ISSN 2007-9737

Knowledge-Based Query Suggestions for Retrieval Improvement

Amir Jamshaid[1], Tenvir Ali[1], Muhammad Sajid[1,2], Maryam Mazher[2], Liliana Chanona-Hernandez[3,*]

[1] The Islamia University of Bahawalpur, Faculty of Computing, Pakistan

[2] Instituto Politécnico Nacional, Centro de Investigación en Computación, Mexico

[3] Instituto Politécnico Nacional, Escuela Superior de Ingeniería de Mecánica y Electrónica Unidad Zacatenco, Mexico

{msajida26, mmazhera26}@cic.ipn.mx, {amir.jamshaid, tenvir.ali}@iub.edu.pk, lchanonah@ipn.mx

Abstract. In recent past entity searches have been in the spotlight. Keyword based entity searches (e.g. Harry Potter) fetch diverse results because of multiple occurrences (ambiguous nature) of entity and show low precision. To address this problem this work presents an Expansion-Based-Query-Suggestion Scheme Entity-to-Entity (E-to-E). Our scheme uses attribute-oriented definitions as a knowledgebase to produce query suggestions. Proposed scheme can distinguish and find ambiguous entities like Harry Potter with high precision. Using the query-URL co-occurrences we evaluate the scheme performance while identifying whether the fetched URL is a good representation of the searched entity or not. We use evaluation matrix based on MAP@k, coverage, success rate, precision, recall and F-measure to prove the effectiveness of the approach. In our experiments, the proposed scheme is compared with three different baselines. Our scheme achieved MAP@9 = 0.7017, Coverage = 100%, Success rate = 89%, and F-measure = 0.72, which shows an edge over others. In general, our experiments show a clear significant advantage of the scheme in a web search for finding ambiguous entities.

Keywords. Query expansion, frequent pattern, query suggestion, information retrieval, random walk, bipartite graphs.

1 Introduction

The universal amount of data is growing rapidly, and the World Wide Web is becoming the ultimate complex repository of information. A study [1] reported that the World Wide Web (WWW) will grow by 4300% by 2020. This growth trend provides new business opportunities as well as raising huge challenges (e.g., the difficulty of web searching) day by day.

These days, search engines are a common source for Information Retrieval (IR). Annually, Google processes 1.2 trillion queries at a rate of 3.5 billion searches per day [2]. Users initially provide a keyword seed query, and if the results do not match their expectations, they submit another query with refined keywords.

This process can be very long and frustrating, as shown in Figure 1, because the effort required to achieve the expected result is unknown. One study in [3] shows that 52% of users reformulate their queries. Two main reasons behind the decline of search engine efficiencies are 1) the use of keywords like “Apple” (due to polysemy) and 2) the

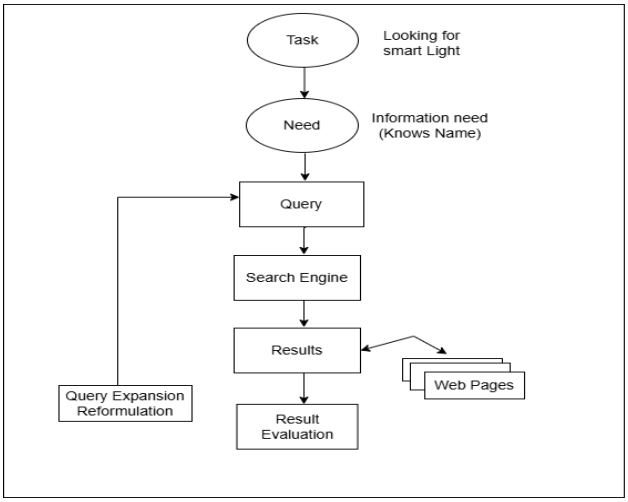

Fig. 1. Common search process

use of the “Bag of words” (BOW) scheme to find such keywords [4-8]. Another factor behind the low efficiency of a search engine is users' limited knowledge of the topic.

Studies in [6, 7] show that most users are unable to submit queries longer than 2 or 3 words. This lack of knowledge while using the BOW scheme results in low precision and accuracy [8]. So far, the proposed solutions to the above problems mostly revolve around query modification techniques (e.g., Query Expansion, Suggestion, and Refinement) [10-43]. Query modification is an IR problem that utilizes seed query and knowledge bases. Since query modification and optimization is an NP-hard problem [9], query suggestions came to the user's rescue. Query suggestion is one of the most popular solutions adopted by many famous search engines. This feature helps users by providing related suggestions that they can accept with a single click. This feature is becoming an essential part of modern search engines. That is why most search engines, such as Google, Yahoo, Live Search, Ask, and Baidu, provide query suggestions through their interactive interfaces. When a user enters a query, the search engine suggests the top-k similar queries to improve the results.

This work presents an Expansion-Based Query Suggestion Scheme (E-to-E) for retrieving search results with high precision when searching for ambiguous entities. In the E-to-E scheme, we map hundreds of attribute-oriented definitions of ambiguous entities as knowledge-based (E) against hundreds of millions of web instances of these entities (E). Our knowledge base contains rich information in the form of attribute-oriented definitions, which can be helpful when searching the web. The proposed E-to-E scheme consists of two phases, the query expansion phase and the suggestion phase. In the first phase, the scheme expands the keyword-based seed query and forms a query set used for suggestions.

To date, proposed methods of query expansion are based on Corpus-Based knowledge [13-15], Relevance Feedback [16-23], or Language Model-Based [24-26] approaches. Our scheme is novel in that it uses definitions of ambiguous entities as a knowledge base for seed query expansion. These attribute-oriented definitions help reduce query drift in most previously proposed schemes. The second phase takes the produced query set as input and suggests an optimal query. To date, proposed methods for query suggestion are based on Click-Through-Based [28], Session-Based (Query log-based) [29-30], and Mining Bipartite Graph-Based [31-43].

We propose a novel query scheme based on Query-Expansion rather than on search history. The proposed scheme suggests queries using

two methods. The first method is the MF-URL tree, and the second is a random walk on the Query-URL bipartite graph. The scheme suggests the query selected by both methods. The details of these methods are given in Section 3. The verification of the proposed E-to-E scheme is performed using an evolution metric based on MAP@k, coverage, success rate, precision, recall, F-measure, and human assessment of the suggested queries. Our experimental results show scores of MAP@9 = 0.7017, Coverage = 100%, Success rate = 89%, and F-measure = 0.72.

This paper is organized as follows. Section 2 reviews related work on query modification. Section 3 describes our proposed E-to-E expansion-based query suggestion scheme and algorithms in detail. Section 4 describes the experimental setup and results, and we conclude our work in Section 5 by outlining future work.

2 Related Work

Query modification is performed primarily due to issues such as ambiguity, vocabulary, and vagueness that commonly occur in IR systems during querying. Query modification techniques such as expansion, suggestion, and refinement are commonly used today. Researchers have proposed many ways to achieve ideal results by expanding a seed query (e.g., Buckley et al. [10] proposed a query expansion method using relevance feedback). Query suggestion is a scheme that helps the user modify a query with a single click (e.g., Velez et al. [11] dynamically combine precomputed suggestions for single-term queries to form multiple-term queries).

2.1 Query Expansion

Researchers believe that low accuracy is a major issue when a query is formed with only a few keywords. Therefore, query expansion is used to remove ambiguity in natural language and add the required details to the original query [12]. Query expansions can be automatic, manual, or user assisted. The main query expansion methods are as follows.

2.1.1 Corpus-Based Knowledge Models

Researchers have argued that words that often co-occur in a document corpus are on the same subject [13] and can be used for expansion. Using a handcrafted thesaurus is a boring process, e.g., WordNet [14]. Hand-crafted thesauri are domain-specific, too general, unable to include new words, and unsuitable for a static document collection. An automatic thesaurus gathers words based on their occurrence pattern in a cluster [15]. Corpus-based query expansion is not good when a user's query is not listed in the thesaurus.

