SIH 2022

Anton
SH1009

First Evaluation

Team Name - Anton

Team Leader - Prakanshul Saxena

Team Member - Prachir Agrawal

Team Member - Harshit Jajodia

Team Member - Harshil Singhal

Team Member - Manas Wadhwa

Team Member - Shaleen Malik

Problem Statement (SH1009)

Title - "Algorithm/Mechanism for analyzing and estimating % match level for similar data column/s values (e.g. Consolidation of multiple Govt. programs implies an individual be uniquely identified in all DBs. To uniquely identify, a location address is provided (e.g. a village name) and then other details such as name, DOB, Mother's, Father's name etc."

Description - "The name itself might not be same in all and can have variations in other DBs such as (Name – Raam, Ram, Rama), (Father's Name – Shyam, Syam, Sham etc.). Algorithm is required such that inference can be drawn as to what percent the field value is matching between entries in other DBs. This will help estimating the overall variation % of field values in other DBs and the accuracy level if these programs are to be converged with each other. Initially it can be one field, then can be extended to combination of multiple fields etc. Due to large volume of data, initially a sample size can be determined and taken to calculate the match. Sample data required: No"

Problem Exploration

Misspelling

Abbreviations in Names

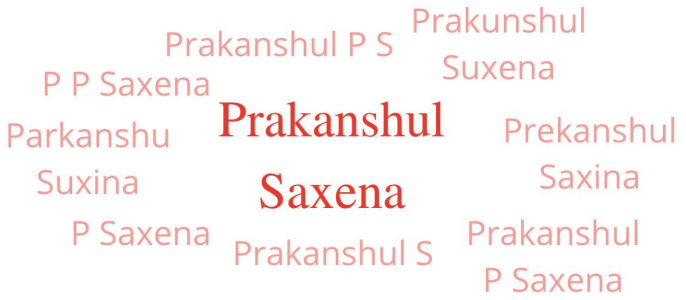
Unordered

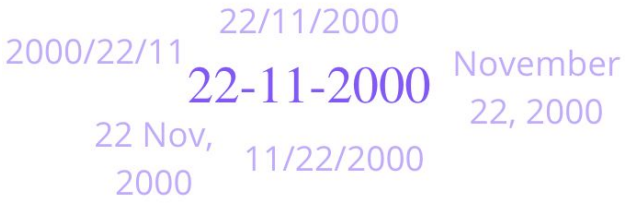
Prefixes

Abbreviations in Addresses

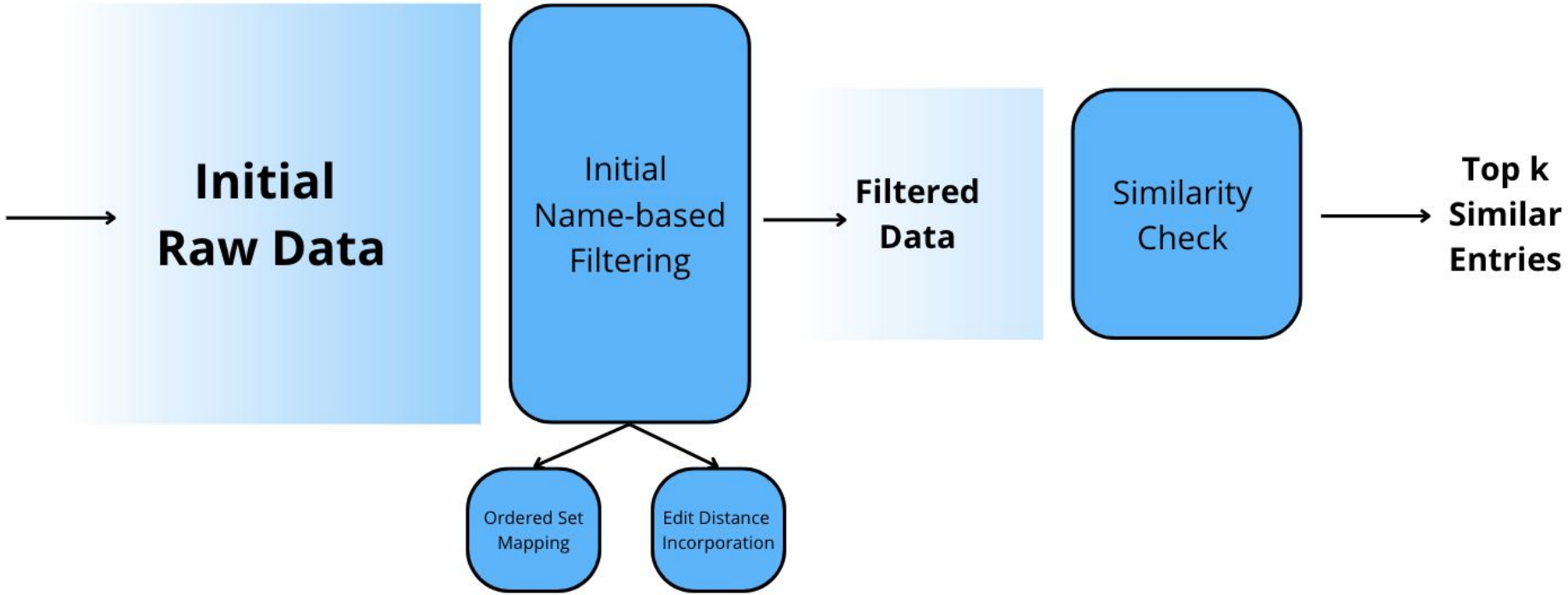
Date of Birth Formats

Missing Fields





Solution Approach



Solution Specifications

Time Complexity

Preprocessing $= \mathcal{O}(n \times 50) \approx \mathcal{O}(n)$

Algorithm (Average Case) $= \mathcal{O}(k \times 8 \times 26 \times 5) \approx \mathcal{O}(k \times 10^3)$ where : k <<< n

