Comparing Top-k XML Lists

Ramakrishna Varadarajan[a], Fernando Farfán[b], Vagelis Hristidis[c]

[a]*Hewlett-Packard,*
Billerica, MA 01821
[b]*Department of Computer Science and Engineering,*
University of Michigan,
Ann Arbor, 48109
[c]*Department of Computer Science and Engineering,*
University of California,
Riverside, 92521

Abstract

Systems that produce ranked lists of results are abundant. For instance, Web search engines return ranked lists of Web pages. There has been work on distance measure for list permutations, like Kendall tau and Spearman's Footrule, as well as extensions to handle top-k lists, which are more common in practice. In addition to ranking whole objects (e.g., Web pages), there is an increasing number of systems that provide keyword search on XML or other semistructured data, and produce ranked lists of XML sub-trees. Unfortunately, previous distance measures are not suitable for ranked lists of sub-trees since they do not account for the possible overlap between the returned sub-trees. That is, two sub-trees differing by a single node would be considered separate objects. In this paper, we present the first distance measures for ranked lists of sub-trees, and show under what conditions these measures are metrics. Furthermore, we present algorithms to efficiently compute these distance measures. Finally, we evaluate and compare the proposed measures on real data using three popular XML keyword proximity search systems.

Keywords: Total Mapping, Partial Mapping, Similarity Distance, Position Distance

1. Introduction

Systems that produce ranked lists of results are abundant. For instance, Web search engines return ranked lists of Web pages. To compare the lists produced by different systems, Fagin et al. [1, 2] present distance measures for top-k lists that extend the traditional distance measures for permutations of objects, like Kendall tau [3] and Spearman's Footrule [4].

Email addresses: `ramakrishna.r.varadarajan@hp.com` (Ramakrishna Varadarajan), `ffarfan@eecs.umich.edu` (Fernando Farfán), `vagelis@cs.ucr.edu` (Vagelis Hristidis)

In addition to ranking whole objects (e.g., Web pages), there is an increasing number of systems, including XRANK [5], XSEarch [6], XKeyword [7], XXL [8], XIRQL [9], rank-aware extensions [9] of XPath that provide keyword search on XML or other semi-structured data, and produce ranked lists of XML sub-trees. The same ranking challenges and principles apply to newer semistructured representation models like JSON (www.json.org).

Below are some *applications* that need XML lists distance measures:

- *Evaluate quality of XML ranking systems given a ground truth ranking*: For instance, how can we quantify how close the results of XSEarch or XRANK are to the "ideal" ranking provided by users?
- *Compare rankings between XML ranking systems*: For instance, if we show that two systems generate very similar rankings, we may choose to deploy the more efficient one.
- *Rank aggregation*: How can we build an XML meta-search engine, by combining the results of individual engines? A solution (see [10] for Web search aggregation) is to create a new "aggregated" list that is as close as possible to the individual lists. To compute that, we need lists distance measures.
- *Cluster or classify queries*: We may cluster XML queries for semantics or performance purposes based on their top-k results, or classify them, e.g., to ambiguous and non-ambiguous.

Unfortunately, previous distance measures are not suitable for ranked lists of sub-trees since they do not account for the possible overlap between the returned sub-trees. That is, two sub-trees differing by a single node would be considered separate objects. For instance, Figure 1 shows two top-3 lists of sub-trees produced by two imaginary XML keyword proximity search algorithms. Trees Ta_3 and Tb_2 only differ by a single node but this is ignored by object-level distance measures.

In this paper, we present the first distance measures for ranked lists of sub-trees, and show under what conditions these measures are metrics or near-metrics. The metric property enables fast algorithms for nearest neighbor searches [11], clustering [12] and rank aggregation [2], which are some of the applications of this work.

The proposed distance measures consist of two components: the tree similarity component and the position distance component. The former captures the similarity between the structures of the returned sub-trees, while the latter captures the distance of the sub-trees in the two lists, similarly to previous object-level distance measures [1, 2].

Intuitively, our distance measures work in two phases. In the first phase, they find the optimal (closest) mapping between the two top-k lists of sub-trees, where the distance between a pair of sub-trees is computed using one of the approaches proposed in previous works, including tree edit distance [13, 14, 15],

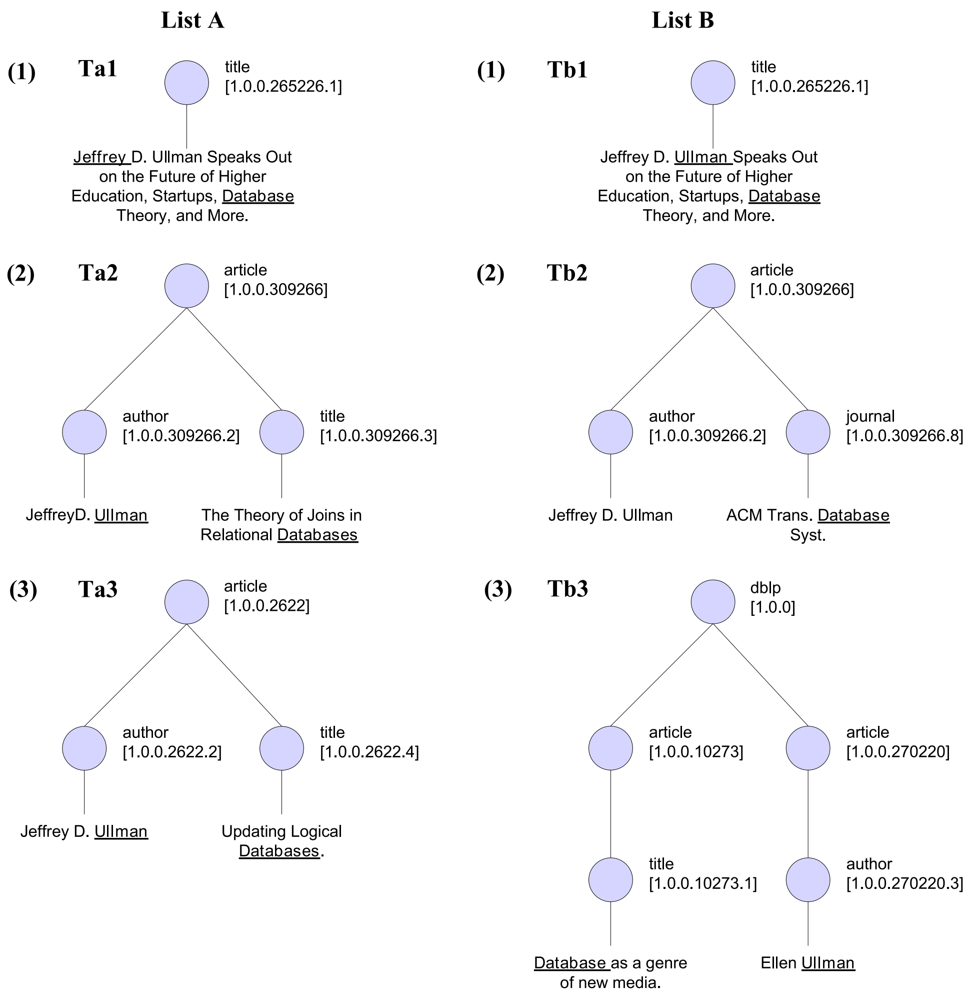


Figure 1: Top-3 tree results for query "Ullman Database" as returned by two imaginary XML keyword proximity search algorithms.

tree alignment distance [13], Fourier transform-based similarity [16], entropy-based similarity [17], tag similarity [18], and path shingle similarity [18]. The cost of the optimal mapping between the two lists of sub-trees represents the tree similarity component.

Next, we compute the position distance component given the optimal mapping, using one of the previously proposed techniques on measuring the distance between top-k (partial) lists [1, 2].

Clearly, the cost of computing all k^2 pair-wise distances between all XML trees is expensive using exact tree edit distance. This is not a problem if the application is comparing the distance between XML search engines or evaluating their quality compared to a ground truth ranking, since these application are not part of the production search engine, with which users interact. For applications where time is important, we can use approximate tree distance measures. For instance, we can use the method described in [19] to convert the trees into strings and then use edit distance approximation algorithms like edit distance with move operations [20].

We make the following contributions:

1. We present the first suite of distance measures for ranked lists of sub-trees. Three variants are presented: XML Lists Similarity based on Total Mapping (*XLS*) where all sub-trees from the first list are mapped to sub-trees in the second; XML Lists Similarity based on Total Mapping with position component (*XLS-P*)which also includes a position component; XML Lists Similarity based on Partial Mapping with position component (*XLS-PP*) where only adequately similar sub-trees are matched to each other.
2. Prove under what conditions these measures are metrics. As we show, the trickiest requirement is the satisfaction of the triangular inequality.
3. We evaluate and compare the proposed measures using real datasets. For that, we implemented three popular XML keyword proximity search systems: XRANK [5], XSEarch [6] and XKeyword [7].

The rest of this paper is organized as follows: Section 2 presents the background. In Section 3 we define the distance measures for lists of XML trees. Section 4 describes the normalization issues when combining the similarity and position component. An experimental evaluation of these algorithms is presented in Section 5. Section 6 expands our initial review of related work. Finally, Section 7 discusses our conclusions and future work.

2. Background

In this section, we briefly discuss various tree similarity measures (Section 2.1), distance measures for lists of objects (Section 2.2) and conditions that a measure must satisfy to be considered a metric (Section 2.3).

2.1. Tree Similarity Measures

In this section, we briefly present state-of-art techniques for measuring similarity between trees, proposed in the literature. Any of these similarity measures can be used in our framework. However, only the measures that are metrics will lead to a distance metric for XML lists, as shown in Section 3.

General Tree Similarity Measures: Several techniques have been proposed in the literature for measuring the similarity between general trees. Tree edit distance [13, 21, 22, 23] measures the minimum number of node insertions, deletions, and updates required to convert one tree into another. Tree alignment distance [13, 24] is a special case of the tree editing problem, in which trees become isomorphic when labels are ignored.

XML-Specific Tree Similarity Measures: Various techniques for measuring the structural similarity between XML trees have been proposed. Nierman and Jagadish [15] introduced a structural similarity distance based on tree edit distance, by adding insert-tree, delete-tree operations. Flesca et al. [16] proposed a Fourier transform technique to compute similarity, while Buttler [18] proposed a similarity metric based on path-shingles in which the structural information is extracted from the documents using the full paths. Entropy-based similarity [17] is a novel technique used to compute the structural similarity based on entropy. Tag similarity is perhaps the simplest metric for structural similarity, as it only measures how closely the set of tags match between two pages. Weis and Naumann [25] proposed a method to identify duplicate entities in a XML document which could be used to enhance the tree mapping step in our distance metrics.

