K-Join: Knowledge-Aware Similarity Join

(Extended Abstract)

Zeyuan Shang Yaxiao Liu Guoliang Li Jianhua Feng
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China.
zeyuanxy@gmail.com; {liguoliang,fengjh}@tsinghua.edu.cn,liuyx12@mails.tsinghua.edu.cn

Abstract—Similarity join is a fundamental operation in data cleaning and integration. Existing similarity-join methods utilize the string similarity to quantify the relevance but neglect the knowledge behind the data, which plays an important role in understanding the data. Thanks to public knowledge bases, e.g., Freebase and Yago, we have an opportunity to use the knowledge to improve similarity join. To address this problem, we study knowledge-aware similarity join, which, given a knowledge hierarchy and two collections of objects (e.g., documents), finds all knowledge-aware similar object pairs. To the best of our knowledge, this is the first study on knowledge-aware similarity join. There are two main challenges. The first is how to quantify the knowledge-aware similarity. The second is how to efficiently identify the similar pairs. To address these challenges, we first propose a new similarity metric to quantify the knowledge-aware similarity using the knowledge hierarchy. We then devise a filter-and-verification framework to efficiently identify the similar pairs. We propose effective signature-based filtering techniques to prune large numbers of dissimilar pairs and develop efficient verification algorithms to verify the candidates that are not pruned in the filter step.

To summarize, we make the following contributions.

1. We propose a knowledge-aware similarity metric to quantify the knowledge-aware similarity and propose a new similarity metric to compute knowledge-aware similarity. We then devise a filter-and-verification framework to efficiently identify the similar pairs. We devise signature-based filtering techniques to prune large numbers of dissimilar pairs and develop efficient verification algorithms to verify the candidates that are not pruned in the filter step. To summarize, we make the following contributions.

2. We propose a filter-and-verification framework to efficiently identify the similar pairs. The filter step prunes many dissimilar pairs and the verification step verifies the candidate pairs that are not pruned in the filter step.

3. In the filter step, we generate high-quality signatures based on the knowledge hierarchy such that if two objects have no common signatures, they cannot be similar. We utilize these signatures to prune dissimilar pairs.

4. It is rather expensive to directly compute the knowledge-aware similarity and we propose an adaptive framework to verify the candidates. We estimate the upper bounds and lower bounds of candidate pairs. We utilize upper bounds to prune dissimilar pairs and use lower bounds to avoid computing the knowledge-aware similarity.

5. We have conducted an extensive set of experiments on real datasets. Experimental results show that our method significantly outperforms the baseline algorithms in terms of both efficiency and effectiveness.

I. INTRODUCTION

As an important operation in data cleaning and integration, similarity join has attracted significant attention from the database community. It has widespread real applications such as web clustering, duplicate detection, and collaborative filtering [1], [4]. Given two collections of objects, similarity join aims to find all similar pairs from the two collections. There are many functions to quantify the similarity between objects, such as Jaccard, Cosine and edit distance [2], [3]. However these functions only utilize the string similarity to quantify the similarity between objects but neglect the knowledge behind the data, which plays an important role in understanding the data. To address this problem, we propose knowledge-aware similarity join, which, given a knowledge hierarchy and two collections of objects (e.g., POIs), finds all knowledge-aware similar object pairs. Note that our method can facilitate many real-world applications. For example, Yelp wants to classify similar restaurants together to improve restaurant recommendations, and Amazon wants to classify similar products together using the knowledge information.

To the best of our knowledge, this is the first study on knowledge-aware similarity join. There are two main challenges to address this problem. The first is how to quantify the knowledge-aware similarity. The second is how to efficiently identify the similar pairs. To address these challenges, we utilize the knowledge hierarchy to quantify the knowledge-aware similarity and propose a new similarity metric to compute knowledge-aware similarity. We then devise a filter-and-verification framework to efficiently identify the similar pairs. We devise signature-based filtering techniques to prune large numbers of dissimilar pairs and develop efficient verification algorithms to verify the candidates that are not pruned in the filter step. To summarize, we make the following contributions.

