Advanced Record Linkage Techniques for Improving the Data Matching between Cultural Heritage Datasets from Different Sources

Jordan Stoikov [1], Alexandra Nikolova [1], Vladimir Georgiev [2]

[1] *Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Bulgaria*
[2] *Computer Science Department, American University in Bulgaria, Bulgaria*

Abstract **– The paper's goal is to investigate a matching algorithm that can be used to connect records of cultural heritage data from various data sources and enhances the precision and effectiveness of the matching operations involved in this process. The foundation of this work is a fuzzy match similarity (FMS) function, which explicitly associates weights with tokens to measure their relative relevance while reading a string as a series of tokens.**

Keywords **– record linkage, probabilistic linkage, similarity function.**

1. Introduction

The process of developing bibliographic databases [1] with cultural heritage data is facing numerous challenges related to data quality and record linkage.

Multiple data sources are one of the factors to take into account; if data is collected from several organizations or systems, in various places, during various time periods, or through inconsistent data input techniques, then it is probable that the data is unreliable. The largest issue is that there are often several input techniques used when entering the same surname and initials into a database since there is no set format for abbreviations, etc.

In Database repositories, the received data flow from peripheral origins should be purged and authenticated in order to assure excellent data quality. The majority of the time, data from external sources that are received at the data warehouse include substantial mistakes, such as misspellings, abbreviations, special characters, variant spellings, and name variations. The clean data set must, in the vast majority of cases, match the permitted dataset values in reference tables. For instance, the fields for the description and name of the artifact in a cultural heritage dataset originating from a new data source must coincide with the name and description fields already recorded in the reference relation for the item. Implementing a precise and effective matching procedure that can properly clean an incoming data set in the event that it does not precisely match with any dataset value in the reference relation would provide a considerable problem in such a circumstance. An effective matching method is developed in this paper using a novel similarity function that substantially reduces the drawbacks of frequently used comparison functions. On actual datasets, the exhibited technique's efficacy is assessed.

The necessity for fuzzy-join parameterization has grown in importance as a result of recent considerable improvements in scalability. The fuzzy-join implementations specifically propose a rich set of complex setup parameters, most of which have to precisely configured in order to be generated precise

DOI: 10.18421/TEM114-59
https://doi.org/10.18421/TEM114-59

Corresponding author: Jordan Stoikov,
Institute of Mathematics and Informatics, Bulgarian Academy of Sciences, Sofia, Bulgaria.
Email: jstoikov@shieldui.com

Received: 01 September 2022.
Revised: 12 October 2022.
Accepted: 08 November 2022.
Published: 25 November 2022.

© 2022 Jordan Stoikov, Alexandra Nikolova & Vladimir Georgiev; published by UIKTEN. This work is licensed under the Creative Commons Attribution-NonCommercial-NoDerivs 4.0 License.

The article is published with Open Access at https://www.temjournal.com/

and accurate results. This is determined by the necessity to amend the linkage quality for diverse entry tables, incoming from disparate datasets. For instance, Data Ladder [2] makes use of a well-liked fuzzy-join capability that is a component of its toolbox. It makes use of a sophisticated setup user interface with 3 dialogs and 19 adjustable settings. 12 of these choices are Boolean (may be true or false), giving rise to 212 = 4096 distinct combinations, which make manual programming difficult. Similar to this, the well-known open-source fuzzy-join tool py_stringmatchin [3] makes use of a total of 92 parameters. You'll see that we haven't yet considered parameters from numeric domains, such a "similarity threshold" that may contain any value I the range of [0, 1].

End-user inquiries asking for an explanation of how fuzzy-joins may be coded in accordance with various use cases, including how to configure parameters such as similarity-thresholds, weights of tokens, distance-functions, etc., have been seen often in places like Power Query [4] user forums. Users with functional rather than technical background (such as those using Tableau or Excel) need to try many different parameter combinations or use the sub-optimal default parameters. Experienced practitioners can examine entry data and utilize their prior knowledge to make refined estimation of appropriate parameters (based on the trials and errors approach). This study's premise is that this is a serious problem that prevents fuzzy-join from being widely used. In this study, the feasibility of auto-programmable fuzzy-joins using suitable parameters specifically designed for the provided input tables is investigated. The suggested method is designed to function independently and without input from consumers (e.g., educational illustrations with labels for non-linked and linked data). It is based on a fundamental characteristic of fuzzy-join operations that mandates the entry tables to serve the function of an "indicative table," or an optimized principal table with limited or no duplicates. It is well known that reference tables are often used in the literature [5] and incorporated into business systems (e.g., SQL Server [6], OpenRefine [7], etc.). Without requiring labelled data, it is feasible to infer superior fuzzy-join algorithms by taking use of this crucial characteristic of reference tables.