As mentioned before, only those tree similarity measures that are metrics may lead to a distance metric or near-metric for XML lists. The following tree similarity measures are metrics: tree-edit distance [13], tree-edit-based structural distance [15], fourier transform-based distance [16, 26], entropy-based similarity [17], and the similarity measure in [14]. In Table 1 we present these results in more detail along with the complexity of calculating each tree similarity measure.

2.2. Distance Measures for Permutations and top-k Lists

In this section, we present some of the most popular and widely used measures for computing distance between complete (permutations) and partial lists of objects.

Measures for permutations: We review Spearman's footrule and Kendall tau distance measures [4, 2, 3]. Spearman's footrule metric is the $L1$ distance between two permutations. Formally, it is defined by

$$F(\sigma_1, \sigma_2) = \sum_{i=1}^{k} |\sigma_1(i) - \sigma_2(i)|$$

where σ_1 and σ_2 are the two permutations of length k, and $\sigma_1(i)$ denotes the ith element in σ_1.

Table 1: Tree Similarity Measures and their Properties.

Tree Similarity Measure	Is Metric?	XML-Specific?	**Time Complexity to compute the measure between pair of trees**
Tree-Edit Distance [13]	Yes	No	$O(Cost(TD(Ta_i, Tb_j)) = O(\|Ta_i\| \cdot \|Tb_j\| \cdot min(leaves(Ta_i), depth(Ta_i)) \cdot min(leaves(Tb_j), depth(Tb_j))$ [15]
Tree-Alignment Distance [13, 24]	No (fails triangle inequality)	No	$O(\|Ta_i\| \cdot \|Tb_j\| \cdot (deg(Ta_i) + deg(Tb_j))^2)$
Tree-Edit-based Structural Distance [15]	Yes	Yes	$O(\|Ta_i\| \cdot \|Tb_j\|)$
Fourier Transform-based Distance [26, 16]	Yes	Yes	$O(N \log N)$ where $N = max(\|Ta_i\|, \|Tb_j\|)$
Entropy-based Similarity [17]	Yes	Yes	$O(N)$ where $N = max(\|Ta_i\|, \|Tb_j\|)$
Path-Shingle-based Similarity [18]	No (since it is based on hashing)	Yes	Linear time in general. $O(\|Ta_i\| + \|Tb_j\|)$
Similarity measure in [14]	Yes	Yes	Linear time in general. $O(\|Ta_i\| + \|Tb_j\|)$
Similarity measure in [22]	No	No	Linear time in general. $O(\|Ta_i\| + \|Tb_j\|)$

Kendall's tau metric between permutations is defined as follows: For each pair $i, j \in P$ of distinct members, if i and j are in the same order in σ_1 and σ_2, then let $\overline{K}_{i,j}(\sigma_1, \sigma_2) = 0$; else $\overline{K}_{i,j}(\sigma_1, \sigma_2) = 1$. Kendall tau is

$$K(\sigma_1, \sigma_2) = \sum_{\{i,j\} \in P} \overline{K}_{i,j}(\sigma_1, \sigma_2)$$

Measures for top-k lists: We now review two well-known measures on top-k lists of objects [2, 1]: Spearman's footrule distance with location parameter l, denoted $F^{(l)}$, and Kendall distance with penalty parameter p, denoted $K^{(p)}$. $F^{(l)}$ is obtained, intuitively, by placing all missing elements in each of the lists at position l and computing the usual footrule distance between them. A natural choice for l is $k+1$, which we use in our experiments. $K^{(p)}$ is obtained by naturally generalizing Kendall tau measure for permutations. Similar to $\overline{K}_{i,j}(\sigma_1, \sigma_2)$ for permutations σ_1, σ_2, a penalty $\overline{K}^{(p)}_{i,j}(\sigma_1, \sigma_2)$ for top-k lists σ_1, σ_2 is defined as shown in [2]. The measures discussed in [2] for top-k lists does not consider ties. Typically, in real world, there are a number of fields with very few distinct values, and hence the corresponding rankings have many ties in them. [1] modify top-k list measures to handle ties by organizing the elements with same score in buckets $\overline{K}_{i,j}$ and taking the average location within each bucket as the position of each element in that bucket. In Section 3.3 we use a combination of [2] and [1] to define the distance between top-k lists with ties.

2.3. When a Distance Measure is a metric

A binary function d is called symmetric if $d(x, y) = d(y, x)$ for all x, y in the domain, and is called regular if $d(x, y) = 0$ if and only if $x = y$. We define a distance measure to be a nonnegative, symmetric, regular binary function. A metric is a distance measure d that satisfies the triangle inequality $d(x, z) \leq d(x, y) + d(y, z)$ for all x, y, z in the domain.

3. Distance Measures for Lists of XML Trees

In this section, we first provide some definitions (Section 3.1), and present the *XLS* measure (Section 3.2), *XLS-P* measure (Section 3.3) and *XLS-PP* measure (Section 3.4). Normalization issues are discussed in Section 4.

3.1. Problem Definition

The goal of this work is to define and compute the distance between two lists La, Lb of XML trees, $La = Ta_1, Ta_2 \cdots Ta_k$ and $Lb = Tb_1, Tb_2 \cdots, Tb_k$, where Tx_i are XML trees. Often, as is the case with XML proximity search systems, all Ta_i, Tb_j are included (obtained by a sequence of deletes) in a tree Ti of a collection $D = T1, \cdots, Tn$. However, this property is not important in our definitions. Note that for the case of complete lists (permutations) of subtrees where each subtree appears in both lists, the problem is reduced to the permutations distance problem which we discussed in Section 2.2. However, this case is not practical since XML search engines return different XML trees. Hence, we focus on top-k lists.

A *total mapping* f from La to Lb is a bijection from La to Lb. Hence, tree Ta_i is mapped to $Tb_j = f(Ta_i)$. Let N be the set of all possible total mappings f from La to Lb. Similarly, a *partial mapping* g is a partial function from La to Lb. Let $TS(T1, T2)$ be the tree similarity between two trees $T1$, $T2$. TS can be the tree edit distance or another measure as discussed in Section 2. TS is normalized in $[0, 1]$ as explained in Section 4.1.

3.2. XML Lists Similarity Distance based on Total Mapping (XLS)

In this section we present our first measure for the distance between two top-k lists of XML trees. The key intuition is that we extend previous list distance measures that only consider exact mappings between the objects of the two lists to also consider approximate mappings. In particular, we compute the closest pair-wise mappings between the XML trees from the two lists. Assuming k elements in each XML list, *XLS* is defined as follows. First we define the total mapping similarity distance $M^T(La,Lb,f)$ between La and Lb for a total mapping f as

$$M^T(La,Lb,f) = \frac{\sum_{i=1\cdots k} TS(Ta_i, f(Ta_i))}{k} \tag{1}$$

That is, M^T is a measure of how "tight" the total mapping f is. Notice that $M^T(La,Lb,f)$ takes values in $[0, 1]$, since TS is also in $[0, 1]$ and we divide by k.

We next define the minimum total mapping $fmin^T_{ab}$ as the total mapping between La and Lb with minimum $M^T(La,Lb,f)$. It is,

$$fmin^T_{ab} = argmin_f M^T(La,Lb,f) \tag{2}$$

that is, $argmin_f$ is the f that minimizes M^T.

Given $fmin^T_{ab}$, we define the *minimum total mapping similarity distance*,

$$MinM^T(La,Lb) = M^T(La,Lb,fmin^T_{ab}) \tag{3}$$

Definition 3.1. *The XML Lists Similarity based on total mapping (XLS) between XML lists La, Lb is the minimum total mapping similarity distance. It is:*

$$XLS(La,Lb) = MinM^T(La,Lb) \tag{4}$$

Notice that $XLS(La,Lb)$ is in [0,1] since $MinM^T(La,Lb)$ is in [0,1].

Measures for $MinM^T(La,Lb)$: The tree similarity TS which is used to compute $MinM^T(La,Lb)$ can be any of the tree or XML similarity measures discussed in Section 2.1. The only constraint (as we show in Theorem 3.1) is that the measure used must be a metric if XLS is to be a metric.

Example 1: *Consider the top-3 lists La and Lb in Figure 1. We will illustrate the steps involved in computing XLS(La,Lb). In this example, we use tree edit distance, TED as the tree similarity measure, TS. We first compute the XML similarity component by finding all possible total mappings,* $N = \{f_1, f_2, f_3, f_4, f_5, f_6\}$*:*

$$\begin{array}{l}
f_1(Ta_1) = Tb_1,\ f_1(Ta_2) = Tb_2,\ f_1(Ta_3) = Tb_3\\
f_2(Ta_1) = Tb_3,\ f_2(Ta_2) = Tb_2,\ f_2(Ta_3) = Tb_1\\
f_3(Ta_1) = Tb_2,\ f_3(Ta_2) = Tb_1,\ f_3(Ta_3) = Tb_3\\
f_4(Ta_1) = Tb_1,\ f_4(Ta_2) = Tb_3,\ f_4(Ta_3) = Tb_2\\
f_5(Ta_1) = Tb_3,\ f_5(Ta_2) = Tb_1,\ f_5(Ta_3) = Tb_2\\
f_6(Ta_1) = Tb_2,\ f_6(Ta_2) = Tb_3,\ f_6(Ta_3) = Tb_1
\end{array}$$

The normalized tree edit distance (see Section 4.1) between each pair of trees in La and Lb is given by the following matrix:

$$\begin{array}{c|ccc}
 & Tb_1 & Tb_2 & Tb_3\\
\hline
Ta_1 & 0.00 & 0.78 & 0.71\\
Ta_2 & 0.71 & 0.58 & 0.20\\
Ta_3 & 0.78 & 0.43 & 0.58
\end{array}$$

The total mapping similarity distance of each total mapping in N is calculated by Eq. 1 as follows:

$$\begin{array}{l}
M^T(La,Lb,f_1) = (0.00 + 0.58 + 0.58)/3 = 0.39\\
M^T(La,Lb,f_2) = (0.71 + 0.58 + 0.78)/3 = 0.69\\
M^T(La,Lb,f_3) = (0.78 + 0.71 + 0.58)/3 = 0.69\\
M^T(La,Lb,f_4) = (0.00 + 0.20 + 0.43)/3 = 0.21\\
M^T(La,Lb,f_5) = (0.71 + 0.71 + 0.43)/3 = 0.62\\
M^T(La,Lb,f_6) = (0.78 + 0.20 + 0.78)/3 = 0.59
\end{array}$$

Hence, f_4 is the mapping with the minimum mapping distance. It is

$$XLS(La,Lb) = MinM^T(La,Lb) = M^T(La,Lb,f_4) = 0.21.\ \square$$

3.2.1. XLS is a metric

Theorem 3.1. *XLS is a metric if the tree similarity measure employed, TS, is a metric.*

Proof. It is straightforward that *XLS* is non-negative ($XLS(La,Lb) \geq 0$), symmetric ($XLS(La,Lb) = XLS(Lb,La)$) and regular ($XLS(La,La) = 0$) since this holds for tree similarity measure, *TS* which is a metric.

