Concept-Based Web Search

Yue Wang[1], Hongsong Li[2], Haixun Wang[2], and Kenny Q. Zhu[3]

[1] University of Massachusetts, Amherst
[2] Microsoft Research Asia
[3] Shanghai Jiao Tong University

Abstract. Traditional web search engines are keyword-based. Such a mechanism is effective when the user knows exactly the right words in the web pages they are looking for. However, it doesn't produce good results if the user asks for a concept or topic that has broader and sometimes ambiguous meanings. In this paper, we present a framework that improves web search experiences through the use of a probabilistic knowledge base. The framework classifies web queries into different patterns according to the concepts and entities in addition to keywords contained in these queries. Then it produces answers by interpreting the queries with the help of the knowledge base. Our preliminary results showed that the new framework is capable of answering various types of concept-based queries with much higher user satisfaction, and is therefore a valuable addition to the traditional web search.

1 Introduction

Keyword based search works well if the users know exactly what they want and formulate queries with the "right" keywords. It does not help much and is sometimes even hopeless if the users only have vague concepts about what they are asking. The followings are some examples of such "conceptual queries":

Q1. database conferences in asian cities
Q2. tech companies slogan

Although the intentions of these queries are quite clear, they are not "good" keyword queries by traditional standard. In the first query, the user wants to know about the database conferences located in Asian cities, without knowing the names of the conferences or cities. In the second query, the user wants to find out the various slogans of tech companies.

Fig. 1 and 2 show the top results returned by Google and Bing respectively. None of these results render meaningful answers to the actual questions implied by the queries. Most of them are likely to be considered irrelevant by the user. For example, the top two results in Fig. 1 have nothing to do with Asian cities, or database conferences. Fig. 2 happens to return a page with a list of company slogans, not because these are tech companies but because one of the company names contains the word "tech".

Apparently, *database conferences*, *asian cities* and *tech companies* are abstract concepts while *slogan* can be regarded as an attribute of tech companies. None of these are

Kenny Q. Zhu was partially supported by NSFC grants No. 61033002 and 61100050.

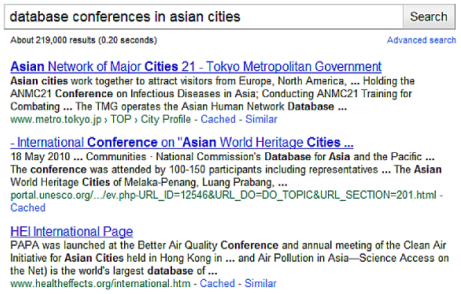


Fig. 1. Google results for Q1

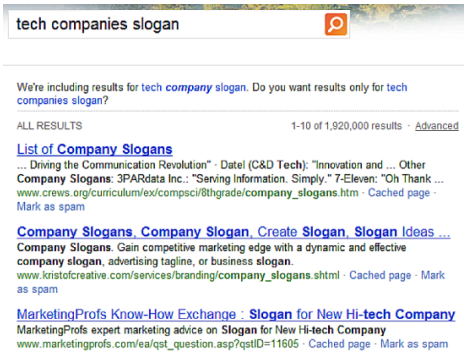


Fig. 2. Bing results for Q2

keywords in the traditional sense. Traditional search engines are good at handling entities in these concepts (e.g. VLDB as an entity of database conferences), But in reality, quite a significant percentage of web queries are not *entity only* queries. Our statistics from the search log of a major search engine in last two years suggests that about 62% of the queries contain at least one conceptual class term (see Fig. 6). To better serve such conceptual queries, we need to understand concepts in web pages.

In this paper, we present a framework that leverages a knowledge base and query interpretation techniques to improve web search on concept-related web queries. This knowledge base, named Probase [1–5], is a general-purpose probabilistic taxonomy, automatically constructed by integrating information from web pages and other more reliable data sources such as Freebase [6] and Wordnet [7]. It contains large number of concepts, entities and attributes, organized in a hierarchical structure with subsumption, similarity and other relations. Here, an *entity* refers to a specific object, a *concept* refers to a collection of things, and an *attribute* refers to a property of one or more objects. With the help of Probase, we can identify concepts and attributes in queries and interpret them by replacing the concepts with their most likely entities, and hence formulate more accurate keyword-based queries. The results of these new queries from the traditional search engines can then be ranked and presented to users, in addition to normal keyword queries.