2.1.2 Relevance Feedback

Chooses terms for expansion from documents that are fetched because of the query and identified as relevant by the users or by the system. It considers top-ranked documents as relevant [16, 17]. So, expansion terms are extracted from top-ranked documents retrieved in response to a user's seed query. Here, effectiveness expansion terms depend on the quality top-ranked documents retrieved from the original query. Cao et al. [18] proposed a supervised learning approach to select terms. Explicit feedback is collected from the explicit evidence that shows relevance [19]. Graded relevance is a form of explicit feedback; it presents the relevance of the document and the query using numbers, letters, or other forms [20, 21]. Implicit feedback infers the user's interest by observing their behavior [22]. Information is collected using a low-cost system without putting any burden on the user [23]. However, the information is difficult to understand due to its noisy nature.

2.1.3 Language-Model-Based Approach

It uses a statistical language model for query expansion; it specifies a probability distribution over the terms [24]. Term selection depends on the language model's probability score. These schemes produce promising results and provide a solid theoretical setting [25]. The work by Buscher et al. [26] showed strong correlations between relevance and gaze-based measures and validated their results through experiments.

2.2 Query Suggestion

Query suggestions help with query reformulation and address ambiguity and inaccuracy in IR systems. The system improves search by providing suggestions after inferring the user's intent from past interactions [18]. Meng et al. [27] found that users do not like using automatic query expansion; instead, they prefer query suggestions.

2.2.1 Click-Through-Based Query Suggestion

The clicked URLs are used to exploit the relationship between different queries. Kenneth et al. [28] proposed a query-suggestion method using a personalized concept-based clustering technique. A personalized query suggestion was given to users based on their past behavior.

2.2.2 Session-Based Query Suggestion

Assumes that search queries in the same session are related to each other [29]. The simple assumptions are first, queries in a session during a specific time interval are submitted by the same user. Second, the user often tries to change the query or submit a new one to get a better result during a single session. Third, the query is about a single topic. The work by Cucerzan and White [30] used a user's previous search experience to generate query suggestions for a new user.

2.2.3 Mining Bipartite Graphs and Random Walk-Based Query Suggestion

Ma et al. in [31] used user-query and query-URL bipartite graphs built on click-through data. Mei et al. in [32] perform a random walk to find queries that are like the query qi and return the smallest hitting time query as a suggestion. Song and He used clicked-URL and skipped-URL information to propose an optimal rare query suggestion [33]. Researchers in [34-36] represent Queries and URLs as nodes in a graph, with edges connecting queries to URLs based on clicks. Chen and Zhang constructed personalized query suggestions using an agent-based approach to query-concept bipartite graphs and concept relation trees [37]. Cao and Xue-Qi proposed a long-tail query suggestion model based on query intent [38]. Gao et al. described a new method for cross-lingual query suggestion by exploiting a wide spectrum of bilingual and monolingual information [39].

Yang et al. proposed a unified strategy for combining query logs and search results to support query suggestion [40]. Strohmaier et al. introduced intentional query suggestion as a novel approach to making the user's intent more explicit during search and presented a prototypical algorithm for it [41]. Sadikov et al. combined the query flow graph approach with clicked URL information to find query suggestions [42]. Hotho et al. improved the bipartite graph and proposed a method for modeling folksonomies [43], which are represented as a tripartite document-user-tag graph.

Most related research focused on query expansion or query-suggestion techniques based on terms extracted from web pages. To explore further please refer to some other recent studies related to query modification [44, 45, 46, 47, 48]. In this paper, we use the E-to-E Expansion-Based-Query-Suggestion approach for search optimization, which is a combination of expansion and suggestion techniques. Our experiment results show improvement in retrieval quality.

2.3 Motivation

Search engines are widely used to locate relevant information on the web; however, keyword-based searching often returns many irrelevant results, especially when the query is short, ambiguous, or represents a popular entity. For example, a user who wants to purchase the online version of the Harry Potter book may search the keyword “Harry Potter” using Bing. As shown in Figure 2, this query returned 49,500,000 results in our observation. However, the results appearing on the first page were not directly related to purchasing the book, and their irrelevance is marked with a cross symbol.

A similar problem arises in institutional information retrieval. Suppose a university or college wants to create an alumni database by collecting former students’ contact information through a search engine. If the institution searches for an alumna named “Sara” on Bing, the query may return millions of results because the name is common and highly ambiguous. In our observation, the query returned 25,900,000

results, while the top ten results were irrelevant to the intended alumna. These examples show that conventional keyword-based search engines may fail to retrieve contextually relevant information when queries are vague, broad, or entity-based. Therefore, there is a need for an improved search and recommendation approach that can better understand user intent, reduce irrelevant results, and retrieve more accurate information from large-scale web data.

Both scenarios relate to polysemy and to searching for ambiguous entities on the web. Ambiguous entity search is non-trivial due to different occurrences and definitions across billions of web pages. A query like a keyword “Harry Potter" is trying to find a book, but because of polysemy and the BOW scheme, the search engines interpret it wrongly, as shown in Figure 2.

Thus, the query returns many irrelevant results, and the precision and accuracy of the search task decrease. Here, a suitable representation for the query is web pages that sell Harry Potter books, rather than a review or blog about the book, a movie link or any other kind of web page. Sometimes, there are too many results, making it difficult to find the desired ambiguous entity on the web.

The user manually determines the relevance of web pages in real time and examines many pages. Therefore, it is a considerably significant effort to filter out irrelevant pages and reduce the number of results returned by using the right query. Nowadays, search engines use query suggestions to help users in this regard, but these suggestions still have a lot of room for improvement.

Our proposed Expansion-Based-Query-Suggestion scheme, E-to-E, like other suggestion methods, helps reduce the number of results and increase the precision of the search process. The idea is simple: to find a link between entities in a knowledge base and their representatives on the public web. The E-to-E scheme consists of two phases, the query expansion phase and the suggestion phase. The first phase adds useful query terms to the seed query after extracting attributes from the knowledge base, forming a query set. Whereas the second phase suggests optimal query forms for this query set.

3 Proposed Methodology

This section presents an overview of our methodology for improving query suggestions using a knowledge-based approach. The first part focuses on constructing and preparing a knowledge base containing attribute-oriented definitions of ambiguous entities. The second part involves generating query sets from seed queries and executing them on search engines to collect relevant URLs. The final part describes two techniques for refining and suggesting optimal queries: the MF-URLs tree method, which ranks queries based on URL frequency and content similarity, and the random walk on a query-URL bipartite graph, which captures relationships between queries and URLs to determine relevance. Each of these steps is elaborated in detail in the following subsections.

3.1 Dataset

To the best of our knowledge, there is currently no standard publicly available corpus specifically designed for query suggestion tasks. Most of the previous research makes use of query logs as stated before. But nowadays because of privacy and some commercial factors, it is problematic to acquire real query logs from a search engine like Microsoft, Yahoo, Google, Bing and so on. Some English logs are available these days, but they are too old (e.g. AltaVista_2003 query log). Most of the pages in this weblog now do not exist. So, we prepared in-house data set. We used our proposed E-to-E scheme and baselines used in the evaluation process. We also use one-month query log of AltaVista for the click-through based suggestion.

In our work for the expansion of seed query and to prepare a query set we use a test knowledgebase which contains attribute-oriented definitions of ambiguous entities. Basically, this knowledgebase is built using the web. We manually explore 200 ambiguous entities and their attribute values and store these definitions in a text file. Each line of the text file represents a new definition and represents query term vector: Q(terms) = t_1(e); t_2(e); … t_m(e). The knowledgebase and its structure are explained in Appendix B.

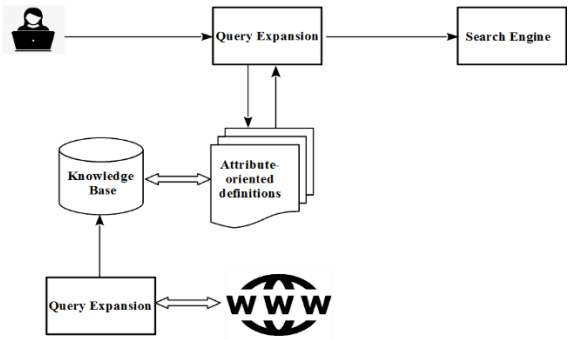

Fig. 2. Query expansion scheme

We use each definition in the knowledgebase for expansion of the query. Using the knowledgebase, a query set Q for each entity instance is formed, which contains 2n queries depending on several attributes. So, our experimental work is based on 30000 queries against 200 entity definitions.

