able to recognize all the named entities if they were part of a given query. In some cases, there may be some meanings for all or part of an named entity. Hence, we would like to use one of the existing NER tools at the level of query pre-processing to understand the query better and tries to keep the named entities as they are. Our study on various work on NER tools revealed that Illinois Extended Named Entity Recognizer [Ratinov and Roth, (2009)] outperforms the other such tools like Stanford NER [Finkel *et al.*, (2005)], OpenCalais WS [Thomson Reuters, (2008)], and LingPipe [Alias-i, (2008)]. Also, Illinois Extended Named Entity Recognizer [Ratinov and Roth, (2009)] is able to recognize more entity types when compared to its counterparts. Hence it is possible to detect many entity types in open domain question answering system.

3.1.2. Keyword extraction and lemmatization

It is known that all the words present in a user question may not contribute much in identifying target set of documents. The reason is that they may contain the words like 'a', 'an', 'who', 'what', etc. These set of words are called as Stop Words. We are eliminating the stop words from the parsed questions. We have used the stop word list that was built at Cornell University for the experimental SMART information retrieval system to eliminate stop words. The words that are left after stop words removal are considered as keywords that might contribute something for searching required data. These keywords may be in several inflected forms. Also, the target set of documents will contain the keywords in different inflected forms. We are projecting these keywords into a term-document matrix during the process of Latent Semantic Analysis. Hence, we are interested in converting the given keywords into its base form, if any. In document retrieval we have two options for finding the base form of word, namely, Stemming and Lemmatization. Though the stemming process is simpler, it failed to operate with the knowledge of the context of a word. We have chosen lemmatization because it can identify the base form of any words with understanding the context [Carvalho *et al.*, (2007)].

3.1.3. Finding synset

We are using WordNet [Miller, (1995)], a large lexical database that stores synonyms of different words as nouns, verbs, adjectives, and adverbs, for finding the synset of all the lemmatized keywords of the user question. WordNet contains more than 118,000 different word forms and more than 90,000 different word senses, or more than 166,000 (form, sense) pairs. We find all the synonymous words of all the lemmatized keywords that are part of the given user question, except Named Entities. We use Named Entities as they are in our query.

3.1.4. Identification of correct synsets using WordNet similarity measure

WordNet contains synonyms for all the words in different senses. A word can be used in appropriate senses to make sentences. While we form a sentence, we would not use any senses of a word. We need to use only appropriate senses that would suit the context of a sentence. Hence, when we create alternate questions that are equivalent to the given user question, we must consider the right senses of all the keywords. We have used here is the knowledge-based disambiguation technique using the semantic network WordNet [Miller, (1995)]. To choose the correct senses of a keyword, we have used the Adapted Lesk [Banerjee, and Pedersen, (2002)] similarity measure, sometimes referred as Extended Lesk, which is the extension of Lesk algorithm to find the relatedness between question and the key word. Lesk [Lesk, (1986)] determines the sense of a word that overlaps the most with its neighboring words. On the other hand, Adapted Lesk similarity measure uses the glosses of two synsets for finding overlaps between them to choose the best sense. Here, we use the same senses of the words to get the relatedness value. We have used the python implementation of Adapted Lesk [Tan, (2014)] and extended it according to our requirement. As per the evaluation done on SENSEVAL-2 English lexical sample data, the overall accuracy of the Adapted Lesk algorithm is around 32%. The table 1 below shows the accuracy on individual POS elements;

Table 1. Accuracy of Adapted Lesk disambiguation algorithm on different POS elements.

Disambiguation Accuracy of Adapted Lesk Algorithm [Banerjee and Pedersen, (2002)]			
Nouns	Verbs	Adjectives	Total
32.2%	24.9%	46.9%	31.7%

3.1.5. Alternate query formation

We have constructed alternate queries by using the keywords and the alternate words that are found through WordNet. Here the alternate words are words with the senses that match with the context of the given user question. Other senses of the keywords are simply eliminated as they are irrelevant or contributing very little in forming the sentence with the correct context. We have tried all the combination of keywords along with the stop words to form different queries that are alternate but similar to the user question.

