Ranking-Constrained Keyword Sequence Extraction from Web Documents

Dingyi Chen[1] **Xue Li**[1] **Jing Liu**[1,2] **Xia Chen**[1]

[1] School of Information Technology and Electrical Engineering
The University of Queensland,
Brisbane, Qld 4072, Australia,
Email: xueli@itee.uq.edu.au

[2] School of Computer Science and Electronic Engineering
Xidian University,
Xi'an, 710071, China,
Email: neouma@163.com

Abstract

Given a large volume of Web documents, we consider problem of finding the shortest keyword sequences for each of the documents such that a keyword sequence can be rendered to a given search engine, then the corresponding Web document can be identified and is ranked at the first place within the results. We call this system as an Inverse Search Engine (ISE). Whenever a shortest keyword sequence is found for a given Web document, the corresponding document can be returned as the first document by the given search engine. The resulting keyword sequence is search-engine dependent. The ISE therefore can be used as a tool to manage Web content in terms of the extracted shortest keyword sequences. In this way, a traditional keyword extraction process is constrained by the document ranking method adopted by a search engine. The significance is that the whole Web-searchable documents on the World Wide Web can then be partitioned according to their keyword phrases. This paper discusses the design and implementation of the proposed ISE. Four evaluation measures are proposed and are used to show the effectiveness and efficiency of our approach. The experiment results set up a test benchmark for further researches.

1 Introduction

Search engine eg., Google, Yahoo, or Live Search, helps user find Web pages on a given subject using keywords. Knowing the right keywords, a user is able to locate relevant Web resources in a short time. However, as Kleinberg points out [1], search engines are not able to provide direct answers if user only knows what he/she wants, but does not know the right keywords to search. This would raise an interesting question: How can we extract a sequence of keywords from a Web document, so that once user knows this keyword sequence, he/she would be able to locate the document immediately? Thus the Web search problem could become a problem of approximating or mapping the keywords specified by user into the keyword sequences that can represent documents uniquely. This kind of ranking-constrained keyword extraction process is termed inverse search.

Inverse search as a common psychological process can be used for users to remember the Web pages that they have visited. For example, a tourist might wish to find the best place to watch insects that emit lights at night in Australia. Neither knowing proper terms of the insects nor the place where those insects inhabit, he/she tried "firefly Australia", but none of the search results is about the insects, because this kind of insects is usually referred as "glowworm" or more specifically, *Arachnocampa*. After many trials, he/she finally learnt that glowworm can be seen in Springbrook National Park near Gold Coast and would want to use a few words to represent these relevant pages. Indeed, the exact URLs of these pages can be recorded as bookmarks in a Web browser. However, long URL addresses are generally difficult to memorise or to speak out, therefore they are not suitable for oral communications. Instead of memorising the whole URL, the tourist can just refer to the keywords "glowworm" and "Springbrook" in case his/her friends are also interested. To an on-line advertiser, this Web site could be uniquely identified and advertised by buying these two words (or the word *Arachnocampa*) from the search engine. So when user searches for these words, the associated Web page will be returned at the first place in the search results. On the other hand, a Web page may be uniquely identified by extracting the features of its content and making an index on it.

An *Inverse Search Engine (ISE)* accepts a Web document as input and returns a shortest sequence of keywords that can be used to uniquely identify this document through a search engine. That is, after querying on the keyword sequence, the given Web document should be returned as the first search result by the search engine. In the rest of this paper, the term *target page* refers to the given input Web document, and the *shortest keyword sequence (shortest KS)* refers to a minimum (in keyword counting) ordered-list of terms that can make the target page ranked top in the search results. We use the terms of 'Web document' and 'Web page' interchangeably when the discussion is focused on their content.

For a keyword sequence being shortest, we define the following three characteristics:

- **Minimum number of words in sequence** — A minimum number of keywords that are extracted from a web document. The words are ordered and used as a query on the Web.

This project is supported by Australian ARC Discovery Project DP0558879.

- **Uniquely identifying the document on the Web** — When a search engine uses this keyword sequence, it will locate it uniquely and rank it as the first one in the search result.
- **Search-engine dependent** — Since different search engines may have different document ranking methods, for a given Web document, we are interested in finding the keyword sequence that can make the given document be ranked at the top of the search results by a given search engine. So, different search engines may have different keyword sequences used as queries to make the given document being ranked top.

It should be pointed out that although Page Rank [2] affects the ranking of Web pages of keyword-based search results, our ISE as a content-based approach is considered to be independent from the URLs that link the given Web page to others.

Typical search engines treat a search query as a sequence of keywords. This is because the order of search keywords may indicate either the relevant importance of keywords or the occurring order of keywords. For example, a query of "search engine" is different from "engine search", for the former is looking for an information system to obtain lists of references matched with specific criteria; while the latter might be finding a Web site that compares mechanical engines.

Keyword sequence is useful in many fields. In scientific publishing, authors are required to provide a list of words which point out the main topics of the paper for searching and indexing purposes. The important topic usually appears first in a keyword sequence.

Automatic keyword extraction from documents has been implemented in a few systems. For example, Microsoft Word can generate keywords from given documents. The extraction techniques suggested by [11, 10] can induce keyword-generating rules from the existing document/keyword pairs. Indeed, the keywords generated by those methods reflect the content of target documents to some extent. However, those extracted keywords cannot be used in a ranking process that can bring a Web-deployed document to the top position in a search. This, however, could be the most desired feature to Web surfers. On the other hand, most of those keyword-extraction methods are based on the supervised learning that prefers to work with a high-quality training set, which in many circumstances, is not available.