We executed these queries on two separate search engines (Yandex and Bing) and extracted 10 unique URLs against each query in common from both search results. E-to-E query suggestion schemes unique URLs are stored in a table for one entity. After that, the frequency of web pages is counted, and URLs with a frequency more than two for an entity were considered, whereas URLs with frequencies of less than or equal to two are left out while analyzing our methodology. Then, we sort and assign variable names for each based on frequency. Let us explain this process by example. Suppose query q_1 fetches URLs u_1, u_2, and u_4 and q_2 fetches URLs u_1, u_2, u_3, and u_8. While URLs are arranged per their frequencies against each query, as shown in the snapshot of the database in Figure 4, this means u_1 has the highest frequency and u_8 has the lowest frequency. Initially, the total URLs extracted was 300000, but after filtering URLs based on frequency, 256000 remained. We extracted text from the pages to check the query-page similarity and to extract co-occurring terms for making corpus for document-centric probability approach of the first baseline. Out of 256000 URLs, there were some that contain images or text in languages other than English. Therefore, we do not consider these kinds of pages to test similarity. We extracted text from 213000 URLs that have a good text.

3.2 Expansion of Query

Generally, queries fall into three categories based on their nature [49]. Ambiguous queries have more than one meaning, e.g., Apple, Sara, and Harry Potter. Polysemy means a name with different meanings (e.g., Harry Potter can be a movie, a book, or a cartoon). Although combining other attributes of the knowledge base can address this problem, we can still derive an optimal query. As mentioned in the introduction and related work, query suggestion methods require information from users' past interactions, click-through data, or session-based data. Rather, we use an expansion-based scheme as a knowledge base. For experimentation purposes, we prepare a test knowledge base for the E-to-E scheme. It contains 200 entities and their attribute-oriented definitions, which are obtained from the web.

Our expansion scheme adds knowledge-full attributes as useful terms to the seed query to remove ambiguity. We made these definitions by exploring the web. For example, Harry Potter illustrates polysemy; it can refer to both a book and a movie. Our knowledge base contains one definition for each of them. Here, Book: Harry Potter, Author: Revenson Jody; ISBN: 978-0-439-10734-1; Book Format: Hardback; Publisher: Scholastic, Incorporated; Publication Date: July 1999; represents one definition and Harry Potter movie can be defined by Director: Chris Columbus Writers: J.K. Rowling (novel), Steve Kloves (screenplay) Stars: Daniel Radchile, Rupert Grint, Richard Harris, released on 2001. These attributes are collected after applying an entity resolution algorithm on the seed query-related web pages.

Users enter a seed query; our scheme then searches the knowledge base for related terms and returns the line numbers that contain these phrases, such as "Harry Potter". Once a phrase is found, then we read these lines and tokenize them in the form of n-grams. Then add these n-grams to the seed query as a useful term for preparing a query set Q. Then we generate a suggestion using this query set Q as input.

We store these computed suggestions in a database for future use too. So, when a user reenter any of these keywords our system can suggest the query. For example, when the user enters Harry Potter, we expand this using attributes values like Author: Revenson Jody; ISBN: 978-0-439-10734-1; Book Format: Hardback; Publisher: Scholastic, Incorporated; Publication Date: July 1999; and Subject: Fiction). This expansion will generate queries for Harry Potter. We combine entity attributes (Eatt) to form a query set, Q, consisting of n Boolean queries for every definition of the knowledgebase, Equation 1 defines the query set mathematically and Equation 2 shows the total number of queries generated:

$$Q = |P(E_{att})|\,, \tag{1}$$

$$|P(E_{att})| = 2^n\,. \tag{2}$$

Set Q is just like a power set, which contains all the subsets or combinations of entity attributes. Each of attribute or their combination represented by qi and each $q_i \in$ Q. Q is computed by equation 1 where $|P(E_{att})|$ is the power set of the entity. $|P(E_{att})|$ Contains 2^n queries as shown in equation 2 in total where n is number of attributes. For the "Harry Potter" case n=8, so the total number of queries is calculated as equation 3:

$$|P(E_{Att})| = 2^8 = 255. \tag{3}$$

Hence, Q contains 255 possible queries, each representing a distinct combination of entity attributes.

Query expansion process is illustrated below. Generally, the user enters initial seed query value like "Harry Potter" and entity type like Book our technique will form queries for it:

Entity= {a_1, a_2, a_3*} a set of QueriesQ: {<*a_1*>, <*a_2*>, <*a_3*>, <* a_1. a_2*>, <*a_2. a_3*>, <*a_3. a_1*>,*

< a_1. a_2. a_3 *>}.*

In mathematical model a_1, a_2, a_3 are attributes while '.' between the attributes represents the Boolean operator AND. The Boolean query is a query that includes Boolean operator AND, OR, and NOT in between the search keyword. In conjunction to the previous illustration about "Harry Potter", while searching with Bing for the keyword "Harry Potter Revenson, Jody" returns 9,070,000 of results while by inserting "AND" operator in between the two keywords such as "Harry Potter Revenson Jody" returns fewer results. We tried simple combining of attributes values with white spaces and with different Boolean operators like (AND, OR, NOT) during experimentation but the use of AND shows best results in query expansion.

Using above given method if α_1, α_2, α_3 represents Title, author and ISBN respectively our query set Q will be:

Q: < Harry Potter>, < Revenson, Jody>, < 978-0-06-240744-3>, < Harry Potter. Revenson, Jody>, < Revenson, Jody. 978-0-06-240744- 3>, < Harry Potter. 978-0-06-240744-3>,

< Harry Potter. Revenson, Jody. 978-0-06-240744-3 >.

After query expansion, we perform basic preprocessing to give all the queries a standard form because the query form affects the result fetched. We have given a certain form to attributes like 'ISBN', 'Pub Date' with data type Integer, date and time also we handle the attributes that can contain NULL values like 'Series'. Also, we remove punctuation marks and to give queries a Boolean form we add 'And' in queries which are formed by combining two are more attributes.

3.3 Execution of Query Set and Result Processing

After expansion, we perform some basic processing on these queries and execute them on selected search engines (Yandex and Bing) to extract the top 10 unique URLs in combined form and store them in a database as shown in Figure 3. Nowadays search engines return results in different sections, like related video, news, blogs, etc. Thus, we filter retrieved URLs and consider only unique web page URLs returned for q_i. Afterward, perform normalization on the retrieved URLs. URL normalization is a process to convert URLs into a standard and consistent format. The purpose of normalization is to convert a URL into a normalized URL, so it is possible to determine

whether the syntax in different URLs is the same or not.

3.4 Query Suggestion Techniques

After preparing the URLs database, we suggest the top-k queries based on the calculated weight for each query q_i. For simplicity, our proposed scheme assumes following assumptions for selecting optimal query suggestions.

If a URL(U_i) is retrieved by k different queries, then

Freq (U_i) = k.

If Freq (U_j) ≤ Freq (U_i), U_i is likely to be more relevant to the entity than U_j.

Thus, our hypothesis is "Query (q_i) that returns the highest frequency URLs is the optimal query". In our E-to-E scheme, we use two methods for query suggestions in parallel and suggest the query which is suggested by both. Our query suggestion schemes are.

3.4.1 MF-URLs Tree

Based on the given hypotheses, we use an approach that is a variation of FP-tree [50] to suggest the optimal query. We name this approach the MF-URLs tree. Algorithm 1, MF-URLs Tree Formation, uses the URLs database created.

Algorithm 1 MF-URLs Tree Formation

Input: The entire fetched URLs Database.

Output: MF-URLs Tree. Step-1:

1: Scan (URLs Database);
// Scan will read the whole URLs database
2: Append Q-list (qi with Sorted URLs fetched (Ui));
// It will generate a Q-list which has all queries from Q for each
qi
Step-2:
3: Create ("Root" for T as first node); 4: Read (Q-list for each (qi);
5: if (R! = empty) then
6: insert-tree(Ui |R; T);
7: if (Curr T = Ui) then
8: add 1 to Curr-T-count;
9: else
10: Create-node (Ui);
11: count = 1;
12: link (T);
13: end if
14: end if

There are two main steps in Algorithm 1. The first scans the whole database of URLs in statement 1 and then appends the URLs fetched from the database in Q-list, as mentioned in statement 2. It adds these queries and their URL variable representations, $U_1 \ldots U_n$, to the Q-list, which contains all queries and variables.

