

Identity Mapping



Dejan Sarka



#538 | SOFIA 2016

Sponsors



Gold sponsors:



In partnership with



Silver sponsors:



Bronze sponsors:



#538 | SOFIA 2016

Introduction

- Dejan Sarka
 - dsarka@solidq.com, dsarka@siol.net, @DejanSarka
 - MCT, SQL Server MVP
 - 30 years of data modeling, data mining and data quality
- 14 books
- 10+ courses



3

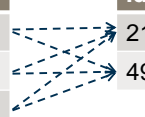
Overview

- The problem
- DQS matching
- SSIS fuzzy transformations
- MDS functions and a custom algorithm

4

The Problem

Id1	FullName1	Id2	FullName2
17	Ken Sanchez	21	Kren Canpez
55	John Campbell	49	Riane Glimpbell
21	Diane Glimp		



A x B

Id1	FullName1	Id2	FullName2	Similarity
17	Ken Sanchez	21	Kren Canpez	0.778
17	Ken Sanchez	49	Riane Glimpbell	0.056
55	John Campbell	21	Kren Canpez	0.211
55	John Campbell	49	Riane Glimpbell	0.445
21	Diane Glimp	21	Kren Canpez	0.138
21	Diane Glimp	49	Riane Glimpbell	0.664

5

No Common Identifier

- Multiple updatable sources of master data (e.g., customer data) exist
- There is no common key for simple joins
- Data merging has to be done based on string similarity
 - Name, address, e-mail, city, etc.
- Which similarity algorithm to use?
- Identity mapping and de-duplicating actually represent a single problem

6

Trade-Offs

- How much data do you want to match automatically?
 - The more data you match automatically, the more errors you can expect
- How many errors can you afford with the automatic algorithm?
 - Impossible to get 100% correct matches automatically
 - For ~100% accuracy, data stewards must do final matching manually

7

Performance Problem

- For approximate merging, any row from one table can be joined to any row from another table
 - Cross join?
 - Even on small data sets, cross joins can cause performance problems
- Need to map data in smaller chunks
 - Partitioning
 - Sorting neighborhood
 - Pruning

8

Authoritative Source

- Can have an authoritative source
 - This could be the MDS source
 - Update other sources to values from the master source
- No master data defined
 - All sources have equal priority
- How do you find the canonical record?
 - You do not have enough data to learn

9

Iterative Approach

- Do matching in iterations
 - Start with exact matches (inner join)
 - Then merge similar strings
 - Then merge less similar strings
 - ...
 - Then merge manually
- Solution depends on the tool
 - DQS: Do the iterations manually
 - SSIS: Base steps on a similarity threshold
- Cleanse the data as much as possible before matching

10

DQS Matching

- DQS matching tokenizes strings to substrings of length n
 - Tokens are called *NGrams*
- DQS matching then checks to see how many NGrams two strings have in common
 - Divides the number of shared nGrams with the number of all possible nGrams to get the similarity coefficient
- You need to prepare a matching KB
 - Matching rules define domains for matching
 - Composite domains can be very useful

11

Building a Matching KB

- Define importance of a domain
 - Weight
 - Request exact match for a domain
 - Degree of similarity
 - Prerequisite for matching
- Use sample data for a KB
 - Test how different matching rules perform
 - Normalize strings and define rules for dates and numbers
 - Define minimum score for matching

12

DQS Matching Project

- Run on the same dataset used when building a KB or a different dataset
- Computer-assisted phase – DQS groups similar records in clusters
- Interactive phase – define the survivorship rule
 - Pivot record
 - Most complete and longest record
 - Most complete record
 - Longest record
- Export results

13

SSIS Fuzzy Transformations

- Fuzzy Lookup and Fuzzy Grouping use a custom, domain- (language-) independent distance function
 - Takes into account the Jaccard and Levenshtein (edit) distance, the number of common tokens (NGrams), token order, and relative frequencies
 - Not as easily misled by transpositions and can detect patterns of a higher level than an approach that uses only edit distance