This paper's goal is to investigate a matching technique that may be used across several domains and does not need explicit parametrization. The similarity function that is employed to compare data sets is a crucial part of a matching procedure. The matching procedure should return the reference data set—an ordered data set in the reference relation—that is most similar to the entry dataset value based on the comparison function and an entry dataset.

The fuzzy match similarity (FMS) [8] function that construes the strings as a sequence of tokens and explicitly associates weights indicating their relevance, is the foundation of this work. Compared to matching tuples on low weight tokens, matching data sets on high weight tokens yields better similarity findings. In order to gauge the token significance concept for this resolution, inverse document frequency (IDF) weights have been applied. Accordingly, the token relevance depends on its occurrence, represented by the frequency of its appearance in the reference relation [9]. In order to achieve the objective of returning the nearest reference data set with a high probability, a probabilistic [10] technique is used. The reference relation is initially processed to create the error tolerant index (ETI) [11] relation, which is utilized to get a limited number of potential reference data sets during runtime and compare them to the input tuple. Instead of needing to examine the whole set of reference relation and compare each input dataset value to it, as is customary for the naive technique, this approach develops an "index" on the reference relation with the purpose to efficiently extract a superset of the goal fuzzy matches.

The sequence of the sections of the paper is as follows. Related works are included in Section 2. Information on the similarity function is provided in Section 3. The method for building the ETI and the algorithm that makes it easier to identify the target reference data set value are detailed in Section 4. Results of an experiment using actual data are shown in Section 5.

2. Informational Origin and Associated Work

When measuring similarity, the majority of approximation string matching algorithms ignore variations in token significance.

By pre-processing the text strings, the approximate string-matching approach matching [12] creates q-gram [13] tables that include tuples for each string of a certain length that appears as a substring of a text string that is use as a reference. The list of string IDs for the strings it is a substring of is also included in the record. Only a portion of all q-grams per tuple are chosen in the method [14] to generate the error tolerance index ETI, and the retrieval technique for locating the target reference data sets (ii) encrypts the limits of columns that are particular to relational area.

The IDF weights have been effectively used in the area of information retrieval to make it easier to distinguish between the significance of tokens or words. The use of the presumption that every input token suggested in a query is accurate, however, does not further address the potential problems. Numerous

well-known search engines have lately begun to consider little spelling mistakes. The processed data sets for the fuzzy match [15] operation include relatively few tokens; thus, the incorrect input tokens cannot be ignored since they may be essential for differentiating among thousands of reference tuples. The flaw with the cosine similarity measure with IDF weighting-based clustering and reference matching algorithms [16] is that they ignore incorrect input tokens. The similarity function, which enhances efficiency by noting the variation in input token weights, is explored in this work without assuming the correctness of the input tokens.

Table 1. Bibliographic Reference Relation

ID	Name	City	State	Born
R1	Sophronius of Vratsa	Kotel	BG	1739
R2	Sofroniy Vrachanski	Kotel	BG	1741
R3	Vrachansky	Kotel	BG	1737

Table 2. Entry Bibliographic Dataset Values

ID	Name	City	State	Born
I1	Sofronius of Vratsa	Kotel	BG	1739
I2	Sophronius of Vr.	Kotel	BG	1739
I3	Sofronii Vratchanski	Kotel	BG	1739
I4	Vratchanski Sofronius	Kotel	NULL	1741

In Certain recent generic metric space efforts [17] are too complicated and perform poorly for high-output systems that manage data from external data sources. Furthermore, many of these solutions demand the retention of certain index structures, which makes it difficult to implement them over modern data warehouses (e.g., M-trees).