For example, the database showed in Figure 3 for seed query "Harry Potter". The query q_1 fetches U_1; U_2; U_4 while q_2 fetches U_1; U_2; U_3 and U_8; by combining these, we form the Q-list. The second part of Algorithm 1 creates the tree using Q-list. It starts with Root, building in statement 3, and inserts each U_i from Q-list as a node in the tree; if a node with the same label already exists, increases its count; otherwise, it inserts a new node in the tree. Statements 5 to 14 are repeated until the Q-list is empty.

Figure 5 is a graphical representation of MF-URLs tree formation against example shown in Figure 4 for seed query "Harry Potter". The MF-URLs example shows first five queries. Each leaf node represents a query path in the tree.

For example, encircled path3 represents q_3 which fetches U_1; U_2; U_3; U_4; U_5 and U_6. Some URLs appear in multiple paths, e.g., U_4, they are represented multiple times only to make it easy to understand, and their multiple occurrences are expressed with dotted lines and dotted ellipses, while their frequency is represented once at the first occurrence.

The second part of Algorithm 1 creates the tree using Q-list. It starts with Root, building in statement 3, and inserts each U_i from Q-list as for a node in the tree; if a node with the same label already exists, increases its count; otherwise, it inserts a new node in the tree.

Statements 5 to 14 are repeated until the Q-list is empty. After computing the sub-factors T_1, T_2, and T_3. for each query path in the MF-URLS tree, we calculate the overall weight of a query path using the following equation 4:

$$Weight(Path|q_i|) = a \times T_1 + \left(\frac{a}{2}\right) \times T_2 + \left(\frac{a}{2}\right) \times T_3 . \qquad (4)$$

URL	Freq	Var Name
http://www.tp-link.com/us/products/details/cat-5609_LB100.html	98	U_1
https://www.amazon.com/TP-Link-Dimmable-Equivalent-Assistant-LB100/dp/B01HXM8XF6	69	U_2
http://uk.tp-link.com/products/details/cat-5609_LB110.html	53	U_3
http://www.homedepot.com/p/TP-LINK-60-Watt-Smart-Wi-Fi-LED-Bulb-with-Energy-Monitoring-LB110/207104829	45	U_4
https://www.pcmag.com/article2/0,2817,2483488,00.asp	31	U_5
….	…	…
http://thewirecutter.com/reviews/best-smart-led-light-bulbs/	5	U_{n-1}
https://en.wikipedia.org/wiki/Smart_lighting	5	U_n

Q	Search Results					
q_1	U_1	U_2	U_4			
q_2	U_1	U_2	U_3	U_8		
q_3	U_1	U_2	U_3	U_4	U_5	U_6
q_4	U_1	U_2	U_3	U_4	U_5	U_7
q_5	U_1	U_2	U_3	U_4	U_6	
…	…	…	…	…	…	…
q_{255}	U_1	U_2	U_3	U_7		

Fig. 3. Snapshot of the database. The upper part represents URLs and their frequencies, and lower part represents the query and variable forms of the URLs after sorting

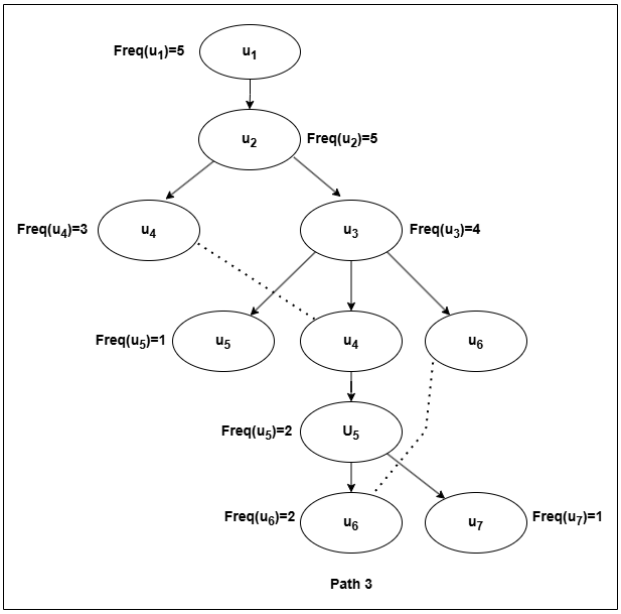

Fig. 4. MF-URLs Tree construction for Harry Potter data. Here dotted lines and dotted ellipses show URLs multiple occurrences. Encircled path3 represents the optimal query q_3

We suggest the top-k queries from the MF-URLs tree using the scoring scheme shown in equation 4. In this scheme, we use the sum of three sub-factors T_1, T_2, and T_3. Here α represents the significance of each factor.

We perform experiments with different values of α and based on the results we decide α = 0.5 is the most suitable value. In the scoring scheme for MF-URLs tree, we have assigned the highest weight to sub-factor T_1, which is based on frequency:

$$T_1 = \frac{\sum_{i=1}^{m} Freq(U_i)}{\sum_{j=1}^{m}\sum_{i=1}^{m}\{(Freq(U_{ij}))\}}. \quad (5)$$

Table 1. Transition matrix M for random walk on the query-URL bipartite graph

	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8
q_1	1/3	1/3		1/3				
q_2	¼	1/4	1/4					1/4
q_3	1/6	1/6	1/6	1/6	1/6	1/6		
q_4	1/6	1/6	1/6	1/6	1/6		1/6	
q_5	¼	1/4	1/4			1/4		

Calculation of T_1 is shown in equation 5, which is based on the MF-URLs value. In equation 5 nominator represents the sum of frequencies of all the pages returned by a query (e.g. Path in MF-URLs tree), and it is normalized by dividing the overall sum of all the URL frequencies in the tree, represented by Freq (U_{ij}):

$$T_2 = avg_s\left(Path_{qi}\ URL\right). \quad (6)$$

The second sub-factor we use for optimal query suggestion is text similarity between query terms, the text of web pages and URLs. In equation 6, T_2 is the average similarity score between the query and all the resulting URLs fetched from it. Uniform resource locators (URLs) themselves comprise a lot of information. For example, the URL "http://www.ebooks.com/1724792/the-wright-brothers/mccullough-david" suggests a home page of eBooks containing the title The Wright Brothers and author "David McCullough". After normalization of URLs, a segment of the URL provides "www.ebooks.com" which becomes "www", "ebooks" and "com", or we get lengths of tokens, orthographic features, sequential n-grams, and sequential bi-grams. In our work, we extract n-grams from URLs to check for similarity to the query:

$$T_3 = avg_s\left(Path_{qi}P_j\right). \quad (7)$$

While in equation 7, T_3 represents the average similarity score between the query terms and the text of URLs. T_2 and T_3 help in calculating the relevance score of a web page and query. We extract the text from all unique URLs 2 and store it as P_j. We use Alchemy API 2 for text extraction. After text retrieval from URLs, we perform some basic text processing on each P_j and URL U_i. We tokenize URLs with the help of non-alphanumeric characters as a boundary. We remove English stop words, including web-specific stop words, as well as file and domain extensions, etc.

Then we generate n-grams of lengths 2 and 3 for URLs and use the sentences found in the web page text. Punctuation marks ("," "." etc.) are removed. N-grams that consist of stop words only or that do not contain at least one alphanumeric character (e.g. n-grams like "at the" or "#, @") are removed.

We construct the entity feature vector as follows: Q(terms) = t_1 (e); t_2 (e); tm(e) where terms represent attributes from knowledge-based means a query can contain m attributes. While t_1 (e) and t_2 (e) n-grams were extracted from attributes of entity. We do not use stemming because stemming is particularly problematic in web search, where users often use special or new words in their queries. (A standard stemmer, such as Porter's, will stem them incorrectly.)

We remove sparsity in our work by constructing a vector for only query terms, not considering the other terms from a web document. The bag of words approaches faces problems if the terms contain any spelling mistakes (e.g. with a unigram).

