WS-Finder: A Framework for Similarity Search of Web Services

Jiangang Ma[1], Quan Z. Sheng[1], Kewen Liao[1],
Yanchun Zhang[2], and Anne H.H. Ngu[3]

[1] School of Computer Science, The University of Adelaide, Australia
{jiangang.ma,michael.sheng,kewen.liao}@adelaide.edu.au
[2] School of Engineering and Science, Victoria University, Australia
yanchun.zhang@vu.edu.au
[3] Department of Computer Science, Texas State University, USA
angu@txstate.edu

Abstract. Most existing Web service search engines employ keyword search over databases, which computes the distance between the query and the Web services over a fixed set of features. Such an approach often results in incompleteness of search results. The Earth Mover's Distance (EMD) has been successfully used in multimedia databases due to its ability to capture the differences between two distributions. However, calculating EMD is computationally intensive. In this paper, we present a novel framework called WS-Finder, which improves the existing keyword-based search techniques for Web services. In particular, we employ EMD for many-to-many partial matching between the contents of the query and the service attributes. We also develop a generalized minimization lower bound as a new EMD filter for partial matching. This new EMD filter is then combined to a k-NN algorithm for producing complete top-k search results. Furthermore, we theoretically and empirically show that WS-Finder is able to produce query answers effectively and efficiently.

1 Introduction

Similarity search of Web services (WSs) is a crucial task in many practical applications such as services discovery, service composition and Web applications like incorporating Google Maps to locate businesses [1–6]. In addition, because WSs have become one of the standard technologies for sharing data and programs on the Web, and also because new paradigm of the pay-per-use is adopted by the recent Cloud Computing, a number of enterprises are developing large-scale software applications by wrapping their data, business processes, and databases applications into WSs. These WSs can be further composed to new services that provide value-added functionalities. For example, according to a recent statistics[1](accessed on 07/05/2012), there are 28,606 WSs available on the Web, provided by 7,739 different providers. As a result, searching desired services is described to be akin to looking for a needle in a haystack [7].

[1] http://webservices.seekda.com

C. Liu et al. (Eds.): ICSOC 2012, LNCS 7636, pp. 313–327, 2012.

Currently, most existing WSs search engines employ simple keyword search on WSs descriptions in databases. This is based on the fact that WSs are syntactically described by Web Services Description Language (WSDL), which makes it feasible to index these WSDL files through using keywords strategy [5].

However, the simple keyword search strategy adopted by these search engines suffer from a limitation. The exact keyword search may miss many relevant search results. This is because exact keyword-based search approaches compute the distance between the query and the services over a fixed set of features. That is, most existing Web-based search engines [1, 4] compute a similarity score by matching each keyword in the query against the keyword at corresponding position in the WSs descriptions in database, not considering the impact of the similarity of the neighboring keywords. We refer to this type of search as one-to-one search. The limitation of this approach is that one-to-one search may not return all relevant results. For example, consider a typical keyword searching query `WHOLESALE` issued against the service database. In this case, WSs whose function descriptions contain keyword `SALE` may not be returned because the keyword `WHOLESALE` in the query are not present in the WSs description, which results in *incompleteness* of search results. However, we can see that there is some partial similarity between `WHOLESALE` and `SALE`.

One potential approach to capture the partial similarity is to integrate the neighboring keywords into the similarity computation by using the term partial match techniques such as the Earth Mover's Distance (EMD)[8]. EMD has been successfully used in multimedia databases due to its ability to capture the differences between two distributions. However, calculating EMD is computationally intensive. Another possible method is to describe services capabilities by using the Semantic Web. For example, some research [9] uses ontology to annotate the elements in WSs for finding common semantic concepts between the query and services advertisements. As we here focus on similarity search of WSs from the point of view of algorithms, Semantic Web is beyond the scope of this paper and will not be discussed further.

To address the challenges involved in similarity search for WSs, in this paper, we have designed and implemented a novel framework called WS-Finder, which improves the existing keyword-based search techniques for WSs. In particular, we employ EMD for many-to-many partial matching between the contents of the query and the service attributes. To facilitate search of WSs in an efficient manner, we design a new WS model. We also develop a generalized minimization lower bound as a new EMD filter for partial matching in order to overcome the complexity of computing EMD. This new EMD filter is then combined to a k-NN algorithm for producing complete top-k search results. Furthermore, we theoretically and empirically show that WS-Finder is able to produce query answers effectively and efficiently. Our key contributions are as the following:

– We propose a novel WS model that incorporates WS architecture information into *records* stored in databases. Such model could not only be searched in WS entries but also mined by the optimization algorithm proposed in the paper.

- We develop a *generalized independent minimization lower bound* (LB_{GIM}) extended from the LB_{IM} lower bound in [10] as a new EMD filter for partial matching. The filter is then incorporated into a *k-NN algorithm* for producing top-k results. To the best of our knowledge, LB_{GIM} is the first effective and easy-to-compute lower bound for the *general* EMD. In particular, it is flexible and suitable for keyword-based content searches where words are always compared with different lengths. Whereas almost all other lower bound filters [8, 10, 11] are only developed for *special* EMD with equal total weights. For this reason, LB_{GIM} also has great potential to be used in other application domains. We compute LB_{GIM} by proposing a *unified greedy algorithm*.
- We conduct an extensive experimental evaluation over a set of real datasets of WSs. Our experimental results show that WS-Finder produces query answers with high recall and precision and low response time.