By using a similarity function and classifying very comparable data sets as duplicates, several recent solutions tackled the related issue of removing "duplicates" in a relation. There are some that employ the notion of edit distance (ED) [18], others are based on IDF weights implemented by cosine similarity [19], some that are grounded on learning similarity functions for derivation of training datasets [20], and others that are based on the use of dimension hierarchies [21]. The productivity demands of the online matching procedure, where input data sets must be swiftly corresponded to the aimed location value set before insertion into the database management system, cannot be answered by any of these approaches since they are all designed for offline usage. A comparable approach to these difficulties is to enhance a connection by removing replacements, and after that commit the supplementary elements via the matching procedure for avoiding the generation of further identical values.

Many successful companies (like Axciom) include address-specific elements into their closed source algorithms for comparing and coupling point of contact information of private or organizational information. The record linkage studies that originate in [22], also take into account the issue of finding identical records across interconnections and utilize a selection similarity functions specific for a given domain.

Amid the unassisted approaches, our assessment assumes that the finely-tuned Excel is a robust reference point as it utilizes a variant of the widespread fuzzy comparison, which is a weighted grouping of several distance methodologies.

Additional entity-matching methods embrace Ditto [23] and AutoEM [24], employing pre-trained models for matching of entities.

Contrary to this the goal of this paper is to progress a strategy that is autonomous of any particular domain.

3. FMS

Times An overview of the FMS function's specification for comparing data sets is provided in this section. The following list provides definitions of the various components:

The minimum amount of character editing actions (remove, addition, and replace) needed in order to convert s_1 to s_2 is identified as the editing distance (ED) among two threads, s_1 and s_2, and it is determined by the maximum lengths of s_1 and s_2. The ED between the strings "Sofroniy" and "Sophronius" for the sample presented in the used figure is 6/10=0.6, and the order of editing actions is shown. The upright lines designate replacements at a rate of 1, or exact matches at a cost of 0. There is always a unit cost associated with the operation of deletion or insertion for italicized letters.

Relation to Reference: Assume that the relation to the reference is R [tid, A_1, ...,A_n], where Ai stands for the i^{th} column. Each A_i is acknowledged to be an attribute of string-value type. The notion that tid, or data set indicator, is a core element of R is also widely accepted. The term "tuple r" refers to a data collection whose tid property takes the value "r". The element ai from the dataset v[r, b_1, ...,b_n] is denoted by the v[i] notation.

Tokenization: The tokenization function, designated by the symbol tok, divides a thread s into a collection of indications, tok(s), that are specified by a set of delimiters. For instance, {Sophronius, Vratsa} is tok(v[1]) of the value set v = [R1, Sophronius of Vratsa, Kotel, BG,]. Please be aware that while creating tokens, case does not matter. The attribute of the column from which a token originates is connected with tokens that are formed from

attribute values of data sets. For instance, col is the tokens' column attribute in tok(v[col]). The sets tok(b_1), ...,tok(b_n) of indications out of the data set v[r, b_1, ...,b_n] are combined into the set tok(v). If a token t occurs in more than one column, it is expected that an individual copy for each column in tok(v) is kept, enabling each copy to be identified by its column attribute. For $1 \leq i \leq n$, a representation is contained in tok(v) if it is in association with the tok(b_i).

Weight Function: With regards to this research, the IDF weight function is modified by considering each data set as a tokenized document. The frequency of occurrence of dataset values v in R, so as to tok(v[i]) includes t will be the existence of the token t in column I, which is signified as freq(t, i). When freq(t, i) > 0, then the IDF element, IDF(t, i) of a token denoted with t, concerning the column signified as i^{th} within the R scheme is calculated appropriately.

$$w(t,i) = IDF(t,i) = \log \frac{|R|}{\text{freq}\ (t,i)}$$

Assuming that the token denoted as t is an unfitting variation of a token from the data set that is used for reference and that the related token is unknown when the token t's occurrence in column i is equal to 0, the token is given the estimated significance of all representations contained the column signified as i^{th} within the R relation.