So, we use n-grams rather than a collection of unigrams, where occurrences of pairs of consecutive words are counted. In our query suggestions, we use selected term-based Jaccard distance on web page text and URLs, first, because of its simplicity and it can be used for BOW, which means we can use it to measure the similarity between sets; and second because we are only interested in shared terms between query and web pages. Jaccard distance is a popular measure, defined as equation 8:

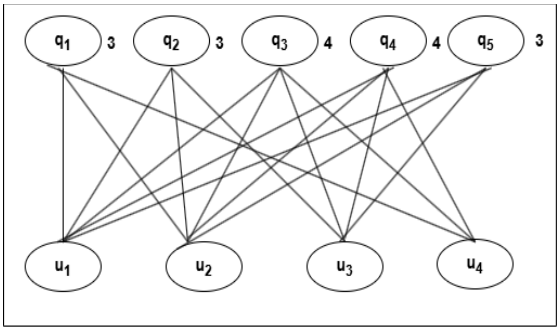

Fig. 5. Query-URL bipartite graph for same data shown in a snapshot of the database for Harry Potter. Here five queries are represented by Q_i and URLs by U_i

$$Jaccard(q_i, P_j) = q_i \cup P_j - q_i \cap \frac{P_j}{q_i} \cup P_j. \quad (8)$$

Here q_i and P_j are queries and pages, the value of the defined similarity between query and page is in the range [0, 1].

3.4.2 Random Walk on a Query-URL Bipartite Graph

The second method we used for top-k query suggestions is based on a random walk on a query-URL bipartite graph. A query is a point in high-dimensional space [51], where each dimension corresponds to a unique URL; i.e. a query can be given a vector representation based on all the different URLs in its cover. Several graph theories are in existence for this purpose. One study identified several types of query graphs [51], the bipartite graph being one of them, widely in use for web data mining and to represent the relationship between queries and pages [52, 53].

Bipartite graphs represent a mapping between the queries set Q and the URL set U. We model the web search as a query-URL bipartite graph G = (Q; U; E), where nodes are divided into two separate sets: Q, the set of queries, and U, the set of URLs. Every edge in set E links a query and a URL. There is an edge between a query and a URL if the query fetches from the web page. Query and URL sets maintain a many-to-many relationship with each other.

Using this relationship among queries and URLs, we create a bipartite graph. This graph represents a natural judgment of the relationships between query and URLs. In our system, each record has a query and URLs with their appropriate degrees. In a bipartite graph, edges in E represent the confidence values of the query and pages in the search results.

Figure 6 represents our proposed bipartite graph for the same database shown in Figure 4. Our query-URL graph gave us a summary of co-occurrence between query-URL pairs. In our probabilistic retrieval model, we fulfill an information need with a query set Q. Given an undirected graph G = (Q, U, E) with l = |Q| queries, m = |U| URLs and n = |E| edges. The natural random walk starts with one of these random queries, q_i, and then selects one of the fetched URLs using the n×n transition matrix M shown in Table 1.

Figure 5 represents this random walk on a bipartite graph of query-URLs where the number of queries is five and there are eight URLs. Per example, our random walk starts by choosing random query q_i from set Q, and then it selects a random URL U_i connected to q_i, and from this URL U_i, we select another connected query, and so on. This process of query-URL and the URL-query transition is repeated, or it stops at a query or URL node based on the value of n. The result is a distribution of queries and web pages, describing how likely they co-occur. Our model follows simple given assumptions. Our random walk follows a memoryless property.

In our random walk model, we limit the number of transitions and keep them to the information needed. It is also a simplifying assumption that our random walk only visits an edge once, meaning that if two nodes are connected by an edge, once we have traversed them, we will choose a new random edge the next time. Our random walk follows Algorithm 2 and executes in the following manner; notations used are heavily influenced by the study from Jaakkola and Szummer [54].

Let U be the set of URLs and let Q be the set of queries. We construct bipartite graph G, where the number of edges, E, assigns weights to the queries and URLs, given by incoming edge counts (degree dq_i; du_i) of each node. We define transition probabilities $Pt+1|t(u_i\ |q_i)$ from query q_i to URL u_i and vice versa, so $Pt+1|t(u_i\ |q_i) = 1/dq_i$, where i ranges over all connected URL nodes with

a given query. The notation $P t_2|t1(q_i|u_i)$ will denote the transition probability from node u_i at time t_1 to node q_i at time t_2, while transition probabilities $Pt+1|t(u_i|q_i)$ generally is not the same because the number of degrees varies across nodes.

In our random walk model on the query-URL bipartite graph, probability of transitioning from a URL u_i to a connected query q is determined by the degree of the URL node. Formally, this transition probability is given by equation 9:

$$P_{t+1|t}(q|u_i) = \left\{\frac{1}{d_{ui}}, if\ uq \in E, 0\ \text{otherwise}\right\}. \quad (9)$$

We organize the transition probabilities as a matrix M whose u: q entry is $Pt+1|t(u_i|q_i)$. The matrix M is row stochastic so that rows sum to 1. Then, we perform the random walk: we calculate the probability of transitioning and degrees from node q_i to node u_i in time t, denoted by $Pt|t-1(u_i|q_i)$, and it is equal to $Pt|t-1(u_i|q_i) = Mt(q_i u_i)$.

Algorithm 2 Random Walk on a Bipartite Graph

Input: q_0=seed query, run size n = |E|.

Output: Sample queries and their degrees; Sample Pages with Degree.

```
1: Matrix R and list Us are empty;
// Matrix R and list Us will store degrees of URLs because
of Random Walk.2: i=1;
3: while i ≤ n do
4:      Ui=random(URL);
// Assign one random URL which is fetched by the query.
5:          if qiUi cell!=Null then
6:              qiUi cell = dui;
//Store degree of URL both in US and R matrix against qiUi.
7:      Else
8:              dom(URL)
9:              Ui cell = dui;
10:             ndom(qi);

// Assign one random Query which is fetching URL.11:
          end if
12: end while=0
```

Algorithm 2 represents the details of our random walk process on a bipartite graph and, according to this; we always visit a single node once, which will populate R and Us.

Us is a list, which will store the degrees of URLs, while matrix R will store the degrees of URLs in reference q_i: u_i transitions. Both these data structures help in the confidence score calculation foreach query and URL.

The confidence of a query or URL is theratio of the co-occurrence of the query/URL pair in the collection of results.

The ratio value is always ≤ 1 which makesit easier to process than showing actual high co-occurrence values.

Our work calculates confidence scores as follows:

$$U_c = \frac{U_f}{|Q|}, \quad (10)$$

$$Q_c = \sum(d_{url}\ of\ q_i) / \sum(d_{url}\ of\ all\ URLS). \quad (11)$$

Equation 10 calculates confidence in the URL. It is represented as U_c (URL confidence) U_f represents the URL fetched by the number of queries in the graph and |Q| is the total number of queries in the query set.

Equation 11 shows the confidence value of query q_i. Here, the numerator is the sum of the degrees of all URLs fetched by q_i and the denominator is the sum of degrees for all URLs fetched by all queries.

3.5 Assessment Method

Next, we generated the query suggestions for each ambiguous entity using both proposed methods (MF-URLs tree and random walk) mentioned in Section III and pooled the results based on ambiguous entities.

To evaluate the results, it is necessary to have the human judgment to check whether the web documents found are suitable representations of the entities.

Therefore, a simple relevance assessment was performed using five assessors. To support the relevancy decision, we provided the assessors with the ambiguous entity's label and a short description taken from our attribute-oriented definitions of knowledgebase.

Each assessor was first presented with 40 entities and URLs at random.

They assessed the pages as relevant (1) or irrelevant (0). This relevance check helps in evaluating the E-to-E scheme and baselines in the comparative study.