Fig. 3 shows the top search results of the query "database conferences in asian cities" from our prototype system. These results do not contain the keywords "database conferences" or "asian cities", but instead directly gives information about three recent VLDB conferences that were actually hosted in Asia. This information is a lot more targeted and relevant from the user perspective. When serving the query "tech companies slogan", our prototype system realizes that the information is immediately available from the taxonomy, and hence presents all the slogans it knows directly in a table, which is shown in Fig. 4.

It is important to note that the lack of a concept-based search feature in all mainstream search engines has, in many situations, discouraged people from expressing their queries in a more natural way. Instead, users are forced to formulate their queries as keywords. This makes it difficult for people who are new to keyword-based search to effectively acquire information from the web.

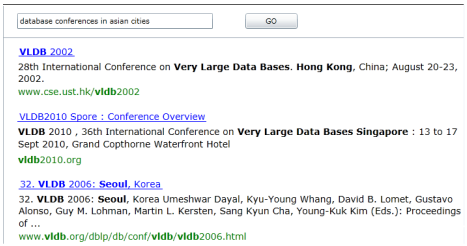


Fig. 3. Our top results for Q1

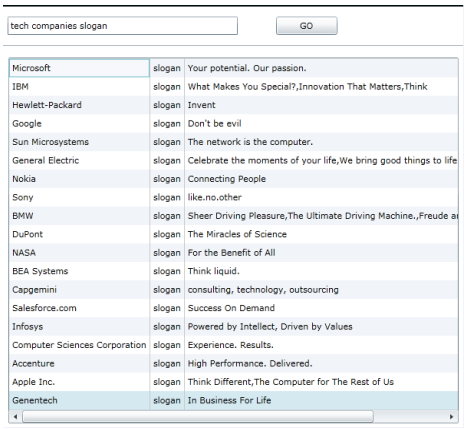


Fig. 4. Our top results for Q2

The proposed framework is not meant to replace the existing keyword based search, but complements keyword search as we can now handle some of "non-keyword" queries as well.

In the remainder of this paper, we will first introduce Probase the knowledge base in Section 2, and then present our framework in Section 3 and some of our experimental results in Section 4, before discussing some related work (Section 5) and concluding the paper (Section 6).

2 The Knowledge Base

In order to better understand queries, the search engine needs to have access to a knowledge base, which knows that, for example, *VLDB* is a database conference, *Hong Kong* is an Asian city, many companies have their *slogans*, and the slogan of Google, a well known tech company, is "Don't be evil." Furthermore, we also need certain meta information, for example, how entities are ranked by their representativeness within a same concept (e.g., What are the top 5 Internet companies?), or how plausible is a claim (e.g., Is Pluto a planet, or a dwarf planet?)

In this work, we take advantage of the Probase taxonomy [1] for rewriting queries. The backbone of Probase is constructed using linguistic patterns such as Hearst patterns [8]. Fig. 5 illustrates a snapshot of the Probase taxonomy which includes the concept "politicians", plus its super-concepts, sub-concepts, entities and similar concepts.

Probase is unique in two aspects. First, the Probase taxonomy is very *rich*. The core taxonomy alone (which is learned from 1.68 billion web pages and 2 years' worth of search log from a major search engine) contains 2.7 million concepts. With 2.7 million concepts obtained directly from Web documents, the knowledge base has much better chance to encompass as many concepts in the mind of humans beings as possible. As shown in Fig. 6, at least 80% of the search contains concepts or entities in Probase.

Second, the Probase taxonomy is *probabilistic*, which means every claim in Probase is associated with some probabilities that model the claim's plausibility, ambiguity, and

Fig. 5. A fragment of Probase taxonomy

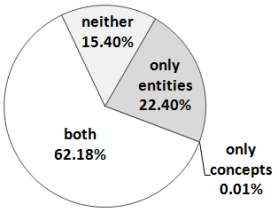


Fig. 6. Queries with concepts or entities

other characteristics. It enables Probase to rank the information it contains. For example, it can answer questions such as "What are the top 5 Internet companies?", or "How likely is Pluto a planet vs. a dwarf planet?"

3 Our Approach

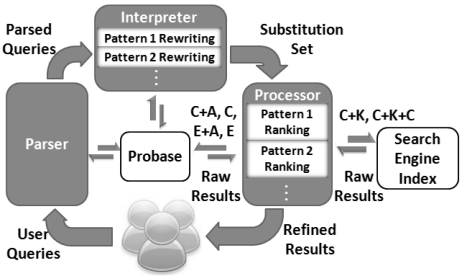


Fig. 7. The framework of concept-based search

Our concept-based search framework contains three main modules. the *parser*, the *interpreter* and the *processor*, illustrated in Figure 7. When a user issues a query, the *parser* uses Probase to decompose the query into possible term sequences, which consist of terms of 4 different types: concepts, entities, attributes and keywords. The *interpreter* identifies the intent or the semantics of the term sequence based on a set of query patterns. It then rewrites the parsed queries into a set of candidate queries by substituting abstract concepts with their specific entities. The *processor* ranks the candidate queries based on their likelihood, which is estimated by word association probabilities, and then it submits top queries either to Probase or to the search engine index to obtain a list of raw results. These results are ranked before being returned to the user. Next, we discuss these modules in more details.