1. We propose a knowledge-aware similarity metric to quantify the knowledge-aware similarity and propose a new similarity metric to compute knowledge-aware similarity. We then devise a filter-and-verification framework to efficiently identify the similar pairs.

II. PROBLEM FORMULATION

Knowledge-Aware Similarity For Elements. We model each object (e.g., a POI) as a set of elements (e.g., tokens) by tokenizing the object. We first discuss how to quantify the similarity between elements and then propose a knowledge-aware similarity metric for objects. We model a knowledge hierarchy as a tree structure $T$ and how to support the directed acyclic graph (DAG) structure is discussed in our full paper [6]. Given two elements $e_x$, $e_y$, we first map them to tree nodes in $T$. Here we assume that each element matches a single node and how to support the case that each element matches multiple tree nodes is discussed in our full paper. If the context is clear, we also use $e_x$ and $e_y$ to denote the corresponding matched nodes. Let $\text{LCA}_{e_x,e_y}$ denote their lowest common ancestor (i.e., the common ancestor of the two
nodes and any of its descendant will not be a common ancestor
of the two nodes), and $d_{x,y}$ denote the depth of node $e_x$ (the
depth of the root is 0). Intuitively, the larger $d_{x,y}$ is, the
two elements are more similar.

**Definition 1 (Knowledge-Aware Similarity for Elements).**
Given a knowledge hierarchy $T$ and elements $e_x$ and $e_y$, their
knowledge-aware similarity is $SIM(e_x, e_y) = \frac{d_{x,y}}{\max(d_{x,y}, 1)}$.

An element may map to multiple tree nodes due to (1)
an element may appear in multiple nodes; (2) an element
can have synonyms; and (3) an element may have typos and
may map to multiple tree nodes that approximately match
the element to tolerate typos). Thus we enumerate every
node of an element and compute the maximum similarity,
\[ \phi(e_x, e_y) = \max_{e'_x, e'_y} \frac{d_{e'_x, e'_y}}{\max(d_{e'_x, e'_y}, 1)} \]
where $e'_x$ and $e'_y$ are mapping nodes of $e_x$ and $e_y$ respectively,
and $\phi(e_x, e_y)$ is the similarity between $e_x$ and $e'_y$. If $e_x = e'_y$
and if they are synonyms, $\phi(e_x, e_y) = 1$; otherwise, we utilize
normalized edit distance (edit similarity) to quantify their
similarity, i.e., $\phi(e_x, e'_y) = 1 - \frac{ED(e_x, e'_y)}{\max(|e_x|, |e'_y|)}$, where $ED(e_x, e'_y)$
is the edit distance of $e_x$ and $e'_y$. $|e_x|$ is the length of $e_x$.

**Knowledge-Aware Similarity For Objects** $S_x$ and $S_y$. We
construct a bigraph $G = ((S_x, S_y), E)$, where $E$ is the edge
set. If an element in $S_x$ is similar to an element in $S_y$, there
is an edge between them whose weight is the knowledge-
aware similarity between the two elements. To avoid involving
dissimilar pairs, we remove all the edges whose weights are
smaller than a given threshold $\delta$. To avoid mapping an element
from one object to multiple elements in the other object, we
use the graph matching to compute the similarity. A matching
in a bigraph is a set of edges without common elements,
and the maximum weight matching is the matching with
the maximum edge weight. We use the maximum weight matching
of $G$ as the fuzzy overlap of $S_x$ and $S_y$, denoted by $S_x\cap_r S_y$.

Using the fuzzy overlap, we define knowledge-aware similarity
on two objects. Here we take Jaccard as an example and how
to support other metrics is discussed in our full paper.

**Definition 2 (Knowledge-Aware Similarity for Objects).**
Given a knowledge hierarchy $T$, objects $S_x$ and $S_y$, and
an element similarity threshold $\delta$, the knowledge-aware similarity
of $S_x$ and $S_y$ is $SIM(S_x, S_y) = \frac{|S_x \cap_r S_y|}{|S_x| + |S_y|}$, where $|S_x|$ is the size of $S_x$ and $|S_x \cap_r S_y|$ is the sum of the weights
of edges in the maximum matching.

