International Conference on "Emerging Trends in Computer Engineering, Science and Information Technology"- 2015
Special Issue of International Journal of Electronics, Communication & Soft Computing Science and Engineering, ISSN: 2277-9477

MVJoin: An Efficient Approach for Record Linkage and Duplication Finding

Ms. Laxmi R. Adhav **Prof. Santosh D. Kumar**

***Abstract* — Duplicate detection is major task in data processing and cleaning. In this paper we discussed about various methods of duplicate detection for a dataset. Calculating Edit Distance is the most preferred approach for duplicate detection. A similarity join calculates different entities from two data sets for their similarity value, not less than a given threshold. The most commonly used approach is based on technique of extracting overlapping characters from strings and considering only strings, that share some decided (q) number of characters as candidates. While calculating edit distance strings are divided into number of small strings called Chunks. The proposed system–Modified VChunk Join (MVJoin) algorithm uses a greedy approach to automatically select a suitable chunking scheme for a given dataset, to find duplicates efficiently. This system implements an efficient approach for finding duplicate records in data sets as well as linking records of various dataset into single one. It is experimentally demonstrated that the MVJoin algorithm is faster than alternative methods occupying less space.**

***Key Words* — Chunks, CDB, Edit Distance, Gram, MVJoin, Similarity Join, Virtual CDB**.

I. INTRODUCTION

Databases are of vital importance to IT industry without which any industry can't exist. To store huge amount of data, Data-Warehouse is the only solution, but the processing of the data is not supported by data-warehouse. The Data Mining has capability to process on huge data effectively. Data mining process that data though ETL (Extraction, Transformation and Loading) techniques. The industries using these databases require quality of information for their smooth functioning and business. Quality of data indicates clean and error free data. Clean data refers the data where duplicate records, null or empty records should be avoided. Hence it becomes important to remove duplicate data. Duplicate data detection is the process of detecting multiple records of single object. It is supported by data preparation stage to store data in uniform manner.

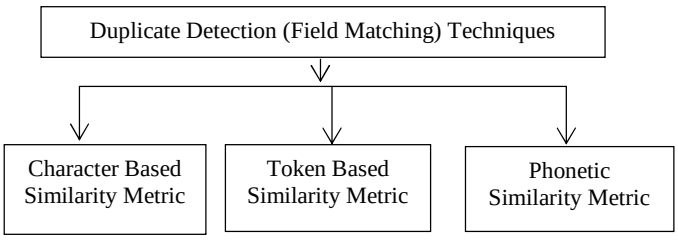

Fig 1: Duplicate Detection (Field Matching) Techniques

This could use method of Field Matching Techniques which contains three distinct methods as shown in Fig. 1.

A. Character-based similarity metrics

In case of character-based similarity metrics, characters are considered as major factor for content identification. It is also divided into various methods as,

i. Edit distance,
ii. Affine gap distance,
iii. Smith-Waterman distance,
iv. Jaro distance metric, and
v. *Q*-gram distance

i) In Edit distance method, edit distance is the number of minimum edit operations required to transform one string to another. The edit operations could be

- Inserting a character
- Deleting a character or
- Replacing a character.

For each of these operations edit cost 1 is considered.

Edit distance has two major advantages over alternative distance measure:

(a) It focus on the ordering of tokens in string

(b) It permits non-trivial alignment of the chunks.

This method is useful for finding typing errors but ineffective for other kind of mismatches.

ii) In Q-gram distance method, Q-grams are considered as small substrings of original strings. This method is used for string approximation in which two strings are said to be equal if they share sufficient number of common Q- grams. Letters in q-grams including unigram, bi-gram, trigrams are used for text recognition and spelling checking.

A similarity join is used to find different objects from two dataset representing same entity. Euclidean space is used to calculate similarities in early research, whereas currently various distance functions are used to calculate similarity. Existing methods classified into three categories:

1. Gram based:

Conventionally, fixed length q-grams are widely used for edit similarity joins or queries. A q-gram or chunk is a contiguous substring of length q. Given a complete string s, we move a sliding window of width q over the string s to extract the q-grams of the string.

2. Tree based:

A tree-based approach builds a tree for the data set and support edit similarity queries by incrementally building the tree. A tree-based approach for edit similarity search has been proposed [11]. It builds a tree for the data set and support edit similarity queries by incrementally probing the tire. Zhang et al. [15] propose an index structure named Bed- tree to support edit similarity selection and join queries by mapping strings into a linear space which is supported by a standard B-tree, together with several

filtering approaches to prune internal and leaf nodes of the B-tree.