3.1 Query Parsing

We regard a search query as a sequence of *terms*. We are interested in four kinds of terms: *entity* terms, *concept* terms, *attribute* terms, and *keyword* terms.Keywords are non-trivial types of words other than concepts and entities. Table 1 gives some examples of each type of terms which are highlighted.

Table 1. Query Terms and Examples

Term Type	Examples
Entity	**Citigroup** **ICDE** in **Hong Kong** **Hong Kong** area
Concept	**companies** **big financial companies** campaign donation **database conferences** in **asian cities**
Attribute	tech companies **slogan** Hong Kong **area** movies **director**
Keyword	Oracle **acquire** Sun big financial companies **campaign donation** **what's the date today**

The first three types are usually noun phrases, while the keyword terms are often verbs, adjectives or any combinations of other terms that are not recognized as one of the first three types.

Formally, we represent a raw query by an array of words, that is, $q[1,n] = (w_1, \cdots, w_n)$. We parse it into a sequence of terms, where each term is an entity, a concept, an attribute, or an attribute value in the Probase taxonomy, or simply a keyword otherwise. Specifically, we represent a parsed query as $p[1,m] = (t_1, \cdots, t_m)$, where each t_k is a consecutive list of words in the raw query, i.e., $t_k = q[i,j] = (w_i, w_{i+1}, \cdots, w_j)$.

Clearly, there may exist many possible interpretations of a query, or multiple different parses. For example, "*president george bush fires general batiste*" can be parsed as

[president] (george bush) fires [general] (batiste)

[president] george [bush fires] [general] (batiste)

(president) (george bush) fires [general] (batiste)

where () denotes an entity, [] a concept, <> an attribute. The reason of multiple parses is because both *george bush* and *bush fires* are valid Probase terms. Further more, *president* can either be a concept, which refers to all presidents, or a specific entity in "political leader" concept. So the parser needs to return all meaningful parses.

For an n-word query, there are at most $2^{(n-1)}$ possible parses which is expensive to compute. We first introduce a greedy algorithm to solve this problem. Then we improve this algorithm using a dynamic programming approach. The greedy algorithm contains three steps: (1) find all possible terms; (2) find all correlations among terms; (3) use a scoring function to find one meaningful parse in a greedy manner.

First, we find all terms in a query. For a sequence of n word, there are $n(n+1)/2$ possible subsequences. We check each of them to see if they are concepts, entities or

attributes. We give a term t a *score* according to its type and length:

$$s_{term}(t) = w_{term}(t) \cdot w_{len}(|t|) \tag{1}$$

where $|t|$ is the number of words in t, $w_{term}(t)$ is the weight function defined as:

$$w_{term}(t) = \begin{cases} w_e, & \text{if t is an entity} \\ w_c, & \text{if t is a concept} \\ w_a, & \text{if t is an attribute} \\ 0, & \text{otherwise} \end{cases}$$

and $w_{len}(x) = x^{\alpha}$, where w_e, w_c and w_a are constants, and $w_e > w_c$. We let $\alpha > 1$ to bias toward longer terms.

Next, we consider the correlations among terms. Currently we focus on three kinds of correlations: Entity-Attribute, Concept-Attribute, and Concept-Entity. We use R_1-R_2 to denote the correlation between one R_1 term and several R_2 terms. For example, "<population> of (china)" is an instance of Entity-Attribute correlation, and "[tech companies] <slogan> and <founder>" is an instance of Concept-Attribute correlation. Note that terms in a correlation do not have to be physically adjacent to each other, which means keywords can be mixed with correlated terms, e.g. "[presidents] and their <wives>".