The remainder of this paper is organized as follows. In Section 2, we describe a Web service model and the WS-Finder architecture and in Section 3, we discuss an EMD-based optimization algorithm. In Section 4, we present experimental results to show the effectiveness and efficiency of our approach. Section 5 is dedicated to related work, and finally, Section 6 concludes the paper and discusses some future research directions.

2 Web Service Model and WS-Finder Architecture

In this section, we briefly describe a model to represent Web services and the architecture of the WS-Finder system.

2.1 Web Service Model

As one of key factors resulting in an effective similarity search of WSs is how WSs are stored [5], we first introduce a new model to represent WSs. The model incorporates WSs' architecture information such as functional and non-functional requirements (FRs and NFRs). The FR attributes of a WS are name, category, function and description while NFR attributes include QoS such as response time (RT), availability (AVA), reliability (REL) and etc. Specifically, we model WSs as *records* stored in a database, where each WS usually has a record with some attributes and values to describe the basic structured information of the entry. Such records could not only be searched in WS entries but also mined by algorithms. Thus, all these records form a WS database **DB**.

Definition 1. *A Web service record* $r \in \mathbf{DB}$ *is represented by a tuple* $r = \langle n^r, c^r, f^r, d^r, \mathbf{QoS}^r \rangle$, *where* n^r, c^r, f^r *and* d^r *contain the content of the attribute name, category, function , description, and* $\mathbf{QoS}^r = \{QoS_1^r, QoS_2^r, \ldots\}$ *contains an ordered set of numerical NFR contents respectively.* □

The examples of WS records are shown in Table 1.

Table 1. Web service records

FR Attributes					NFR Attributes		
ID	Name	Category	Function	Description	RT (s)	AVA (%)	REL (%)
1	B.S. Stillwell Ford	New cars	getCarPrice	Finance facilities at competitive rates	1.2	86	88
2	City Holden	Used cars	getCarColor	Minimise the hassle of a second hand vehicle	0.8	92	80

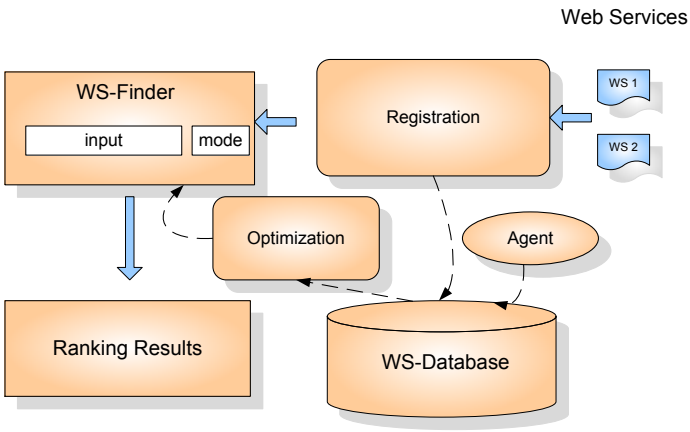

Fig. 1. WS-Finder Architecture

2.2 WS-Finder Architecture

The proposed architecture consists of three main components (see Figure 1). The registration component deals with the registration of WSs. Service providers can register their WSs through this component. The registered WSs are pre-processed to remove common terms and stopwords by using word stemming and the stopwords removing approaches. Then the processed WSs are stored in service database in terms of records, with the techniques and approaches introduced in this paper for enabling similarity search. The registration process can be performed offline.

The optimization component handles the query optimization for similarity search. It enables effective and efficient search by using EMD techniques. In this paper, we mainly concentrate on this optimization component.

3 EMD-Based Optimization Algorithm

In this section, we discuss an EMD-based optimization algorithm to support similarity search. Formally, given a database **DB** containing WS records and a query q, similarity search returns all records $r \in \mathbf{DB}$, such that $dis(q, r) \leq \varepsilon$, where $dis(q, r)$ is a distance function, and ε the specified similarity threshold. We use EMD as a measure for the distance $dis_{EMD}(q, r)$ between a keyword and an attribute word sequences $\mathbf{kw}^q$ and $\mathbf{aw}^{r,M^q}$ from a query q and a record r, which describes their similarity. The problem we focus on here is the design of

efficient computing EMD methods that will minimize the total number of actual EMD computations.

3.1 Earth Mover's Distance

The Earth Mover's Distance was first introduced in computer vision for improving distance measure between two distributions. As it can effectively capture the differences between two distributions and allow for partial matching, EMD is successfully exploited for various applications like phishing detection [12], document retrieval [13], and databases [10, 14].

Traditional distance measure in computer vision is based on computing the similarity between histograms of two objects. Given two objects histograms x=$\{x_i\}$ and $y = \{y_i\}$, the L_p distance, defined as $L_p(x, y) = (\sum_{i=0}^{n} |x_i - y_i|^p)^{\frac{1}{p}}$, is used to compute the distance of the histograms. Due to its rigid binning distance measure, a small shift of bins in the histograms often results in a large distance in L_P. To overcome the limitation of bin-by-bin approach, EMD incorporates information cross bins into the definition of the distances between two objects. This can be achieved with the help of the solution of solving transporting problem, where the distribution of one object is regarded as a mass of earth with quality W and the other as a collection holes with a given limited capacity U. Mathematically, the EMD involves in the solution of transportation problem in linear programming.