3. Enumeration based:

Neighborhood generation-based methods count all possible strings obtained by the edit operations. While naive enumeration method only works in theory, recent proposals using deletion neighborhood [9] and partitioning can work well with small edit distance thresholds.

In calculating edit distance, focus is on edit similarity joins, i.e., similarity joins with an edit distance condition of distance no more than a constant threshold τ. The edit distance also called as *Levenshtein distance* between two strings s1 and s2 is the minimum number of single character edits operations such as insertion, deletion, and substitution that are needed to change s1 to s2. The major technical difficulty is the fact that the edit distance metric is complex and costly to compute. Edit distance has two distinct advantages over alternative distance measures: (a) it reflects the ordering of tokens in the string; and (b) Edit distance allows non-trivial alignments of the strings in dataset.

An n-gram of any string is a substring of length n that can be used as identification for the string. Naturally, for each string present in dataset, we divide it into several different substrings (or chunks) and it only needs to be processed and indexed these different chunks. The frequencies of variable-length grams in strings are analyzed, and a set of grams is selected, called gram dictionary, such that each selected gram in the gram dictionary is not too frequent in the strings of the dataset.

The rest of the paper is organized as section II Literature Review, section III Implementation details, Section IV Results Analysis, Section V Conclusion and Future scope, and Section VI References.

II. LITERATURE REVIEW

There are various algorithms that use Q-gram technique to find similar strings for the purpose of finding Duplicate records or to find similar text in given dataset. Ed-Join, Winnowing, VGRAM, etc. are the methods that work on Q-gram technique. VChunkJoin algorithm works with edit similarity join.

Edit based similarity measures considers difference between two objects as the number of edit operations required to convert one string object into another. Similarity and number of edit operations are reciprocal of each other i.e. similarity measure decreases as number of edit operations increases. A similarity join finds pairs of objects from two data sets such that the pair's similarity value is no less than a given threshold [15]. *Edit similarity join* is a similarity join with considering Edit based similarity measures.

A. Ed-Join

In Ed-Join algorithm all positional q-grams are extracted and ordered by decreasing order of their id values and increasing order of their locations [14]. They called the sorted array, the 'q-gram' array of the string. This technique improves speed of similarity joins.

Three filtering approach were currently followed as Count filtering, Position filtering and Length filtering. In Ed-Join algorithm new filtering technique called Prefix filtering is introduces as

Prefix Filtering- Let x and y be two q-gram arrays and $ed(str(x), str(y)) \leq \tau$. Then the $(q \cdot \tau + 1)$-prefix of x and the $(q \cdot \tau + 1)$-prefix of y must have at least one matching q-gram [14].

Location based mismatch filtering technique is used to detect errors that are within distance τ. In Content based mismatch filtering, they are selecting probing window and looks for the contents of the string. If the content difference in probing window is appropriate edit distance measure, similarity between two strings can be decided.

B. VGRAM

This algorithm mainly focuses on choosing high quality grams of variable length from collection of strings for providing support on queries.

This algorithm uses two variable q_{min} and q_{max} such that $q_{min} < q_{max}$. The grams are selected between q_{min} and q_{max}.

Gram Dictionary is set of sub-strings that are not too frequent. Using gram dictionary set of variable length grams is created. At each step grams are generated for longest substring for matching gram from the dictionary.

If no such gram exists in gram dictionary, then gram is generated of length q_{min}. Gram dictionary is stored as trie. The trie is a tree structure having each edge is labeled as character. According to this algorithm if two strings are within edit distance *k*, then their set of grams also has relevant similarity to *k*. Each, end of gram, is recognized by adding special symbol that does not belong to alphabet set. Path traversal from root to leaf node gives gram in Gram Dictionary. While constructing a gram Dictionary, gram frequencies are collected first then out of frequent grams high quality grams are selected.

C. Winnowing

Document fingerprinting is technique used to detect small partial copied content in a large set of documents. A fingerprint also contains information about position to describe location from where fingerprint is arrived. In practice, the set of fingerprints is obtained from a small subset of all k-gram hashes.

In this algorithm 'k-gram' is considered as contiguous substring of length k. Document is divided into k-grams, the value of k is accepted from user.

