Similarity Joins as Stronger Metric Operations

Wei Wang
University of New South Wales, Australia
weiw@cse.unsw.edu.au

1 Introduction

Similarity joins between two sets of records return pairs of records whose similarity is no less than a given threshold. More specifically, consider two sets of records, \( R \) and \( S \), a similarity function \( \text{sim}(\cdot, \cdot) \) and a threshold \( t \), a similarity join between \( R \) and \( S \) is defined as \( \{ (r, s) \mid (r, s) \in R \times S, \text{sim}(r, s) \geq t \} \). A similarity join is a generalization of the traditional equality join commonly found in database systems. A variant of the similarity join is to use a distance threshold to replace the similarity threshold. It is generally expected that the similarity threshold is close to the maximum possible value (usually 1.0), and the distance threshold is close to the minimum possible value (usually 0). For example, we may find near-duplicate documents in a document repository using a cosine similarity threshold of 0.9, or we may find pairs of incorrectly spelt queries and their correct versions in a query log with an edit distance threshold of 2.

Similarity joins have many applications. For example, similarity joins are useful to data integration and data cleaning applications. This is mainly because records representing the same real-world object may have different digital representations, due to reasons such as typographical errors and lack of standardization [9]. Similarity joins can be used to detect near-duplicate Web documents [4], to perform collaborative filtering [2] in the Web environment. They also benefit data mining tasks such as clustering [3].

A straightforward method to perform similarity join is to use the nested loop algorithm, i.e., for each record in one dataset, find all the records in the other dataset that satisfy the similarity threshold; the latter operation may be performed efficiently if there is a suitable index for the given similarity metric. However, the weakness of this method is that it misses many valuable optimization opportunities among repeated similarity searches.

In this article, we will briefly summarize the state-of-the-arts in different areas of similarity joins, and then present several future research challenges.
2 Exact Similarity Joins

2.1 Similarity Joins in the Euclidean and Metric Spaces

Similarity joins with Euclidean distance are also known as spatial distance joins. Efficient algorithms for both low and high dimensional spaces have been proposed [15, 11, 17, 12, 6]. The main idea is to partition the objects into different groups according to their spatial locations, which helps to reduce the candidates an objects has to be compared with. However, the efficiency of these methods decreases quickly with the increase of dimensionality, as a consequence of the curse of dimensionality.

The Euclidean space just is a special case of the general metric space. Similarity joins in a general metric space is even harder, as there are only limited means to prune candidates based on the triangular inequality. Existing methods exploit specialized indexes [7, 16], or adopt the divide-and-conquer paradigm without using any index [10].

2.2 Similarity Joins for Specific Similarity/Distance Functions

Recently progress has been made that results in fast algorithms to perform similarity joins for several simple yet widely used similarity or distance functions, including Jaccard, cosine, set overlap, Dice similarities, Hamming distance, and edit distance [5, 1, 2, 14, 23, 22, 21].

The majority of the work is based on the filter-and-refine framework. The basic idea of filtering is that similar objects should share some identical feature values. If the filtering can be performed in an efficient manner, we can generate candidate pairs of records whose similarities are likely to be no less than the given threshold. In the refinement stage, the similarities of these candidate pairs are actually calculated and are compared with the threshold. An effective filter developed recently is the prefix filter [5], and it is later improved and generalized by [2, 23]. This filter enables us to eliminate non-promising candidate pairs by looking into only part of the records.

Another important insight is that similarity joins with above-mentioned similarity or distance functions can be converted into similarity joins with another functions. This means we can focus on optimizing one type of similarity join and it will automatically benefit others [5, 1, 2, 23]. This is especially advantageous to integrating various kinds of similarity join operators into an existing system [5].

2.2.1 Edit Similarity Joins

An edit similarity join is a similarity join with an edit distance threshold. A popular method is to reduce the problem to a set overlap similarity join on the q-gram representation of the input strings. Unlike other conversions between similarity joins with different similarity functions, this conversion introduces false positives, i.e., not all the results obtained by the set overlap similarity join satisfy the edit distance constraint. Additional filters hence can be introduced and further improves the performance, including the length filter, position filter [8], location-based and content-based mismatch filters [22]. Instead of using fixed length q-grams, there are also method
based on variable length q-grams [13], signatures based on domain partitioning [1], and deletion neighborhood [18, 21].

3 Future Directions

Many research efforts have been focused on devising new methods for similarity joins or improving their performance. In addition to this, we list two important research challenges that call for further research.

**Large-scale Similarity Joins** With the rapid increase of data, there is a need to scale the similarity join algorithms to a very large scale. In particular, the prevalent method to perform similarity joins on large datasets is to use approximate methods. The ability to perform the join exactly is important, as it will reveal many more interesting results. It seems promising to adapt and implement similarity join algorithms on the Map-Reduce framework [19, 20].

**Support More Applications** In addition to the existing applications, it is important to consider using similarity joins in supporting more applications. The key issue is to support application-specific similarity/distance functions, e.g., the Earth mover’s distance in multimedia applications, and bio-sequence scoring functions with various scoring matrices.

References


