Fast Record Linkage using Suffix Arrays

Author: Timothy de Vries, tide2817@mail.usyd.edu.au
Supervisor: Dr Sanjay Chawla
School of Information Technologies

1. Aims of the Project

- To avoid the completely infeasible quadratic O(n^2) computational requirements of full scale record linkage
- To avoid the loss of true matches when optimising record linkage to discard sets of records that are dissimilar
- Using arrays of suffixes that have been grouped to efficiently index records

2. Introduction

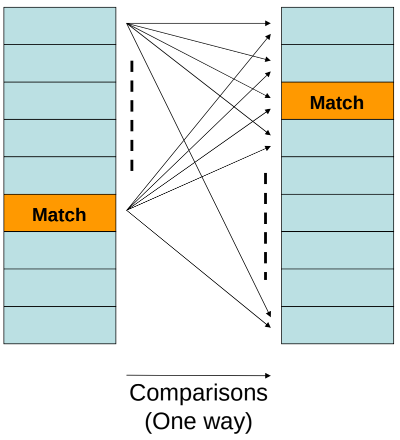
Data is being collected and stored all the time, usually resulting in many separate databases, all with their own data format, that nevertheless deal with the same kind of data. As there is no primary key structure to link these databases, approximate techniques must be used instead.

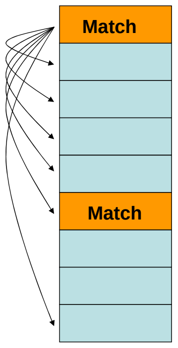
Linking records across these separate databases usually allows for much more extensive and accurate analysis to be carried out. For example: a large company has taken over smaller companies, each of which use their own customer record. Linking customers across the business allows for much more information to be gathered.

Uses of this extra information include improving products to cater to specific customer needs, analysing regional business performance, and predicting behaviour (for insurance purposes), among many others.

3. The Record Linkage Problem

Given two databases, each with n records, the unoptimised matching process requires n^2 comparisons.



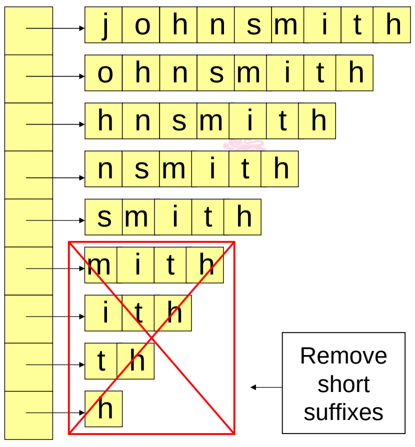


Record Linkage may also be used to find duplicates within the same data set. 'Dirty' data is a common situation. This step is also of n^2 complexity in the number of rows of data.

4. Suffix Arrays for Indexing

First we create a key string by selecting key information from the records and merging it together. Suffix arrays are created from this. Here is an example:

"JohnSmith"



Creating the index:

Add the remaining suffixes for each record to a **sorted index**. E.g:

Original records:

1	John Smith, born 1 Jan 1980
2	Peter Jones, born 1 Jan 1970
3	Jon Smit, born 1 Jan 1980

Sorted **inverted index** (partly):

erjones	2
eterjones	2
hnsmith	1
jonsmit	3
nsmit	3
nsmith	1
ohnsmith	1
onsmit	3
rjones	2

Compare for similarity among adjacent suffixes

Highly similar

Record identifier

Concatenated key fields

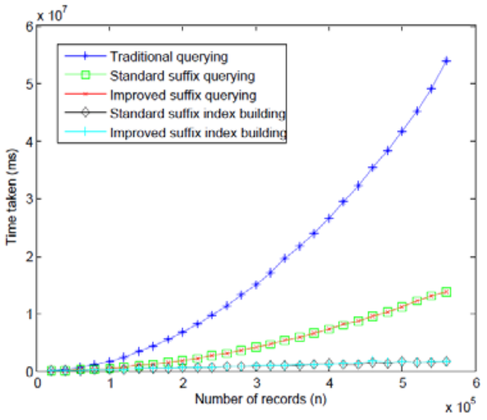
5. Grouping Similar Suffixes

We can group together suffixes that are similar in some **section** of their key value string. Since the overall method takes many suffixes into account, it is very robust to errors and typos in any area of the key value string.

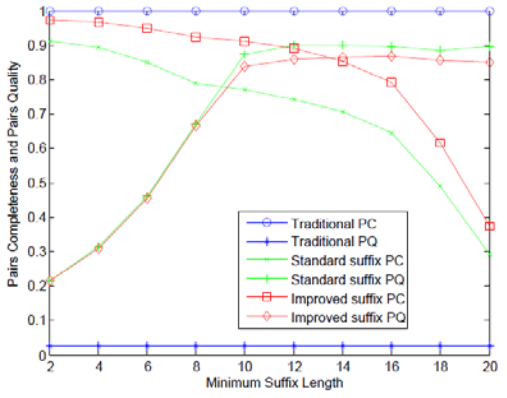
Querying the index structure is then as easy as creating suffixes from the query key string, and querying the inverted index structure to gather a list of record numbers.

6. Results from real data

- Very large scale (~90 million) record linkage problem in an insurance company.
- Scalability is paramount:



- Always a tradeoff between accuracy and performance
- Accuracy is important too:



- PC = Pairs Completeness, the proportion of true matches found out of all true matches (accuracy)
- PQ = Pairs Quality, the proportion of similar records found that are true matches (efficiency)

- Timothy de Vries, Hui Ke, Sanjay Chawla and Peter Christen. Robust Record Linkage Blocking using Suffix Arrays. In Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09), Hong Kong, China.

This research is sponsored by the Capital Markets Cooperative Research Centre.

The University of Sydney