If hash function is chosen such that probabilities of collision of these hash functions is very small. In such case if two documents share common fingerprint then they share a k-grams as well. Larger value of k indicates matches in the document are not coincidental. So the algorithm avoids matching of strings below the *noise threshold*. The upper bound on performance is expressed as trade-off between the numbers of fingerprints that must be selected and the shortest match that are guaranteed to detect. Substrings are considered as matching if they satisfy two properties as

1. If there is a substring match, at least, as long as the *guarantee threshold*, *t*, then this match is detected and

2. We do not detect any matches shorter than the *noise threshold k*.

The constants *t* and $k \leq t$ are chosen by the user [3]

This algorithm can't ensure copied contents detected completely. Note that there are almost as many *k*-grams as there are characters in the document, as every position in the document (except for the last $k - 1$ positions) marks the beginning of a *k*-gram. Now hash each *k*-gram and select some subset of these hashes to be the document's *fingerprints*. [16]

D. VchunkJoin

All the previous algorithms for similarity joins, compute edit distance to find similarity join which is very costly to compute. They all uses filter and verify techniques for eliminating non matching strings first, then performs verification of matching strings by calculating edit distance. The filter method used, is relatively inexpensive to compute.

Most widespread filtering approach is based on computing GRAMS. Computing Grams can be categorized as fixed length grams and variable length grams. Working with grams results in space and time overhead as computing grams results in large index size which can't entirely accommodated in memory and requires high query processing cost. This algorithm mainly focuses on edit similarity joins. VChunkJoin algorithm divides string into several small substrings, called as chunks. Only chunk's index is stored and processed hence very less space is required as compared to all previous duplicate detection algorithms. This algorithm uses all previous filtering techniques to find out matching strings like length, count, prefix, location-based mismatch, and content based mismatch filtering.

It also uses new filtering approach as rank, chunk number and virtual CDB filtering. A positional q-gram is a q-gram with its position represented in algorithm as of (qgram, pos). For matching q-grams they should have content and position within edit distance τ. This algorithm also uses a class of strong chunking schemes that applies tight lower bound on the number of chunks shared by similar strings. Good chunking scheme guarantees that avalanching effect is not generated. *Avalanching effect* is destruction of chunks while finding similarity distance in single edit distance.

A chunking scheme of this algorithm divides a string into different non-overlapping substrings, each called a chunk. A Vchunk is a chunk with its position and rank information, represented as (chunk, pos, rank).

a) The VChunkJoin Algorithm

The basic version of VChunkJoin algorithm is a chunk-based counterpart of basic All- Pairs-Ed algorithm with three major modifications as:

1. This algorithm replaces q-grams with Vchunks. The prefix length is shortened to $2\tau+1$, and significantly reduce inverted index size when $q > 2$.

2. Rank and chunk number filters are used in the algorithm

3. An improved verification algorithm VerifyVChunk is used.

Two Vchunks u and v are said to be matching (with respect to τ) if

- Their contents are the same, and
- Their positions are within τ, and
- Their ranks are within τ [13].

Chunk Number filtering is the unique filtering technique used for effective filtering. A previous filtering technique like location based mismatch filtering is also used. Content based mismatch filtering could not be directly applied. A new technique called virtual CBD filtering is applied. In case of virtual CBD, only resulting chunk numbers are stored for additional CDBs instead of storing and indexing the resulting chunks. Hence additional virtual CBD storage overhead per string is less.

III. IMPLEMENTATION DETAILS

Information obtained from different data sources can have various inconsistencies. The real world data obtained from different sources can be represented in slightly different variations, for example "PO Box No. 35, Main Street." and "P.O. Box NO. 35, Main Street", or "Tom Cruse" and "Cruse Tom". There could be errors in the information due to the process it was collected, for example while taking address for any data set it consist of "Address Line 1" and "Address Line 2", but the conventions for that is not specified and could collect wrong information . The quality of data in a data set suffers from typing mistakes, deficiency of standards for entering database fields, non-enforced integrity constraints, inconsistency in data mappings, etc. For such many reasons, data cleaning often needs to find entities representing similar information that is repetitive in a given data set.

A. Block Diagram

The proposed system is composed of architecture as shown in Fig. 2. The architecture consist of modules like Cleaning and standardisation, indexing, record pair comparison, similar vector classification, clerical view to database administrator, Evaluation and Database linkage.

