String Similarity Search and Join : A Survey

written by M. Yu, G. Li, D. Deng and J. Feng
What is ER?

Table 1: A table of products.

<table>
<thead>
<tr>
<th>ID</th>
<th>Product Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>iPad Two 16GB WiFi White</td>
<td>$490</td>
</tr>
<tr>
<td>$r_2$</td>
<td>iPad 2nd generation 16GB WiFi White</td>
<td>$469</td>
</tr>
<tr>
<td>$r_3$</td>
<td>iPhone 4th generation White 16GB</td>
<td>$545</td>
</tr>
<tr>
<td>$r_4$</td>
<td>Apple iPhone 4 16GB White</td>
<td>$520</td>
</tr>
<tr>
<td>$r_5$</td>
<td>Apple iPhone 3rd generation Black 16GB</td>
<td>$375</td>
</tr>
<tr>
<td>$r_6$</td>
<td>iPhone 4 32GB White</td>
<td>$599</td>
</tr>
<tr>
<td>$r_7$</td>
<td>Apple iPad2 16GB WiFi White</td>
<td>$499</td>
</tr>
<tr>
<td>$r_8$</td>
<td>Apple iPod shuffle 2GB Blue</td>
<td>$49</td>
</tr>
<tr>
<td>$r_9$</td>
<td>Apple iPod shuffle USB Cable</td>
<td>$19</td>
</tr>
</tbody>
</table>
Challenges

There are several challenges in string similarity join and search.

1) How to quantify the similarity between two objects.

2) How to efficiently support similarity join and search and achieve high performance and scalability.
Threshold-based Similarity Join

**String Similarity Join.** Given two sets of strings R and S, a similarity function Sim and a threshold $\tau$, string similarity join is to find a set of string pairs whose similarity scores are not smaller than $\tau$, i.e., \{$(r, s) \mid r \in R, s \in S \text{ and } Sim(r, s) \geq \tau$\}.

In practice, however, it is rather hard to get an appropriate threshold, because a large threshold returns a larger numbers of dissimilar results and a small threshold returns few and even empty result. To address this problem, the top-$k$ similarity search and join are proposed.

**Top-k Similarity Join.** Given two string sets S and R and a similarity function Sim and an integer $k$, the top-$k$ similarity join problem aims to find the $k$ most similar string pairs form S and R which have the highest similarity score based on Sim, i.e., $A \subseteq S \ast R$, and $\forall (s, r) \in A, (s', r') \in S \ast R - A$, $Sim(s, r) \leq Sim(s', r')$. 
Similarity functions

• 1. Token-based similarity functions
• 2. Character-based similarity functions
• 3. Hybrid similarity functions.
Token-based similarity functions

The Overlap similarity (OLP) takes the size of the overlap of their token sets as their similarity, i.e., \( OLP(r, s) = |r \cap s| \).

- Jaccard Similarity: \( JAC(r, s) = \frac{|r \cap s|}{|r \cup s|} = \frac{|r \cap s|}{|r| + |s| - |r \cap s|} \)
- Cosine Similarity: \( COS(r, s) = \frac{|r \cap s|}{\sqrt{|r| \cdot |s|}} \)
- Dice Similarity: \( DICE(r, s) = \frac{2|r \cap s|}{|r| + |s|} \)

For example, given two strings \( r = \{\text{frontier, computer, science}\} \) and \( s = \{\text{computer, science}\} \), their similarity scores based on above similarity functions are

\[ OLP(r, s) = 2 \]
\[ JAC(r, s) = 2/3 \]
\[ COS(r, s) = 2/\sqrt{6} \]
\[ DICE(r, s) = 4/5 \]
Character-based Similarity

The character-based similarity takes each string as a sequence of characters.

It measures the similarity by counting the number of different characters in these two sequences.

The most representative character-based function is Edit Distance (ED). Other similarity functions are also used, such as Hamming distance.
# Edit Distance

Given two words word1 and word2, find the minimum number of steps required to convert word1 to word2. (each operation is counted as 1 step.)

You have the following 3 operations permitted on a word:

- a) Insert a character
- b) Delete a character
- c) Replace a character

Distance Mesurement:

1. \(d[0, j] = j\);
2. \(d[i, 0] = i\);
3. \(d[i, j] = d[i-1, j - 1] \text{ if } A[i] == B[j]\)
4. \(d[i, j] = \min(d[i-1, j - 1], d[i, j - 1], d[i-1, j]) + 1 \text{ if } A[i] != B[j]\)

\(EDS (r, s) = 1 - ED(r,s)/\text{Max}(|r|,|s|)\).
transfer A(kitten) to B(sitting):

sitten (k→s) replacement
sittin (e→i) insertion
sitting (→g) insertion

So \( ED(A, B) = 3 \)
\[ EDS(A, B) = 1 - \frac{3}{7} = \frac{4}{7} \]

The bigger the EDS is, the two strings are more similar.
Hamming distance

Hamming distance counts the number of mismatched characters in every position of two strings.

For example, the hamming distance of “karolin" and “kathrin" is 3 because they have 3 different characters in positions 3, 4, 5.

Hamming distance is widely used in telecommunication to estimate errors by counting the number of flipped bit in a fixed-length binary word.
Jaro distance

The Jaro distance $d_j$ of two given strings $s_1$ and $s_2$ is

$$d_j = \begin{cases} 
0 & \text{if } m = 0 \\
\frac{1}{3} \left( \frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m} \right) & \text{otherwise}
\end{cases}$$

Where:

- $|s_i|$ is the length of the string $s_i$;
- $m$ is the number of matching characters (see below);
- $t$ is half the number of transpositions (see below).

Two characters from $s_1$ and $s_2$ respectively, are considered matching only if they are the same and not farther than $$\left[ \frac{\max(|s_1|, |s_2|)}{2} \right] - 1.$$
Jaro–Winkler distance

Jaro–Winkler distance uses a prefix scale $p$ which gives more favourable ratings to strings that match from the beginning for a set prefix length $\ell$. Given two strings $s_1$ and $s_2$, their Jaro–Winkler distance $d_w$ is:

$$d_w = d_j + (\ell p \cdot (1 - d_j)),$$

where:

- $d_j$ is the Jaro distance for strings $s_1$ and $s_2$
- $\ell$ is the length of common prefix at the start of the string up to a maximum of four characters
- $p$ is a constant scaling factor for how much the score is adjusted upwards for having common prefixes. $p$ should not exceed 0.25, otherwise the distance can become larger than 1. The standard value for this constant in Winkler's work is $p = 0.1$
Example

*MARTHA* and *MARHTA*

Jaro Distance between them:

\[
d_j = \frac{1}{3} \left( \frac{6}{6} + \frac{6}{6} + \frac{6 - 1}{6} \right) = 0.944
\]

Jaro-Winkler Distance:

\[
d_w = 0.944 + (3 \times 0.1(1 - 0.944)) = 0.961
\]
Hybrid-based Similarity

- Fuzzy Overlap: $OLP(r, s) = |r \cap s|$
- Fuzzy Jaccard: $JAC(r, s) = \frac{|r \cap s|}{|r + s| - |r \cap s|}$
- Fuzzy Cosine: $COS(r, s) = \frac{|r \cap s|}{\sqrt{|r| \cdot |s|}}$
- Fuzzy Dice: $DICE(r, s) = \frac{2|r \cap s|}{|r| + |s|}$
The End

- Thanks very much.