3.2 Defining EMD

EMD describes the normalized minimum amount of work required to transform one distribution to the other. Computing the exact EMD requires solving the famous *transportation problem* [8] in operations research.

In our context, the *subtask* is to find EMD between a keyword and an attribute word sequences $\mathbf{kw}^q$ and $\mathbf{aw}^{r,M^q}$ from a query q and a record r, which describes their similarity. For simplicity, let any subtask to involve two word sequences **kw** and **aw** instead of $\mathbf{kw}^q$ and $\mathbf{aw}^{r,M^q}$, where $|\mathbf{kw}| = n_1$ and $|\mathbf{aw}| = n_2$. Finding the minimum work of the subtask to transform **kw** to **aw** through the flow $\mathbf{f} = \{\forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij}\}$ is equivalent to computing the optimal solution to the following linear program (LP) with variable f_{ij}:

$$
\begin{array}{ll}
\text{minimize} : \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} d_{ij} & \\
\text{subject to} : & \\
\forall 1 \le i \le n_1 : \sum_{j=1}^{n_2} f_{ij} \le w_{kw_i} & (1.1) \\
\forall 1 \le j \le n_2 : \sum_{i=1}^{n_1} f_{ij} \le w_{aw_j} & (1.2) \\
\forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij} \ge 0 & (1.3) \\
\sum_{i=1}^{n_1} \sum_{j=1}^{n_2} f_{ij} = \min \left(\sum_{i=1}^{n_1} w_{kw_i}, \sum_{j=1}^{n_2} w_{aw_j} \right) & (1.4)
\end{array}
\tag{1}
$$

where $\mathbf{d} = \{\forall 1 \le i \le n_1, 1 \le j \le n_2 : d_{ij}\}$ is the ground distance matrix between **kw** and **aw**; $\mathbf{w_{kw}}$ and $\mathbf{w_{aw}}$ are weight vectors of keywords and attribute words respectively. These distances and weights are necessary *input values* to LP

(1) that will be defined later. Furthermore, constraint (1.1) restricts the total outgoing flow from any keyword not to exceed the corresponding weight; (1.2) limits the total incoming flow to any attribute word to be no larger than the weight; (1.3) ensures the positiveness of all flows; (1.4) defines the amount of total flow which is equal to the minimum of keyword and attribute word sequences' total weights. Note that unlike in [10, 14, 15], the flow system we consider here is more general, which means the total weights of word sequences may not be equal. Therefore, the system can be modeled as a directed *complete bipartite graph* with n_1 keyword nodes and n_2 attribute word nodes as two parts. For simplicity, let the total number of nodes $n = n_1 + n_2$ and total number of edges (representing flows) $m = n_1 \times n_2$.

Now if we assume the optimal flow $\mathbf{f}^*$ is found for the above LP, the EMD is then the corresponding work *normalized* by the amount of total flow:

$$EMD\left(\mathbf{kw},\, \mathbf{aw}\right) = \frac{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{ij}^* d_{ij}}{\sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{ij}^*}$$

which has the capability to avoid favoring shorter queries in our partial words matching context.

3.3 Web Service Query Processing

WS-Finder processes queries in a multi-step manner. In particular, there are two processing paths depending on the query type: *basic* or *advanced*. On both paths, content filtering is the main technique for efficient query computation.

EMD-Based Query Processing without Filter. For each query q and its k required answers, WS-Finder is able to operate the search process in database with a *boolean function* $\phi^{q,k}$: $r \to \{0,\, 1\}$, where $\phi^{q,k}\left(r\right) = 1$ if $EMD\left(\mathbf{kw}^q,\, \mathbf{aw}^{r,M^q}\right) \leq \mathbf{EMDs}^{q,\mathbf{DB}}\left(k\right)$[2]. EMD function computes EMD between keyword and attribute word sequences, and the *ascendingly sorted* set of all EMD values is defined as : $\mathbf{EMDs}^{q,\mathbf{DB}} = \left\{\forall r \in \mathbf{DB} :\, EMD\left(\mathbf{kw}^q,\, \mathbf{aw}^{r,M^q}\right)\right\}^{+}$. This process can then be translated into the following SQL query:

$$\begin{aligned} &SELECT\ *\ FROM\ \mathbf{DB} \\ &WHERE\ \phi^{q,k}\left(r\right) \\ &ORDER\ BY\ \bar{EMD} \end{aligned}$$

where $\phi^{q,k}\left(r\right)$ is a boolean predicate in SQL and $\bar{EMD}$ denotes the extra created attribute name storing contents of $\mathbf{EMDs}^{q,\mathbf{DB}}$.

EMD-Based Query Processing with Filter. Because generating the set $\mathbf{EMDs}^{q,\mathbf{DB}}$ may take too long for a large number of exact EMD computations, we use LB_{GIM} filter for basic query uses $\hat{EMD}$ function to approximate EMD, and computes the

[2] The k-th element in the set $\mathbf{EMDs}^{q,\mathbf{DB}}$.

set : $\hat{\mathbf{EMDs}}^{q,\mathbf{DB}} = \left\{\forall r \in \mathbf{DB} : \hat{EMD}\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right)\right\}^+$ in a much shorter time. This set together with range and top-k queries is able to generate a filtered set $\mathbf{DB}' \subseteq \mathbf{DB}$ for processing the final top-k query. Therefore, the boolean function which describes the search process is revised to be : $\phi^{q,k}$: $r \in \mathbf{DB}' \rightarrow \{0, 1\}$, where $\phi^{q,k}(r) = 1$ if $EMD\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right) \leq \mathbf{EMDs}^{q,\mathbf{DB}'}(k)$.