We need to prove the triangular property, that is, for any tree lists *La*, *Lb*, *Lc* prove that:

$$XLS(La,Lc) \leq XLS(La,Lb) + XLS(Lb,Lc) \tag{5}$$

To do so, we will prove the triangular property for $MinM^T(\cdot,\cdot)$. That is we need to prove that:

$$MinM^T(La,Lc) \leq MinM^T(La,Lb) + MinM^T(Lb,Lc) \tag{6}$$

- *Prove triangular property for $MinM^T$:* From Eqs. 1[1] and 3:

$$\begin{aligned}
MinM^T(La,Lb) &= M^T(La,Lb,fmin^T_{ab})\\
&= \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ab}(Ta_i))
\end{aligned} \tag{7}$$

[1]We skip k in denominator of Eq. 1 throughout the proof, as it is for normalization purposes and does not affect the proof correctness.

where $fmin_{ab}$ is the minimum total mapping from La to Lb. Similarly:

$$\begin{aligned} MinM^T(Lb,Lc) &= M^T(Lb,Lc,fmin^T_{bc}) \\ &= \sum_{j=1\cdots k} TS(Tb_j, fmin^T_{bc}(Tb_j)) \end{aligned} \tag{8}$$

$$\begin{aligned} MinM^T(La,Lc) &= M^T(La,Lc,fmin^T_{ac}) \\ &= \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ac}(Ta_i)) \end{aligned} \tag{9}$$

Hence, from Eqs. 7, 8 and 9, proving Eq. 6 is equivalent to proving:

$$\begin{aligned} \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ac}(Ta_i)) \leq & \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ab}(Ta_i)) + \\ & \sum_{j=1\cdots k} TS(Tb_j, fmin^T_{bc}(Tb_j)) \end{aligned} \tag{10}$$

Since the tree similarity measure $TS(\cdot,\cdot)$ is a metric, it satisfies the triangular property. Consider a tree Ta_i in La that is mapped to $Tb_j = fmin^T_{ab}(Ta_i)$ in Lb, which is in turn mapped to tree $Tc_s = fmin^T_{bc}(Tb_j) = fmin^T_{bc}(fmin^T_{ab}(Ta_i))$ in Lc. The triangular property for Ta_i, Tb_j, Tc_s can be written as:

$$\begin{aligned} TS(Ta_i, fmin^T_{bc}(fmin^T_{ab}(Ta_i))) \leq & TS(Ta_i, fmin^T_{ab}(Ta_i)) + \\ & TS(Tb_j, fmin^T_{bc}(Tb_j)) \end{aligned} \tag{11}$$

Summing Eq. 11 over all Ta_i's in La, and keeping in mind that $fmin_{ab}, fmin_{bc}$ are bijections, we get

$$\begin{aligned} \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{bc}(fmin^T_{ab}(Ta_i))) \leq & \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ab}(Ta_i)) + \\ & \sum_{j=1\cdots k} TS(Tb_j, fmin^T_{bc}(Tb_j)) \end{aligned} \tag{12}$$

The left hand side of Eq. 12 is the total mapping similarity distance $M^T(La,Lc,f')$, where $f'(\cdot) = fmin^T_{bc}(fmin^T_{ab}(\cdot))$. We know from Eq. 9, that $fmin^T_{ac}$ gives the minimum total mapping similarity distance between La,Lc. That is

$$M^T(La,Lc,fmin^T_{ac}) \leq M^T(La,Lc,f') \tag{13}$$

Hence,

$$\sum_{i=1\cdots k} TS(Ta_i, fmin^T_{ac}(Ta_i)) \leq \sum_{i=1\cdots k} TS(Ta_i, fmin^T_{bc}(fmin^T_{ab}(Ta_i))) \tag{14}$$

From Eqs. 11 and 14 we get Eq. 10 which was our goal. □

3.2.2. XLS Computation

To calculate *XLS* between any two top-k XML lists La and Lb, we start by precomputing the tree similarity measure between each tree pair across the two lists. There are k^2 such pairs, hence the complexity of this step is $k^2 \cdot Cost(TS(Ta_i, Tb_j))$ where $Cost(TS(Ta_i, Tb_j))$ is the complexity of computing the tree similarity between the two trees Ta_i and Tb_j. We use the dynamic programming algorithm by Zhang and Shasha [23], in our experiments, to compute the edit-distance between ordered trees [13]. Then, we apply a minimum cost perfect matching algorithm (we use the Hungarian algorithm [27]) to compute all minimum mappings fmin^T_{ab} between the two lists. The complexity of this step is $O(k^3)$. Then, we compute $\mathit{XLS}(\mathit{La},\mathit{Lb})$ using Eq. 4. The total complexity of *XLS* computation is $O(k^2 \cdot \mathit{Cost}(\mathit{TS}(\mathit{Ta}_i, \mathit{Tb}_j)) + k^3)$.

3.3. XML Lists Similarity Distance based on Total Mapping with Position Component (XLS-P)

As we described in Section 3.2, *XLS* considers the similarity of XML trees across each list. This works well in computing a reasonable similarity distance between top-k XML Lists where k is relatively small. When k is large, it is important to also consider the position of the mapped trees in each list. We define XML Lists Similarity Distance based on Total Mapping with Position Component (*XLS-P*), which includes the mapping position distance in addition to the mapping similarity distance. The key intuition is that *XLS-P* estimates the cost to transform the one list into the other list. In particular, we first compute the closest pair-wise mappings between the XML trees from the two lists (as in XLS) and then view these mappings as exact mappings and apply list permutation distance measures.

Definition 3.2. *The XML Lists Similarity Distance based on Total Mapping with Position Component (XLS-P) between XML lists La, Lb has two components:*

a. The XML similarity component $\mathit{MinM}^T(\mathit{La},\mathit{Lb})$.
b. The total mapping position distance component $P^T(\mathit{La},\mathit{Lb},\mathit{fmin}^T_{ab})$*, which is also referred as the position component in this section.* P^T *is defined using one of the well known metrics on permutations as discussed below.* P^T *is in* $[0,1]$ *as discussed in Section 4.2.*

It is

$$\mathit{XLS\text{-}P}(\mathit{La},\mathit{Lb}) = \mathit{MinM}^T(\mathit{La},\mathit{Lb}) + P^T(\mathit{La},\mathit{Lb},\mathit{fmin}^T_{ab}) \tag{15}$$

Note that $\mathit{XLS\text{-}P}(\mathit{La},\mathit{Lb})$ *is in* $[0,2]$ *since* $\mathit{MinM}^T(\mathit{La},\mathit{Lb})$ *and* $P^T(\mathit{La},\mathit{Lb},\mathit{fmin}^T_{ab})$ *are in* $[0,1]$.

We choose fmin^T_{ab} to minimize the XML similarity component and not the whole *XLS-P*, because we believe it is more intuitive to compute the distance component based on the tightest XML similarity mapping rather than mixing the two components. Note that other functions can be used to combine the contribution of the two components, as we discuss below.

Measures for XML Similarity component, $\mathit{MinM}^T(\mathit{La},\mathit{Lb})$*:* The tree similarity *TS* which is used to compute $\mathit{MinM}^T(\mathit{La},\mathit{Lb})$ can be any of the tree or XML similarity measures discussed in Section 2.1.

Measures for Position component, $P^T(\mathit{La},\mathit{Lb},\mathit{fmin}^T_{ab})$*:* Note that list permutation distance metrics (not top-k list distance measures) are used in *XLS-P*. Given the mapping $fmin^T_{ab}$, we naturally extend the Spearman's footrule distance and Kendall tau distance for permutations with ties [1, 2, 4, 3] as follows:

Position distance (P^{TF}) based on Spearman's footrule metric for permutations, is given by:

$$P^{TF}(La,Lb,fmin^{T}_{ab}) = \sum_{i=1}^{k} \left| pos_{La}(Ta_i) - pos_{Lb}(fmin^{T}_{ab}(Ta_i)) \right| \tag{16}$$

where $pos_{La}(Ta_i)$ is the position of tree Ta_i in list La. This formula is extended as follows to consider ties. A set of trees with the same score is called a bucket. The ranked list of results can be then viewed as ranked list of buckets $B_1, B_2, \cdots, B_n$. The position of bucket B_i, denoted $pos(B_i)$ is the average result location within bucket B_i. We assign $pos_{La}(Ta_i) = pos(B(Ta_i))$ where $B(Ta_i)$ is the bucket of Ta_i.

Position distance (P^{TK}) based on Kendall tau metric for permutations considering ties, is given by:

$$P^{TK}(La,Lb,fmin^{T}_{ab}) = \sum_{\{i,j\}\in S} \overline{K}_{i,j}(La,Lb') \tag{17}$$

where Lb' is constructed from list Lb when element Tb_j is replaced by $Ta_i = (fmin^{T}_{ab})^{-1}(Tb_j)$, that is, $Tb_j = fmin^{T}_{ab}(Ta_i)$. That is, we assume that an element Ta_i in La and its corresponding element Tb_j in Lb are the same. Hence, we just have k distinct elements $\{1, 2, \cdots, k\}$ in both lists, and the problem of computing $P^{TK}(La,Lb,fmin^{T}_{ab})$ of the two XML lists is same as computing the Kendall Tau metric of two permutations. S is the set of all unordered pairs of the k distinct elements.