The main idea of this paper is to combine the keyword extraction with a document ranking process. In order to test the keyword sequence extracted by the system, a search engine will be used to feed with the extracted keyword sequence and to obtain a list of Web pages that the given Web page is included and ranked high. To this end, the implementation of ISE faces a twofold-problem. Firstly, many search engines limit their total number of daily accesses for an automatic client application. Yahoo allows 5,000 queries per day, while Google and Live Search (successor of MSN Search) only allow 1,000. So the number of queries made by ISE should be as small as possible. Secondly, search engines also limit the character-length of queries for security and performance reasons. For example, Google allows at most 2048 characters and MSN allows only 250[1]. Thus, the ISE-generated keyword sequence should contain as fewer words as possible.

To find out the shortest KS, there is always a naive way by using a brute-force method to exhaustively search for the solution. However, this will be computationally very expensive and infeasible. We propose a heuristic method to discover the shortest KS. Our method consists of three stages: embracing, expanding, and eliminating. Firstly the target page is embraced by a seed, i.e., an initial candidate KS that ranks target page in a work range such as top 100, or a larger number to be decided experimentally. Then the candidate KS is extended in order to improve the ranking of the page in that search engine. Finally, the terms in the candidate KS is reordered and surplus terms are eliminated. In this case, the result of ISE is search engine dependent, that is, the shortest KS that tops the target page in Google does not necessarily top the target page in Yahoo.

We have developed a framework of four measures for the evaluation of the effectiveness and efficiency of ISE. These measures are: (1) the *success rate* that is a count of the shortest KSs that can be obtained from different Web pages; (2) the *top-one rate* which tells the percentage of the obtained shortest KSs that actually top the target pages; (3) the *shortness* that reflects whether the ISE is KS-size efficient; and (4) the *impoliteness* that indicates whether the ISE sends too many queries to bother a search engine.

Keyword sequence extracted by ISE has three advantages. (1) Keyword sequence can be used as a digest of corresponding Web content. It can be a phrase that captures the topic of document. It can also be used as a query to get the target page from the Web. In this case, the best representative keywords become the best query words. (2) The training to the keyword extraction process is now performed by a search engine that provides feedback through its ranking process. So ISE does not need to collect a large volume of training data set for keyword extraction. (3) Keyword sequence as a shorthand of Web document can be used to index the Web content so to improve the search engine efficiency. It can be used by online advertisement or other Web-based applications where the key phrases are uniquely associated with certain services or functions.

This paper is organised as follows. Section 2 addresses the influential related work. Section 3 explains our proposed approach. Section 4 illustrates our experimental results. Section 5 provides the conclusions.

2 Related Work

There are two types of keyword-extraction approaches: (1) domain-dependent methods that are based on the supervised machine-learning models and require large training corpora, and (2) domain-independent methods that do not require training corpora.

A keyword is a meaningful term that has some importance in a document. It can be identified using the term frequency (TF) [8]. The intuition is that the important concepts are likely to be referred to more times than others. However, this might not be true in the situation that the terms are frequent in all documents that have the similar content. In this case, the documents cannot be differentiated from each other. As an alternative, we can rank the candidate keywords based on the inverse document frequency (IDF) [9].

Frank *et al.*, [3] introduced an automatic keyword extraction algorithm namely KEA, based on a domain-specific machine learning model. It employs lexical and information retrieval methods to identify candidate keyphrase from document. It calculates the feature values for each candidate and uses Naive Bayes machine to predict the overall probability of

[1]From the HTML source code of http://www.google.com and http://www.msn.com

keyphrases. The features used in the algorithm include $TF \times IDF$ and the positions of their first occurrences.

Kelleher *et al.*, [5] enhanced the KEA by introducing a feature of "Semantic Ratio (SR)" which makes KEA adapted to Web corpus. The SR of a phrase is calculated by dividing the number of occurrences of the phrase in the current document by the number of times it occurs in all documents directly hyperlinked to that document. The idea is based on the assumption that the semantics connection between Web documents is measurable by counting the neighbours of a Web document (in a way similar to the Page Rank) and that the subject matter (identified by the keyphrases) of the document is therefore in some way related to their content. They concluded that the hyperlink information can be used to improve the effectiveness of automatic keyphrase extraction by 50%.

Yih *et al.*, [10] demonstrated that by using extra features, such as the $TF \times IDF$ vectors, the meta data in Web pages, and the query-frequency information from Microsoft MSN query logs, their learning algorithm can substantially outperform KEA.

In the context of extracting keywords from Web documents, it is impossible to collect a large enough training data set for all possible types of Web content. So the domain-independent keyword extraction which does not require training corpus is considered in our approach.

Matsu and Ishizuka [6] proposed a keyword extraction algorithm from a single document without using any training corpus. The method firstly extracts a set of frequent terms from the given document. Then a set of co-occurrences between a term and a set of frequent terms is generated. If the probability distribution of a co-occurrence between term t and a set of frequent terms is biased to a particular subset of frequent terms, then term t is believed to be a keyword. In this case, the degree of bias of the co-occurrence between a term and a set of frequent terms is regarded as an indicator of the term importance.

Our work shares a similar idea from the Implicit Query System [4] and the Robust Hyperlinks [7]. The Implicit Query System can automatically generate query words or phrases from an email and send them to an Internet search engine in order to find documents that are relevant to the email. Their method can extract the special features from emails, such as the words used in subject line. The system also uses query logs from the Microsoft MSN Search to avoid picking up the words or phrases that would never be queried by real-life users. In this way, the system can dramatically reduce the total number of candidate queries. Their method uses a training data set with a logistic regression training process.

The Robust Hyperlinks is another keyword extraction method that extracts text signatures from Web pages. These signatures serve as search queries to locate Web pages once the hyperlinks fail. In [7], text signatures are acquired through the TF-IDF vectors. This approach can be very effective for Web directory maintenance and digital libraries, because the total number of documents is known in advance. However, it is difficult, if not impossible, to provide a sufficient large collection of Web pages to produce IDF based on the unlimited number of Web documents.

