ACM SIGMOD Programming Contest 2022

Team: RUPikachu

Members: Ezhil Nikhilan C, Aditya Maheshwari

Contact: {ezhil.nikhilan, aditya.maheshwari}@rutgers.edu

Task Overview

Task: To perform blocking for Entity Resolution in a limited time (35 minutes) i.e, filter out obvious non-matches.

Dataset	Description	Expected # of pairs
D1	Notebook Specifications	1000000
D2	Product Specifications	2000000

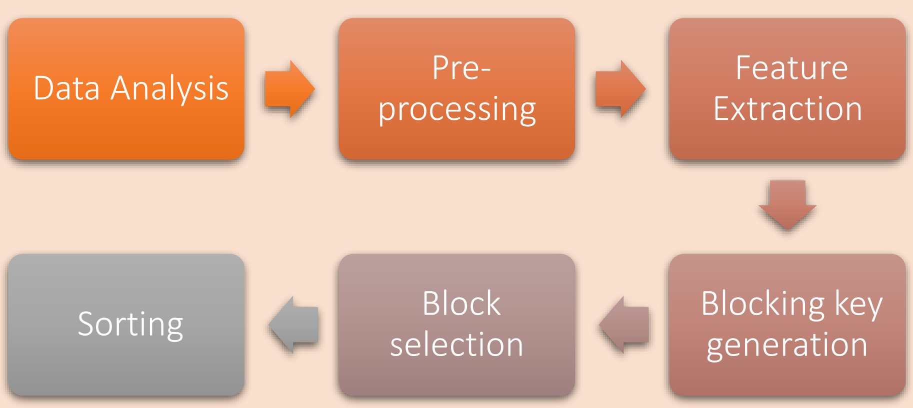
Evaluation Metric: Recall & Runtime. Trivial equi-joins not to be included, and output pairs to be transitively-closed.

Evaluation Environment: 16 CPU x 2.7 GHz, 32 GB Main, 32 GB Storage, Ubuntu 20.04.3 LTS

Methodology

We used a non-learning, schema-aware method to generate hash-based blocking keys.

- Our solution involved 4 major steps:
  1. **Data analysis:** To understand the data.
  2. **Data preprocessing:** To clean the data.
  3. **Blocking key generation:** From the extracted features
  4. **Postprocessing:** To select the most relevant pairs.



Data Analysis

We identified dominant patterns in the data using tokenization and TF-IDF. Our analysis focused on identifying,

- **Product types:** Like Laptops, SD cards etc.
- **Product identifiers:** Like brands, specs etc.
- **Nature of the noise:** Errors, inconsistencies, language differences, missing information etc.

Data Preprocessing

We used regex and python string manipulation to standardize the data for feature extraction. This involved,

- **Standardization:** Convert to lowercase, remove irrelevant special characters.
- **Error Correction:** Correct the errors and inconsistencies identified during Data analysis. Eg: datattraveler/data traveler → *datatraveler*
- **Semantic mapping:** Map words of similar meaning to a single identifier. Eg: class, clase, klase → class

Blocking Key Generation

Features were extracted from the preprocessed data using **Regex**. The extracted features were visualized through different data visualization tools.

Based on the findings from the above step, we identified the best feature combinations to create keys.

- **Loose keys** (such as *brand+model*) were used to capture the less frequently occurring matches.
- **Specific feature combinations** to capture the more common patterns.

Post Processing

Block Selection: Extremely common patterns were filtered out by limiting the block size.

Sorting: The selected candidate pairs were sorted using Jaccard and Overlap similarity to determine the top 3 million pairs.

Results

We achieved a better recall in the relatively smaller Dataset 1 with a significant margin for improvement in the Dataset 2.

#	Dataset 1	Dataset 2	Overall
Recall	0.772	0.241	0.507

Discussion of results and Conclusion

Discussion: Multilingual nature, and the highly variable representation of specifications were the primary challenges in designing a time-constrained blocking system for Dataset 2. Besides, the relatively small sample set X2, could not provide a complete representation of the massive D2 dataset. It is notable that, despite achieving 0.9+ recall on the sample set, the final recall did not cross the 0.25 mark.

Conclusion: Data analysis and visualization proved to be efficient in deriving insights about real-world data even using a small sample set. Our choice of a hash-based method was useful in escaping the quadratic complexity of set similarity join techniques, although the runtime can be further improved using multithreading.
A low correlation between the recall for the sample and the actual dataset hints that a more generic blocking system could achieve a better recall. Efficient means of translating multilingual data could be useful for time-constrained blocking of real-world data.