Hence, there are two variants of *XLS-P*:

$$XLS\text{-}P^{F}(La,Lb) = MinM^{T}(La,Lb) + P^{TF}(La,Lb,fmin^{T}_{ab}) \tag{18}$$

$$XLS\text{-}P^{K}(La,Lb) = MinM^{T}(La,Lb) + P^{TK}(La,Lb,fmin^{T}_{ab}) \tag{19}$$

Example 1 (cont'd): Consider the top-3 lists La and Lb in Figure 1. We will illustrate the steps involved in computing *XLS-P*(La,Lb). As before, we first compute f_4 –the total mapping with the minimum mapping similarity distance. It is $MinM^{T}(La,Lb) = M^{T}(La,Lb,f_4) = 0.21$. The normalized Spearman's footrule position component is $P^{TF}(La,Lb,f_4) = 2.0/4.0 = 0.5$. Hence, $XLS\text{-}P^{F}(La,Lb) = 0.21 + 0.5 = 0.71$. If the position distance is calculated using normalized Kendall tau, then $P^{TK}(La,Lb,f_4) = 1.0/3.0 = 0.33$ and $XLS\text{-}P^{K}(La,Lb) = 0.21 + 0.33 = 0.54$. The difference in the two scores is due to inherent differences between the Spearman's footrule and Kendall tau metrics. □

3.3.1. XLS-P is a near-metric

XLS-P is not a metric because the total mapping position distance component $P^{T}(La,Lb,fmin^{T}_{ab})$ is not a metric. In particular, $P^{T}(La,Lb,fmin^{T}_{ab})$ does not satisfy the triangular inequality property. This is because the mapping $fmin^{T}_{ab}$ is computed by comparing XML trees (accounting for possible tree overlaps) and not by comparing whole objects. To be more specfic, if we consider three lists of (whole) objects *Wa, Wb* and *Wc*, then $f^{T}_{ac}(\cdot) = f^{T}_{bc}(f^{T}_{ab}(\cdot))$ (where

f^T_{ac} is a total mapping between Wa and Wc) since we can only have "exact" matches. But if we consider three lists of XML trees *La, Lb* and *Lc*, typically $fmin^T_{ac}(\cdot) \neq fmin^T_{bc}(fmin^T_{ab}(\cdot))$ since we could have "partial" matches.

The following example illustrates this scenario: Let *La, Lb* and *Lc* be the following top-2 lists of XML trees. $La = (Ta_1,\ Ta_2)$, $Lb = (Tb_1,\ Tb_2)$ and $Lc = (Tc_1,\ Tc_2)$. Now, suppose that $TS(Ta_1,\ Tb_1) = TS(Ta_2,\ Tb_2) = TS(Tb_1,\ Tc_1) = TS(Tb_2,\ Tc_2) = TS(Ta_1,\ Tc_2) = TS(Ta_2,\ Tc_1) = 0.4$ and all other distances (between the remaining pairs across the different lists) are 0.6 (and for all x, $TS(x,x) = 0$). Then, the following would be the minimum total mappings between each list *La, Lb* and *Lc*:

$$fmin^T_{ab}(Ta_1) = Tb_1,\ fmin^T_{ab}(Ta_2) = Tb_2,$$
$$fmin^T_{bc}(Tb_1) = Tc_1,\ fmin^T_{bc}(Tb_2) = Tc_2,$$
$$fmin^T_{ac}(Ta_1) = Tc_2,\ fmin^T_{ac}(Ta_2) = Tc_1$$

Then $MinM^T(La,Lb) = MinM^T(Lb,Lc) = MinM^T(La,Lc) = (0.4 + 0.4)/2 = 0.4$ (after normalization). But, $fmin^T_{ab}$ and $fmin^T_{bc}$ preserve order (i.e., Ta_1 is mapped to Tb_1, Ta_2 is mapped to Tb_2 and so on), whereas $fmin^T_{ac}$ does not preserve order (it maps Ta_1 to Tc_2 and Ta_2 to Tc_1). Hence we have $P^T(La,Lb,fmin^T_{ab}) = P^T(La,Lb,fmin^T_{bc}) = 0.0$ and $XLS\text{-}P(La,Lb) = XLS\text{-}P(Lb,Lc)$. Now, since $fmin^T_{ac}$ does not preserve order, $P^T(La,Lb,fmin^T_{ab}) > 0$ (in fact the actual value would be 1.0 as it maps the elements in reverse order). So, $XLS\text{-}P(La,Lc) = (0.4 + 0.4)/2 + 1.0 = 1.4$. This breaks the triangular inequality property since $XLS\text{-}P(La,Lc) > XLS\text{-}P(La,Lb) + XLS\text{-}P(Lb,Lc)$, as $1.4 > 0.4 + 0.4 = 0.8$.

A measure is a near-metric when it satisfies all properties of a metric, but instead of the triangular property, it satisfies the near-triangular inequality: $d(x,z) \leq r \cdot (d(x,y) + d(y,z))$, where r is a positive constant [28]. Note that Fagin et al. [2] proved that most of their measures (for whole objects, that is, no objects' overlap considered) are near-metrics.

Theorem 3.2. *XLS-P is a near-metric if the tree similarity measure TS employed is a metric (or near-metric).*

Proof. We must prove that there is constant $r > 0$, such that

$$XLS\text{-}P(La,Lc) \leq r \cdot (XLS\text{-}P(La,Lb) + XLS\text{-}P(Lb,Lc)) \tag{20}$$

Note that we omit the K (Kendall tau) or F (Footrule) superscript because the same proof applies to both.

From Eq. 18, we have

$$m + v \leq r \cdot (M + V) \tag{21}$$

where $m = MinM^T(La, Lc)$, $v = P^T(La, Lc)$, $M = MinM^T(La, Lb) + MinM^T(Lb, Lc)$, $V = P^T(La, Lb) + P^T(Lb, Lc)$.

If $M = 0$ then La, Lb, Lc have exactly same set of trees, and hence XLS-P reduces to top-k Footrule or Kendall tau, which we know that they are near-metrics [2].

If $M > 0$, we know from Eq. 1 that $M > w/k$, where w is the minimum possible normalized tree similarity distance $TS(Ta, Tb)$ between two trees Ta, Tb.

E.g., for tree edit distance, it is $w = 1/(2 \cdot \mathit{maxTreeSize})$ assuming unit node deletion and insertion cost (see Section 4.1).

We also know that $0 \leq m, v \leq 1$, $V \geq 0$. That is, the maximum value of the left hand side of Eq. 21 is 2. If we pick $r = 2 \cdot k/w$, we guarantee that Eq. 21 is satisfied.

$\square$

3.3.2. XLS-P Computation

The computation of *XLS-P* is similar to the computation shown in Section 3.2.2, except for a few changes as we will describe. fmin^T_{ab} between the two lists is computed as before and then the position distance $P^T(\mathit{La},\mathit{Lb},\mathit{fmin}^T_{ab})$ is computed using Eqs. 16 and 17 for Spearman's footrule and Kendall tau position component respectively. Then, *XLS-P* is computed using Eq. 18 or 19. The total complexity *XLS-P* computation is $O(k^2 \cdot \mathit{Cost}(\mathit{TS}(\mathit{Ta}_i, \mathit{Tb}_j)) + k^3)$.

3.4. XML Lists Similarity Distance based on Partial Mapping with Position Component (XLS-PP)

The total mapping distance measures in Sections 3.2 and 3.3 have the drawback that two totally irrelevant trees from the two lists may be mapped to each other, given that all trees must be mapped between the two lists. This is unintuitive and may lead to confusing results, especially for the position component of the measure. Hence, we propose to only maps trees if they are adequately similar.

Similarity Threshold: In order to partially map the two XML lists, we specify a threshold ω in $[0, 1]$. Intuitively, we only create mappings between trees whose tree similarity (*TS*) is up to ω. For example, if we want to create only the mappings between trees that are at most 40% different, then we set $\omega = 0.4$. Notice that *TS* is also in $[0, 1]$ as described in Section 4.1. We consider various values for ω in Section 5 and show that the relative distance between two pairs of XML lists is not so sensitive on the choice of ω. We observe that $\omega = 0.5$ gives a nice balance between the position and tree similarity components of *XLS-PP*. Note that for $\omega = 1$, *XLS-PP* reduces to *XLS-P*.

Assuming k elements in each XML list, *XLS-PP* is defined as follows. First we define the partial mapping g for a total mapping f and threshold ω. g is a partial function defined only for XML trees Ta_i with $\mathit{TS}(\mathit{Ta}_i, f(\mathit{Ta}_i)) \leq \omega$. Then $g(\mathit{Ta}_i) = f(\mathit{Ta}_i)$. Let La^g be the subset of La that contains the XML trees that have a mapping for g. Next we define the *partial mapping similarity distance* $M^P(\mathit{La},\mathit{Lb},g)$ between La and Lb given a partial mapping g as:

$$M^P(\mathit{La},\mathit{Lb},g) = \frac{\sum_{\mathit{Ta}_i \in \mathit{La}^g} \mathit{TS}(\mathit{Ta}_i, g(\mathit{Ta}_i)) + \sum_{\mathit{Ta}_i \in \{\mathit{La}\text{-}\mathit{La}^g\}} c}{k \cdot \mathit{max}(c, \omega)} \tag{22}$$

where XML trees that do not get mapped incur a penalty cost $0 \leq c \leq 1$. Notice that $M^P(\mathit{La},\mathit{Lb},g)$ is also in [0,1] since *TS* is in [0,1] and we divide by $k \cdot \mathit{max}(c, \omega)$. c is naturally between ω and 1; we set $c = \omega$ in our experiments. The rationale is similar to [2], where an element not in the top-k is assumed to be in the $k+1$-th position (in our case where c is a real number it would be $c = \omega + \epsilon \simeq \omega$).

We next define the minimum partial mapping $gmin^P_{ab}$ between La and Lb given a threshold, ω as the partial mapping that has a corresponding total mapping f for threshold ω and has the minimum $M^P(La,Lb,g)$. That is,

$$gmin^P_{ab} = argmin_g M^P(La,Lb,g) \tag{23}$$

We emphasize that g must come from a total mapping, in order for the metric properties defined below to hold.

Given $gmin^P_{ab}$, we define the *minimum partial mapping similarity distance*

$$MinM^P(La,Lb) = M^P(La,Lb,gmin^P_{ab}) \tag{24}$$

Definition 3.3. *The XML Lists Similarity Distance based on Partial Mapping with Position Component (XLS-PP) has two components:*

a. The XML partial similarity component $MinM^P(La,Lb)$.

b. The partial mapping position distance component $P^P(La,Lb,gmin^P_{ab})$, which is also referred as the position component. P^P can be one of the well known measures (some are not metrics) on top-k lists as discussed below. P^P is in $[0,1]$ as discussed in Section 4.2.

It is:

$$XLS\text{-}PP(La,Lb) = MinM^P(La,Lb) + P^P(La,Lb,gmin^P_{ab}) \tag{25}$$

Notice that XLS-PP(La,Lb) is in $[0,2]$ since $MinM^P(La,Lb)$ and $P^P(La,Lb,gmin^P_{ab})$ are in $[0,1]$.