In our work, the keyword learning is not based on a large collection of documents, but on the feedback given by a search engine in its ranking process. For a given keyword sequence extracted from a document, the higher ranking of the document has, the better that the keyword sequence represents the document. The system can learn directly from the feedback supplied by search engine to find out the best query words or phrases for a given Web page without training on a corpus. The next section discusses the design of Inverse Search Engine (ISE).

3 Inverse Search Engine (ISE)

A query rendered to a search engine is in the form of a text string (keyword sequence) k. A sorted list of Web documents D will be returned as the search result of k. The relationship between search result list D and the search engine function SE can be expressed as:

$$D = [d_1, d_2, \ldots d_i, d_n] \leftarrow SE(k) \tag{1}$$

where the i-th result is referred as d_i and n is the maximum search range, which defines the maximum number of returned Web documents.

In contrast to SE , an inverse search engine (ISE) accepts a target page $\check{d}$ as input and returns the shortest keyword sequence $\check{k}$ which makes the target page $\check{d}$ be ranked at the top of the results of a search engine. In other words, the problem is: Given a target document $\check{d}$ and a search engine $SE(k)$, find a *target keyword sequence* $\check{k}$, which is the shortest keyword sequence that makes the target document $\check{d}$ be ranked at the first ($d_1 = \check{d}$) of the search results, as shown in Formula 2:

$$\begin{array}{l} \check{k} \leftarrow ISE(\check{d}), \quad \text{such that} \\ Rank(\check{k}, \check{d}) = 1 \end{array} \tag{2}$$

where

$$Rank(k, d) = \begin{cases} i & \text{if } d \in D,\ d_i = d \\ -1 & \text{if } d \notin D. \end{cases} \tag{3}$$

The ISE architecture is shown in Figure 1. As we can see that the input of ISE is a Web document $\check{d}$. The final output of ISE is the shortest keyword sequence $\check{k}$. During the process, an initial keyword sequence k is fed into a Search Engine then a list of documents D is returned from the Internet with $\check{d}$ is one of them and ranked high. This process is repeated until $\check{d}$ is ranked at the top. After a further process that eliminates any superfluous words, k becomes the final output $\check{k}$.

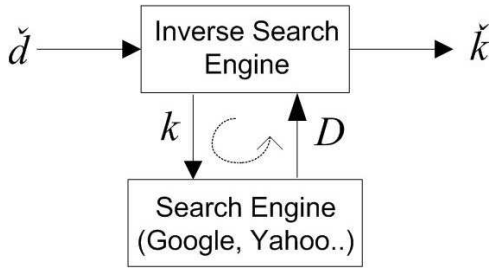

Figure 1: ISE Architecture

There are two naive methods that can exhaustively search for the resulting shortest KS. One is a bottom-up approach namely incremental brute-force. This approach tries all the possible word permutation from uni-gram to n-gram until the shortest keyword sequence is found. The other is a top-down approach namely shrinking brute-force. This approach starts from the keyword sequence that represents the whole content of the target page. Then, the system iteratively removes the "less informative" words from the candidate keyword sequence until the target page falls from its first rank in the search results. Clearly, exhaustive search for the shortest keyword sequence is

expensive. The both brute-force methods therefore, are not desirable due to their time complexity.

We consider a heuristic approach to determining an initial candidate KS, so that the search results will contain the target page. The system then makes the target page progressively ranked higher by refining the candidate KS. Our ISE strategy consists of three stages: *embracing*, *expanding* and *eliminating*. At embracing stage, ISE finds an initial candidate KS, whose results contain the target page; then at expanding stage, the target page is topped by adding the words that can highly differentiate the target page from others; and finally at eliminating stage, the candidate KS is shrunk to a minimal size, and yet it can still make the target page ranked at the first place in the search results.

3.1 Embracing Stage

The objective of embracing stage is to determine a *seed* KS that would qualify the target page entering into a ranking process. Two simple heuristic strategies are used for the initial seed generation: *from title* and *from single important words*.

Intuitively, seed contains words that describe the target page. Such important words can be found in the HTML keyword fields, URL anchor text, and *emphasising HTML tags* such as <title>, headings (<h1> or <h2>), bold/italic (<b>/<i>) tags, or the text in larger fonts. Seed might be among terms in emphasised text fragments because those terms imply either their importance or relevance. The pseudo code of embracing strategy "From_Single_Word" is listed in Algorithm 1. The input of the algorithm is the target page $\check{d}$; while the out of the algorithm is a seed s and a list of important terms T.

Algorithm 1 Embracing stage with strategy "From_Single_Word".

function EMBRACE$_{FromSingleWord}(\check{d})$
  $\check{d} \mapsto T$ ▷ Terms extracted from $\check{d}$ is stored in term list T.
  for $\forall t : t \in T$ **do**
    if t appears between emphasising tags **then**
      $t.w \leftarrow 10$ ▷ Emphasised terms weight: 10
    else
      $t.w \leftarrow 1$ ▷ Normal terms weight: 1
    end if
  end for
  Sort T from "heavy" to "light"
  $s \leftarrow \emptyset$
  for $\tau \leftarrow 1$ to ℓ **do** ▷ τ: length of seed.
    for $\theta \leftarrow 1$ to Θ **do**
      Set the candidate seed $\hat{s}$ by choosing τ terms $([t_\theta, t_{\theta+1}, \ldots, t_{\theta+\tau-1}])$ from T
      $D \leftarrow SE(\hat{s})$
      if $1 \geq Rank(\hat{s}, \check{d}) < Rank(s, \check{d})$ **then**
        $s \leftarrow \hat{s}$
      end if
    end for
    if $s \neq \emptyset$ **then**
      return (s,T) ▷ Seed found.
    end if
  end for
  return (s,T) ▷ $s = \emptyset$, Seed not found.
end function

To hedge against exhaustive search, three constant control parameters are applied to ISE algorithms. They are the maximum search range ϵ, which limits the number of maximum returned results; the maximum trial keyword sequence Θ, which limits the number of trials for different size; and the maximum keyword sequence length ℓ. If no seed can be found in the iteration, we consider this stage fail and will not proceed further.