3.2. *Document processing*

3.2.1. URL weight calculation

All the alternate queries that are generated during the query processing along with the original query are used to search for the target set of documents using Google. For each query, we accept only the top 100 suggestions by the search engine. In some cases, there may be less than 100 suggestions. The results for different alternate queries are

considered as lists of result sets. Then we are looking for the presence of all the URLs in the result sets of other alternate queries. If an URL is present in a list, we count the presence and the position (rank) of that URL to find its new weight. To increase the confidence value, we are taking into account only the URLs that are present in at least half and more of the number of result sets. For example, we have found 6 alternate queries for the query "What is the length of border between the Ukrine and Soviet Russia?" That means we get six result sets and we consider any URL that is present in 3 or more results. The weight w_{ij} of an URL i of search result set j is then calculated and normalized using the Eq. (1).

$$w_{ij} = \left\{ \left\{ \frac{|x_i| + 1 - rank_{ij}}{|x_i|} \right\}_{j=1}^{m} \right\}_{i=1}^{n}$$

$$|x_i| = \begin{cases} 100, if\ number\ of\ URLs\ in\ the\ result\ set\ i \geq 100 \\ number\ of\ URLs\ otherwise \end{cases} \quad (1)$$

Then we extracted and arranged all unique URLs into a separate set. For each URL in the new set, we have normalized its weight by finding their presence and weight in each search result set. To include the importance of an URL, we have taken into account the URLs that are present in at least half and more of the total number of search result sets. If they are not suggested by at least half and more of the alternate queries including the user query, we simply discarded those URLs. The new weight for each URL is calculated using Eq. (2).

$$new\_weight_{URL} = \frac{\sum_{i=1}^{n} \sum_{j=1}^{m} WEIGHT_{URL}}{n}$$

$$WEIGHT_{URL} = \begin{cases} weight\ of\ the\ given\ URL\ if\ that\ URL\ \in result\ set\ i \\ 0\ otherwise \end{cases} \quad (2)$$

Here, n is the total number of result sets generated (number of alternate queries including user query), m is the total number URLs in a given result set i, $WEIGHT_{URL}$ is the weight of an URL if that URL present in a result set, and new_weight_{URL} is the final weight of an URL. The proposed algorithm for calculating the final weight of each URL is shown in Figure 2.

Algorithm 1 WEIGHT_CALCULATION
INPUT : Ranked URLs L_j with weights W_{ij} of candidate query result set R_i where i in 1 to n and j in 1 to 100 OUTPUT : Re-ranked list of URLs with new weights 1: for i in 1 to number of result sets 2: for j in 1 to number of URLs in result set i 3: $W_{ij} \leftarrow$ rank of individual URL L_j of R_i 4: end for 5: end for 6: for each result set R_i do 7: for each URL L_j Є R_i do 8: if L_j of R_i presents in at least half of other R_i then

9: $W_{ij} \leftarrow W_{ij}$ + (Rank of matching URL of other R_i / Number of URLs in matching result set R_i)
10: end if
11: end for
12: end for
13: Repeat steps 4 to 10 for all R_i
14: Arrange all URLs of all R_i into one set in descending order of new weights W_{ij}
15: Eliminate duplicate URLs from this list
16: Display the ranked probable of URLs

Fig. 2. Algorithm for weight calculation

3.2.2. Latent semantic analysis

We extract the texts of all the top 100 URLs that we have identified in the URL weight calculation process. We project all the top 100 documents to the process of Latent Semantic Indexing (LSI). LSI was first proposed by Deerwester *et al.*, (1990). The idea of LSI is to eliminate the keyword comparing nature of any retrieval techniques [Deerwester *et al.*, (1990)]. LSI applies the vector space model and singular value decomposition to find the hidden relationship with the words and documents by applying the concept of higher order co-occurrences of the words in documents. Application of LSI reduces the importance given to the set of keywords that co-occur very frequently and thus it eliminates or reduces the noise in the data for further processing [Kontostathis and Pottenger, (2006)][Kumar, (2009)].