Worst Case $= \mathcal{O}(n \times 10^3)$

Space Complexity $= \mathcal{O}(n)$ (used to create dictionary)

k - Number of Indices filtered based on the ordered set and edit distance method

n - Total number of entries in the second database

k or n	$\sim 10^3$	$\sim 10^4$	$\sim 10^5$	$\sim 10^6$
Time Taken	$\sim$0s	$\sim$10s	$\sim$1.6 mins	$\sim$16 mins

Solution Specifications (Similarity Metric)

$$\text{Similarity Metric} = \frac{\sum_{i}\left(\frac{\sum_{j}((X_{gt})_{j}^{i} == (X_{d})_{j}^{i})}{n(X_{gt}^{i})}\right)}{N} \times 100$$

All the values compared are - Sorted Ordered Sets

gt - Ground Truth

d - Given Data

X - One Specific Field

N - Total Number of Fields in the Dataset

Datasets

1st Dataset - 50 Entries (Self Curated)

2nd Dataset - 234 Entries (Self Curated and Modified Accordingly)

Thank You!!

Open for Questions & Suggestions

Second Evaluation

Current Progress

Base algorithm running on a sample of dataset to provide similarity matching.

New Additions & Experiments

Frontend and Backend

Soundex

Regex Incorporation

Parallelization

Data Ingestion

Datasets

LSH (On base Algorithm)

Combining Different Databases + U_id

Sound based implementation

Dedupe

Soundex

Phonetic Similarity

Generates four-character codes based upon the pronunciation of words.

Hari = Harie = Haree = Hurry = H600

Enlarges the first screen through names

Leads to better accuracy and **robustness**

Regex Incorporation

Similarity Calculation and Accuracy Improves

Increases Algorithm **robustness**

[Flat D-1303, AVJ Heights, Sector @ Zeta-1, Gr. Noida, Uttar Pradesh]

[‘Flat’ ,‘D’,’1303’, ‘AVJ’,’Heights’, ‘Sector’,’Zeta ‘1’, ‘Gr’, ‘Noida’, ‘Uttar’,’Pradesh’]

Parallelization/Multiprocessing

Overall Net Time reduced to one-third of its initial value

Hardware dependency

Different thread utilization

Contributes to algorithm **efficiency**

Parallelization/Multiprocessing Metrics

The function below is the bottleneck or the most time consuming function in our approach, and thus we have implemented threading upon this function.

```
for idx in df_modified_pruned.index:      You, 13 hours ago • add api for multiple db …
    number_of_fields,cumulative_percentage_score=0,0
    for field in common_fields:
        number_of_fields+=1
        cumulative_percentage_score+=find_percentage_match(df_original[field][idx_to_check],df_modified_pruned[field][idx])
    overall_percentange_match_value=cumulative_percentage_score/number_of_fields
    df_modified_pruned.loc[idx, 'Percentange Match %'] = overall_percentange_match_value
```

k or n	$\sim 10^3$	$\sim 10^4$	$\sim 10^5$	$\sim 10^6$
Time Taken	$\sim$0s	$\sim$10s	$\sim$1.6 mins	$\sim$16 mins
Time Taken after Threading (Parallelisation)	$\sim$0s	$\sim$2s	$\sim$15s	$\sim$3 mins

LSH (Locality Sensitive Hashing) (OSS)

Suitable for massive datasets

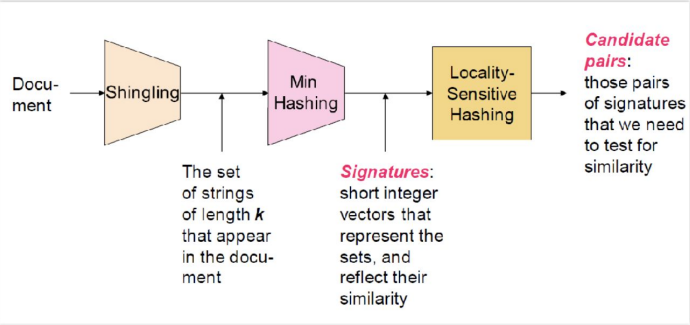
The space complexity is $O(\ell nk)$, since we have $O(\ell)$ hash tables, each with n points and for each point we store a k-dim hash vector.

Ignoring lower order terms, the space complexity is $O(n^{1+\rho})$.

query time is $O(n^{\rho})$

we need $O(n^{1.5})$ memory to achieve $O(\sqrt{n})$ query time.

we need $O(n^{1.25})$ memory to achieve $O(n^{1/4})$ query time



A natural similarity measure is the Jaccard similarity:

- $Sim(D_1, D_2) = \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$

Jaccard distance:

- $d(C_1, C_2) = 1 - \frac{|C_1 \cap C_2|}{|C_1 \cup C_2|}$

Dedupe

Entity Resolution

A OSS tool that learns the best way to find similar rows in our data using machine learning.

Uncertain pairs determined using blocking, affine gap distance, and active learning.

Active Learning is a loss.

Thank You!!

Open for Questions & Suggestions