The title of Web page is usually important because it is either a summary of Web page content, or the relative paths from the home Web page. It is possible to locate the target page by merely using its titles as the initial keyword sequence. Though the HTML specification does not mention the limitation of the title length, in practice, titles are seldom longer than one hundred characters because the title bar of Web browser windows usually does not have a sufficient space to display long titles. For the Web pages with no title, the first sentence within the HTML typesetting/emphasising tags can be treated as a title. If such feature still does not exist, then we use the first sentence of body text. The pseudo code of embracing strategy "From_Title" is listed in Algorithm 2. The input of the algorithm is the target page $\check{d}$ (as well as the title of the target page $d.t$). The output of the algorithm is a seed s and a list of important terms T.

Algorithm 2 Embracing stage with strategy "From_Title".

function EMBRACE$_{FromTitle}(\check{d})$
  $\check{d} \mapsto T$ ▷ Terms extracted from $\check{d}$ is stored in term list T.
  for $\forall t : t \in T$ **do**
    if t appears between emphasising tags **then**
      $t.w \leftarrow 10$ ▷ Emphasised terms weight: 10
    else
      $t.w \leftarrow 1$ ▷ Normal terms weight: 1
    end if
  end for
  Sort T from "heavy" to "light"
  $\hat{s} \leftarrow \check{d}.t$
  $s \leftarrow \emptyset$
  if $Rank(\hat{s}, \check{d}) >= 1$ **then**
    $s \leftarrow T$
    return (s,T) ▷ Seed found.
  end if
  for $\tau \leftarrow 1$ to $\ell-$ length of $\check{d}.t$ **do** ▷ τ: length of seed.
    for $\theta \leftarrow 1$ to Θ **do**
      Append τ terms $([t_\theta, t_{\theta+1}, \ldots, t_{\theta+\tau-1}])$ from T to candidate seed $\hat{s}$
      $D \leftarrow SE(\hat{s})$
      if $1 \geq Rank(\hat{s}, d) < Rank(s, d)$ **then**
        $s \leftarrow \hat{s}$
      end if
    end for
    if $s \neq \emptyset$ **then**
      return (s,T) ▷ Seed found.
    end if
  end for
  return (s,T) ▷ $s = \emptyset$, Seed not found.
end function

To choose a seed for it being more readable, popular, or user acceptable, the weight for each term in the word list can be multiplied by a word importance function $M(t)$ when an importance word list T is available, where t is a word. The higher return value of the function indicates the more likely the word is to be chosen for seeding. For example, 'glowworm' should have a higher weight than 'Arachnocampa', as 'glowworm' is a word easier to remember. A typical implementation for word importance is the word fre-

quencies, except for the words in stopword list which should return 0. School teachers may like to set the return values of coarse words to 0 to prevent students from viewing inappropriate Web pages. The important word list is re-sorted after the weight of words in the list are updated. However, there is no such a universal word importance function currently available, we treat each word equally in our current design and the tests of the effectiveness of word importance function is not included in our experiments.

3.2 Expanding Stage

The objective of expanding stage is to find a candidate KS that makes the target page be ranked at the top of the search results. The seed derived at embracing stage is used as the initial candidate KS for the expanding. New terms from the important term list T shall be added to candidate KS until the candidate KS tops the target page.

The current selection method of appending new terms is based on the inverse document frequencies (IDFs) of those terms appeared within the documents of set D. The terms with low document frequencies (DFs) are more likely to improve the rank of the target page. The main reasons are that (1) they narrow down the range of documents to search, (2) they provide the search engine with indexes of all terms, and (3) they return only the documents that contain the keywords in query.

Document frequency is calculated based on Formula 4:

$$DF(t) = |d : d \ni t, d \in D - \check{d}| \tag{4}$$

After DF of each term in the target page is calculated, the term with the lowest DF is appended to the candidate KS. If the rank of the target page improves after appending term t, the term appending is repeated until the candidate KS tops the target page. Otherwise the second lowest DF term is appended and tested, and so forth.

During the process of expanding, a phenomenon called *Search Engine Shading* might occur such that none of the terms in $T_{\check{d}}$ would improve the rank of the target page at the expanding stage. The Search Engine Shading refers that the target document $\check{d}$ is "shaded" by other documents whose ranks are ahead of $\check{d}$ and whose terms form the supersets of that of the target document. For instance, two documents, d_1: "Cats are better than dogs" and d_2: "Cats are not better than dogs". are accessible for a search engine. The term sets of d_1 and d_2 are: { 'Cats', 'are', 'better', 'than', 'dogs'} and { 'Cats', 'are', 'better', 'than', 'dogs', 'not' } . That means $T_{d_1} \subseteq T_{d_2}$. If d_2 is ranked ahead of d_1, there might be no way to rank d_1 ahead of d_2 in the process. So when encountering Search Engine Shading, we need to relax the constraints and to return the candidate KS in order to make the target page ranking higher.

The pseudo code of expanding stage is listed in Algorithm 3. The input parameters of the algorithm are the target page $\check{d}$, the term list T, and the seed s from embracing stage. The algorithm returns a candidate KS k.