3.4 Defining Input Values

Before computing the EMD, we need to decide the input values $\mathbf{w_{kw}}$, $\mathbf{w_{aw}}$ and $\mathbf{d}$. Intuitively, the weight of a word can be calculated by its length, so $\mathbf{w_{kw}}$ and $\mathbf{w_{aw}}$ can be easily computed. Note that $\mathbf{w_{aw}}$ is computed offline while $\mathbf{w_{kw}}$ is online. However, the ground distance is not so obvious because it needs to reflect the similarity between a pair of words with possibly different lengths. Assume the words to compare are kw_i and aw_j with lengths w_{kw_i} and w_{aw_j} respectively, our first adopted distance metric SED compares them character-wise[3] in $O\left(\min\left(w_{kw_i}, w_{aw_j}\right)\right)$ time and gets the number of different characters $diff$. The value of SED is therefore:

$$SED(kw_i, aw_j) = diff + \left|w_{kw_i} - w_{aw_j}\right|$$

For example, SED("CAR", "CITY") $= 2+1 = 3$. The advantage of this metric is the linear time complexity and no required extra space. However it is not robust enough to capture the distance between words having similar structures. For instance, SED("SALE", "WHOLESALE") $= 9$, but these two words have the similar suffixes. Our second chosen metric LD [16] instead represents the *edit distance* between two words. LD is a classical distance metric that has been used in many important applications such as spell-checkers, Unix diff command and DNA sequence alignment. The following definition explains LD in our context.

Definition 2. *The edit distance (LD) between a keyword and an attribute word is the least total number of character insertions, deletions and substitutions required to transform one to the other.* □

LD can be computed in $O\left(w_{kw_i} w_{aw_j}\right)$ time using dynamic programming (dp) to solve an equivalent sequence alignment problem. Also, there is an efficient way in maintaining the dp table that only requires $O\left(\min\left(w_{kw_i}, w_{aw_j}\right)\right)$ space. For the details, refer to the Hirschberg's algorithm [17]. Adopting this metric, LD("SALE", "WHOLESALE") $= 5$ since a minimum of five insertions are required. Despite that LD provides more robustness than SED in partial matching between words, its computation takes quadratic time and requires linear extra space which is more expensive. In Section 4, we also empirically compare the impacts of SED and LD on the performance of WS-Finder.

[3] Compare capitalized characters at the same index.

3.5 The EMD Filter: LB_{GIM}

We now discuss the generation of the EMD filter LB_{GIM} for word sequences partial matching. We will prove that LB_{GIM} lower bounds the general EMD and is suitable for EMD computation with unequal total weights case.

There are many ways in computing exact EMD with *arbitrary* distance metric. The streamlined approach we adopted is the transportation-simplex method [18] due to its supercubic (between $\Omega\left(n^3\right)$ and $O\left(n^4\right)$) empirical performance [8, 19]. Also, this algorithm exploits the special structure of the transportation problem and hence runs fast in practice.

Even though the algorithms for calculating the exact EMD are deemed to be efficient, as the amount of WS contents continue to increase, the framework's response time for top-k queries may be increased quickly. Hence we use the filtering idea similar to [8, 10, 11] to minimize the total number of actual EMD computations. However, their lower bound filters are only for EMD with equal weights ($\sum_{i=1}^{n_1} w_{kw_i} = \sum_{j=1}^{n_2} w_{aw_j}$ in our case), which is clearly not suitable for the comparison of word sequences that requires solving the general EMD (including the unequal total weights case). For this reason, we develop the LB_{GIM} filter generalized from LB_{IM} in [10] for word sequences partial matching. We will also see in the next subsection how this filter is incorporated into the k-NN algorithm for producing the final results.

Our LB_{GIM} is a *conditional lower bound* depending on three cases of LP (1): 1)$\sum_{i=1}^{n_1} w_{kw_i} < \sum_{j=1}^{n_2} w_{aw_j}$; 2)$\sum_{i=1}^{n_1} w_{kw_i} > \sum_{j=1}^{n_2} w_{aw_j}$; 3) $\sum_{i=1}^{n_1} w_{kw_i} = \sum_{j=1}^{n_2} w_{aw_j}$. In particular, case 3) is the same as LB_{IM}. For cases 1) and 2), we develop new lower bound LPs (3) and (2) respectively in the following and LB_{GIM} can be either of their normalized optimal solutions.

