Parallel corpus approach for name matching in record linkage

Jeff Sukharev, Leonid Zhukov, Alexandrin Popescul
60 million user generated family trees
16 billion records
Person and record search

• Search query

First & Middle Name(s) | Last Name
---|---
Robert | Johnson

Name a place your ancestor might have lived | Estimated birth year
---|---
New York, USA | 1869

Year | Location
---|---
1949 | New York, USA

Add life events (birth, marriage, death, and more)

Matches 1-20 of 247,347 Sorted By Relevance

- Sylvester Robert Johnson
  - SPOUSE: Rosa Anna Ebel Johnson
  - MOTHER: Gertrude Lansing Johnson
  - FATHER: Jay Johnson
  - Search options:
  - See all information...

- Robert J Johnson
  - SPOUSE: Anna Johnson
  - BIRTH: abt 1868 - New York
  - RESIDENCE: North Salem, Westchester, New York

- Robert Johnson
  - SPOUSE: Lena Johnson
  - BIRTH: abt 1870 - New York

- Robert Johnson Junior
  - BIRTH: abt 1870 - New York
  - RESIDENCE: White Plains, Westchester, New York

- Robert M Johnson
  - SPOUSE: Louise W Johnson
  - BIRTH: abt 1863 - New York
  - RESIDENCE: 1930 - Oakland, Alameda, California

View Image

Searching for...
Names: "Robert Johnson"
Birth: 1869, New York, USA
Death: 1949, New York, USA

Narrow by Category
- All Categories
- Census & Voter Lists 5,000+
- Birth, Marriage & Death 5,000+
- Military 5,000+
- Immigration & Travel 2,328
- Schools, Directories & Church Histories 5,000+
- Tax, Criminal, Land & Wills 3,631
- Reference, Dictionaries & Almanacs 1,905
Motivation

• Name matching is a key operation in genealogy search
• Databases often contain alternative name spellings
  – MacDonald → McDonald
  – Clark → Clarke
  – Riedmueller → Reidmiller
  – Arsenault → Arseneau
  – Schumacher → Schumaker
• Number of algorithms exist. Which ones are better?
  – The literature is contradictory on this subject and we could not get a single/acceptable answer
• Can we use Ancestry.com data to improve existing methods?
Goals

- Produce data-driven method for finding high quality lists of alternative name spellings
- Generate dataset that can be used by wider community
- Compare existing name matching methods using unified framework
• Motivated by previous applications of collaborative filtering

• User actions suggest labels
  – Use search reformulations and record attachments
  – Names come in different forms: Shepherd, Sheppard, Shepard, Shephard, Shepperd, Sheperd

• Use Parallel Corpora for training machine translation methods

• No need to separate name pairs into negative/positive classes
Where do we get our data?

- “Records dataset”
  - Source: User data from family trees and attached user records
  - <Name1, Name2>
    - Name1 from the tree node
    - Name2 from the attached record

- “Search dataset”
  - Source: Search logs
  - <Name1, Name2>
    - Name1 from an earlier search performed by a user
    - Name2 from search reformulation from the same user
Machine translation

\[ \arg \max_{t_{\text{name}}} P(t_{\text{name}} | s_{\text{name}}) = \arg \max_{t_{\text{name}}} P(t_{\text{name}}) * P(s_{\text{name}} | t_{\text{name}}) \]

- The goal is finding K top (ranked by probability) target name spellings given source name.

- We adapted machine translation at character level for translating last names into other last names using training set of directed name pairs.
  - In our adaptation sentence consist of a single word (last name) and words are characters.
Name Model

\[
\arg \max_{t_{name}} P(t_{name} | s_{name}) = \arg \max_{t_{name}} P(t_{name}) \times P(s_{name} | t_{name})
\]

- “Name model” is a probability of encountering target names
- Underneath this model is implemented using N-grams

\[
P(c_1 c_2 \ldots c_m) = \prod_{i=1}^{m} P(c_i | c_{max}(1, i-(n-1)), \ldots, c_{max}(1, i-1))
\]
Alignment model

\[
\arg \max_{t_{name}} P(t_{name}|s_{name}) = \arg \max_{t_{name}} P(t_{name}) \times P(s_{name}|t_{name})
\]

- An “alignment model” is used in generating translational correspondences between names in our context.
- Alignment rules and alignment model is computed using Expectation Maximization method.
Other methods

• Compare within the same framework
  – Phonetic algorithms (codes): e.g. Soundex, Dmetaphone, NYSIIS
  – String similarity measures: e.g. Jaro-Winkler, Levenshtein
  – Machine translation method: (using Moses library)

• Phonetic methods:
  – Rule-based methods where similar sounding names get assigned the same code
  – Example: (Soundex)
    ▫ Maps consonants to codes and ignores vowels
    ▫ Similarly sounding names map into the same code
    ▫ Example: Smith – Smythe (code s530)
Results (Records data) 10-fold confidence intervals

The diagram illustrates the precision-recall curves for different algorithms used in records data analysis. Each line represents a different algorithm:
- **dmetaphone**
- **fuzzy_soundex**
- **mod_soundex**
- **nysiis**
- **phonex**
- **phonix**
- **soundex**
- **jaro**
- **winkler–jaro**
- **levenshtein**
- **moses 5–gram**

The curves show the relationship between precision and recall for each algorithm, indicating performance and confidence intervals.
Results (Search data) 10-fold confidence intervals
Tools Used

• Moses software package for machine translation
  – We used IRSTLM, statistical language model for generating 2-gram through 6-gram language models (6 was the maximum possible)
  – Alignment model building: Moses uses the GIZA++ package for statistical character-alignment character (Word)-alignment tools typically implement one of Brown’s IBM generative models.

• Febrl library (by Peter Christen)
  – For similarity measures and phonetic algorithm implementations
Conclusions

• Our machine translation method for finding ranked list of alternative last name spellings far-outperformed all other methods we tried

• NYSIIS phonetic method significantly outperformed other phonetic algorithms and the Phonex phonetic method did not perform as well on our data

• Additionally, Jaro-Winkler similarity method together with the Levenshtein edit distance method performed better than the Jaro method

• Our training sets are available for download from GitHub: https://github.com/jeffsicdm14/name_pairs.git