LSI basically requires the preprocessed text. Literature shows that the use of raw-terms as input did not work well. Different works by authors used different techniques on data preprocessing for LSI. Some of them are removal of stop words, application of stemming, raw-term frequency, term-frequency with inverted document frequency (TF-IDF) etc. In our proposal, we have removed stop words. We have projected the top 100 documents that are produced in the last step to LSI in different ways; ie., without stemming, with stemming, with raw-term frequency and with TFIDF. TFIDF is a statistical measure that is used to find out the importance of a word with respect to a document in our result set. It is calculated as given in Eq. (3)

$$TFIDF_{t,d} = tf_{t,d} * idf_t \tag{3}$$

Here, $tf_{t,d}$ is the number of occurrences of term t in document d and is calculated in such a way to normalize the term count in the larger documents using Eq. (4); idf_t is the inverse document frequency used for identifying the rarity of a keyword in the set of all documents and calculated using Eq. (5).

$$tf_{t,d} = \frac{frequency\ of\ term\ t\ in\ document\ d}{total\ number\ of\ terms\ in\ document\ d} \tag{4}$$

$$idf_t = log \frac{Total\ number\ of\ documents\ in\ the\ collection}{Number\ of\ documents\ with\ term\ t} \tag{5}$$

LSI uses Singular Value Decomposition (SVD) to factorize the term-by-document matrix. SVD of a matrix is given in Eq. (6).

$$A = U\Sigma V^T \tag{6}$$

Here, A is the term-by-document matrix, U is the term-by-term unitary matrix contains the coordinates of term vectors, V is the document-by-document unitary matrix contains the coordinates of document vectors, and $\sum$ is the rectangular diagonal matrix contains the dimensions. Next we have truncated the matrices U, V and $\sum$ to k dimensions as given in Eq. (7) below.

$$A_k = U_k \Sigma_k V_k^T \tag{7}$$

Here, U_k is the term-by-dimension matrix, V_k is the dimension-by-document matrix and $\sum_k$ is the dimension-by-dimension matrix. Major issue in the application of LSA is the selection of the k value, that is the dimension to which the actual term-by-document matrix to be reduced. As we use small set of documents, we have applied the following equation to find the k value. If we have n singular values such that S[1], S[2], …, S[n], then we can find the optimal k values using Eq. (8) as follows;

$$k = \left\{ max_i \mid S[i] > \left(\frac{\sum_{i=1}^{n} S[i]}{n} \right) \right\} \tag{8}$$

It is observed from our experiments that the k value chosen this way performs well for small set of documents. Finally, after the SVD process, we used the cosine similarity measure given in Eq. (9) to find the similarity between each of the ranked documents d and the query q.

$$Sim(q,d) = \frac{\sum_{i=1}^{n} q_i d_i}{\sqrt{\sum_{i=1}^{n} {q_i}^2} \sqrt{\sum_{i=1}^{n} {d_i}^2}} \tag{9}$$

4. Experimental Evaluation

We have used Text REtrieval Conference's (TREC) QA track dataset [Voorhees and Tice, (2000)] to evaluate the performance of the proposed approach. We have taken a sample set of 200 factoid and list questions. We have calculated the precision at the top 5, and 10 web documents from our results. The performance of some sample questions is listed in Table 2 along with the reciprocal rank. Fig. 3 shows the performance of TREC sample factoid questions. We have taken top 10 or 5 documents due to the reason that our method includes as low as 5 documents as the final result. The final result that consists of less than 5 have not included in our study. We are looking for the set of web documents that contain the required answers directly in it. Our method shows difference in precision values for different questions. We found that the reason for such difference is due to one or more of the following; our method might have missed some top documents, the search

engine has not listed the required document in the top 10, the WWW may not have the required document (for example, the web document that possibly contain the answer for the question "Who received the Will Rogers award in 1989?" is not listed in search engines Google and Bing), or the search engine may include some other document which might be very relevant to a single entity of the given question given (for example, for the question "How many islands does Fiji have?" is understood by search engine as different entities like the number of islands, the islands of Fiji, Fiji itself and so on. Our method included one such result about Fiji and not about the count on islands in Fiji). We have analyzed the performance of LSI for finding the ranks of documents using two different approaches. In the first, we tried to find the similarity between the result documents and the question keywords as they are. In the second, to find the similarity we used all the question keywords along with their alternate keywords that we found using WordNet similarity measures. The later showed some improvement over the former in many questions. Hence, we have used the later to produce the final ranked results.

Table 2. Performance of our approach for some sample TREC questions.