Based on terms and term scores, we define *block* and *block score*. A block is either a correlation or a single term. The block score is defined as:

$$s_{block}(q[i,j]) = \max_{i',j'} \{ w_{block}(p[i',j']) \cdot \sum_{k=i'}^{j'} s_{term}(t_k) \} \tag{2}$$

where $p[i',\ j']$ is a term sequence parsed from $q[i,\ j]$, and

$$w_{block}(p[i',j']) = \begin{cases} w_{e-a}, & \text{if p[i$'$,j$'$] is an E-A correlation} \\ w_{c-a}, & \text{if p[i$'$,j$'$] is a C-A correlation} \\ w_{c-e}, & \text{if p[i$'$,j$'$] is a C-E correlation} \\ 1, & \text{otherwise} \end{cases}$$

where w_{e-a}, w_{c-a} and w_{c-e} are all greater than 1. The above formula rewards blocks with a term correlation, and if there is no correlation, the block score is equal to the sum of term scores. (Now we empirically decide w_e, w_c, w_a, w_{e-a}, w_{c-a} and w_{c-e}, which could be obtained in a more precise way, e.g. machine learning methods, in the future.)

Finally, after finding all possible terms and blocks, we greedily select the block with the highest score. Once a block is selected, all blocks it overlaps with are removed.

We show the three-step greedy algorithm for parsing query "*president george bush fires general batiste*" in Fig. 8. In step (1), we identify all terms in the query and score them according to Eq. 1. In step (2), we generate blocks based on these terms. Each term becomes a block and their block scores equal to their term scores. At the same time, we notice that (george bush) is an entity of concept [president], so we build a C-E correlation block. Same goes for (batiste) and [general]. In step (3), we perform greedy

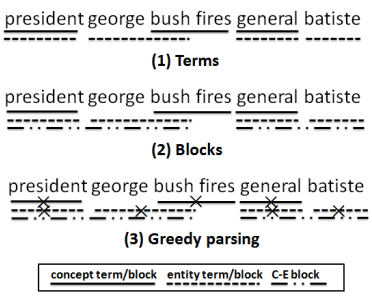


Fig. 8. Example of greedy parsing

selection on blocks. We identify "[president] (george bush)" as the best block among all, and remove other overlapping blocks such as "(president)" and "[bush fires]". Similarly we keep block "[general] (batiste)" and remove its overlapping blocks. Finally, the algorithm returns "[president] (george bush) fires [general] (batiste)" as the best parse.

However, the greedy algorithm does not guarantee an optimal parse, and it cannot return a list of top parses. As an improvement, we propose the following dynamic programming algorithm. We define a *preference score* $s_{pref}(n)$ to represent the quality of the best parse of a query of n words:

$$s_{pref}(n) = \begin{cases} \max_{i=0}^{n-1}\{s_{pref}(i) + s_{block}(q[i+1,n])\}, & \text{if n} > 0 \\ 0, & \text{if n} = 0 \end{cases}$$

By memoizing the sub-solutions of $s_{pref}(n)$, one can produce the parse with highest preference score. Moreover, when defining $s_{pref}(n)$ as a score set of top parses, one can also obtain the top k parses.

3.2 Query Interpretation

Table 2. Query Patterns and Examples

Patterns	Queries
E	(VLDB) (Citigroup)
C	[database conferences] [big financial companies]
E+A	(Apple) <slogan> [Hong Kong] <country> <area>
C+A	[tech companies] <slogan> [films] <language> <tagline>
C+K	[big financial companies] campaign donation [species] endangered
C+K+C	[database conferences] in [asian cities] [politicians] commit [crimes]

In this module, we classify the input parsed queries into different patterns. Our analysis on the search log during the period of Sep. of 2007 to Jun. of 2009 of a major search engine (see Fig. 6) shows that about 62% of the queries contain at least one concept term. More detailed analysis revealed that common web queries can be classified into a

number of different patterns. The following six basic patterns account for the majority of all the queries during that period: *Single Entity (E)*, *Single Concept (C)*, *Single Entity + Attributes (E+A)*, *Single Concept + Attributes (C+A)*, *Single Concept + Keywords (C+K)*, and *Concept + Keywords + Concept (C+K+C)*. Table 2 lists some example queries of each pattern.

Once the system determines the pattern of each parsed query, it starts interpreting them using the following strategies. The general approach is substituting the abstract concepts in a query with more specific search terms such as their associated entities which are more suitable for traditional keyword search.

For *E* and *E+A* queries, no further interpretation is necessary since this type of queries are already specific and can be searched directly in both Probase and the search engine index.

For a *C* or *C+A* query, it substitutes a list of top entities associated with that concept in Probase for the concept term to form a list of E or E+A queries.

For a *C+K* query, it replaces the concept with its associated entities to form a list of Entity + Keywords queries which require no further interpretation.

For a *C+K+C* query, it is considered as an extended form of C+K queries. We replace both concepts with their associated entities to form a list of Entity + Keywords + Entity queries. Note that the number of such queries can be very large but we will show in the next subsection how to reduce them to only relevant queries.

