Australian National University

COMP3430 / COMP8430
Data wrangling

Lecture 14: Blocking / indexing (2)
(Lecturer: Peter Christen)

Lecture outline

- Improving traditional blocking
- Phonetic encoding techniques
- Alternative blocking techniques

Improving traditional blocking

- Problems with traditional blocking:
  - An erroneous value in a blocking key variable results in a record being inserted into the wrong block
  - Attributes that are know to change will mean true matching record pairs are potentially missed (for example, *surname* or *postcode*)
  - Missing values mean BKV cannot be generated
  - Frequency distribution of a BKV influences the number of record pairs generated
- Some of these problems can be overcome by careful selection of the attributes to be used as blocking keys
- Others can be addressed by using *phonetic encoding*

Phonetic encoding

- Techniques that convert strings (assumed to be names) into some form of code according to how a string is pronounced
  - Generally assuming English language
  - Variations of techniques for other languages exist
- The earliest and still commonly used technique is *Soundex*
  - Patented in 1918 and 1922, used for analysis of older censuses
- Various other techniques improve upon drawbacks of Soundex
  - NYSIIS (New York State Identification and Intelligence System), 1970
  - Metaphone and Double-Metaphone, 1990 and 2000
  - Phonex (direct improvement of Soundex), 1993
  - Phonix, Fuzzy Soundex, Oxford name Compression Algorithm (ONCA), etc.

Phonetic encoding for blocking

- Name variations are common in databases used for linkage (because generally linkage requires names and addresses)
- Many name variations are valid (not errors)
- For example:
  – “gail” versus “gayle” versus “gale” versus “gaile”
  – “albert” versus “alberta” or “alva” versus “alvie”
  (all these are valid suburb names in Australia – check Australia Post!)
- Names are often recorded how people think they should be spelled (based on their experiences)
- Phonetic encoding can help bring together spelling variations of the same name for improved blocking

Soundex algorithm

- A simple algorithm to convert (name) strings into codes made of one letter and three digits
- Steps:

1) Keep first letter of a string
2) Remove all following occurrences of: a, e, i, o, u, y, h, w
3) Replace all consonants from position 2 onwards with digits using these rules:

b, f, p, v → 1	c, g, j, k, q, s, x, z → 2
d, t → 3	l → 4
m, n → 5	r → 6

4) Only keep unique adjacent digits
5) If length of a code is less than 4 add zeros, if longer truncate at length 4

Soundex examples

- “gail” → g400 “gayle” → g400
- “christine” → c623 “christina → c623
- “kristina” → k623 “kirstin” → k623
- “peter” → p360 “christen” → c623

- Online converter:
  http://sites.rootsweb.com/~kyhickma/soundex_converter.htm

- **Questions**: *What is the Soundex code of your first and last name?*
  What blocks do we get if we use Soundex of our names compared to our names directly?
  What are some of the problems with Soundex?

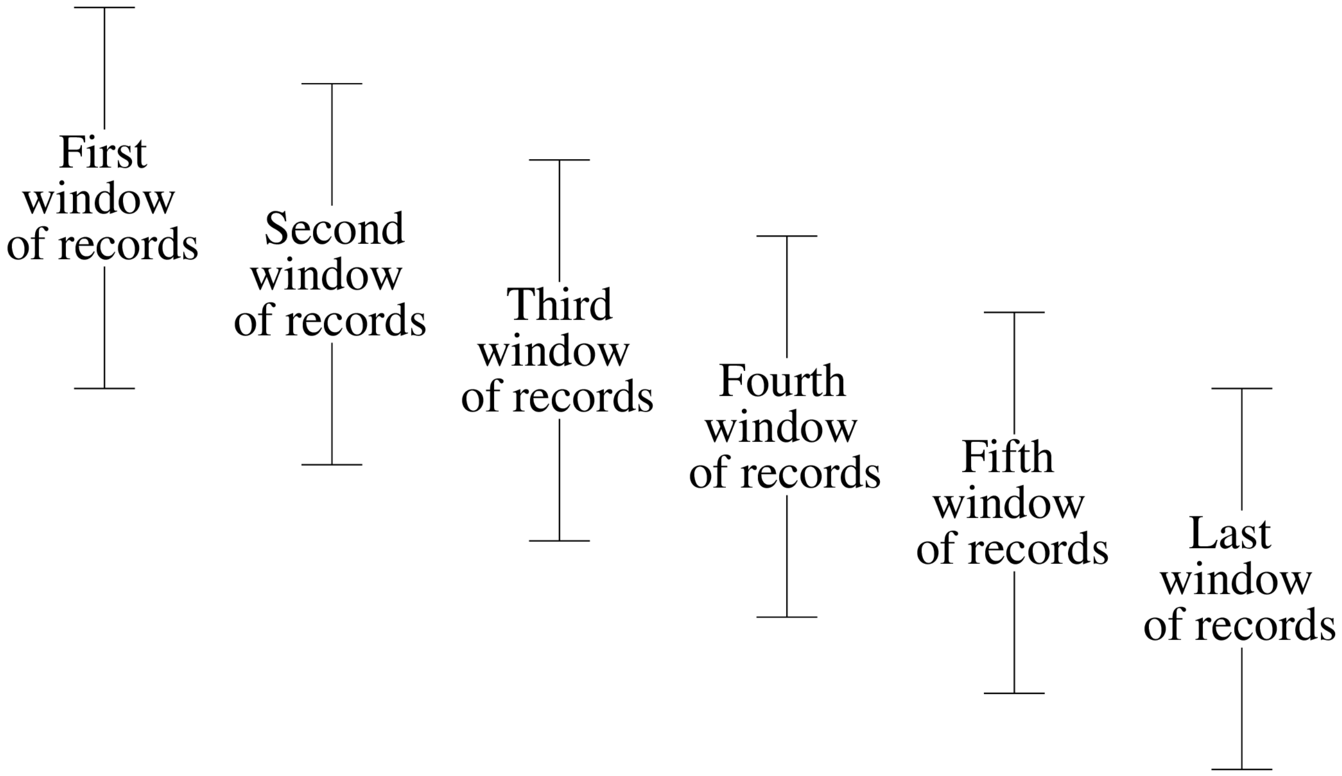
Alternative blocking: Sorted neighbourhood