14

Fuzzy Transformations Features

- Fuzzy Lookup and Fuzzy Grouping can be parameterized to run in loops
 - Similarity threshold
 - Token delimiters
 - For Fuzzy Lookup, can also specify the maximum number of matches to return per input row
- Store index to speed up further matching
- Fuzzy Grouping also selects the *canonical row*

15

Added Columns

- Fuzzy Lookup adds two columns to the SSIS buffer
 - `_Similarity`
 - `_Confidence`
- Fuzzy Grouping adds multiple columns, including
 - `_key_in`
 - `_key_out`
 - `_score`
 - Columns for clean values from the canonical row

16

MDS Similarity Algorithms

- Master Data Services implements some known similarity algorithms
 - Levenshtein distance (also known as edit distance)
 - Jaccard index
 - Jaro-Winkler distance
 - Simil algorithm (also known as Ratcliff/Obershelp)
 - NGrams
- Implemented in `mdq.NGrams` and `mdq.Similarity CLR` functions

17

Levenshtein Distance

- Levenshtein (edit) distance measures the minimum number of edits needed to transform one string into another
 - For example, the distance between kitten and sitting is 3
 - **k**itten → **s**itten (substitution of 's' for 'k')
 - **s**itten → sittin (substitution of 'i' for 'e')
 - sittin → sitt**ing** (insert 'g' at the end)
- Similarity is normalized between 0 and 1

18

Jaccard Index

- The Jaccard index (similarity coefficient) measures similarity between sample sets
- The size of the intersection is divided by the size of the union of the sample sets

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|}$$

19

Jaro Distance

- Jaro distance combines matches and transpositions

$$d_j = \frac{1}{3} \left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m - t}{m} \right)$$

- m is the number of *matching characters*
- t is the number of *transpositions*
- Characters are matching if no farther than

$$\left(\frac{\max(|s_1|, |s_2|)}{2} \right) - 1$$

20

Jaro-Winkler Distance

- Jaro-Winkler distance uses a prefix scale p which gives more favorable ratings to strings with matching beginnings

$$d_w = d_j + (l_p * (1 - d_j))$$

- d_j is the Jaro distance
- l_p is the length of the common prefix at the start of the string up to a maximum of 4 characters
- p is a scaling factor for common prefixes
 - p should not exceed 0.25; otherwise the distance can become larger than 1 (usually p is equal to 0.1)

21

Simil Algorithm

- The Simil algorithm searches for the longest common substring of two strings
 - Then searches for next longest common substring in the remainders from left and right
 - Continues recursively until no more common substrings are found
- Calculates coefficient as a value between 0 and 1 by dividing the sum of the lengths of the substrings by the lengths of the strings themselves

22

Combination of Algorithms

- To improve results, combine two or more algorithms
- Example combination: Jaro-Winkler Simil Normalized (JWSN) coefficient **q** (similarity threshold)
- One of the parameters

$$JWSN = \frac{JW * S}{JW * S + (1 - JW) * (1 - S)}$$

23

Smart Pre-Selection

- Tokenize strings to substrings of length **n**
 - Tokens are called NGrams
 - MDS provides the mdq.NGrams function
- Calculate overall token frequency
- Compare strings that have at least **m** common NGrams with a frequency less than **p**
- Can tweak m, n, p, and q in a loop
 - Start with stricter matching

24

Comparing Solutions

- DQS matching
 - Simple, suitable for smaller datasets
 - Improves with KB improvement
- SSIS Fuzzy Lookup
 - Gives very accurate results
 - Might not match enough data
- Custom algorithm
 - Can be tweaked the most
 - Can get maximum number of matches

25



Course: Accelerated SQL Server 2012-2016 Integration Services

November 7, 2016 – November 10, 2016 - SOFIA

Course: SQL Server 2016 for Business Intelligence Experts

January 23, 2017– January 25, 2017 - SOFIA

SQL Saturday Ljubljana, December 10th, 2016!



Sponsors

Gold sponsors:



In partnership with



Silver sponsors:



Bronze sponsors:



#538 | SOFIA 2016