Search Engine Shading can be detected if the candidate KS k cannot top the target document when the algorithm stops.

3.3 Eliminating Stage

The objective of eliminating stage is to delete superfluous terms in the candidate KS (denoted as k in the algorithm). A naive way to discover superfluous terms is to test every permutation of terms in k. However the time complexity of this method is $\sum_{i=0}^{|k|} P_i^{|k|}$, which is not desirable for a middle sized candidate KS. For instance, there could have 9,864,100 permutations for a 10-word candidate KS, and every permutation would issue a query to search engine.

In order to reduce the number of tests, we start elimination from the leading terms in KS until the elimination would make the target page fall from its top rank, or the terms in the candidate KS are all tested. The pseudo code of elimination process is listed in Algorithm 4. The input parameters of the algorithm are the target page $\check{d}$ and the candidate KS k from expanding stage. The algorithm returns the shortest KS $\check{k}$.

Algorithm 3 Expanding stage.

```
1: function EXPANDING(ď,T,s)
2:     k ← k̂ ← s
3:     for t : t ∈ T do k̂ ← k̂ + t
4:         if Rank(k̂, ď) = 1 then
5:             k ← k̂
6:             return k
7:         else if  then1 < Rank(k̂, ď) < Rank(s, ď)
8:             k ← k̂
9:             return k ←Expanding(ď,T,k̂)
10:        end if
11:    end for
12:    return k                    ▷ Search Engine Shading encountered.
13: end function
```

Algorithm 4 Eliminating stage.

```
1: function ELIMINATION(ď,k)
2:     ǩ ← k.
3:     Let T_k be the list of terms in k.
4:     for t : t ∈ T_k do k' ← Elimination(ď,ǩ)
5:         if 1 ≤ Rank(k', ď) ≤ Rank(ǩ, ď) and k' is shorter than ǩ then
6:             ǩ ← k'                  ▷ Shorter KS found.
7:         end if
8:     end for
9:     return ǩ                    ▷ the shortest KS found.
10: end function
```

There are still some cases that the found KS is not the shortest. Other than Search Engine Shading, a possible reason of this problem is that there is a limit of the query length for the search engine efficiency and its security control. The process of finding the shortest KS may also fail if the character-length of KS is longer than the length-limit of a query.

3.4 Limitations

The proposed approach would not work in following circumstances [7]:

Non-indexed Web page: Embracing stage will fail if the target page is not searchable by a search engine.

Dynamic Web page: Page content is subject to constant changes. These pages include service pages, dynamic HTML and search result pages from other search engines or from a database.

Non-textual resource: If the target page does not contain textual information, then it is impossible for embracing stage to extract keyword.

Modern mainstream search engines have many advanced search functions such as phrase search, in-link, regular express, and non-negative search. However, these innovative search methods are hardly applicable to digital libraries or on intranets. For a wider applicability of ISE the above advanced features are not considered in the design of ISE.

4 Experiments

The experiments are designed to evaluate the effectiveness and efficiency of our ISE implementation. The ISE effectiveness is considered as whether a shortest KS can be successfully obtained. And if it is able to top the given target document. While the ISE efficiency is considered as whether the number of queries is reasonably small in order to obtain a shortest KS. It should be pointed out that the actual runtime analysis is not suitable here for ISE efficiency evaluation, as the runtime of ISE can heavily rely on the network traffic, which may not be always stable.

The effectiveness and efficiency of embracing strategies "FromSingleWord" and "FromTitle" are examined and compared with different search engines. The data set preparation, the evaluation methodology, and the results of experiments are shown in the rest of this section.

4.1 Experimental Data

The ISE implementation is tested with two major search engines: Live Search and Yahoo. We have also performed the experiments with Google, however, after we finished the experiments, we found out that Google's terms of service [2] states that the automatic querying clients are not allowed without separate permission. Since our application for the permission has not yet been granted by Google, we cannot provide our experimental results on Google in this paper. Though the results are consistent to the other two search engines.

To verify our proposed approach, a shortest KS is required for every target page. But only a brute-force search can guarantee to found such shortest KSs. Here we provide *pseudo shortest keyword sequences* for each target page in test. A pseudo shortest KS is generated from randomly concatenated terms which were selected from the titles of category and sub-category of a very large and well-known project namely, Open Directory Project (ODP) [3]. In this way their titles and category/subcategory can form a unique identification of their corresponding Web documents. Therefore the criteria of the effectiveness are set objectively. All Web documents of the Open Directory Project can be potentially used as target pages and used as the initial search result regarding to the given pseudo shortest KSs. In other words, a pseudo shortest KS can guarantee that it will top the target page, even it is not always the actual one that could be found by a brute-force approach.

Since the evaluation sets from search engine vendors are not available yet, a pseudo shortest KS is a necessary devil. Indeed, the web pages indexed by pseudo shortest KSs tend to rank higher, but evaluation cannot proceed without knowing the existence of the shortest KS. Pseudo shortest KS also ensures the target pages be searchable. Establishing an experimental search engine seems plausible, however, a nearly enterprise-level search engine is required, otherwise the conclusions might be misleading because a shortest KS can be found in a few steps, not to mention an ad-hoc ISE function need to be developed for the experimental search engine.

For each search engine, 100 pseudo shortest KSs have been automatically generated in 10 different sizes (from one-word to 10-word). After excluding the non-HTML files (e.g., PDF or Microsoft Word files), 96 Web documents are used as the testing target pages for Yahoo and 99 for Live Search.

Other types of files are to be supported in future, as non-HTML files such as pdf, postscript, Microsoft Word documents, and Powerpoint slides, are now all searchable in modern search engines. Different parsers should be used to handle those formats.