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{ij}d_{ij} \\
\text{subject to} & \forall 1 \le i \le n_1 : \sum_{j=1}^{n_2} f_{ij} = w_{kw_i} \quad (2.1) \\
& \forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij} \le w_{aw_j} \quad (2.2) \\
& \forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij} \ge 0 \quad (2.3)
\end{array} \tag{2}
$$

$$
\begin{array}{ll}
\text{minimize} & \sum_{i=1}^{n_1}\sum_{j=1}^{n_2} f_{ij}d_{ij} \\
\text{subject to} & \forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij} \le w_{kw_i} \quad (3.1) \\
& \forall 1 \le j \le n_2 : \sum_{i=1}^{n_1} f_{ij} = w_{aw_j} \quad (3.2) \\
& \forall 1 \le i \le n_1, 1 \le j \le n_2 : f_{ij} \ge 0 \quad (3.3)
\end{array} \tag{3}
$$

Theorem 1. *LB_{GIM} lower bounds the general EMD.*

Proof. For case 3), the theorem holds as the proof in [10] states that LB_{IM} lower bounds the EMD with equal total weights. For case 1), the constraint (1.4) in LP (1) immediately becomes constraint (2.1) that also subsumes constraint (1.1). Also, constraint (2.2) provides a more relaxed search space of f_{ij} than (1.2), since satisfying (1.2) $\Rightarrow$ satisfying (2.2) but not vice versa. So LP (2) will attain the same or a smaller optimal solution than LP (1). After being normalized by the amount of total flow, we obtain LB_{GIM} as a lower bound. Finally, because case 2) with LP (3) is similar to the situation of case 1) we analyzed, we have LB_{GIM} lower bounds the general EMD in all cases.

Although LB_{GIM} seems to involve 3 cases, we can still provide a fast *unified greedy algorithm* to resolve this. Moreover, the algorithm enables us to avoid explicitly solving these LPs which take much longer time. The main idea for developing the greedy algorithm is to look at LB_{GIM} from the point of view of algorithms rather than restricting ourselves to its mathematical view (the LP formulations). The algorithmic view of the problem is explained by Figure 2. In this problem, we are asked to find the minimum cost flow subject to the constraints in LPs. If case 1) happens (the upper figure), we can consider each of keyword *in turn* due to the relaxed constraint (2.2). For each keyword, we first ascendingly sort all its edges by distances. Then the *greedy strategy* is always trying to assign a *larger flow* value along the edge with the *shorter distance* until constraint (2.1) is fulfilled. Afterwards the unassigned edges associated with the keyword will have zero flow value. LB_{GIM} can then be easily calculated from assigned flows and edge distances. If case 2) happens (the lower figure), without loss of generality, we can reverse the direction of flow, consider each attribute word in turn instead and do the same thing as case 1). Finally, if case 3) happens, we consider both case 1) and case 2) and take the larger LB_{GIM} value which is closer to the exact EMD value.

For example, as shown in Figure 3, **kw**={ *"HOLDEN", "CAR", "SERVICES"*} and **aw**={ *"CITY", "HOLDEN"*}. Also, for simplicity if we use SED as the distance metric, then $\mathbf{d} = \begin{pmatrix} 6 & 0 \\ 3 & 6 \\ 8 & 8 \end{pmatrix}$, $\mathbf{w_{kw}} = \{6, 3, 8\}$ and $\mathbf{w_{aw}} = \{4, 6\}$ and the problem falls into the case 2) of LB_{GIM}. So we reverse the flow direction with $\mathbf{d}^T = \begin{pmatrix} 6 & 3 & 8 \\ 0 & 6 & 8 \end{pmatrix}$ and follow the greedy algorithm to get the total cost of assigned flows $TC = 3 \times 3 + (4-3) \times 6 + 6 \times 0 = 15$, where the total flow is $TF = 4 + 6 = 10$. Hence $LB_{GIM} = \frac{TC}{TF} = 1.5$. In the following, we also provide the greedy algorithm's optimality proof and the runtime complexity analysis.

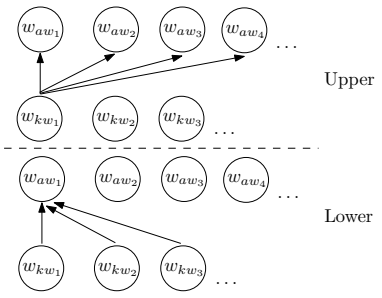

Fig. 2. Algorithmic view of LB_{GIM}

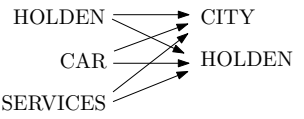

Fig. 3. An example of computing LB_{GIM}

Lemma 1. *The unified greedy algorithm obtains the normalized optimal solutions of LPs, i.e. the LB_{GIM}, and it runs in $O\left(n^2 \log n\right)$ time.*

Proof. The optimality of the algorithm is quite clear, since the relaxed constraints (2.2) and (3.1) enable us to treat either keywords or attribute words independently. Therefore the local greedy optimal choices for the words will combine to the global optimal choice. In terms of the runtime complexity, for every independent keyword/attribute word, in the worst case the algorithm needs to compare it with all attribute words/keywords, which takes $O\left(n^2\right)$ in total. However, we need to sort every word's associated distances first, which totally takes $O\left(n^2 \log n\right)$ that dominates the runtime.