Finally, the output of the Interpreter module is a set of substituted queries of the following 4 patterns: E, E+A, E+K and E+K+E.

3.3 Query Processing

The Processor module takes as input a set of substituted candidate queries, submits some or all of these queries to Probase or a search index, and presents a final set of ranked results to the user.

For E and E+A pattern queries, the processor queries the Probase taxonomy for all the detailed information about this particular entity. This information is returned as a table which will eventually be presented to the user as an info-box (e.g. Fig. 4).

In the rest of this subsection, we will focus on E+K and E+K+E queries which are more interesting and challenging to process. One naive approach is to submit all these substituted queries to a traditional search engine index, combine the results, and present ranked results to the user. However, number of such queries can be prohibitively large because many concepts are associated with large number of the entities. For example, Probase contains hundreds of *politicians* and thousands of *crimes*. For query "*politicians commit crimes*", the system would generate millions of candidate substitutions even though most of these, such as "*obama commit burglary*", are not relevant at all.

Our proposed technique to address the above problem is to compute the *word association* values between an entity and a sequence of keywords, and multiply that with the typicality score of this entity in the concept it belongs to, to get a *relevance score* for a given query. The typicality score is the conditional probability $P(e|c)$, where e is an entity and c is the parent concept.

A word association value is used to measure the associativity of a set of words. To compute the word association value between two words, which we call *two-way asso-*

ciation, we measure the frequency of *co-occurrence* of the two words in a document among all documents or in a sentence among all sentences in the all documents.

In this paper, we approximate multi-way association by combining the values of two-way association. In general, given a set of m words: $W = \{w_1, w_2, ..., w_m\}$, it is impossible to compute the exact word association $wa(W)$ based only on two-way association $wa(\{w_i,\ w_j\})$, where w_i and w_j are any two distinct words in the W. However, because an co-occurrence of W together implies a co-occurrence of (w_i, w_j), for any $w_i, w_j \in W$, we have $wa(W) \leq \min\ wa(\{w_i, w_j\})$.

In other words, the minimum two-way association value provides an upper bound to the m-way association value. In this computation, we approximate the m-way association, by the minimum value of the two-way association between a word in the entity and the key word, or

$$wa(\{e, w_{key}\}) \approx \min_{w_i \in e}\ wa(\{w_i, w_{key}\}). \tag{3}$$

This technique is based on the notion of *pivot words*. A pivot word is the most informative and distinguishing word in a short phrase or term. Given that W contains the words from an entity term and a keyword, if two words w_i and w_j from W co-occur the minimum number of times, we argue that there is a high probability that one of them is the pivot word of the entity term and the other is the keyword. This is because the pivot word appears less frequently than the other more common words in the entity. It is even rarer for the pivot word to appear with an arbitrary keyword than with the other words in the entity. Therefore $wa(\{e_{pivot}, w_{key}\})$ is likely to be minimum, and can be used to simulate $wa(\{e, w_{key}\})$.

We can further extend (3) to compute the association of an entity term and a sequence of keywords:

$$wa^+(\{e, k_1, \ldots k_n\}) \approx \min_{i \in [1,n]}\ wa(\{e, k_i\}). \tag{4}$$

To get the word association values, we first obtain a list of most frequently used words on the web (excluding common stop words) as our word list and then compute the sentence-level pairwise co-occurrence of these words. We chose to count the co-occurrence within sentences rather than documents because this gives stronger evidence of association between two words.

We first group the E+K and E+K+E queries by their prefix E+K. As a result, each E+K query form a group by itself; and E+K+E queries with the same prefix form a group. Next, we compute a relevance score for each group G as

$$\max_{q \in G}(wa^+(q_{e1}, q_k, q_{e2}) \times rp(q))$$

where q_{e1}, q_k and q_{e2} are the first entity, keywords and the second entity of query q (q_{e2} may be null if q is an E+K query), and $rp(q)$ is the representativeness score of q which can be obtained from Probase or any other possible sources.

We then select the top n groups with the best relevance scores. Finally we collect all results from the top n groups and rank them according to some common search engine metric such as PageRank before returning to the user.

4 Evaluation

In this section, we evaluate the performance of online query processing and offline pre-computation (word association). To facilitate this evaluation, we create a set of benchmark queries that contain concepts, entities, and attributes, for example, "politicians commit crimes" (C+K+C), "large companies in chicago" (C+K), "president washington quotes" (E+A), etc. For a complete list of the queries and their results, please see [9].