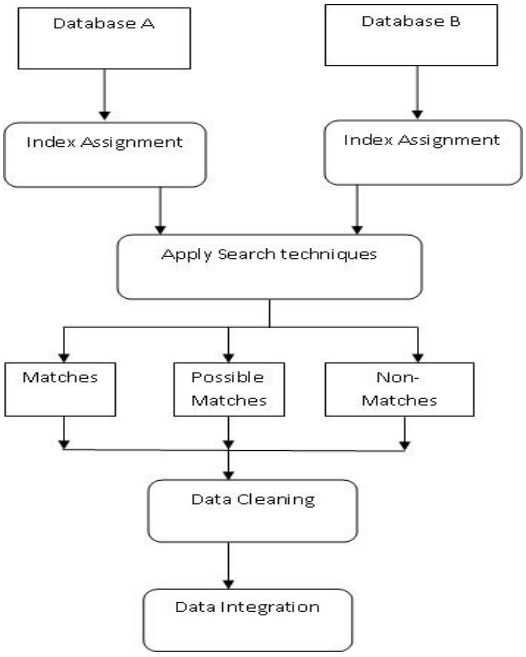

Fig.2: Block diagram of proposed system

The system will consist of many Databases or tables of database say Database A and B. The task of proposed system is to make the available Database to Standard format. After this records are assigned with index number for faster searching. Record comparison is done for checking if similar kinds of records are present in database or not. If records are found matching or possible matching then its Clerical view will be generated to confirm about duplicate records. If the records are similar and confirmed by database administrator, he can delete the duplicate record. Database will be arranged to standard format again. If administrator wants to link/merge the database into single database, he is allowed to do so with proposed system.

B. Modules Used

The proposed system is divided into following modules

- *Database Connection Manager*

In this module, user can create and store, edit and manage all database connection information. This will allow users to access their database in a single click.

- *Retrieving Schema and Content Information*

The system will collect or get all information like schema (about database info – driver name, database version, username, password, connection timeout, available table names, etc.) and Content Information like (about table columns and rows, as well as table data, etc.) to a specific database, which is selected by user from connection manager.

- *Duplication Finder*

In this module, user can find their duplicate or redundant data from their database. The module will produce output in three categories, which is

1. Complete Matches
2. Possible Matches
3. Non Matches

So user can get information about their data and redundant.

- *Data Cleaning*

User can clean or remove their unwanted data (NULL value, invalid data, repetitive data etc.) from their database. So user can keep their database in a well form, which will produce an accurate output for user.

- *Data Integration or Record Linkage*

In this module, user can perform actions on their duplicate data's. To clean and standardize their data, user can integrate or link their data's from one to another.

C. Algorithm Used

The MVJoin algorithm is faster and efficient than alternate algorithms. It will follow greedy approach for duplicate finding in the given dataset.

The system is also finding modified *pruning* approach to get best possible matches in the dataset. Matching is followed by deletion of data to avoid repetitions in a dataset or multiple datasets.

The algorithm is

1. Select Database and their tables
2. Display columns to user and select all/some of them as reference
3. Search options
   a) column options
   b) chunk choice
   c) Matching technique should include columns separately or not
4. Display original, Complete match & non-matching records
5. Apply action selected by user for data cleaning and integration

D. Platform Used

Software

NetBeans IDE 7 .0.1 and above

JDK 1.6 and above

Windows XP and above

MySQL

Hardware
Processor - Pentium IV and above
Ram – 1 GB
Hard Disk – 80 GB

IV. Results Analysis

For experimental analysis this algorithm uses several publically real datasets out of which one is demonstrated here

– IMDB is a collection of actor names downloaded from IMDB website.

For comparison following parameters are considered as

– The average length of the prefixes;

– The running time.

Table 1: Result analysis on IMDB dataset

Sr. No.	Name of Dataset	No of records	Data size (KiB)	No of Duplicate found		Time taken (ms)	
				Vchunk Join	MVJoin	Vchunk Join	MVJoin
1	IMDB	417	64	1	1	78	47
		281	48	0	0	48	16
		879	80	1	1	79	16

Table 1 shows comparison of time required to compute duplicate records. It is observed that MVJoin algorithm has smaller time computation requirement than VChinkJoin.

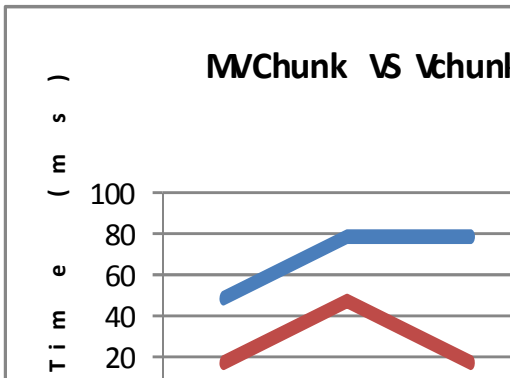

Fig. 3: Time efficiency graph

From the Fig. 3 the speed performance of MVJoin algorithm is increased for larger amount of data. The proposed system is useful and efficient for duplicate detection of huge dataset. The MVJoin and VChinkJoin algorithm works same for smaller dataset. This system also gives result in the form of possible matches for relatively matching records whose decision must be taken by dataset administrator. This feature improves accuracy of duplicate detection.