The same tree similarity measures as in *XLS* can be used for the XML partial similarity component.

Measures for Position component, $P^P(La,Lb,gmin^P_{ab})$: We need to use partial (top-k) list distance measures. Given the partial mapping $gmin^P_{ab}$, we naturally extend the Spearman's footrule distance and Kendall tau distance for top-k lists with ties by combining previous works [1, 2, 4, 3], which separately tackle the top-k [2] and the ties [1] issues, as follows:

Position distance $P^{PF(l)}$ based on Spearman's footrule for top-k lists with location parameter l considering ties is computed as follows. We place all trees in both lists whose tree similarity TS is greater than threshold ω at position l. Let list Lb be a list constructed by Lb by replacing each element Tb_i by $Ta_j = (gmin^P_{ab})^{-1}(Tb_i)$, if this mapping exists (recall that $gmin^P_{ab}$ is a partial function). Then,

$$P^{PF(l)}(La,Lb,gmin^P_{ab}) = F^{(l)}(La,Lb') \tag{26}$$

where $F^{(l)}(\cdot,\cdot)$ is the footrule function for top-k lists defined in [2]. We extend this formula to account for ties by considering buckets for computing the position as explained in Section 3.2.

Position distance $P^{PK(p)}(La,Lb,gmin^P_{ab})$ based on Kendall tau metric for top-k lists with penalty parameter p, considering ties, is given by:

$$P^{PK(p)}(La,Lb) = \sum_{\{i,j\} \in La \,\cup\, Lb'} \overline{K}^{(p)}_{(i,j)}(La,Lb') \tag{27}$$

where Lb' is defined as in Section 3.2, and $\overline{K}^{(p)}_{(i,j)}(\cdot,\cdot)$ is defined as in [2].

Hence, we have two variant of *XLS-PP*:

$$XLS\text{-}PP^F(La,Lb) = MinM^P(La,Lb) + P^{PF(l)}(La,Lb,gmin^P_{ab}) \tag{28}$$

$$XLS\text{-}PP^K(La,Lb) = MinM^P(La,Lb) + P^{PK(p)}(La,Lb,gmin^P_{ab}) \tag{29}$$

Example 1 (cont'd): Consider again the lists La and Lb in Figure 1. Assuming $\omega = c = 0.4$, we get the following partial mappings from the previous total mappings:

$$\begin{aligned} g_1(Ta_1) &= Tb_1. \\ g_2,\ g_3 & \quad \textit{are empty mappings.} \\ g_4(Ta_1) &= Tb_1, g_4(Ta_2) = Tb_3. g_5 \textit{ are empty mappings.} \\ g_6(Ta_2) &= Tb_3. \end{aligned}$$

The mapping distance of each partial mapping is as follows:

$$\begin{aligned} M^P(La,Lb,g_1) &= (0.00 + c + c)/(3 \cdot max(c,\omega)) = 0.66 \\ M^P(La,Lb,g_2) &= (c + c + c)/(3 \cdot max(c,\omega)) = 1.00 \\ M^P(La,Lb,g_3) &= 1.00 \\ M^P(La,Lb,g_4) &= (0.00 + 0.20 + c)/(3 \cdot max(c,\omega)) = 0.50 \\ M^P(La,Lb,g_5) &= 1.00 \\ M^P(La,Lb,g_6) &= (c + 0.20 + c)/(3 \cdot max(c,\omega)) = 0.83 \end{aligned}$$

g_4 is the mapping with the minimum mapping distance. $MinM^P(La,Lb) = M^P(La,Lb,g_4) = 0.50$.

The Spearman's footrule position component $P^{PF}(La,Lb,g_4) = 4.0/12.0 = 0.33$. $XLS\text{-}PP^F(La,Lb) = 0.50 + 0.33 = 0.83$. If the position distance is calculated using normalized Kendall tau, then $P^P(La,Lb,g_4) = 2.0/12.0 = 0.17$. $XLS\text{-}PP^K(La,Lb) = 0.50 + 0.17 = 0.67$. Notice that the normalized position component in *XLS-P* is smaller than in *XLS-PP*, even though two trees do not match in *XLS-PP*. The reason is that the maximum value (used in normalizing as we describe in Section 4.2) of position distance (P^P) is larger in *XLS-PP*.□

3.4.1. XLS-PP is a near-metric

XLS-PP is not a metric because M^P is not a metric. The reason is that the triangular property does not hold for any choice of ω and c.

Theorem 3.3. *XLS-PP is a near-metric if the tree similarity measure employed, TS, is a metric (or near-metric).*

Proof. The proof is identical to that of Theorem 3.2, except that we pick $r = 2 \cdot k/w'$, where $w' = min(c,w)/max(c,w)$, according to Eq. 22. □

3.4.2. XLS-PP Computation

The computation of *XLS-PP* between two top-k XML lists La and Lb is similar to the procedure sketched in Section 3.2.2, except for a few changes as we will describe. After computing the tree similarity measure between each tree pair across the two lists, we ignore the similarity between pairs that exceed ω and then apply a minimum cost perfect matching algorithm to find $gmin^{P}_{ab}$ with the minimum mapping similarity distance, $MinMSD^{P}(La,Lb)$. Then, we compute the position distance $P^{P}(La,Lb,fmin^{P}_{ab})$ using Eqs. 26 or 27. Finally, *XLS-PP* is computed using Eq. 28 and 29 for Spearman's footrule and Kendall tau respectively. The total complexity of *XLS-PP* computation is $O(k^2 \cdot Cost(TS(Ta_i, Tb_j)) + k^3)$ (which is same as the one for *XLS* or *XLS-P*).

4. Normalization

In this section we discuss how we normalize the XML similarity component and the position components of *XLS-P* and *XLS-PP*, in Sections 3.3 and 3.4 respectively. Normalization of XML similarity component depends on the tree similarity measure (*TS*) employed. In Section 4.1, we discuss the normalization steps when tree edit distance (*TED*) is used as the tree similarity measure. Section 4.2 discusses the normalization of the position component.

4.1. Normalize Tree-Edit Distance based XML Similarity Component

Let $TED(T_1, T_2)$ be the tree edit distance between two rooted, ordered and labeled XML trees –T_1 and T_2. Let $TED_{max}(T_1, T_2)$ be the maximum cost among the costs of all possible sequences of tree-edit operations that transform T_1 to T_2 (notice that the tree edit distance, $TED(T_1, T_2)$ is the minimum cost among the costs of all possible sequences of tree-edit operations). We normalize the tree edit distance by dividing the tree edit distance, $TED(T_1, T_2)$ by $TED_{max}(T_1, T_2)$.This normalized $TED(T_1, T_2)$ is also called Structural Distance in [29, 30]. To calculate $TED_{max}(T_1, T_2)$, we calculate the cost to delete all nodes from T_1 and insert all nodes from T_2. That is, $TED_{max}(T_1, T_2) = size(T_1) \cdot D_p + size(T_2) \cdot I_p$ where D_p and I_p are the delete and insert penalties and $size(T_1)$ is the number of nodes present in tree T_1.

We use unit delete and insert penalties in our experiments. The normalized $TED(T_1, T_2)$ is low when the trees have similar structure and high percentage of matching nodes, and high when the trees have different structure and low percentage of matching nodes (0 [1] is the min [max] value).

4.2. Normalize Position Component

In XLS-P: To normalize the position component in *XLS-P*, we refer to the metrics on permutations presented in [2]. We observe the maximum value of $P^{TK}(La,Lb,f)$ is $k(k-1)/2$, which occurs when La is the reverse of Lb. The maximum value of $P^{TF}(La,Lb,f)$ is $k^2/2$ when k is even and $(k+1)(k-1)/2$ when k is odd. As with Spearman's footrule, the maximum occurs when La is the reverse of *Lb*. Hence, to normalize we divide the metrics by these maximum values.

In XLS-PP: To normalize the position component in *XLS-PP*, we refer to the metrics on top-k lists presented in [2]. In order to normalize the position components of two top-k lists, we divide them by their maximum values which occur when there are no mappings between Lists *La* and *Lb*.

Theorem 4.1. *The maximum value of top-k Spearman's Footrule* $P^{PF(l)}(La,Lb)$ *is* $2k(l-(k+1)/2)$ *where l is the location parameter.*

Proof. Since there are no mappings between the top-k lists, all k elements of each of the list get mapped to location l. Hence,

$$\begin{aligned} PR^{F(l)}_{max}(La,Lb) &= 2(|1-l|+|2-l|+\cdots+|k-l|) \\ &= 2k(l-(k+1)/2) \end{aligned}$$

For a natural choice of $l = k+1$, the maximum value is $k(k+1)$, which we use in our experiments. □

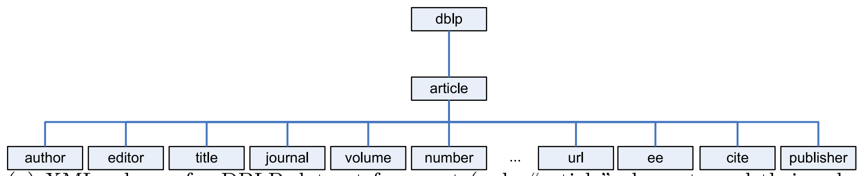
Theorem 4.2. *The maximum value of top-k Kendall tau,* $PR^{k(p)}(La,Lb)$ *is* $pk(k-1)+k^2$ *where p is the penalty parameter.*

Proof. Since there are no mappings between the top-k lists, there are $2k$ distinct elements in $La \cup Lb$. For the unordered pairs within each list, $\bar{K}^{(p)}_{(i,j)}(La,Lb') = p$ since these pairs do not appear in the other list. There are $k(k-1)/2$ such pairs and considering both the lists, there are $k(k-1)$ such pairs, each with penalty p. Hence the total penalty is $pk(k-1)$. For the unordered pairs across each of the two lists, $\bar{K}^{(p)}_{(i,j)}(La,Lb') = 1$ since one element in each pair does not appear in the other list. There is k^2 such pairs, each with penalty 1. Hence the total penalty in this case is k^2. Adding them together, we get the maximum value which is $pk(k-1)+k^2$. □

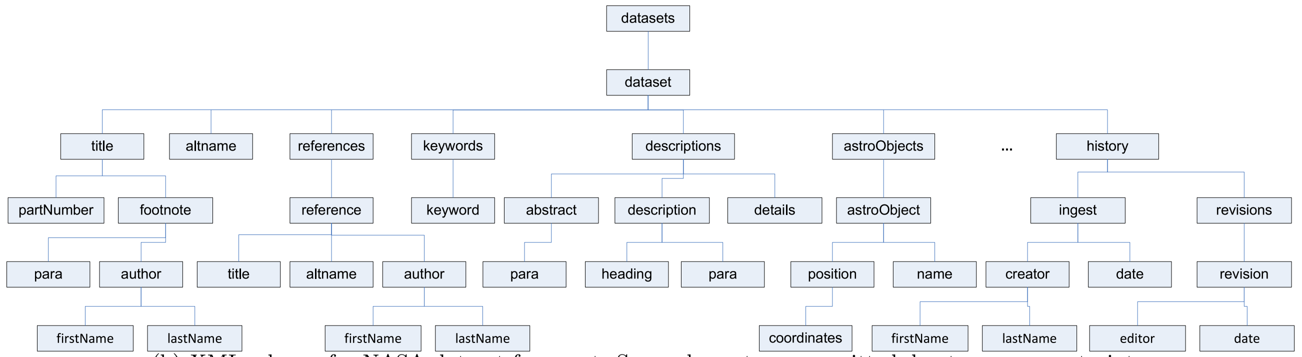
We use $p = 0.5$ in our experiments.