4.1 Semantic Query Processing

The experiment is done on a workstation with a 2.53 GHz Intel Xeon E5540 processor and 32 GB of memory running 64-bit Microsoft Windows Servers 2003. In all search quality related experiments, we asked three human judges to rate each search result as being "relevant" or "not relevant", and we record the majority vote.

Quality of C+K & C+K+C queries In the first experiment, we compare the user ratings of the top 10 search results returned by our prototype system with that of Bing, Google, Hakia [10], Evri [11] and Sensebot [12], as illustrated in Fig. 9. The last three are popular semantic search engines. It shows that our prototype has a clear advantage at answering concept related queries. In addition, it also shows that results for C+K+C queries have lower relevance than C+K queries across the three systems, because C+K+C queries often involve more complicated semantic relations than C+K queries. The semantic engines have even fewer relevant results than Google and Bing do, because they index limited number of documents. For example, we found SenseBot's top 10 results are often extracted from 3∼5 documents.

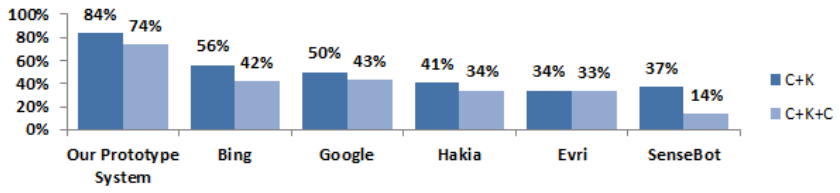
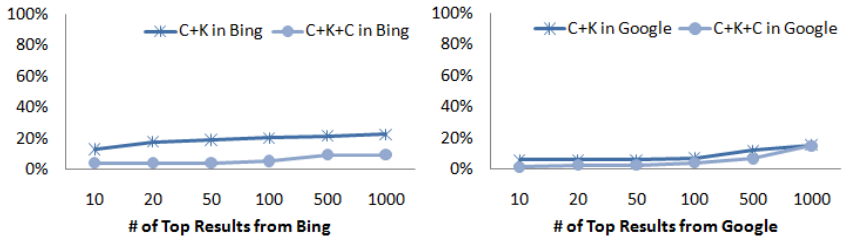


Fig. 9. % of relevant results for C+K & C+K+C queries

In the second experiment, we show the ability of our prototype system to pick up relevant results that keyword based search engines would miss. To see this, we focus on relevant results in the top 10 results returned by our prototype system. Among the top 10 results for the 10 C+K queries we use, 84 are judged relevant. The number for C+K+C queries is 74. We check whether these results will ever show up in Google or Bing. Fig. 10 shows that at least 77% of the relevant C+K results and 85% of the relevant C+K+C results in our top 10 could not be found in Bing or Google even after scanning the first 1000 results.



(a) Bing (b) Google

Fig. 10. Bing/Google miss the relevant results in our top 10

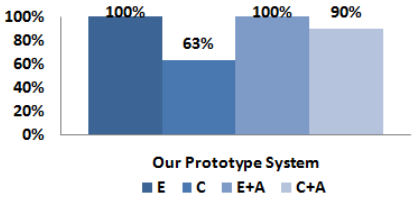


Fig. 11. % of relevant results

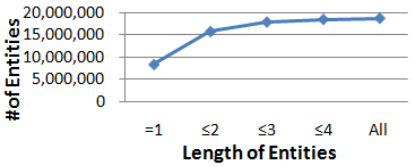


Fig. 12. Number of words in Probase entities

Quality of E, C, E+A & C+A queries For E, C, E+A, and C+A queries (see [9]), we return tables containing relevant entities instead of hyperlinks. We ask human judges to rate the relevancy of each returned table. Fig. 11 shows the percentages of relevant results. All results are relevant for E and E+A queries while some results are wrong for C and C+A queries, because Probase contains some erroneous concept-entity pairs. For example, it doesn't know if "George Washington" is a book or a person.

Time Performance We next evaluate the efficiency of our prototype system. Table. 3 shows the average running time of the benchmark queries in different categories. Note that the system is configured to return only the top 10 results for C+K, C+K+C, C and E patterns, and all results for the other two patterns.

Table 3. Execution Time (secs)

Pattern	Pr.	It.	Pc.	First Result
E	0.06	0.16	0.16	0.38
C	0.33	0.23	0.32	0.88
E + A	0.15	0.16	0.08	0.39
C + A	0.16	0.66	0.15	0.97
C + K	0.12	0.50	5.24	0.62
C + K + C	0.36	1.22	13.21	2.83

* Pr. = Parsing, It. = Interpretation, Pc. = Processing.