4.2 Evaluation Methodology

The effectiveness of ISE is measured by the success rate and top-one rate. The success rate (SR) shows whether ISE can successfully get the shortest KSs. It is defined as:

$$SR = \frac{N_{\check{k}}}{N} \tag{5}$$

where $N_{\check{k}}$ is the number of the shortest KSs found, and N is the number of the target pages. The top-one rate (TR) shows whether a obtained shortest KS can top the corresponding target page. It is defined as:

$$TR = \frac{T_{\check{k}}}{N_{\check{k}}} \tag{6}$$

where $T_{\check{k}}$ is the number of obtained shortest KSs that top the target pages. The top-one rate is also inversely proportional to the occurrence of the Search Engine Shading problem. An ISE implementation is considered effective if both the SR and TR are in high percentage (i.e., close to 100 percent).

The efficiency of ISE is measured by the average shortness and average impoliteness. The shortness (SH) shows whether the obtained shortest KSs are shorter than the pseudo shortest KSs. It is defined as:

$$SH = \frac{|k|}{|\check{k}|} \tag{7}$$

where $|k|$ and $|\check{k}|$ are the numbers of terms in the pseudo shortest KS and in the shortest KS found. ISE is efficient if the computed shortness is greater than 1. The impoliteness (IP) shows whether ISE is polite to the search engine, that is, ISE is impolite when it sends a large number of queries to a search engine. IP is an important measure because it implies not only the inefficiency of ISE, but also the extra network traffic and search engine load. Moreover, some search engines may have daily query-quota for their client applications. The impoliteness is defined as:

$$IP = \log_{10} q \tag{8}$$

where q is number of queries sent to search engine to obtain a shortest KS. For a single automatic client like our ISE implementation, the daily query-quota stated by most search engines is 1000. Therefore, our ISE implementation is deemed to be acceptable if the impoliteness score is below 3 (i.e., the magnitude of the query-quota).

4.3 Results

4.3.1 Effectiveness Evaluation

The effectiveness of ISE with different embracing strategies is compared in this section. Table 1

[2] http://www.google.com/accounts/TOS

[3] see http://dmoz.org/. "The Open Directory is the most widely distributed data base of Web content classified by humans. The Open Directory powers the core directory services for the Web's largest and most popular search engines and portals, including Netscape Search, AOL Search, Google, Lycos, HotBot, DirectHit, and hundreds of others."

Table 1: Success Rate of Embracing Strategies

	FromSingleWord	**FromTitle**
Live Search	94.95 %	98.99 %
Yahoo	90.63 %	89.58 %

Table 2: Top-one Rate of Embracing Strategies.

	FromSingleWord	**FromTitle**
Live Search	100.00 %	100.00 %
Yahoo	100.00 %	100.00 %

presents the success rate (SR) of ISE with various embracing strategies in Live Search and Yahoo. High success rates suggest that both the seed and shortest KSs can be found in most cases.

Table 2 presents the top-one rate (TR) of ISE with various embracing strategies in Live Search and Yahoo. The results are perfect. These results show that whenever a shortest KS is found, it tops the given target page. In other words, if a seed can be obtained at embracing stage, then the shortest KS based on the seed will top the target page. Results in Table 2 also show that there is no occurrence of Search Engine Shading. However, we cannot safely conclude that Search Engine Shading will never happen, because we have only tested a limited number of target pages (i.e., 96 for Yahoo and 99 for Live Search).

4.3.2 Efficiency Evaluation

The efficiency of ISE of different embracing strategies are compared in this section. Figures 2 and 3 illustrate the comparisons of the sizes of candidate KSs (a seed is also a candidate KS) at every stage. In these figures, the solid and shading bars on the left represent the sizes of the candidate KSs of the "FromSingleWord" strategy, while bars on the right are for the "FromTitle" strategy at embracing stage. In order to compare the sizes among that of pseudo shortest KSs and candidate KSs, the sizes of pseudo shortest KSs are shown as the hollow wide bars on the left. A shorter KS size suggests a better KS-size efficiency. From these figures, it can be seen that the sizes of candidate KSs are seldom larger than that of the pseudo shortest KSs. This implies that most of the ISE-generated shortest KSs are not very long. We also observed that the sizes of the final shortest KSs are almost identical to their corresponding seeds in "FromSingleWord" strategy, while the sizes of the final shortest KSs are slightly shorter than that of their

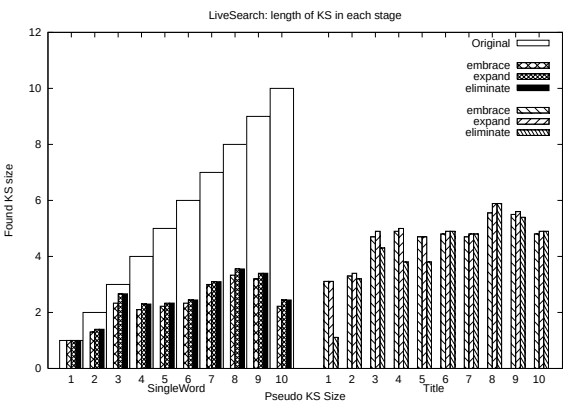

Figure 2: Live Search: Size of Candidate KS at Each Stage.

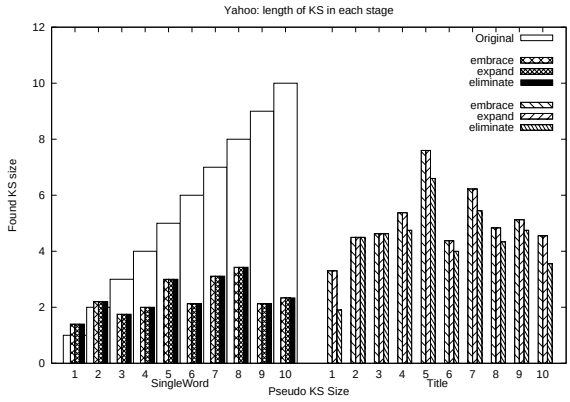

Figure 3: Yahoo: Size of Candidate KS at Each Stage.