5. Experimental Evaluation

In this section we experimentally evaluate the measures presented in the previous sections by comparing three popular XML keyword search algorithms. We use tree edit distance (*TED*) as the XML tree similarity measure (*TS*). We selected to evaluate our proposed measures on the problem of keyword search in XML databases, which we believe is an important and representative application where systems produce subjective ranked lists of XML trees. Clearly, any other application that returns such lists could also be used for evaluation.



(a) XML schema for DBLP dataset fragment (only “article” elements and their subtrees from original dataset are included)



(b) XML schema for NASA dataset fragment. Some elements were omitted due to space constraints.

Figure 2: XML schemas for DBLP and NASA datasets.

5.1. Datasets & Experimental Setup

Datasets: We use two real datasets: the DBLP Bibliography dataset [31], and the GSFC/NASA XML dataset available at [32]. Figure 2 shows a reduced version of the schemas for both datasets and Table 2 summarizes their characteristics.

Table 2: XML Datasets Used in the Experiments.

Dataset	# Elems.	# Attrs.	Max. Depth	Avg. Depth
DBLP Bibliography	3 332 130	404 276	6	2.90228
GSFC/NASA XML	476 646	56 317	8	5.58314

Experimental Setup: We implemented the following XML keyword proximity search systems: XRANK [5], XSEarch [6] and XKeyword [7]. These three algorithms take as input a corpus of XML documents and a keyword query, and return as output an ordered list of XML fragments that satisfy the query by containing all the keywords. All three algorithms favor minimal and compact sub-trees that satisfy the query, but use different ranking functions and pruning rules. In particular, while XKeyword ranks its answers by the size of the resulting sub-tree, XRANK and XSEARCH also utilize Information Retrieval (IR) score functions based on $tf \cdot idf$. XSEarch prunes result paths that repeat the same tag in internal nodes, while XRANK prunes results if there is a more specific result in the same element. Also, XRANK returns whole sub-trees while XSEarch and XKeyword return paths. We used the IR score provided by the `CONTAINSTABLE` function of Microsoft SQL Server 2000 to compute the IR components of both XRANK and XSEARCH ranking functions.

The experiments were performed on a PC with an Intel Pentium Core 2 Duo, 2.00 GHz processor, 2GB RAM, running Windows Vista Business. All algorithms were developed in Java (JDK version 1.6.0_06), use the Document Object Model (DOM) for XML parsing and navigation, and Microsoft SQLServer 2000 for the persistent storage of indexes. The tree similarity (TS) measure we use in our experiments is the dynamic programming algorithm by Zhang and Shasha [23] which computes the tree-edit-distance between ordered trees [13] whose complexity is $Cost(TED(Ta_i, Tb_j)) = O(|Ta_i||Tb_j| \cdot min(leaves(Ta_i), depth(Ta_i)) \cdot min(leaves(Tb_j), depth(Tb_j))$. We refer to a detailed survey of tree edit distance algorithms [13].

In Section 5.2, we analyze the results of a single query to show the intuition of our evaluation scheme. In Section 5.3 we report average XML Lists Distance values over many experiments on the two datasets. In Section 5.4, we report performance (time) experimental results.

5.2. Analyzing a Single Query

To illustrate our measures, we present an analysis for the keyword query "*database retrieval language*" over the DBLP XML dataset. Figure 3 shows the top-3 search results output by each of the three XML search algorithms. Table 3 presents the *XLS*, *XLS-P* and *XLS-PP* measures between every pair of XML lists from Figure 3. Notice that *XLS* takes values in [0,1] while *XLS-P* and *XLS-P* in [0,2].

Notice that in Table 3 we only present *XLS-PP* measures for $\omega = 0.7$ and 0.9 as *XLS-PP* values for $\omega = 0.5$, 0.3, 0.1 for the top-3 lists in Figure 3 are the same as for $\omega = 0.7$ (we explain why later). Note that we use the penalty constant c equal to ω.

Table 3: XML List distances based on Total and Partial mappings for top-k lists in Figure 3.

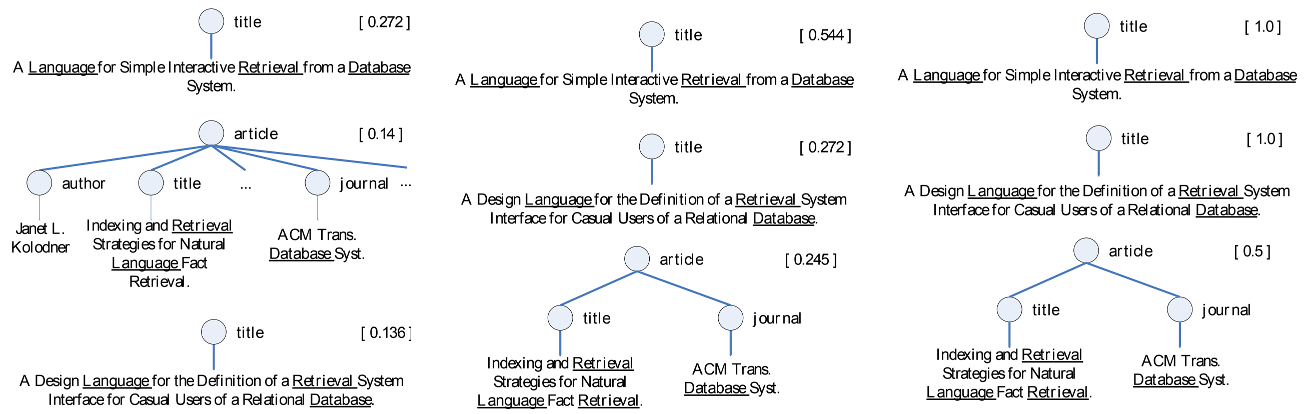
		XRANK	XSEARCH	XKEYWORD
XRANK	*XLS*	0.00	0.30	0.30
	XLS-P^F, *XLS-P*^{K}	0.00, 0.00	0.80, 0.30	1.05, 0.46
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.7$)	0.00, 0.00	0.67, 0.50	0.75, 0.54
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.9$)	0.00, 0.00	0.49, 0.41	0.58, 0.45
XSEARCH	*XLS*	0.30	0.00	0.00
	XLS-P^F, *XLS-P*^{K}	0.80, 0.30	0.00, 0.00	0.25, 0.17
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.7$)	0.67, 0.50	0.00, 0.00	0.08, 0.04
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.9$)	0.49, 0.41	0.00, 0.00	0.08, 0.04
XKEYWORD	*XLS*	0.30	0.00	0.00
	XLS-P^F, *XLS-P*^{K}	1.05, 0.46	0.25, 0.17	0.00, 0.00
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.7$)	0.75, 0.54	0.08, 0.04	0.00, 0.00
	XLS-PP^F, *XLS-PP*^{K} ($\omega = 0.9$)	0.58, 0.45	0.08, 0.04	0.00, 0.00

Let La, Lb and Lc be the top-3 lists of XRANK, XSEarch and XKeyword algorithms respectively as shown in Figure 3. The associated tree edit distance values between every pair of XML trees in each of the lists as follows:

$$AB = \begin{array}{c c} & \begin{array}{ccc} Tb_1 & Tb_2 & Tb_3 \end{array} \\ \begin{array}{c} Ta_1 \\ Ta_2 \\ Ta_3 \end{array} & \left[\begin{array}{ccc} 0.00 & 0.50 & 0.71 \\ 0.98 & 0.98 & 0.89 \\ 0.50 & 0.00 & 0.71 \end{array}\right] \end{array} \quad AC = \begin{array}{c c} & \begin{array}{ccc} Tb_1 & Tb_2 & Tb_3 \end{array} \\ \begin{array}{c} Ta_1 \\ Ta_2 \\ Ta_3 \end{array} & \left[\begin{array}{ccc} 0.00 & 0.50 & 0.71 \\ 0.98 & 0.98 & 0.89 \\ 0.50 & 0.00 & 0.71 \end{array}\right] \end{array}$$

$$BC = \begin{array}{c c} & \begin{array}{ccc} Tb_1 & Tb_2 & Tb_3 \end{array} \\ \begin{array}{c} Ta_1 \\ Ta_2 \\ Ta_3 \end{array} & \left[\begin{array}{ccc} 0.00 & 0.50 & 0.71 \\ 0.50 & 0.00 & 0.71 \\ 0.71 & 0.71 & 0.00 \end{array}\right] \end{array}$$

First of all, notice that the top-3 lists of XSEarch and XKeyword are identical (and hence the tree edit distance matrices AB and AC are identical), except that the first two results of XKeyword have the same score. This is the reason that the distances between XSEarch and XKeyword, for total mapping, are small (but not zero) in Table 3, since the position components consider ties. Note that XRANK returns a different sub-tree as its second result, since the XRANK function ranks the total score for this sub-tree higher than the score of the single element that appears in the other two lists. In this sub-tree, the keyword "*Retrieval*" appears twice within the "*title*" element, which increases its IR score. In addition, the third element in the XRANK list was penalized by its length and as a result. Between XRANK and XSEarch, two results are identical, and Ta_2 is mapped to Tb_3 in total mapping, even though they are very different. This irrelevant mapping is removed in the partial mapping measures.


title [0.272]
A Language for Simple Interactive Retrieval from a Database System.
article [0.14]
author
title
...
journal
...
Janet L. Kolodner
Indexing and Retrieval Strategies for Natural Language Fact Retrieval.
ACM Trans. Database Syst.
title [0.136]
A Design Language for the Definition of a Retrieval System Interface for Casual Users of a Relational Database.
title [0.544]
A Language for Simple Interactive Retrieval from a Database System.
title [0.272]
A Design Language for the Definition of a Retrieval System Interface for Casual Users of a Relational Database.
article [0.245]
title
journal
Indexing and Retrieval Strategies for Natural Language Fact Retrieval.
ACM Trans. Database Syst.
title [1.0]
A Language for Simple Interactive Retrieval from a Database System.
title [1.0]
A Design Language for the Definition of a Retrieval System Interface for Casual Users of a Relational Database.
article [0.5]
title
journal
Indexing and Retrieval Strategies for Natural Language Fact Retrieval.
ACM Trans. Database Syst.