V. Conclusion and Future Scope

In this paper, we discussed about various methods of finding duplicates in a given dataset. The gram based methods like Ed-Join, Winnowing, VGRAM gives results in terms of duplicate detection. Out of these old methods VGRAM method is more efficient. VChunkJoin was a novel approach for duplicate detection for edit similarity joins. The proposed system is aimed at to reduce redundancy and repetition in the huge amount of data set so as get clean and non-repeating data. Using MVJoin algorithm, redundancy in the data set is reduced at the best possible level and quick and accurate searching is achieved so that user will get the results expected in less time.

References

[1] C. Xiao, W. Wang, X. Lin, and J.X. Yu, "Efficient Similarity Joins for Near Duplicate Detection," Proc. 17th Int'l Conf. World Wide Web (WWW), 2008.

[2] R.J. Bayardo, Y. Ma, and R. Srikant, "Scaling up all Pairs Similarity Search," Proc. 16th Int'l Conf. World Wide Web (WWW), 2007.

[3] C. Xiao, W. Wang, and X. Lin, "Ed-Join: An Efficient Algorithm for Similarity Joins with Edit Distance Constraints," Proc. VLDB Endowment, vol. 1, no. 1, pp. 933-944, 2008.

[4] A. Chandel, O. Hassanzadeh, N. Koudas, M. Sadoghi, and D. Srivastava, "Benchmarking Declarative Approximate Selection Predicates," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 353-364, 2007.

[5] C. Li, B. Wang, and X. Yang, "VGRAM: Improving Performance of Approximate Queries on String Collections using Variable-Length Grams," Proc. 33rd Int'l Conf. Very Large Databases (VLDB), 2007.

[6] X. Yang, B. Wang, and C. Li, "Cost-Based Variable-Length-Gram Selection for String Collections to Support Approximate Queries Efficiently," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 353-364, 2008.

[7] S. Chaudhuri, V. Ganti, and R. Kaushik, "A Primitive Operator for Similarity Joins in Data Cleaning," Proc. 22nd Int'l Conf. Data Eng. (ICDE), 2006.

[8] L. Jin and C. Li, "Selectivity Estimation for Fuzzy String Predicates in Large Data Sets," Proc. Int'l Conf. Very Large Databases (VLDB), pp. 397-408, 2005.

[9] A. Mazeika, M.H. Bo¨hlen, N. Koudas, and D. Srivastava, "Estimating the Selectivity of Approximate String Queries," ACM Trans. Database Systems, vol. 32, no. 2, article 12, 2007.

[10] M. Hadjieleftheriou, X. Yu, N. Koudas, and D. Srivastava, "Hashed Samples: Selectivity Estimators for Set Similarity Selection Queries," Proc. VLDB Endowment, vol. 1, no. 1, pp. 201-212, 2008.

[11] H. Lee, R.T. Ng, and K. Shim, "Similarity Join Size Estimation Using Locality Sensitive Hashing," Proc. VLDB Endowment, vol. 4, no. 6, pp. 338-349, 2011.

[12] B.S.T. Bocek and E. Hunt, "Fast similarity Search in Large Dictionaries," Technical Report ifi-2007.02, Dept. of Informatics, Univ. of Zurich, Apr. 2007.

[13] Wei Wang, Jianbin Qin, Chuan Xiao, Xuemin Lin, and Heng Tao Shen, Senior Member, IEEE, "VChunkJoin: An Efficient Algorithm for Edit Similarity Joins." IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING, VOL. 25, NO. 8, AUGUST 2013

[14] C. Xiao, W. Wang, and X. Lin, "Ed-Join: An Efficient Algorithm for Similarity Joins with Edit Distance Constraints," Proc. VLDB Endowment, vol. 1, no. 1, pp. 933-944, 2008.

[15] S. Schleimer, D.S. Wilkerson, and A. Aiken, "Winnowing: Local Algorithms for Document Fingerprinting," Proc. ACM SIGMOD Int'l Conf. Management of Data, pp. 76-85, 2003.

Author's Profile

Ms. Laxmi R. Adhav

She received her Bachelors of Engineering [Computer] degree from University of Pune in 2008. She is currently working as Lecturer in Computer Department at Sandip Foundation and pursuing Masters of Engineering at Pune University.

Prof. Santosh Kumar

He is working as Asst. Professor, at Computer Department, SITRC, Nashik.
He is also working as ME Coordinator for Savitribai Phule Pune University
He is member of ISTE.