Table 2. Assessment of suggested queries by all schemes for seed query Harry Potter. Assessors rate query 1(useful), 2 (somewhat useful) and 3 (not useful)

Scheme	Suggested Query	1	2	3
E-to-E	Harry Potter & Revenson, Jody & 978-0-06-240744-3 & Hardback. (Path 3)	Y		
DCPM	Harry Potter movies		Y	
GAC	Harry Potter World			Y
CTBQS	Harry Potter Jody Revenson	Y		

Table 3. Top 2 query suggestions for MF-URL tree-based scheme. Here, q_3 and q_4 are suggested for seed query Harry Potter based on score shown by weight column

Query	T_1	T_2	T_3	Weight (Path)
Path (q_3)	0.91	0.56	0.56	0.74
Path (q_4)	0.87	0.46	0.52	0.68
Path (q_5)	0.70	0.38	0.42	0.55
Path (q_2)	0.65	0.45	0.30	0.51
Path (q_1)	0.57	0.41	0.45	0.50

Table 4. Top 2 query suggestions using Random walk on the bipartite based scheme. Here too, q_3 and q_4 are suggested for seed query Harry Potter shown by right most columns

Query#	u_1	u_2	u_3	u_4	u_5	u_6	u_7	u_8	Σ(*d*)	Σ(*d*)/Σ(*E*)
***q*₃**	5	5	4	3	2	2			21	**0.91**
***q*₄**	5	5	4	3	2		1		20	**0.87**
q_5	5	5	4		2				16	0.70
q_2	5	5	4					1	15	0.65
q_1	5	5		3					13	0.57

Then, assessors also evaluated the suggested queries by E-to-E and baselines and for each ambiguous entity. The assessors were asked to judge the suggested queries as useful (1), somewhat useful (2), or not useful (3) as shown in Table 2.

We provided them with the definition of ambiguous entities and the query suggestions of schemes.

A very broad instruction was given: a useful suggested query is such that, if the user submits it to the search engine, it provides results with high precision.

3.6 Suggested Queries

We first discuss the suggestion results for the MF-URLs tree explained in section III. We cannot include the complete MF-URLs tree in this paper due to space limitations, so we present a sample tree for only five queries as shown in table 3.

Table 3 shows the weights for queries using all sub-factors, which represent the importance of the query and provides the base for queries suggestions. Here, these results show that the top query is represented by the path (q_3) with the highest weight 0.74, while the second top query is a path (q_4) with a weight of 0.68. In an MF-URLs tree, every leaf node represents a query path. Then, we applied the weighting scheme shown in (3) to assign the appropriate weight to the query paths. We calculated T_1; T_2 and T_3 for each of the five queries.

In the result, T1 is a value that is based on the frequency of URLs. It clearly supports our hypothesis that most-frequent URLs can play a vital role in ranking search results, and the query

that represents most of the high-frequency pages is an optimal query, as shown by query 3. This means the page withthe high frequency can be presented as the most relevant page. Results show that we can use only frequency factors, but by using T_2 and T_3, we ensure the content-based similarity of query and pages. In the second method, we applied random walk on a query-URL bipartite graph on the prepared database.Figure 6 represents the sample random walk for seed query HarryPotter. Here, we set n=23 which is the total number of edges in table 4.

In each iteration, the random walk will visit one of the nodes from the query or the URLs and either update the Us list or matrix R. Here, the scheme will fill the cells using source and destination node name qiuJ and will enter the degrees of the URL in the appropriate cell. Using these degrees shown by R, we calculate confidence in the query and using the Us list we check the confidence in the URL and rank both per their confidence scores. According to table 4, the top 2 queries are q_3 and q_4, with confidence scores is 0.91 and 0.87. While ranking, the URLs based on degree, the order is u_1; u_2; u_3; u_5 and so on.

During the experiments, we observed that the queries thatconsist of one attribute fetches the highest number of results, but as we combine more attributes, the number of results decreases, which is the main motivation behind the expansion technique we use in this paper. While combining attributes, wealso observed that after a certain length, the queries fetch 0 results, so there should be a limit when combining attributes. After executing both the query suggestion methods in parallel, now we can select the query for the suggestion. From the resultsshown above, we can choose query 3 for the suggestion as both methods show agreement on that. For 200 ambiguous entities, more than 80% both methods suggest the same query. But sometimes they differ because of similarity factors of MF-URL method, for those cases scheme suggests 2nd or 3rd best queryon which they agree.

3.7 Proposed Technique

As we described multiple times in my research, we present a novel scheme E -to-E in this paper which is per our knowledge first scheme based on the expansion of query. So, it is difficult to compare it with existing which is mostly based on based Corpus-based knowledge [13-15], Relevance feedback [16-23] and Language model-based [24-26]. But we take three methods to show the worth of our proposed scheme.

Document-Centric Probabilistic Mechanism (DCPM): It is documented corpus-based approach based on probabilistic values of extracted phrases [55]. For comparison purposes, we test this probability-based approach on text corpus, which we formed by extracting text from all web pages for similarity check.

Click-Through-Based Query Suggestion (CTBQS): This approach is based on Query Logs [56]. For comparison purpose, we extract one month AltaVista query log and apply click-through-based approach on it for suggesting queries for ambiguous entities.

State of the Art Auto-completion method (GAC): This state-of-the-art method is used by Google and Bing. This scheme is based on real searches (means query logs) and ranking based on popularity factors and personalization [57]. Here we perform searches for ambiguous entities on Google and Bing and find their suggestion and compare it with our scheme.

3.8 Evaluation Matrix

No standard evaluation matrix has been proposed so far for query suggestion schemes. For our proposed scheme, we use following evaluation measures: (Precision-recall and F-measure) k, Coverage, Success Rate and MAP (mean average precision)@9.

4 Results and Discussions

The human assessment result shows that 87% of the query suggestions of our scheme E-to-E are useful for ambiguous entity search, as shown in Table 5. While 7.3% of suggestions are not useful these are mostly those queries where our scheme found disagreements between the suggestions. While 3% are not useful because of poor definition formation of the ambiguous entity in the knowledgebase.

Table 5. Quality Distribution of Assessment for Suggested Queries of E-to-E Scheme

Assessment	Percentage
Useful	87
Somewhat Useful	7.3
Not Useful	3
Cannot assess	2.7

Table 6. Results in comparison between the proposed scheme and baseline

Scheme	MAP@9	Coverage %	Success Rate %
E-to-E	0.7017	100	89
DCPM	0.4436	63	50
GAC	0.1642	87	50
CTBQS	0.4218	60	35

Table 6 shows the results of MAP@9 query suggestions, coverage of the schemes and success rate for them. Mean Average Precision (MAP) is the typical solitary numeral measure for comparing query suggestions. During the experiments, we found that main reason behind this huge decline in search engine efficiencies is because of the drift created by adding accurate terms than query intent. The entire baseline most of the time adds terms which show the drift from the user intent. While proposed E-to-E scheme provides terms with the help of attribute-oriented definitions, that is why it does not suffer from drift issues and shows good results.

Our Scheme E-to-E shows edge on other schemes with a MAP score of 0.7017 for top 9 query suggestions. This shows that queries suggested by our scheme fetch a high number of relevant pages than others. Our scheme E-to-E shows the higher MAP, first because accuracy provided by the attributed oriented definitions of the knowledge base and second because we have two-fold validation of suggested query. DCPM is not performing well because of corpus quality. DCPM extract expansion terms based on the probability of their co-occurrence with the keyword of seed query in the built corpus. Hence, there is a probability that a non-required term is co-occurring mostly with seed query. For example, “movie" is a most occurring term with “Harry Potter" keyword in the corpus so it suggested query “Harry Potter movie" which is not relevant to the context. While GAC works on real searches (query log) and suggests based on popularity and personalization factors, therefore it suggests query “Harry Potter world". This is searched by most users using Google. When we use this suggestion, it fetches pages from Harry Potter and world related terms, consequently, have low precision.

CTBQS is a click-through-based scheme. It performs worse because the available query log AltaVista_2003 is very old and does not contain up to date query URL clicking through the record. Coverage is the number of queries for which scheme suggests at least one query, regardless they are related to user intent or not. Our scheme E-to-E suggested queries for almost every ambiguous entity we searched for. However, this totally depends on the formation of knowledgebase.

DCPM and CTBQS, unable to find co-occurring terms because of completeness issues with document corpus and out-of-date query-log. Thus, they cannot produce query suggestions. GAC performs better than other two because of live search and better weblog but is unable to produce queries for seed queries which are longer than 3 terms or numbers like ISBN while our scheme also suggests the same query for any attribute term, thus shows 100% coverage for the given entities. Success rate shows the ratio of meaningful query suggestions by schemes. DCPM and CTBQS do not have a good success rate because these techniques select poor terms for expansion from a corpus and query log for the suggestion. While GAC failed to do so because of popularity and region-based auto-completion method and if we are searching for any old information this scheme will not produce any meaning full suggestion.