QId	**Sample Question**	**Question Type**	**TREC**	**Precision @ 5 (in %)**	**Precision @ 10 (in %)**	**Reciprocal Rank**
48.4	In what countries as Abu Nidal operated from?	List	2004	80	70	1
41.4	Who were the major players involved in the Teapot Dome scandal?	List	2004	100	80	1
14	What country is the biggest producer of tungsten?	Factoid	1999	80	80	1
200	How tall is the replica of the Matterhorn at Disneyland?	Factoid	1999	80	50	0.5
70	How many lives were lost in the China Airlines' crash in Nagoya, Japan?	Factoid	1999	100	90	1
15	When was London's Docklands Light Railway constructed?	Factoid	1999	60	60	0.33
295	How many films did Ingmar Bergman make?	Factoid	2000	80	40	1
235	How many astronauts have been on the moon?	Factoid	2000	80	70	1
689	How many islands does Fiji have?	Factoid	2000	80	80	1

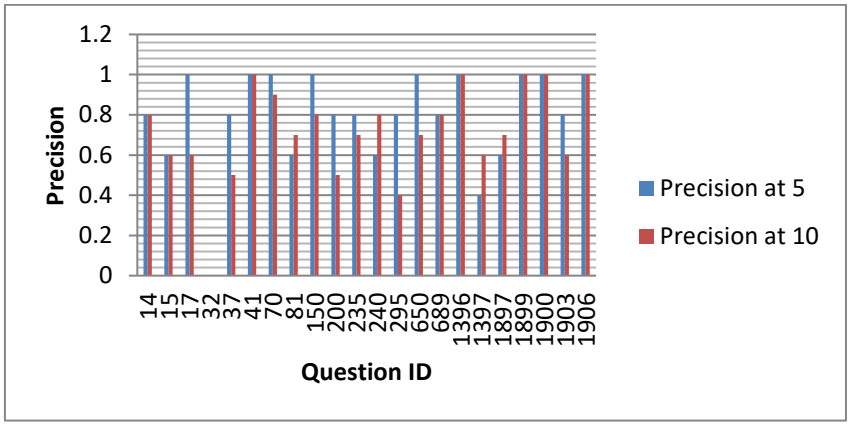

Fig. 3. Precision at 5 and 10 for sample factoid questions with question ID

It is found that the proposed system is not performing well in case of list questions that demands for a list as answer for some general question. For example, the question "List 13 countries that export lobster" from TREC 2002 is not properly answered. But, the question "Who were the major players involved in the Teapot Dome scandal?" from TREC 2004 is answered perfectly. The problem is that the former question is about a list in general, and the later is about a list in particular. We are able to get answer from a document that talks about the 'Teapot Dome Scandal' for the later question, but the document that is about 'Lobster' does not contain the answer for the former question.

Table 3. Performance of proposed approach on TREC factoid questions.

Question Type	**Precision @ 5**	**Precision @ 10**	**MRR**
Factoid Questions	0.773	0.742	0.79
List Questions	0.432	0.480	0.42

Table 3 shows the precision value at top 5 documents, top 10 documents and the Mean Reciprocal Rank of factoid questions. Table 4 compares the MRR of our proposed work with that of AskMSR+ and QuASE results on factoid questions from TREC data set.

Table 4. Comparison of different information retrieval approaches

Method	**MRR**
Proposed	0.79
AskMSR+	0.62
QuASE	0.65

5. Conclusion

We have proposed a novel approach to find the relevant documents for open domain question answering system. The method proposed here to find the top relevant documents for further processing of answer extraction showed better results. We are able to find the

best document with the required answer at the top level. The reciprocal rank for most of the factoid questions is 1. In case of list questions, many times we end up in documents that contain one of the answers in the expected list unless otherwise listed completely in a document. We found that list questions were not performed well especially when the individual elements in a list are part of different documents. The proposed approach is definitely helping us in finding the target documents that contains the answers. Hence, we are able to apply the complex natural processing techniques on the small set of suggested documents to extract the relevant answers. As a future work we would like to extend the work so that the list questions may also be answered well.

References

Alias-i. (2008). *LingPipe 4.1.0*. Available: http://alias-i.com/lingpipe

Androutsopoulos, I.; Ritchie, G.; Thanisch, P. (1993): MASQUE/SQL - An Efficient and Portable Natural Language Query Interface for Relational Databases. *Database technical paper, Department of AI, University of Edinburgh*.