We currently use only one machine to communicate with Bing's public API to support our system. The API accepts only 2 queries per second. So we can see that C+K & C+K+C queries take much more time on processing than other queries. In this paper, we focus on improving user experience by presenting the first result as soon as it be-

comes available instead of showing all results at the end. We can also find that C+K+C queries take more time to process than C+K ones because two round trips to Bing are required for each query – one for filtering and one for final results.

4.2 Offline Precomputation

We precompute a word association matrix using a map-reduce framework and a distributed storage system on a cluster with 30 machines. Each machines has an 8-core 2.33 GHz Intel Xeon E5410 CPU and 16 GB of memory, and is running 64-bit Microsoft Windows Servers 2003. However, our program is single threaded and hence uses only one core at a time.

One assumption we made in this paper is that we can estimate the association between an entity term and a keyword using simple two-way word association. The following experiments verify this assumption.

We sampled 9,154,141 web pages (25GB on disk) from a web corpus snapshot of 366,185,148 pages for counting co-occurrences of words. We first examine the length of all the entities in Probase. Fig. 12 shows that 44.36% of the entities contains just one word, almost 84% have 2 or fewer words, and over 95% have 3 or fewer words. One-word entities, which account for almost half of all entities, are straight-forward to process using 2-way word association with another keyword. For the other half of the entities which have more than 1 word, the next experiment indicates that there indeed exists pivot words in many such entities and 2-way association results are very similar to the exact results using multi-way association.

We take the 10 benchmark C+K queries [9], and for each query generate a set of E+K candidate queries by substituting the concept with its associated entities. We next rank this candidate set using two different methods and compare the results. In the first method, which serves as a baseline, we rank the E+K queries by the actual association values of all the words in these queries. In other words, we compute the exact multi-way association of the entity term and the keywords In the second method, we rank the same set of E+K queries by computing 2-way association of the pivot word and the keywords in each query using (4) in Section 3.3. To find out the effectiveness of pivot words and 2-way association, we compute the similarity between the two rankings for each of the 10 benchmark queries using *normalized Kendall's Tau* [13]:

$$\bar{K}(\tau_1, \tau_2) = 1 - \frac{K(\tau_1, \tau_2)}{n(n-1)/2}$$

where $K(\tau_1, \tau_2)$ is Kendall's tau distance between two equi-length sequence τ_1 and τ_2.

Fig. 13 shows that the two rankings are similar enough across all the benchmark queries (see [9] for the the pivot words we discovered in these queries). Fig. 14 shows that the algorithm for multi-word association scales well with the size of the word list.

Accessing the document index directly and concurrently, or offering the user the rewritten queries to let he/she click on them, would definitely improve timing.

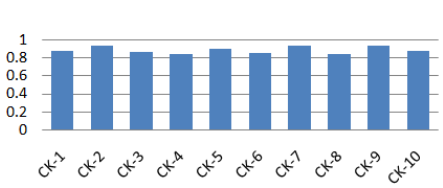


Fig. 13. Normalized similarity between ideal and simulated rankings

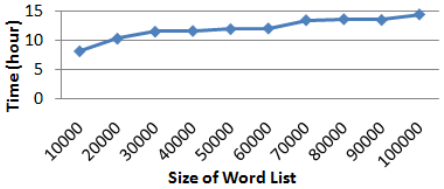


Fig. 14. Word association scaling

5 Related Work

There have been some attempts to support semantic web search using some form of a knowledge base. A well-known example is PowerSet [14] which maintains huge indexes of entities and their concepts. This approach, however, will not scale because updates on the entities can be extremely expensive. Another noteworthy general semantic search engine is Hakia [10] which leverages a commercial ontology and the QDex technology. The QDex technology is unique in that it indexes paragraphs by the embedded frequent queries (sequence of words). Hakia's search results on many of our benchmark queries were similar to keyword search results, which suggest the coverage of their ontology is too limited to help understanding the queries. Other semantic engines include WolframAlpha [15], Evri [11], SenseBot [12] and DeepDyve [16], etc. These engines exploit human curated or automatically extracted knowledge in specific domains such as science, news and research documents. Qiu and Cho proposed a personalized topic search [17] using topic-sensitive PageRank [18], which emphasizes on the analysis of user interests and the disambiguation of entities.

Understanding users' queries and helping users formulate "better" queries is important to document retrieval and web search. Work in this space can be collectively referred to as *query rewriting*. The relevant techniques include query parsing [19], query expansion [20–22], query reduction [23, 24], spelling correction [25] and query substitution [26, 27].

