How to Merge Noisy Datasets

Johannes Ahlmann

scrapinghub

About Johannes

- Over a decade coding in python
- Team Lead Data Science
- NLP, automated crawls, automated extraction
- spark, dask, tensorflow, sklearn

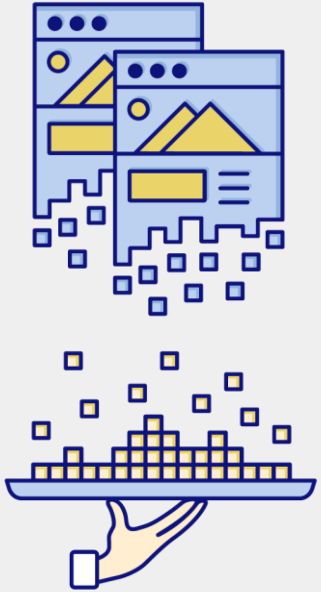
About Scrapinghub

We turn **web content** into **useful data**

Our platform is used to scrape **over 4 billion web pages a month.**

We offer:

- Professional Services to handle the web scraping for you
- Off-the-shelf datasets so you can get data hassle free
- A cloud-based platform that makes scraping a breeze



Who Uses Web Data?

Used by everyone from individuals to large corporates:

- Monitor your competitors by analyzing product information
- Detect fraudulent reviews and sentiment changes by mining reviews
- Create apps that use public data
- Track criminal activity

Joining Datasets

- Acquiring large datasets is quite simple these days on the internet
- Data is often noisy and most of the value often lies in combining, connecting and merging multiple datasets from different sources without unique identifiers
- This talk gives an overview of Probabilistic Record Matching, i.e. the challenges posed when dealing with noisy data, how to normalize data and how to match noisy records to each other

Datasets Example

Data set	Name	Date of birth	City of residence
Data set 1	William J. Smith	1/2/73	Berkeley, California
Data set 2	Smith, W. J.	1973.1.2	Berkeley, CA
Data set 3	Bill Smith	Jan 2, 1973	Berkeley, Calif.

Problem Definition

Situation - Multiple datasets without common unique identifier

- Products, people, companies, hotels, etc.
- “Fluquid Ltd.”
  “Harty’s Quay 80, Rochestown, Cork, Ireland”
  “0214 (303) 2202”
- “Fluquid Ireland Limited”
  “The Mizen, Hartys Quay 80, Rochestown, County Cork”
  “+353 214 303 2200”

Objective - Find likely matching records

- Near-Duplicate Detection
- Record Matching
- Record Linkage

Challenges

- Comparing 1M * 1M elements pairwise would require 1 trillion pairwise comparisons
- Connected Components / Community Detection is computationally expensive
- Same entity can be represented in different ways, different entities may have similar representations
- Different field types need to be compared differently

Examples of Field Types

- Name: Person, Company, University, Product, Brand
- String: Product Description
- Number: Price, …
- Identifier: UPC, ASIN, ISBN, …
- Postal Address
- Geolocation (latitude, longitude)

Data in Noisy

- Data is noisy (typos, free text, etc.) ("Mnuich", "   Munich", "munich")
- Data can vary syntactically ("12.00", 12.00, 12)
- Many ways to represent the same entity ("Munich", "München", "Muenchen", "Munique", "48.1351° N, 11.5820° E", "zip 80331–81929", "[ˈmʏnçn]", "Minga", "慕尼黑")
- Entity representations are ambiguous

  <Munich City, Germany>
  <Munich County, Germany>
  <Munich, North Dakota>

- [Wikipedia disambiguation]

Available Libraries

- Addresses – pypostal, geonames (zip codes, geocodes, etc.)
- Persons – probablepeople
- Date – dateparser, heideltime
- Companies – cleanco
- Entity aliases – Wikipedia redirects/ disambiguation pages
- Deduplication (active learning) – dedupe
- Record Matching (simplistic) – Duke (java), febrl, relais
- Data Exploration – OpenRefine
- Approximate String Comparison

  Simstring (Jaccard similarity)

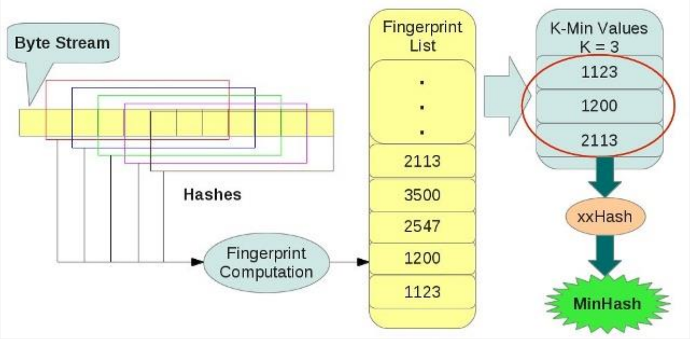
  Simhash, Minhash
- Connected components, community clustering, etc.

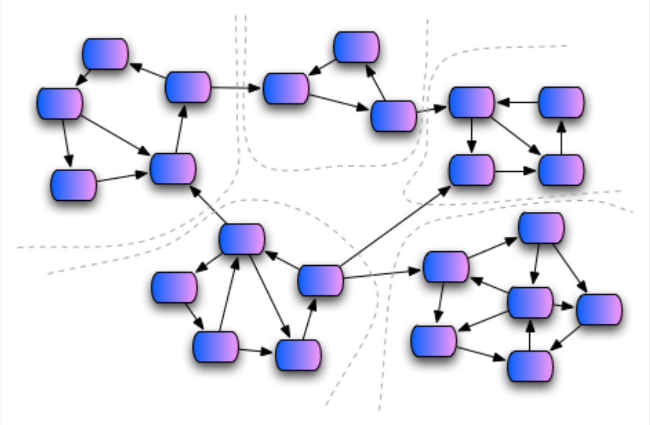
  igraph.fastgreedy.community()

  spark.graphx.connectedComponents()

Approach

1. Standardize and Normalize fields as much as possible
2. Find fingerprint function for each field type
3. Fingerprint each field into high-dimensional space
4. Use nearest-neighbors algorithm to find candidate matches (based on fingerprint)
5. Calculate pair-wise similarity of candidates
6. Use connected components / "community detection" to find likely matches





Thank you!

Johannes Ahlmann
johannes@scrapinghub.com

scrapinghub

scrapinghub.com