Aswani Kumar, C.; Srinivas, S. (2006): Latent semantic indexing using eigenvalue analysis for efficient information retrieval. *International Journal of Applied Mathematics and Computer Science*, *16*, 551-558.

Aswani Kumar, C.; Srinivas, S. (2009): On the performance of latent semantic indexing-based information retrieval. *Journal of Computing and Information Technology*, *17*(3), 259-264.

Banerjee, S.; Pedersen, T. (2002): An adapted Lesk algorithm for word sense disambiguation using WordNet. In *International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 136-145). Springer Berlin Heidelberg.

Bilotti, M. W.; Nyberg, E. (2008): Improving text retrieval precision and answer accuracy in question answering systems. In *Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering* (pp. 1-8). Association for Computational Linguistics.

Bing, L. *et al.*, (2015): Web query reformulation via joint modeling of latent topic dependency and term context. *ACM Transactions on Information Systems (TOIS)*, *33*(2), 6.

Brill, E.; Dumais, S.; Banko, M. (2002): An analysis of the AskMSR question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10* (pp. 257-264). Association for Computational Linguistics.

Carvalho, G.; de Matos, D. M.; Rocio, V. (2007): Document retrieval for question answering: a quantitative evaluation of text preprocessing. In *Proceedings of the ACM first Ph. D. workshop in CIKM* (pp. 125-130). ACM.

Chen, D.; Manning, C. D. (2014): A Fast and Accurate Dependency Parser using Neural Networks. In *EMNLP* (pp. 740-750).

Colace, F. *et al.*, (2015): Weighted word pairs for query expansion. *Information Processing & Management*, *51*(1), 179-193.

Deerwester, S. *et al.*, (1990): Indexing by latent semantic analysis. *Journal of the American society for information science*, *41*(6), 391.

Dos Santos, J. C. A.; Favero, E. L. (2015): Practical use of a latent semantic analysis (LSA) model for automatic evaluation of written answers. *Journal of the Brazilian Computer Society*, *21*(1), 21.

Dwivedi, S. K.; Singh, V. (2013): Research and reviews in question answering system. *Procedia Technology*, *10*, 417-424.

Finkel, J. R.; Grenager, T.; Manning, C. (2005): Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 363-370). Association for Computational Linguistics.

Green Jr *et al.*, (1961): Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference* (pp. 219-224). ACM.

Hahm, G. J. *et al.*, (2014): A personalized query expansion approach for engineering document retrieval. *Advanced Engineering Informatics*, *28*(4), 344-359.

Hazrina, S. *et al.*, (2017): Review on the advancements of disambiguation in semantic question answering system. *Information Processing & Management*, *53*(1), 52-69.

He, Y. *et al.*, (2016): A framework of query expansion for image retrieval based on knowledge base and concept similarity. *Neurocomputing*, *204*, 26-32.

Heie, M. H., Whittaker, E. W., & Furui, S. (2012): Question answering using statistical language modelling. *Computer Speech & Language*, *26*(3), 193-209.

Hovy, E. H. *et al.*, (2000): Question Answering in Webclopedia. In *TREC* (Vol. 52, pp. 53-56).

Karisani, P., Rahgozar, M., & Oroumchian, F. (2016): A query term re-weighting approach using document similarity. *Information Processing & Management*, *52*(3), 478-489.

Katz, B. (1997): Annotating the World Wide Web using natural language. In *Computer-Assisted Information Searching on Internet* (pp. 136-155). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Katz, B. *et al.*, (2002): Omnibase: Uniform access to heterogeneous data for question answering. In *International Conference on Application of Natural Language to Information Systems* (pp. 230-234). Springer Berlin Heidelberg.

Khodadi, I.; Abadeh, M. S. (2016): Genetic programming-based feature learning for question answering. *Information Processing & Management*, *52*(2), 340-357.

Kontostathis, A.; Pottenger, W. M. (2006): A framework for understanding Latent Semantic Indexing (LSI) performance. *Information Processing & Management*, *42*(1), 56-73.

Kumar, A. C. (2009): Analysis of unsupervised dimensionality reduction techniques. *Computer science and information systems*, *6*(2), 217-227.

