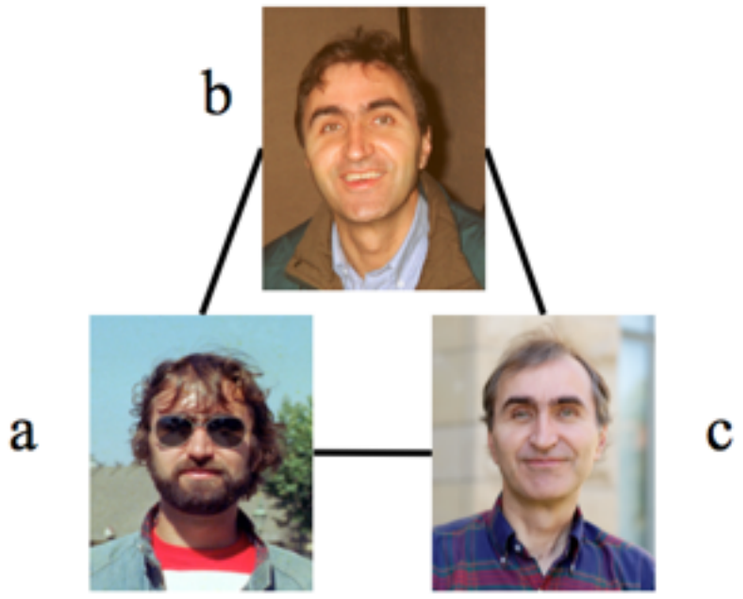
String Similarity Search and Join : A Survey

written by M. Yu, G. Li, D. Deng and J. Feng

What is ER?

Table 1: A table of products.

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



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* τ*, string similarity join is to find a set of string pairs whose similarity scores are not smaller than* τ*, i.e.,* $\{(r, s)|r \in R, s \in S \text{ and } Sim(r, s) \geq \tau\}$.

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,* $i.e., A \subseteq S * R$, *and* $\forall (s,r) \in A, (s',r') \in S * R - A$, $Sim(s,r) \leq Sim(s',r')$.

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 $r = \{frontier, computer, science\}$ and $s = \{computer, science\}$, 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] if A[i] == B[j]
4. d[i, j] = min(d[i-1, j - 1], d[i, j - 1], d[i-1, j]) + 1   if A[i] != B[j]
```

$$EDS(r, s) = 1 - \mathrm{ED}(r,s)/\mathrm{Max}(|r|,|s|).$$

Edit Distance Example

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)=1-3/7=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 $\left\lfloor \frac{\max(|s_1|, |s_2|)}{2} \right\rfloor - 1$.

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 ℓ. Given two strings s_1 and s_2, their Jaro–Winkler distance d_w is:

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

where:

- d_j is the Jaro distance for strings s_1 and s_2
- ℓ 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:

$$d_w = 0.944 + (3 * 0.1(1 - 0.944)) = 0.961$$

Hybrid-based Similarity

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

The End

- Thanks very much.