- Various alternatives to standard blocking have been developed, a popular approach is the *sorted neighbourhood* method
- Basic idea:
  - Merge databases and sort them according to a *sorting key*
  - *S*lide a window over the sorted databases
  - Compare records in the window
  - Use several passes with different sorting criteria
  - Window size can be fixed or adaptive (based on similarities between records)
  - For a window of fixed size *w*, the number of comparisons becomes $w * (|D_1| + |D_2|)$, where $|D|$ is the number of records in a database

Sorted neighbourhood example

- For example, a database is sorted using first and last name:

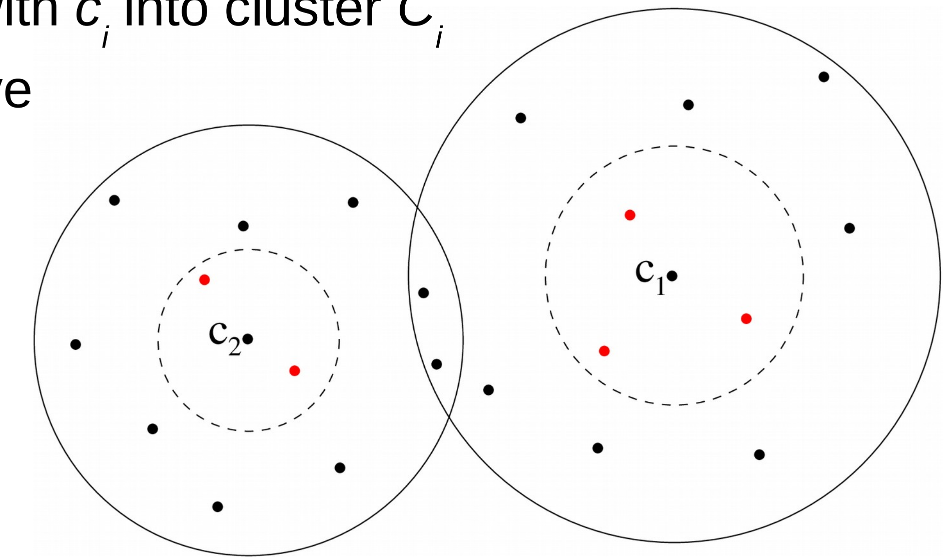
abbybond	r5
paulsmith	r2
pedrosmith	r4
pedrosmith	r9
percysmith	r1
petersmith	r7
petersmith	r10
robinstevens	r3
sallytaylor	r6
sallytaylor	r8



Alternative blocking: Canopy clustering

- Based on a computationally 'cheap' similarity measure such as *Jaccard* (set intersection based on q-grams)
- Records will be inserted into several clusters / blocks
- Algorithm steps:

  1) Randomly select a record in database *D* as cluster centroid c_i , $i = 1, 2, \ldots$
  2) Insert all records that have a similarity of at least s_{loose} with c_i into cluster C_i
  3) Remove all records $r_j \in C_i$ (including c_i) from *D* that have a similarity of at least s_{tight} with c_i, with $s_{tight} \geq s_{loose}$
  4) If database *D* not empty go back to step 1)



Other blocking techniques

- Q-gram based blocking (e.g. 2-grams / bigrams)
  - Convert attribute values into q-gram lists, then generate sub-lists:
    "peter" →['pe','et','te','er'], ['pe','et','te'], ['pe','et','er'], …
    "pete" → ['pe','et','te'], ['pe','et'], ['pe','te'], ['et','te'], …
  - Records with the same sub-list value are inserted into the same block
  - Each record will be inserted into several blocks
  - Works well for 'dirty' data but has high computational costs
- Mapping-based blocking
  - Map strings into a multi-dimensional space such that distances between strings are preserved
- Many more advanced blocking techniques developed in recent years, still an active research area