**Definition 3 (Knowledge-Aware Similarity Join).**
Given a knowledge hierarchy $T$, two object sets $R$ and $S$, an
element similarity threshold $\delta$ and an object similarity threshold $\tau$, a
knowledge-aware similarity join finds all similar pairs $(r, s) \in R \times S$, such that $SIM(r, s) \geq \tau$.

**III. THE K-JOIN FRAMEWORK**

**Framework.** We propose a filter-verification framework. The
filter step generates signatures for each object (if two objects
have no common signatures, they cannot be similar). We take
the objects pairs with common signatures as candidates. The
verification step verifies the candidates. The

**Filtering.** Given a knowledge hierarchy $T$ and an element
similarity threshold $\delta$, for any two similar elements, we can
estimate the minimum depth of their lowest common ancestor
(LCA). Suppose $e_x$ and $e_y$ are two different elements, and their
element similarity is $SIM(e_x, e_y) = \frac{d_{e_x, e_y}}{\max(d_{e_x, e_y}, 1)}$. If $e_x$ and $e_y$ are
similar, we have $d_{e_x, e_y} \geq \frac{1}{1+\delta}$, and if $d_{e_x, e_y} \geq \frac{1}{1+\delta}$. Thus if two
different elements are similar, the depth of their LCA is at least
$\delta = \lceil \frac{1}{1+\delta} \rceil$. For any element $e$ with depth $d_e$, if $d_e < \delta$, we
select $e$ as its node signature, i.e., $g_e = e$. If $d_e \geq \delta$, we choose the ancestor of $e$ whose depth is $d_\delta$,(denoted by $e\delta$) as its signature, i.e., $g_e = e\delta$. Given an object $S$, we
generate its node signature set $G_S = \{g_e | e \in S\}$. Then, we fix a
global order for the node signatures of all the elements. Let
$G_S = G_r[1, |S| - (\tau - 1)]$, which is the subset of $G_S$ with
the first $|S| - (\tau - 1)$ node signatures. We call $G_S$ the node
prefix of node signatures of $S$. Then if $G_S \cap G_{S'} = \phi$, $S_x$ and $S_y$ cannot be similar. Thus we can use $G_S$ to do filtering.

**Verification.** If $\frac{|S_x \cap_r S_y|}{|S_x| + |S_y|} \geq \tau$, $|S_x \cap_r S_y| \geq \tau |S_x|$, and $|S_x \cap_r S_y| \geq |S_y|$, we prune the pair. Otherwise, we compute the real
similarity. If the similarity exceeds the threshold, the pair is
similar; the pair is dissimilar otherwise.

**IV. EXPERIMENTS**

**Datasets.** We used four real-world datasets: Pub, Res, POI
and Tweet. Pub contained 1879 papers and Res contained
864 restaurants. These two datasets had ground truths [8].
We used them to evaluate the effectiveness. POI contained 1
million POIs and Tweet contained 1 millions crawled tweets.
We used them to evaluate efficiency.

**Baseline.** We compared with state-of-the-art methods, approximate
string join FastJoin [7], a synonym based method SyNonyn [5], and a crowdsourcing based method Crowd [8].
Experimental results showed that our method significantly
outperformed baselines in both quality and efficiency.

**ACKNOWLEDGMENT**

This work was supported by 973 Program of China (2015CB358700), NSF of China (61422205, 61373024, 61632016,
61472198), FDCT/116/2013/A3, and FDCT/007/2016/AFJ.

**REFERENCES**

[1] R. J. Bayardo, Y. Ma, and R. Srikant. Scaling up all pairs similarity
[2] D. Deng, G. Li, J. Feng, and W.-S. Li. Top-k string similarity search with
1 keyword search method for unstructured, semi-structured and structured
token matching based string similarity join. In ICDE, pages 458–469,
2011.
transitive relations for crowdsourced joins. In SIGMOD, pages 229–240,
2013.