(a) XRANK Top-3 (b) XSEARCH Top-3 (c) XKEYWORD Top-3

Figure 3: Top-3 search results for query *"database retrieval language"* over DBLP dataset in three different keyword proximity search engines.

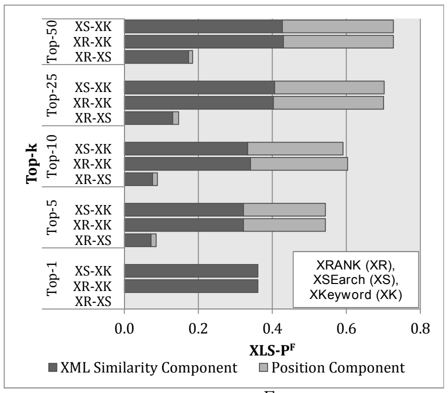
We computed *XLS-PP* for various thresholds, $\omega = 0.9$, 0.7, 0.5, 0.3 and 0.1, and found that *XLS-PP* distance values are identical for thresholds 0.7, 0.5, 0.3 and 0.1. This is because we have at least two identical trees between every pair of list (three identical trees in case of XSEarch and XKeyword) and they always get mapped between them, while the third unmapped result gets mapped with the unmapped result in the other list depending on the threshold. Between XRANK & XSEarch, and XRANK & XKeyword, the tree edit distance of this third mapping is 0.89 and hence the results are identical for thresholds 0.1, 0.3, 0.5, 0.7.

5.3. Quantitative Results over Multiple Queries

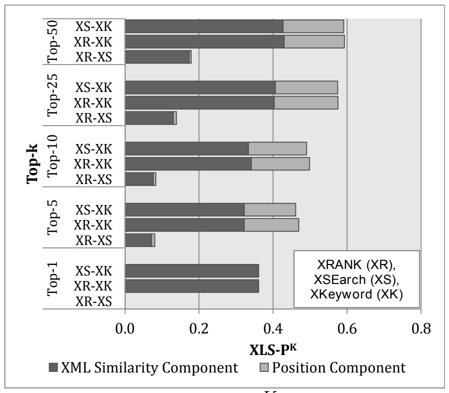
Figures 4(a) and 4(b) show the total similarity distances (split into the two components) between the result lists produced by the three search algorithms on the DBLP dataset averaged over 50 two-keyword queries, using *XLS-P*^{F} and *XLS-P*^{K}, respectively. Notice that the XML Similarity Distance component in *XLS-P* is equal to *XLS*. The queries used include: "*artificial intelligence*", "*xml indexing*", "*text mining*", "*image retrieval*", "*OLAP mining*". Notice that the distance increases as k increases because as the trees get larger, the results become more disparate due to the pruning rules of the algorithms that go in effect for larger trees. As mentioned before, XKeyword ranks its answers by the size of the resulting sub-tree, while XRANK and XSEARCH also utilize Information Retrieval (IR) score functions based on $tf \cdot idf$. The reason that XKeyword has large distance to the other two rankings is that it does not have an IR component in its ranking function. Hence, when multiple trees have the same size, they are ranked arbitrarily. XRANK and XSEarch have smaller distance between them because their rankings are more similar given that the results were mostly single-node trees.

Figures 4(c) and 4(d) show the distances between the results of the three search algorithms for the DBLP dataset, averaged over 50 two-keyword queries using *XLS-PP*^{F} and *XLS-PP*^{K}, for varying thresholds, for top-50 results respectively. Recall that for $\omega = 1.0$, *XLS-PP* (partial mapping) reduces to *XLS-P* (total mapping). Notice that for $\omega = 1.0$, *XLS-PP* will have the same XML Similarity component as *XLS-P* although the position component would be different. We use the penalty constant c equal to ω. In Figures 4(c) and 4(d) we see that the normalized distances increase as ω decreases. The reason is that for small ω there are few matches which lead to large position distance components. Note that for XRANK-vs.-XKeyword and XSEarch-vs.-XKeyword, for $\omega = 0.9$ we get slightly smaller distances than total mapping ($\omega = 1.0$). The reason is that almost all tree pairs in the top-50 results of these rankings have normalized tree edit distance up to 0.7, while for $\omega = 1.0$, we divide by a larger number (than for $\omega = 0.9$) to normalize the XML similarity component. On the other hand, for XRANK-vs.-XSEarch, the distance keeps reducing as ω increases from 0.1 to 0.9 and this is because there are some tree pairs in the top-50 results of these rankings with normalized tree edit distance greater than 0.9.

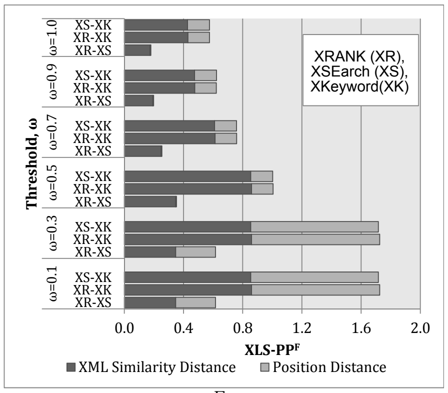
Figure 5 repeats the set of experiments of Figure 4 on the NASA dataset. Some sample two-keyword queries used in these experiments are: *"arcmin-*



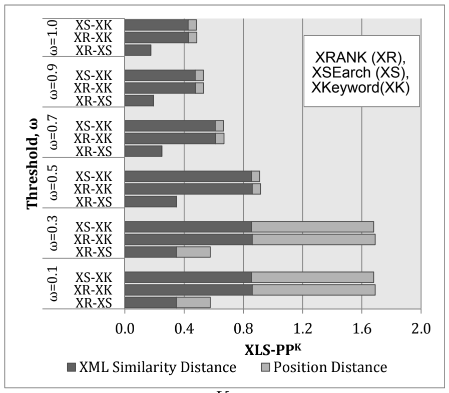
(a) Average $XLS\text{-}P^F$ vs. Top-k



(b) Average $XLS\text{-}P^K$ vs. Top-k

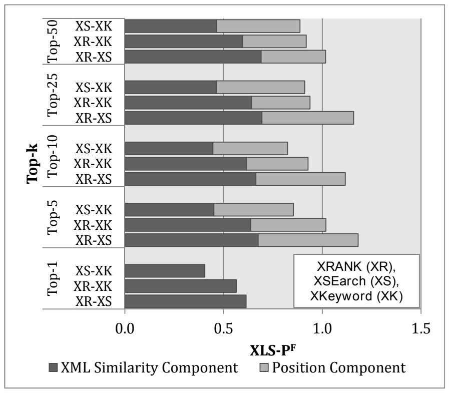


(c) Average $XLS\text{-}PP^F$ vs. threshold, ω, $k = 50$

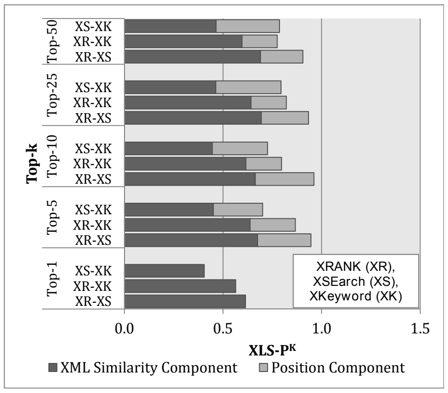


(d) Average $XLS\text{-}PP^K$ vs. threshold, ω, $k = 50$

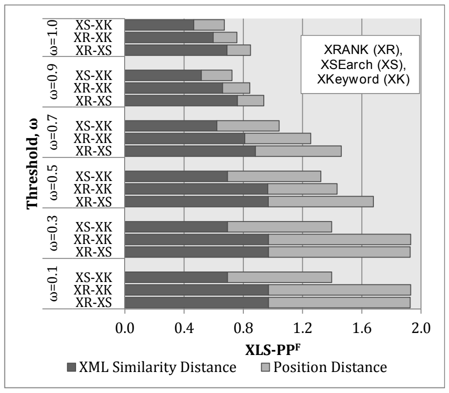
Figure 4: Experiments on DBLP Dataset.



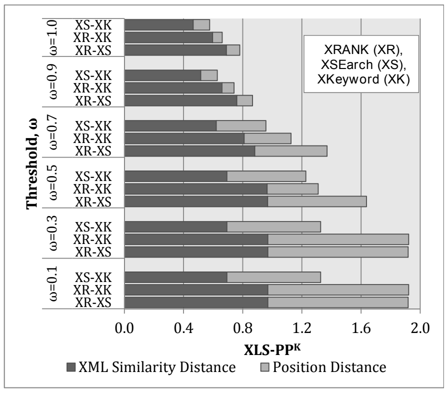
(a) Average *XLS-P*^{F} vs. Top-k



(b) Average *XLS-P*^{K} vs. Top-k



(c) Average *XLS-PP*^{F} vs. threshold, ω, $k =$ 50



(d) Average *XLS-PP*^{K} vs. threshold, ω, $k =$ 50

Figure 5: Experiments on NASA Dataset.

utes magnitude", "astrographic motion", "equinox culmination", "photo-graphic wavelengths", "oxford zone". Some important observations on the results of NASA dataset are (a) Distance between XML lists is generally larger for NASA dataset because of its larger depth. (b) In contrast to Figure 4, XSEarch and XKeyword have the smallest distance because both algorithms return paths as result. This factor was less important in Figure 4 because most results were single-node. In contrast, XRANK has large distance to the other two rankings because it returns whole sub-tree as result. (c) XRANK is very close to XSEarch in DBLP, but very far in NASA dataset. The reason is that the XRANK and XSEarch pruning conditions rarely occur in very shallow sub-trees (DBLP) but appear more frequently in deeper sub-trees (NASA dataset). The latter also leads to unpredictable fluctuations to the distances for increasing k (Figure 5), in contrast to the linear increase in the DBLP dataset (Figure 4).