3.6 Top-k Records Retrieval

In this section, we would like to study how this filter is incorporated into the k-NN algorithm for producing the final results. There are two common query types in any retrieval systems, namely the *range query,* and the *top-k query* that is also known as the k nearest neighbor (k-NN) query. A range query is associated with a specified *metric* and a *threshold*. The answer to this query is the set of all objects within the threshold after measuring by the metric. On the other hand, a top-k query is more flexible, since the threshold which is sometimes hard to decide is not needed anymore. Instead, it requires an input k that specifies the *cardinality* of the result set. In our problem, the two metrics are EMD and its lower bound filter, which are calculated by the EMD and $\hat{EMD}$ functions respectively as introduced in Section 3.3. Continuing from that section, we then define the queries in our framework as follows:

Definition 3. *A range query q to a domain $\mathbb{D} \subseteq \mathbf{DB}$ with metric c and a threshold ϵ asks for a range set of records* $\mathbf{RS}_{c,\epsilon}^{q,\mathbb{D}} = \left\{\forall r \in \mathbb{D} \,|\, c\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right) \leq \epsilon\right\}$ *, where c is either EMD or $\hat{EMD}$.* □

Definition 4. *A top-k query q to a domain $\mathbb{D} \subseteq \mathbf{DB}$ with metric c asks for a top-k set of records* $\mathbf{TKS}_{c}^{q,\mathbb{D}} = \left\{\forall r \in \mathbb{D} \,|\, c\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right) \leq \epsilon'\right\}$ *, where c is either EMD or $\hat{EMD}$, and* $\epsilon' = \mathbf{EMDs}^{q,\mathbb{D}}(k)$. □

The *domain* $\mathbb{D}$ for a query in our framework is either **DB** or its subset and $\mathbf{EMDs}^{q,\mathbb{D}}(k)$ was defined similarly in Section 3.3. A *range set* of records are returned for a range query and a *top-k set* of records for a top-k query. Now, in order to correctly utilize an $\hat{EMD}$ filter to reduce the number of exact EMD calculations in the framework, it is necessary to combine range and top-k queries together as shown in Algorithm 1. In the algorithm, step 1 and 2 issue a top-k query to **DB** with $\hat{EMD}$ as the metric. In the framework, assume we use LB_{GIM} with LD for $\hat{EMD}$ and there are totally R records in domain **DB**, then for a query these two steps take $O\left(RN^2 \log N + R \log R\right)$ time where N is the maximum number of words ($\geq$ any n) contained in a pair of matching word sequences. Step 3 and 4 then computes the exact EMDs for the set of k records obtained in step 2, and set the maximum one as the threshold to issue a range query to **DB** with $\hat{EMD}$ as the metric. These steps take $O\left(kN^3 \log N + R\right)$ time. Finally, step 5 and 6 issue another top-k query to the remaining records

Algorithm 1. k-NN Algorithm with LB_{GIM}

Input: query q with $\mathbf{kw}^q$ and M^q, input number k
Output: set $\mathbf{TKS}^{q,\mathbf{DB}}_{EMD}$.

1. $\forall r \in \mathbf{DB}$: compute $E\hat{M}D\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right)$.
2. Construct the set $\mathbf{I} = \mathbf{TKS}^{q,\mathbf{DB}}_{E\hat{M}D}$.
3. $\forall r \in \mathbf{I}$: compute $EMD\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right)$. Set $\epsilon = \max_r EMD\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right)$.
4. Construct the set $\mathbf{DB}' = \mathbf{RS}^{q,\mathbf{DB}}_{E\hat{M}D,\epsilon}$.
5. $\forall r \in \mathbf{DB}'$: compute $EMD\left(\mathbf{kw}^q, \mathbf{aw}^{r,M^q}\right)$.
6. Construct the result set $\mathbf{TKS}^{q,\mathbf{DB}'}_{EMD}$ which will be proved to be the same as the required output $\mathbf{TKS}^{q,\mathbf{DB}}_{EMD}$.

($\mathbf{DB}'$ obtained in step 4) with EMD as metric for getting the result set. Assuming there are total R' records left after filtering in step 3 and 4, the final steps then take $O\left(R'N^3 \log N + R' \log R'\right)$ time. Note that following Algorithm 1, we first show an important property of the $E\hat{M}D$ metric w.r.t. range queries.

Lemma 2. *Range sets obtained from domain* $\mathbb{D}$ *satisfy* $\mathbf{RS}^{q,\mathbb{D}}_{E\hat{M}D,\epsilon} \supseteq \mathbf{RS}^{q,\mathbb{D}}_{EMD,\epsilon}$.

The last thing left is to verify the *completeness* of the result set $\mathbf{TKS}^{q,\mathbf{DB}'}_{EMD}$, i.e. this set does not lose any actual k-NN records from $\mathbf{DB}$ using EMD as the metric. This is shown through the following lemma.

Lemma 3. *Top-k set* $\mathbf{TKS}^{q,\mathbf{DB}'}_{EMD}$ *produced from Algorithm 1 guarantees no false drops of the actual k-nn records.*

The proofs of Lemma 2 and 3 are omitted due to the constraint of the space.

4 Experiments

4.1 Experimental Setting

Dataset and preprocessing. The experiments were conducted over four datasets which were synthetically generated from the two publicly accessible WS data colletions[4,5] The WS data in these collections are gathered from real world service sites. Also, they are classified into different categories which are suitable for our requirements of the empirical study. Our created datasets include contents of both FR and NFR attributes (as described in Section 2.1) for query processing of our framework. Each dataset contains 139, 218, 315 and 404 WS

[4] `http://www.uoguelph.ca/~qmahmoud/qws/dataset`
[5] `http://www.andreas-hess.info/projects/annotator/ws2003.html`.

records respectively that covers different WS categories. In particular, we focus on `business`, `communication` and `countryInfo` categories. The raw WSs information is then preprocessed, transforming from datasets into formatted WS records in the database of our framework.