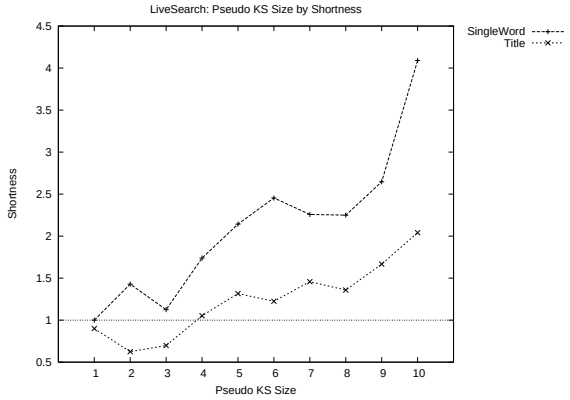

Figure 4: Live Search: Shortness Comparison between Embracing Strategies.

corresponding seeds in "FromTitle" strategy.

The shortness analysis is shown in Figures 4 and 5, and Table 3. A greater-than-one shortness score indicates that ISE is KS-size efficient. From these results, it can be seen that the both of our embracing strategies are KS-size efficient. Moreover, the embracing strategy "FromSingleWord" is better than "FromTitle" in terms of KS-size efficiency.

Figures 6 and 7 compare the numbers of queries required to obtain candidate KSs at every stage. In these figures, the solid and shading bars on the left show the numbers of queries required to obtain the candidate KSs for the embracing strategy "FromSingleWord", while the bars on the right are for the embracing strategy "FromTitle". A small number of

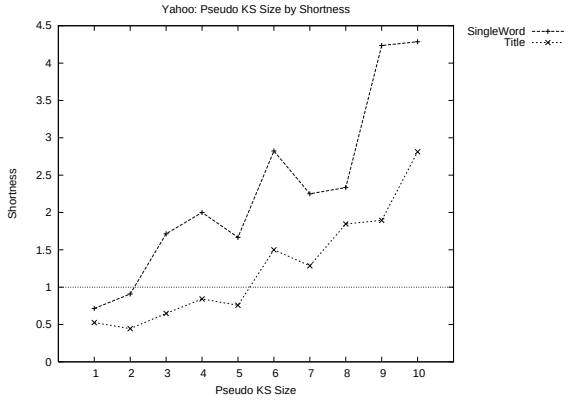

Figure 5: Yahoo: Shortness Comparison between Embracing Strategies.

Table 3: Contingency Table for Average Shortness of Search Engines.

	FromSingleWord	**FromTitle**
Live Search	2.39	1.88
Yahoo	2.72	1.72

Table 4: Average Impoliteness of Search Engines.

	FromSingleWord	**FromTitle**
Live Search	1.59	1.27
Yahoo	1.26	1.01

queries suggests a better query efficiency. In these figures, most queries are issued at embracing stage, followed by eliminating stage, and in most cases, the expanding stage is rarely performed. This implies that the seeds could have already topped the target pages in nearly all cases. Figures 6 and 7 also show that the most shortest KSs can be obtained within 1000 queries, which is below the daily query-quota of the most search engines.

The impoliteness analysis is shown in Figures 8 and 9, and Table 4. A small score of impoliteness indicates that ISE is query-efficient. From these results, the embracing strategy "FromTitle" is more query-efficient than "FromSingleWord" in most cases.

So far no sign of the Search Engine Shading occurred in our experiments. As there exists a shortest KS for each target Web page in our experiments, we have not encountered the circumstances that Search Engine Shading would occur.

To sum up, our ISE implementation is effective and efficient, as the success rates and top-one rates are demonstrated successfully high ($> 89\%$), the shortness scores are large (> 1), and the impoliteness scores are below the threshold (i.e., below the magnitude of the query-quota). Our experiments have effectively set up a benchmark for testing all other future ISE implementations and strategies. The experiment results also show that the embracing stage (seed generation) is the most important stage among the three, as the other two stages make relatively smaller contributions towards the final shortest KS. Thus, finding the better embracing strategies becomes a key to significantly improve the ISE effectiveness and efficiency.

5 Conclusions

Traditional keyword extraction algorithms do not consider the ranking of documents when keyword is extracted. This paper has defined a new type of system namely, Inverse Search Engine (ISE) to extract the shortest sequence of keywords from a Web page such that the keyword sequence can be used in a Web search and the Web page will be ranked as the first result by the given search engine. A number of challenges are addressed and the algorithms are proposed. The main contribution of this paper is the idea of the ranking-constrained keyword sequence extraction as well as the construction of ISE that can be used to discover the shortest keyword sequence for the unique identification of Web documents. The proposed three-stage algorithms are tested for their effectiveness and efficiency. An evaluation framework is proposed and the significant experiment results are demonstrated. Our experiment results can be regarded as a new benchmark for the further research in this direction.

One important issue still remains: 'shortest' is defined as the only one factor. In the real world, metrics such as the length of the keyword and popularity of the keyword would factor in. For example, one would prefer "insect lights Australia" over "Arachnocampa Springbrook". The shortest keyword sequence extracted from Web documents can be useful in many situations, such as Web indexing, content summary, Web advertisement. However, this usage should not prevent users from choosing words that they would prefer or know about, to search for the Web pages that they want.