Our scheme produces best results because it always matches the seed query with appropriate attribute-oriented definitions of knowledgebase, which suggests meaningful query mostly. Most common measures for checking the effectiveness of search strategies are recall and precision. These measures are introduced to summarize

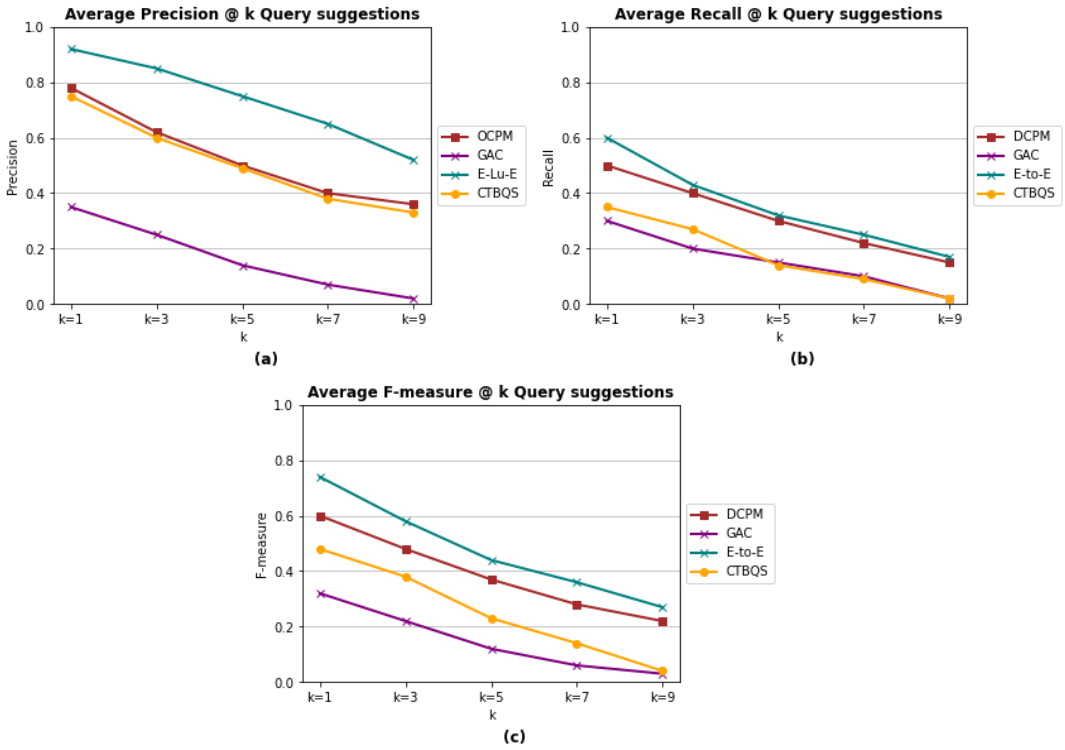

Fig. 6. Average precision recall and F-measure comparison for top 9 suggestions between the E-to-E scheme and baseline measures

and compare search results mentioned by Saracevic [58].

Naturally, recall checks how good the search engine is doing when searching all the relevant URLs for a suggested query, and precision calculates how well it is rejecting irrelevant URLs. Our focus is on improving precision when searching ambiguous entities using a search engine. Labeling of web pages with relevant or irrelevant is performed by assessors in our case. A natural question is; why do we use precision? The answer given by Croft et al. [59] is very simple, “most users are interested in precision only while using a search engine". For example, a precision of 0.7 means 70% of the retrieved URLs are relevant, and there are more chances that the user will get the information needed. We use an evaluation metric based on precision, recall, and F-measure (F-1 score) are used with the number of URLs fetched to compare the performance of top-k suggested queries. We use F-measure because we are interested in checking the average performance of our query suggestion methods for entity search.

Figure 6 represents the comparison for the top 9 queries in the form of average precision, recall and F-measure between the baseline and proposed scheme. All sub-Figures of Fig 6 show that E-to-E scheme gives the highest precision, recall, and F=measure during experiments. For top 1 query suggestion, CTBQS and DCPM schemes achieve precision ≥ 0:70, recall ≥ 0:35 and F-measure ≥ 0:48. While our proposed scheme achieves precision ≥ 0:90, recall ≥ 0:59 and F-measure ≥ 0:70. This difference in results is because of query expansion methods and accuracy of terms provided by our scheme. The experiments show that as we increase the value of K average precision, recall and F-measure of each scheme decrease due to the irrelevancy of

suggested terms for expansion. However, still, our proposed scheme able to achieve the precision ≥ 0:50, recall ≥ 0:15 and F-measure ≥ 0:25 for top 9 queries which are higher than others. While from Fig 6 it can be clearly observed that our proposed scheme easily outperformed the baselines.

In last we compare seed queries and the suggested queries for all entities. The result shows that there is a huge improvement in the F-measure if we use the suggested query proposed by our E-to-E scheme, based on the average, we can say that there is approximately a 57% improvement.

5 Conclusions and Future Work

In this work, we proposed an Expansion-Based-Query-Suggestion Scheme Entity-to-Entity (E-to-E) to fetch search results with higher precision. The proposed work distinguishes and finds ambiguous entities. Our scheme suggested queries after expanding the seed query with the help of attribute-oriented definitions of a knowledgebase. We verified our proposed scheme E-to-E with the help of evolution metric based on MAP@k, coverage, success rate, precision, recall, and F-measure with the help of human assessment for suggested queries. In this work, we compare our results with three different baselines. Our experimental results showed that our suggestions achieve scores equal to MAP@9 = 0:7017, Coverage = 100, Success rate = 89 and F−measure = 0.72. The results endorse the effectiveness of our proposed scheme in helping the user to access relevant information. In the future, we plan to expand our work for general search and to clustering techniques.

References

1. **Stats, I.L. (2017).** Internet Live Stats, Internet Live Stats. internetlivestats.com (2002–2017).
2. **Jansen, B.J., Spink, A., Pedersen, J. (2005**). A Temporal Comparison of AltaVista Web Searching, Journal of the American Society for Information Science and Technology, Vol. 56, No. 6, pp. 559–570.
3. **Wen, J.R., Nie, J.Y., Zhang, H.J. (2001).** Clustering User Queries of a Search Engine, Proceedings of the 10th International Conference on World Wide Web, pp. 162–168.
4. **Cui, H., Wen, J.R., Nie, J.Y., et al. (2002).** Probabilistic Query Expansion Using Query Logs, Proceedings of the 11th International Conference on World Wide Web, pp. 325–332.
5. **Spink, A., Jansen, B.J. (2004). A** Study of Web Search Trends, Webology, Vol. 1, No. 2, pp. 4.
6. **Song, R., Luo, Z., Nie, J.Y., et al. (2009).** Identification of Ambiguous Queries in Web Search, Information Processing & Management, Vol. 45, No. 2, pp. 216–229.
7. **Jansen, B.J., Spink, A., Bateman, J., et al. (1998).** Real Life Information Retrieval: A Study of User Queries on the Web, ACM SIGIR Forum, Vol. 32, No. 1, pp. 5–17. New York, NY, USA: ACM.
8. **French, J.C., Brown, D.E., Kim, N.H. (1997).** A Classification Approach to Boolean Query Reformulation, Journal of the American Society for Information Science, Vol. 48, No. 8, pp. 694–706.
9. **Buckley, C., Singhal, A., Mitra, M., et al. (1995). New** Retrieval Approaches Using SMART: TREC 4, Proceedings of the Fourth Text Retrieval Conference (TREC-4), pp. 25–48.
10. **Vélez, B., Weiss, R., Sheldon, M.A., et al. (1997).** Fast and Effective Query Refinement, ACM SIGIR Forum, Vol. 31, No. SI, pp. 6–15. New York, NY, USA: ACM.
11. **Krovetz, R., Croft, W.B. (1992).** Lexical Ambiguity and Information Retrieval, ACM Transactions on Information Systems (TOIS), Vol. 10, No. 2, pp. 115–141.
12. **Peat, H.J., Willett, P. (1991).** The Limitations of Term Co-Occurrence Data for Query Expansion in Document Retrieval Systems, Journal of the American Society for Information Science, Vol. 42, No. 5, pp. 378–383.
13. **Fellbaum, C. (1998).** A Semantic Network of English: The Mother of All WordNets,

Computers and the Humanities, Vol. 32, No. 2, pp. 209-220.