Implementation. We used C++ programming language to implement the algorithms described in Section 3.6. All of our experiments were conducted on a low to middle sized laptop computer running Windows Vista with 3GB RAM and 1.66GHz Intel(R) Core(TM)2 Duo CPU.

Similarity Search Measure. In order to evaluate the effectiveness of EMD for similarity search of WSs, we use the widely adopted standards in information retrieval: *recall* and *precision* to measure the overall performance of the framework. In particular, given a query q, let C be the total number of retrieved services, A be the total number of relevant services in the service collection and B be the number of relevant services retrieved. The recall and precision of WS search are defined as Recall=B/A and Precision=B/C respectively. Both recall and precision values are in the range between 0 and 1. The higher value of the measure indicates the more effective search results.

Variable Parameters. We identify 3 variable parameters in the experiment and investigate their impact on the experimental results. These parameters are the size of WS database, value of k in top-k queries, and the ground distance used in computing EMD which is either SED or LD.

4.2 Experimental Results

We conducted 3 groups of experimental studies. The first two explored the effectiveness of our EMD approach according to the recall and precision measurements respectively. The last group demonstrates the efficiency of similarity search using EMD and the lower bound filter LB_{GIM}.

Measuring Recall. We measured the recall of EMD-based similarity search against two variable parameters: the DB size and the ground distance. For a test query, we fixed the other variable parameter k to be the total number of relevant services in the respective database. We also compared recalls on three chosen WS categories as mentioned in the dataset description. For instance, for a query q if we know there are total p relevant service records from the business category in a database with size s, then k is fixed to be p for q searching against this particular database's business category. Therefore, for queries q's against different categories and databases with different sizes, we need to measure different p's in order to determine respective values of k.

Figure 4 shows the impact of DB sizes and ground distances. Overall, the impact of DB size on the recall of EMD-based WS search is not significant, which indicates a good property of EMD approach. Recall values in both diagrams are well above 0.6 for all DB sizes. For the impact of ground distances chosen for EMD, LD in general outperforms SED, especially in the case of larger DB sizes.

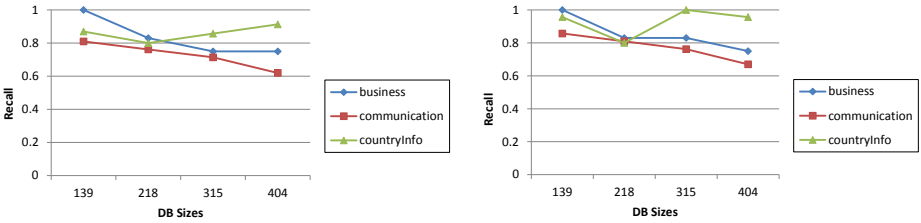

Fig. 4. EMD recall (left with SED, right with LD) v.s. DB Sizes

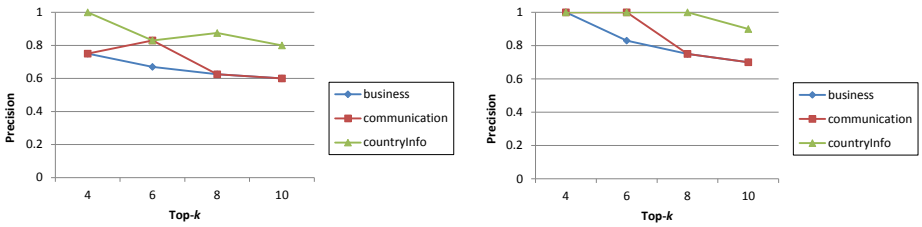

Fig. 5. EMD precision (left with SED, right with LD) v.s. Top-k queries

Measuring Precision. For a top-k query, the precision measures the value $\frac{p}{k}$ where p is the number of relevant services retrieved out of k services. Therefore this experiment is irrelevant to the parameter of DB size, but relevant to the other two parameters: k and the ground distance. For k, we chose the value of 4, 6, 8 and 10 to see its impact. The ground distance has still two choices: SED or LD. For simplicity, we again focused on the `business`, `communication` and `countryInfo` categories for retriving WSs and measuring the respective precisions.

Figure 5 presents the results from the precision measurements. Regarding the ground distances, LD outperforms SED significantly. Most noticeably, for the category `countryInfo` values of precisions are ones for $k = 4, 6, 8$. LD also provides much higher precision than SED for matching individual words having similar structures. For the impact of k, we can generally conclude that the precision descreases as the value of k increases. Nevertheless, even for top-10 queries precision values in both diagrams are still well above 0.6.

5 Related Work

Dong et al. [2] put forward a valuable similarity search approach to find Web services based on the keyword strategy. With the help of the co-occurrence of the terms appearing in service inputs and outputs, names of operation and description in services, the similarity search approach employs the agglomerative clustering algorithm to match WSs. There are also a number of work focusing on services matching and selection based on information retrieval (IR), which

analyses the one-to-one relationship between keywords. A very recent work [4] has proposed a QoS aware search engine for WSs. Although these approaches work well in searching WSs, they ignore many-to-many similarity and partial matching. Our work shares some flavour with the optimal WSs matching search proposed in [20]. However, our work differs from these works in that we employ EMD to capture partial similarity for searching WSs.