Computing a two-way association matrix for a set of N words generally takes $O(N^2)$ time, while computing a multi-way matrix is more costly. Li [28] proposed a method to estimate multi-way association. It uses an improved sketch method when sampling inverted index list of the words and then use maximum likelihood estimation to solve the problem. Consequently, one only needs to store the sampled inverted list of each word, instead of all the association values of different word combinations. However, scanning the inverted list remains to be costly if the number of documents is too large, especially at the scale of the entire web.

6 Conclusion

In this paper, we propose an idea to support concept-based web search. Such search queries contains abstract or vague concepts in them and they are not handled well by traditional keyword based search. We use Probase, an automatically constructed general-purpose taxonomy to help understand interpret the concepts in the queries. By

concretizing the concepts into their most likely entities, we can then transform the original queries into a number of entity-based queries which are more suitable for traditional search engines.

References

1. Wu, W., Li, H., Wang, H., Zhu, K.Q.: Probase: a probabilistic taxonomy for text understanding. In: SIGMOD. (2012)
2. Song, Y., Wang, H., Wang, Z., Li, H., Chen, W.: Short text conceptualization using a probabilistic knowledgebase. In: IJCAI. (2011)
3. Lee, T., Wang, Z., Wang, H., Hwang, S.: Web scale taxonomy cleansing. In: VLDB. (2011)
4. Zhang, Z., Zhu, K.Q., Wang, H.: A system for extracting top-k lists from the web. In: KDD. (2012)
5. Liu, X., Song, Y., Liu, S., Wang, H.: Automatic taxonomy construction from keywords. In: KDD. (2012)
6. Bollacker, K., Evans, C., Paritosh, P., Sturge, T., Taylor, J.: Freebase: a collaboratively created graph database for structuring human knowledge. In: SIGMOD. (2008)
7. Fellbaum, C., ed.: WordNet: an electronic lexical database. MIT Press (1998)
8. Hearst, M.A.: Automatic acquisition of hyponyms from large text corpora. In: COLING. (1992) 539–545
9. Wang, Y.: Supplementary material for topic search on the web. `http://research.microsoft.com/en-us/projects/probase/topicsearch.aspx` (2010)
10. Hakia: Hakia. `http://www.hakia.com` (2011)
11. Evri: Evri. `http://www.evri.com` (2011)
12. SenseBot: Sensebot. `http://www.sensebot.net` (2011)
13. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. SIAM Journal on Discrete Mathematics **17** (2003) 134–160
14. Powerset: Powerset. `http://www.powerset.com` (2011)
15. Alpha, W.: Wolfram alpha. `http://www.wolframalpha.com` (2011)
16. DeepDyve: Deepdyve. `http://www.deepdyve.com` (2011)
17. Qiu, F., Cho, J.: Automatic identification of user interest for personalized search. In: WWW. (2006) 727–736
18. Haveliwala, T.H.: Topic-sensitive pagerank: A context-sensitive ranking algorithm for web search. IEEE Trans. Knowl. Data Eng. **15**(4) (2003) 784–796
19. Guo, J., Xu, G., Cheng, X., Li, H.: Named entity recognition in query. In: SIGIR. (2009) 267–274
20. Lesk, M.E.: Word-word associations in document retrieval systems. American Documentation **20** (1969) 27–38
21. Salton, G., Buckley, C.: Improving retrieval performance by relevance feedback. JASIS **41**(4) (1990) 288–297
22. Fonseca, B.M., Golgher, P.B., Pôssas, B., Ribeiro-Neto, B.A., Ziviani, N.: Concept-based interactive query expansion. In: CIKM. (2005) 696–703
23. Jones, R., Fain, D.C.: Query word deletion prediction. In: SIGIR. (2003) 435–436
24. Kumaran, G., Carvalho, V.R.: Reducing long queries using query quality predictors. In: SIGIR. (2009) 564–571
25. Durham, I., Lamb, D.A., Saxe, J.B.: Spelling correction in user interfaces. Commun. ACM **26**(10) (1983) 764–773
26. Radlinski, F., Broder, A.Z., Ciccolo, P., Gabrilovich, E., Josifovski, V., Riedel, L.: Optimizing relevance and revenue in ad search: a query substitution approach. In: SIGIR. (2008) 403–410
27. Antonellis, I., Garcia-Molina, H., Chang, C.C.: Simrank++: Query rewriting through link analysis of the click graph. In: VLDB. (June 2008)
28. Li, P., Church, K.: A sketch algorithm for estimating two-way and multi-way associations. Computational Linguistics **33**(3) (2007) 305–354