14. **Zohar, H., Liebeskind, C., Schler, J., et al. (2013).** Automatic Thesaurus Construction for Cross Generation Corpus, Journal on Computing and Cultural Heritage (JOCCH), Vol. 6, No. 1, pp. 1–19.
15. **Jones, K.S., Willett, P. (Eds.). (1997**). Readings in Information Retrieval. Morgan Kaufmann.
16. **Manning, C.D. (2008).** Introduction to Information Retrieval. Syngress Publishing.
17. **Cao, G., Nie, J.Y., Gao, J., et al. (2008).** Selecting Good Expansion Terms for Pseudo-Relevance Feedback, Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 243–250.
18. **Saneifar, H., Bonniol, S., Poncelet, P., et al. (2014).** Enhancing Passage Retrieval in Log Files by Query Expansion Based on Explicit and Pseudo Relevance Feedback, Computers in Industry, Vol. 65, No. 6, pp. 937–951.
19. **Preston, C.C., Colman, A.M. (2000).** Optimal Number of Response Categories in Rating Scales: Reliability, Validity, Discriminating Power, and Respondent Preferences, Acta Psychologica, Vol. 104, No. 1, pp. 1–15.
20. **Chapelle, O., Metlzer, D., Zhang, Y., et al. (2009).** Expected Reciprocal Rank for Graded Relevance, Proceedings of the 18th ACM Conference on Information and Knowledge Management, pp. 621–630.
21. **Kelly, D., Teevan, J. (2003).** Implicit Feedback for Inferring User Preference: A Bibliography, ACM SIGIR Forum, Vol. 37, No. 2, pp. 18-28. New York, NY, USA: ACM.
22. **Joachims, T., Granka, L., Pan, B., et al. (2017).** Accurately Interpreting Clickthrough Data as Implicit Feedback, ACM SIGIR Forum, Vol. 51, No. 1, pp. 4-11. New York, NY, USA: ACM.
23. **Carpineto, C., Romano, G. (2012).** A Survey of Automatic Query Expansion in Information Retrieval, ACM Computing Surveys (CSUR), Vol. 44, No. 1, pp. 1-50.
24. **Song, D., Peter, B. (2005).** Query Expansion Using Term Relationships in Language Models for Information Retrieval, Proceedings of ACM CIKM, pp. 688-695.
25. **Buscher, G., Dengel, A., Biedert, R., et al. (2012).** Attentive Documents: Eye Tracking as Implicit Feedback for Information Retrieval and Beyond, ACM Transactions on Interactive Intelligent Systems (TiiS), Vol. 1, No. 2, pp. 1-30.
26. **Meng, L. (2014).** A Survey on Query Suggestions, International Journal of Hybrid Information Technology, Vol. 7, No. 6, pp. 43-56.
27. **Leung, K.W.T., Ng, W., Lee, D.L. (2008).** Personalized Concept-Based Clustering of Search Engine Queries, IEEE Transactions on Knowledge and Data Engineering, Vol. 20, No. 11, pp. 1505-1518.
28. **He, Q., Jiang, D., Liao, Z., et al. (2009).** Web Query Recommendation via Sequential Query Prediction, 2009 IEEE 25th International Conference on Data Engineering, pp. 1443-1454. IEEE.
29. **Cucerzan, S., White, R.W. (2007).** Query Suggestion Based on User Landing Pages, Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 875–876.
30. **Ma, H., Yang, H., King, I., et al. (2008).** Learning Latent Semantic Relations from Clickthrough Data for Query Suggestion, Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 709–718.
31. **Mei, Q., Zhou, D., Church, K. (2008).** Query Suggestion Using Hitting Time, Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 469–478.
32. **Song, Y., He, L.W. (2010).** Optimal Rare Query Suggestion with Implicit User Feedback, Proceedings of the 19th International Conference on World Wide Web, pp. 901–910.
33. **Tong, H., Faloutsos, C., Pan, J.Y. (2008).** Random Walk with Restart: Fast Solutions and Applications, Knowledge and

Information Systems, Vol. 14, No. 3, pp. 327–346.

34. **Craswell, N., Szummer, M. (2007).** Random Walks on the Click Graph, Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 239–246.
35. **Baluja, S., Seth, R., Sivakumar, D., et al. (2008).** Video Suggestion and Discovery for YouTube: Taking Random Walks Through the View Graph, Proceedings of the 17th International Conference on World Wide Web, pp. 895–904.
36. **Chen, Y., Zhang, Y.Q. (2009).** A Personalised Query Suggestion Agent Based on Query-Concept Bipartite Graphs and Concept Relation Trees, International Journal of Advanced Intelligence Paradigms, Vol. 1, No. 4, pp. 398–417.
37. **Bai, L., Guo, J.F., Cao, L., et al. (2013).** Long Tail Query Recommendation Based on Query Intent, Jisuanji Xuebao (Chinese Journal of Computers), Vol. 36, No. 3, pp. 636–642.
38. **Gao, W., Niu, C., Nie, J.Y., et al. (2010).** Exploiting Query Logs for Cross-Lingual Query Suggestions, ACM Transactions on Information Systems (TOIS), Vol. 28, No. 2, pp. 1–33.
39. **Yang, J.M., Cai, R., Jing, F., et al. (2008).** Search-Based Query Suggestion, Proceedings of the 17th ACM Conference on Information and Knowledge Management, pp. 1439–1440.
40. **Strohmaier, M., Kröll, M., Körner, C. (2009).** Intentional Query Suggestion: Making User Goals More Explicit During Search, Proceedings of the 2009 Workshop on Web Search Click Data, pp. 68–74.
41. **Sadikov, E., Madhavan, J., Wang, L., et al. (2010).** Clustering Query Refinements by User Intent, Proceedings of the 19th International Conference on World Wide Web, pp. 841–850.
42. **Hotho, A., Jäschke, R., Schmitz, C., et al. (2006).** Information Retrieval in Folksonomies: Search and Ranking, European Semantic Web Conference, pp. 411–426. Berlin, Heidelberg: Springer Berlin Heidelberg.
43. **Song, J., Xiao, J., Wu, F., et al. (2017).** Hierarchical Contextual Attention Recurrent Neural Network for Map Query Suggestion, IEEE Transactions on Knowledge and Data Engineering, Vol. 29, No. 9, pp. 1888–1901.
44. **Liu, Q., Gao, Y., Zhou, L., et al. (2017).** IS2R: A System for Refining Reverse Top-k Queries, 2017 IEEE 33rd International Conference on Data Engineering (ICDE), pp. 1371–1372. IEEE.
45. **Cai, F., Liang, S., de Rijke, M. (2016).** Prefix-Adaptive and Time-Sensitive Personalized Query Auto Completion, IEEE Transactions on Knowledge and Data Engineering, Vol. 28, No. 9, pp. 2452–2466.
46. **Zhou, D., Wu, X., Zhao, W., et al. (2017).** Query Expansion with Enriched User Profiles for Personalized Search Utilizing Folksonomy Data, IEEE Transactions on Knowledge and Data Engineering, Vol. 29, No. 7, pp. 1536–1548.
47. **Bhattacharya, N., Arpinar, I.B., Kursuncu, U. (2017).** Real Time Evaluation of Quality of Search Terms During Query Expansion for Streaming Text Data Using Velocity and Relevance, 2017 IEEE 11th International Conference on Semantic Computing (ICSC), pp. 280–281. IEEE.
48. **Song, R., Luo, Z., Wen, J.R., et al. (2007).** Identifying Ambiguous Queries in Web Search, Proceedings of the 16th International Conference on World Wide Web, pp. 1169–1170.
49. **Han, J., Pei, J., Yin, Y. (2000).** Mining Frequent Patterns Without Candidate Generation, ACM SIGMOD Record, Vol. 29, No. 2, pp. 1–12.
50. **Baeza-Yates, R. (2007).** Graphs from Search Engine Queries, International Conference on Current Trends in Theory and Practice of Computer Science, pp. 1–8. Berlin, Heidelberg: Springer Berlin Heidelberg.
51. **Beeferman, D., Berger, A. (2000).** Agglomerative Clustering of a Search Engine Query Log, Proceedings of the Sixth ACM

SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 407–416.

Article received on 21/01/2026; accepted on 11/04/2026.
**Corresponding authors is Liliana Chanona-Hernandez.*