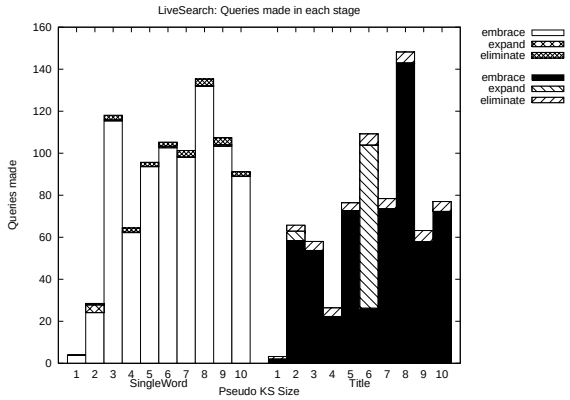

Figure 6: Live Search: Pseudo Shortest KS Size by Queries.

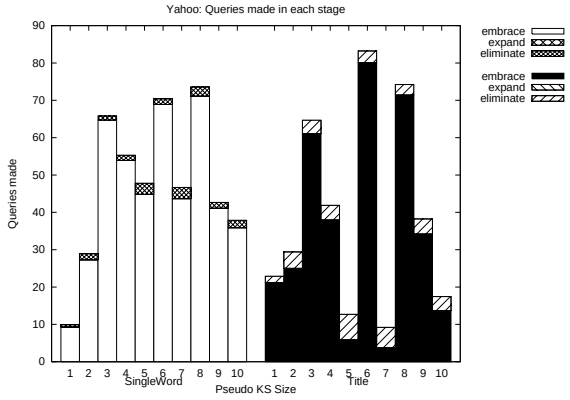

Figure 7: Yahoo: Pseudo Shortest KS Size by Queries.

Further work will focus on above issues and the popularity of keywords will be considered. The system performance regarding the scalability and supporting different file types will also be considered. In this case, the so-called *Search Engine Shading* would occur. The further experiments on the *Search Engine Shading* phenomenon will be given. As the initial embracing strategies can influence the effectiveness and efficiency significantly, a better embracing strategy might be designed to improve the ISE performance.

The idea of design and implementation of ISE has also raised another broad issue that if the whole Web-searchable documents on the World Wide Web can be partitioned into individuals according to their keyword phrases, the certain combination of keyword phrases could be 'owned' by the Web document authors.

References

[1] J. Battelle. John battelle's searchblog: Being jon kleinberg. Battellemedia.com (http://battellemedia.com/archives/000304.php), February 2004.

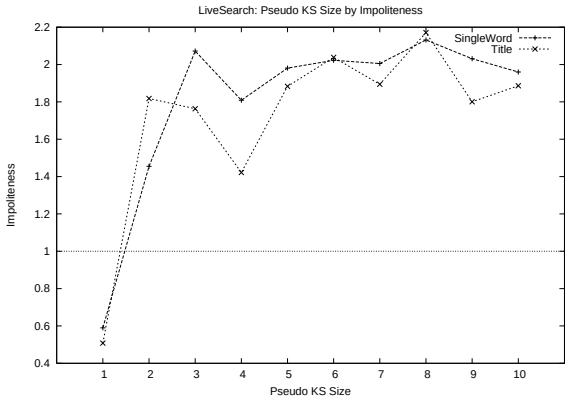

Figure 8: Live Search: Impoliteness Comparison between Embracing Strategies.

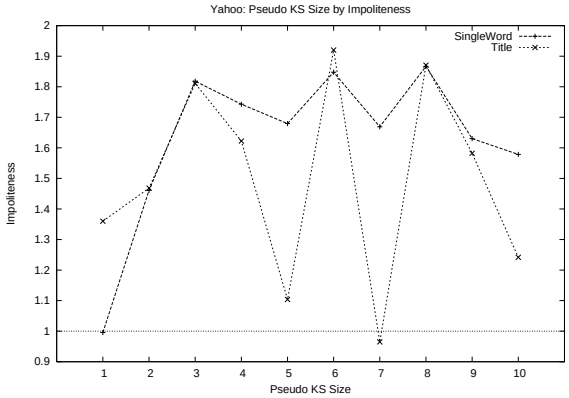

Figure 9: Yahoo: Impoliteness Comparison between Embracing Strategies.

[2] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

[3] E. Frank, G. W. Paynter, I. H. Witten, C. Gutwin, and C. G. Nevill-Manning. Domain-specific keyphrase extraction. In *IJCAI 1999: Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 668–673, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.

[4] J. Goodman and V. R. Carvalho. Implicit queries for email. In *CEAS 2005: 2nd Conference on Email and Anti-Spam*, 2005.

[5] D. Kelleher and S. Luz. Automatic hypertext keyphrase detection. In *IJCAI 2005: Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1608–1609, 2005.

[6] Y. Matsuo and M. Ishizuka. Keyword extraction from a single document using word co-occurrence statistical information. *International Journal on Artificial Intelligence Tools*, 13(1):157–169, 2004.

[7] T. Phelps and R. Wilensky. *Robust Hyperlinks Cost Just Five Words Each.* University of California, Berkeley, Computer Science Division, 2000.

[8] C. Salton, G.and Yang. On the specification of term values in automatic indexing. *Journal of Document*, 29(4):351–372, Dec. 1973.

[9] K. Sparck Jones. A statistical interpretation of term specificity and its application to retrieval. *Journal of Document*, 28(1):11–20, March 1972.

[10] W. tau Yih, J. Goodman, and V. R. Carvalho. Finding advertising keywords on web pages. In *WWW 2006: Proceedings of the 15th international conference on World Wide Web*, pages 213–222, New York, NY, USA, 2006. ACM Press.

[11] P. D. Turney. Learning algorithms for keyphrase extraction. *IR: Information Retrieval*, 2(4):303–336, 2000.