Lesk, M. (1986): Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. In *Proceedings of the 5th annual international conference on Systems documentation* (pp. 24-26). ACM.

Lin, J.; Katz, B. (2003): Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management* (pp. 116-123). ACM.

Miller, G. A. (1995): WordNet: a lexical database for English. *Communications of the ACM*, *38*(11), 39-41.

Mishra, A.; Mishra, N.; Agrawal, A. (2010): Context-aware restricted geographical domain question answering system. In *Computational Intelligence and Communication Networks (CICN), 2010 International Conference on* (pp. 548-553). IEEE.

Mollá, D.; Vicedo, J. L. (2007): Question answering in restricted domains: An overview. *Computational Linguistics*, *33*(1), 41-61.

Monz, C. (2004): Minimal span weighting retrieval for question answering. In *Proceedings of the SIGIR Workshop on Information Retrieval for Question Answering* (Vol. 2).

Negri, M. (2004). Sense-based blind relevance feedback for question answering. In *Proceedings of SIGIR-04 Workshop on Information Retrieval For Question Answering (IR4QA)*.

Oh, H. S.; Jung, Y. (2015): Cluster-based query expansion using external collections in medical information retrieval. *Journal of biomedical informatics*, *58*, 70-79.

Patwardhan, S.; Pedersen, T. (2006): Using WordNet-based context vectors to estimate the semantic relatedness of concepts. In *Proceedings of the eacl 2006 workshop making sense of sense-bringing computational linguistics and psycholinguistics together* (Vol. 1501, pp. 1-8). Trento.

Popescu, A. M.; Etzioni, O.; Kautz, H. (2003): Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on Intelligent user interfaces* (pp. 149-157). ACM.

Ratinov, L.; Roth, D. (2009): Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning* (pp. 147-155). Association for Computational Linguistics.

Saravanakumar, K.; Cherukuri, A. K. (2014): Optimized web search results through additional retrieval lists inferred using WordNet similarity measure. In *Data Mining and Intelligent Computing (ICDMIC), 2014 International Conference on* (pp. 1-7). IEEE.

Sharma, P.; Tripathi, R.; Tripathi, R. C. (2015): Finding Similar Patents through Semantic Query Expansion. *Procedia Computer Science*, *54*, 390-395.

Sun, H. *et al.*, (2015): Open domain question answering via semantic enrichment. In *Proceedings of the 24th International Conference on World Wide Web* (pp. 1045-1055). ACM.

Tan, L. (2014): Pywsd: Python implementations of word sense disambiguation (wsd) technologies [software].

Thomas, N. T.; Kumar, A.; Bijlani, K. (2015): Automatic Answer Assessment in LMS Using Latent Semantic Analysis. *Procedia Computer Science*, *58*, 257-264.

Thomson Reuters. (2008): *Calais Web Service*. Available: http://www.opencalais.com/

Tsai, C. T. *et al.*, (2015): Web-based question answering: Revisiting AskMSR.

Voorhees, E. M. (2001): The TREC question answering track. *Natural Language Engineering*, *7*(04), 361-378.

Voorhees, E. M.; Tice, D. M. (2000): Building a question answering test collection. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 200-207). ACM.

Wang, J.; Peng, J.; Liu, O. (2015): A classification approach for less popular webpages based on latent semantic analysis and rough set model. *Expert Systems with Applications*, *42*(1), 642-648.

Woods, W. A. (1973): Progress in natural language understanding: an application to lunar geology. In *Proceedings of the June 4-8, 1973, national computer conference and exposition* (pp. 441-450). ACM.

Yen, S. J. *et al.*, (2013): A support vector machine-based context-ranking model for question answering. *Information Sciences*, *224*, 77-87.

Zhao, G. *et al.*, (2016): Entity disambiguation to Wikipedia using collective ranking. *Information Processing & Management*, *52*(6), 1247-1257.

Zheng, Z. (2002): AnswerBus question answering system. *In Proceedings of the second international conference on Human Language Technology Research* (pp. 399-404). Morgan Kaufmann Publishers Inc.

Zhu, G.; Iglesias, C. A. (2017): Computing Semantic Similarity of Concepts in Knowledge Graphs. *IEEE Transactions on Knowledge and Data Engineering*, 29(1), 72-85.