Discussion: An important conclusion from all these experiments is that the relative lists distance does not depend much on which of the three variants we use when k is up to 50, because the XML similarty component is dominant. Hence, if k is reasonably small, one can safely pick XLS if the metric property is important for that application.

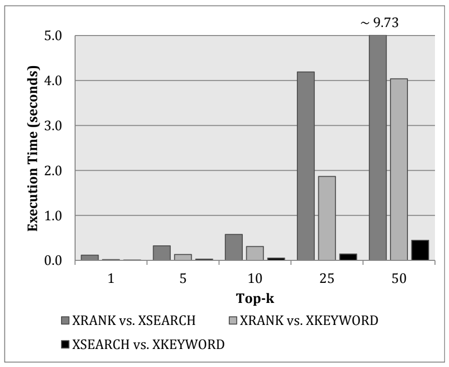
5.4. Performance Results

Due to space constraints and negligible execution times for the DBLP dataset (always less than one second), we only present results on the deeper NASA dataset. Figure 6(a) shows the average execution time to compute *XLS-P* for various values of k, over the same 50 two-keyword queries used in the distance experiments. As expected, the average execution time increases superlinearly as k increases because there are more results in the top-k lists under comparison. Figure 6(b) shows the average execution time to compute *XLS-PP* for various values of the threshold ω, for fixed $k = 50$. Notice that the execution times are different for the three pairs of search algorithms. The reason is that XRANK produces the largest size of results as it returns whole XML elements, while XKeyword produces concise results by returning paths. XSEarch produces results of intermediate size by returning paths like XKeyword but has different pruning rules. Thus, the execution times of XRANK vs. XSEarch are the highest, while XSEarch vs. XKeyword is the lowest.

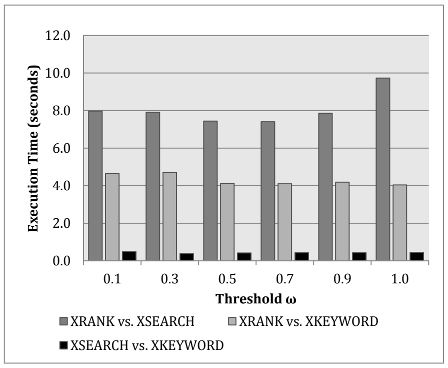
6. Other Related Work

Much of the related work was presented in Section 2.

XML Retrieval Evaluation: The INitiative for the Evaluation of XML Retrieval (INEX) [33] has provided since 2002 the infrastructure and means for evaluating the effectiveness of content-oriented XML search systems. Our work can benefit this initiative of INEX by providing appropriate evaluation measures for lists of XML fragments. Clarke [34] and Kazai et al. [35] present techniques to incorporate the overlap between XML fragments when evaluating XML search



(a) Avg. execution time to compute *XLS-P* vs. Top-k ($\omega = 1.0$)



(b) Avg. execution time to compute *XLS-PP* vs. ω (k=50)

Figure 6: Performance Experiments on NASA Dataset.

algorithms. They are complementary to our work since their techniques can be applied on our measures to account for overlap between the XML results.

Matching in Relational Databases: Guha et al. [36] address the problem of merging approximate attribute rankings produced by executing a query on a "dirty" relational database. To do so, they propose a modification to the Hungarian Algorithm to identify a set of top ranking results. In our case, the top-k lists are fairly small and hence memory-based matching techniques like the Hungarian algorithm are more appropriate.

7. Conclusions

We have presented, to the best of our knowledge, the first suite of distance measures for ranked lists of sub-trees. We also showed under what conditions these measures are metrics. To evaluate our distance measures on real test beds, we implemented three popular XML keyword proximity search systems and compared their results using our novel distance measures.

In the future, we plan to work on customizing these measures for different problems. For instance, for the XML keyword search problem we will differentiate between path-as-result and sub-tree-as-result cases. We will also consider implementing metrics that would consider a multi-way mapping between XML trees instead of just considering a binary mapping, as an XML tree could possibly be mapped to multiple smaller sub-trees and vice versa.

Acknowledgments

This research was partially supported by NSF Awards IIS-0952347, IIS-0811922, CNS-1126619 and a Google Research Award.

References

[1] R. Fagin, R. Kumar, M. Mahdian, D. Sivakumar, E. Vee, Comparing and aggregating rankings with ties, in: Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, ACM, 2004, pp. 47–58.

[2] R. Fagin, R. Kumar, D. Sivakumar, Comparing Top k Lists, SIAM Journal on Discrete Mathematics 17 (2003) 134.

[3] M. Kendall, J. D. Gibbons, Rank Correlation Methods, 5th Edition, A Charles Griffin Title, 1990.

[4] P. Diaconis, Group Representation in Probability and Statistics, volume 11 of IMS Lecture Series, Institute of Mathematical Statistics.

[5] L. Guo, F. Shao, C. Botev, J. Shanmugasundaram, XRANK: Ranked keyword search over XML documents, in: Proceedings of the 2003 ACM SIGMOD international conference on Management of data, ACM, 2003, pp. 16–27.

[6] S. Cohen, J. Mamou, Y. Kanza, Y. Sagiv, XSEarch: A semantic search engine for XML, in: Proceedings of the 29th international conference on Very large data bases-Volume 29, VLDB Endowment, 2003, pp. 45–56.

[7] V. Hristidis, Y. Papakonstantinou, A. Balmin, Keyword proximity search on XML graphs, in: Proceedings. 19th International Conference on Data Engineering, 2003., IEEE, 2003, pp. 367–378.

[8] A. Theobald, G. Weikum, The index-based XXL search engine for querying XML data with relevance ranking, in: Advances in Database Technology EDBT, Springer, 2002, pp. 311–340.

[9] N. Fuhr, K. Großjohann, XIRQL: A query language for information retrieval in XML documents, in: Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2001, pp. 172–180.

[10] C. Dwork, R. Kumar, M. Naor, D. Sivakumar, Rank aggregation methods for the web, in: Proceedings of the 10th international conference on World Wide Web, ACM, 2001, pp. 613–622.

[11] P. Yianilos, Data structures and algorithms for nearest neighbor search in general metric spaces, in: Proceedings of the fourth annual ACM-SIAM Symposium on Discrete algorithms, Society for Industrial and Applied Mathematics, 1993, pp. 311–321.

[12] C. Elkan, Using the triangle inequality to accelerate k-means, in: MACHINE LEARNING-INTERNATIONAL WORKSHOP THEN CONFERENCE-, Vol. 20, 2003, p. 147.

[13] P. Bille, A survey on tree edit distance and related problems, Theoretical computer science 337 (1-3) (2005) 217–239.

[14] W. Lian, D. W. Cheung, I. C. Society, N. Mamoulis, S. ming Yiu, An efficient and scalable algorithm for clustering XML documents by structure, IEEE Transactions on Knowledge and Data Engineering 16 (2004) 82–96.

[15] A. Nierman, H. Jagadish, Evaluating structural similarity in XML documents, in: Proc. of the 5th Int. Workshop on the Web and Databases, 2002, pp. 61–66.

[16] S. Flesca, G. Manco, E. Masciari, L. Pontieri, A. Pugliese, Fast detection of XML structural similarity, IEEE Transactions on Knowledge and Data Engineering (2005) 160–175.

[17] S. Helmer, Measuring the structural similarity of semistructured documents using entropy, in: Proceedings of the 33rd international conference on Very large data bases, VLDB Endowment, 2007, pp. 1022–1032.

[18] D. Buttler, A short survey of document structure similarity algorithms, in: International Conference on Internet Computing, Citeseer, 2004, pp. 3–9.

[19] T. Akutsu, D. Fukagawa, A. Takasu, Approximating tree edit distance through string edit distance, Algorithms and Computation (2006) 90–99.

[20] D. Shapira, J. Storer, Edit distance with move operations, in: Combinatorial Pattern Matching, Springer, 2002, pp. 85–98.

[21] K. Tai, The tree-to-tree correction problem, Journal of the ACM (JACM) 26 (3) (1979) 422–433.

[22] R. Yang, P. Kalnis, A. Tung, Similarity evaluation on tree-structured data, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 754–765.

[23] K. Zhang, D. Shasha, Simple fast algorithms for the editing distance between trees and related problems, SIAM J. Comput. 18 (6) (1989) 1245–1262.

[24] T. Jiang, L. Wang, K. Zhang, Alignment of Trees - An Alternative to Tree Edit, in: CPM '94: Proceedings of the 5th Annual Symposium on Combinatorial Pattern Matching, Springer-Verlag, London, UK, 1994, pp. 75–86.

[25] M. Weis, F. Naumann, DogmatiX tracks down duplicates in XML, in: Proceedings of the 2005 ACM SIGMOD international conference on Management of data, ACM, 2005, pp. 431–442.

[26] S. Flesca, E. Masciari, L. Pontieri, A. Pugliese, Detecting Structural Similarities Between XML Documents, in: In Proc. of the 5th Intl. Workshop on the Web and Databases, 2002, pp. 55–60.

[27] J. Munkres, Algorithms for the assignment and transportation problems, Journal of the Society for Industrial and Applied Mathematics (1957) 32–38.

[28] R. Fagin, L. Stockmeyer, Relaxing the triangle inequality in pattern matching, International Journal of Computer Vision 30 (3) (1998) 219–231.

[29] T. Dalamagas, T. Cheng, K. Winkel, T. Sellis, Clustering XML documents by structure, Methods and Applications of Artificial Intelligence (2004) 112–121.

[30] T. Dalamagas, T. Cheng, K. jan Winkel, T. Sellis, A methodology for clustering XML documents by structure, Information Systems 31 (2006) 187–228.

[31] Trier University, DBLP XML records, http://dblp.uni-trier.de/xml/ (Accessed January 23 2012).

[32] University of Washington, XML Data Repository, http://www.cs.washington.edu/research/xmldatasets/www/repository.html (Accessed January 23 2012).

[33] INitiative for the Evaluation of XML Retrieval., http://www.informatik.uni-trier.de/ ley/db/conf/inex/ (Accessed January 23 2012).

[34] C. Clarke, Controlling overlap in content-oriented XML retrieval, in: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2005, pp. 314–321.

[35] G. Kazai, M. Lalmas, A. de Vries, The overlap problem in content-oriented XML retrieval evaluation, in: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval, ACM, 2004, pp. 72–79.

[36] S. Guha, N. Koudas, A. Marathe, D. Srivastava, Merging the results of approximate match operations, in: Proceedings of the Thirtieth international conference on Very large data bases-Volume 30, VLDB Endowment, 2004, pp. 636–647.