EMD is first proposed in [8] to model similarity measure between two distributions. Because of its various advantages such as allowing for partial matching and matching human perception better than other simiarity measures, it is widely exploited for various applications like phishing detection [12], document retrieval [13], and databases [10, 14]. However, the computation cost of EMD is expensive because of its principle based on linear programming. For this reason, there are a number of research [10] focusing on designing effective lower bound on EMD to improve the efficiency of computing EMD. Nevertheless, their lower bound filters are only for EMD with equal weights ($\sum_{i=1}^{n_1} w_{kw_i} = \sum_{j=1}^{n_2} w_{aw_j}$ in our case), which is clearly not suitable for the comparison of word sequences that requires solving the general EMD (including the unequal total weights case). For this reason, we develop the LB_{GIM} filter generalized from LB_{IM} in [10] for word sequences partial matching.

6 Conclusions

In this paper, we propose a new EMD-based approach for effective Web services search. The proposed EMD filter is the first effective and fast-to-compute lower bound for the general EMD. Our approach is flexible and suitable for keyword-based content searches where words with different lengths are compared. We conducted experiments over a set of real datasets of Web services and the experimental result show that WS-Finder produces query answers with high recall and precision.

The proposed approach and techniques open up some interesting directions for future research. For example, by integrating with Semantic Web technique, we can perform similarity serach for distinguishing words such as *post* and *mail*. To further improve the perfromance of EMD search, we can leverage various index structures such as inverted lists, signature files or R Tree.

References

1. Platzer, C., Dustdar, S.: A vector space search engine for web services. In: Proceedings of the 3rd European IEEE Conference on Web Services (ECOWS 2005), pp. 14–16. IEEE Computer Society Press (2005)
2. Dong, X., Halevy, A., Madhavan, J., Nemes, E., Zhang, J.: Similarity search for web services. In: Proceedings of the Thirtieth International Conference on Very Large Data Bases (VLDB 2004), pp. 372–383. VLDB Endowment (2004)
3. Ma, J., Zhang, Y., He, J.: Efficiently finding web services using a clustering semantic approach. In: Proceedings of the 16th International Workshop on Context Enabled Source and Service Selection, Integration and Adaptation: Organized with the 17th International World Wide Web Conference (WWW 2008). ACM (2008)

4. Zhang, Y., Zheng, Z., Lyu, M.: Wsexpress: a qos-aware search engine for web services. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2010), pp. 91–98. IEEE (2010)
5. Al-Masri, E., Mahmoud, Q.: Investigating web services on the world wide web. In: Proceeding of the 17th International World Wide Web Conference (WWW 2008), pp. 795–804. ACM (2008)
6. Ma, J., Zhang, Y., He, J.: Web services discovery based on latent semantic approach. In: Proceedings of the IEEE International Conference on Web Services (ICWS 2008), pp. 740–747. IEEE (2008)
7. Garofalakis, J., Panagis, Y., Sakkopoulos, E., Tsakalidis, A.: Web service discovery mechanisms: Looking for a needle in a haystack? In: Proceedings of the International Workshop on Web Engineering (2004)
8. Rubner, Y., Tomasi, C., Guibas, L.: The earth mover's distance as a metric for image retrieval. International Journal of Computer Vision 40(2), 99–121 (2000)
9. Paolucci, M., Kawamura, T., Payne, T.R., Sycara, K.: Semantic Matching of Web Services Capabilities. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 333–347. Springer, Heidelberg (2002)
10. Assent, I., Wenning, A., Seidl, T.: Approximation techniques for indexing the earth mover's distance in multimedia databases. In: Proceedings of the 22nd International Conference on Data Engineering (ICDE 2006), pp. 11–22. IEEE (2006)
11. Ljosa, V., Bhattacharya, A., Singh, A.K.: Indexing spatially sensitive distance measures using multi-resolution lower bounds. In: Proceedings of the 10th International Conference on Advances in Database Technology (EDBT 2006), pp. 865–883. ACM (2006)
12. Fu, A., Wenyin, L., Deng, X.: Detecting phishing web pages with visual similarity assessment based on earth mover's distance (emd). IEEE Transactions on Dependable and Secure Computing, 301–311 (2006)
13. Wan, X.: A novel document similarity measure based on earth mover's distance. Information Sciences 177(18), 3718–3730 (2007)
14. Xu, J., Zhang, Z., Tung, A., Yu, G.: Efficient and effective similarity search over probabilistic data based on earth mover's distance. Proceedings of the VLDB Endowment 3(1-2), 758–769 (2010)
15. Assent, I., Wichterich, M., Meisen, T., Seidl, T.: Efficient similarity search using the earth mover's distance for large multimedia databases. In: Proceedings of the 24th International Conference on Data Engineering (ICDE 2008) (2008)
16. Navarro, G., Raffinot, M.: Flexible pattern matching in strings: practical on-line search algorithms for texts and biological sequences. Cambridge Press (2002)
17. Hirschberg, D.S.: A linear space algorithm for computing maximal common subsequences. Communications of ACM 18, 341–343 (1975)
18. Hillier, F., Liberman, G.: Introduction to mathematical programming. McGraw-Hill, New York (1991)
19. Ling, H., Okada, K.: An efficient earth mover's distance algorithm for robust histogram comparison. IEEE Transactions on Pattern Analysis and Machine Intelligence, 840–853 (2007)
20. Srivastava, U., Munagala, K., Widom, J., Motwani, R.: Query optimization over web services. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 355–366. VLDB Endowment (2006)