3.1. Fuzzy Similarity Function

The cost of converting the input dataset from the reference dataset in this research is based on how similar the two datasets are; the greater the similarity, the lower the cost. The three conversion operations—token replacement, insertion, and deletion—are being taken into account. The weight of the token that was changed determines how much each operation will cost. In the context of this investigation, the datasets u and v have the scheme R[A_1, ...,A_n]. Only the case where u signifies the entry value and v is a reference dataset value will be taken into account, with a conversion of u into v as the desired outcome.

(i) Token replacement: It costs ed(t_1, t_2).w(t_1, i) to replace a representation t1 within tok(u[i]) with a represenation t_2 in tok(v[i]). The cost is evaluated as unlimited if t_1 and t_2 come from distinct columns.

(ii) Inserting a token: Entering a representation in u[i] outlays cins.w(t, i) where the representation cins entering component is a fixed cost among 0 and 1.

(iii) Deleting a representation: It costs w(t,i) to eliminate a token t from u[i].

The datasets are compared without taking the tid attribute into account. Each column in u must be changed into v by a series of transformation operations, the cost of which is determined by adding up the expenses of each operation in the sequence. The rate of the lowest cost of the alteration order to change u[i] into v[i] is represented by the modification fee, or tc(u[i], v[i]). The total of the expenses tc(u, v) of transforming u into v in all strings signified as i is the cost tc(u[i], v[i]) of transforming u[i] to v[i].

$$tc(u,v) = \sum_i tc(u[i], v[i])$$

Be By aiding the dynamic programming technique used for the calculation of ED, it is possible to determine the least cost of transformation tc(u[i], v[i]).

The input dataset tuple [Sofronii Vratchanski, Kotel, BG, 1739] in Table 2 and the associated reference tuple [Sofroniy Vrachanski, Kotel, BG, 1741] will be subject to analysis. Two operations must be performed in order to convert u[1] into v[1] at the lowest possible cost: replacing "Sofronii" with "Sofroniy" and "Vratchanski" with "Vrachanski." The costs of these two operations are added up in the function tc(u[1], v[1]), which, when accounting for the unit weights on all tokens, yields a result of 0.97 by adding 0.34 for changing "Sofronii" to "Sofroniy," which has an ED of 0.34, and 0.63 for changing "Vratchanski" to "Vrachanski," which has an ED of 0.63. In this example, the only column-related alteration cost that is not zero is tc(u[1], v[1]).

The FMS between an entry value u and a reference dataset value v is described in this section with regards to the conversion fee tc(u, v). It is expected that the level of importance of all the tokens in the entry dataset value u's token set, tok(u), will be added together to form w(u). What is meant by u and v being similar is:

$$fms(u,v) = 1 - \text{m}\ \left(\frac{tc(u,v)}{w(u)}, 1.0\right)$$

Subsequently six tokens are contained in tok(I1) and every token has a weight of 1, the aforementioned sample involving I3 and R1 has w(I3) = 6.00. Consequently, fms(I3, R1) = 1 - 0.97/6.0 = 0.838. The fee for modification of a filthy entry dataset value into a correct reference dataset value, differentiates from the cost of the opposite transformation, hence the FMS is defined asymmetrically.

3.2. ED and FMS

In order to distinguish between cases where they consent or contradict on the logic for fuzzy matching, this research compares the implied weight assignment technique taken by the ED with that of the FMS for a large subclass of mistakes. The comparison also supports our belief that FMS is the better option in reality.

Subsequently, every entry token is plotted to the token used for reference that matches it the closest, and after every entry token is translated to its equal in the reference dataset value, an input dataset value and its aimed reference dataset value are reliable in the positioning amidst tokens under this errors class. If the list of representations in the entry data set u is $u_1,\ldots,u_m$, sorted by their location in u. In the category of the errors related to order-preserving, the entry representation u_i is changed to the associating representation v_i and $v_1,\ldots,v_m$ is the uniformly arranged set of tokens in the reference tuple v.

Recognize that the $u_i \rightarrow v_i$ alignment is given a significance index similar to max (|ui|, |vi|)/L(u). Since longer tokens have larger weights, ed automatically allocates the alignment between a token and its proportionally to the respective lengths. Because "Sofronii" to "Sofroniy," for instance, is given more weight than "Vratchanski" to "Vratsa," this explains why ed correlates input dataset value R1 (in Table 1) with I3 (in Table 2) rather than the intended aim, R2, which is the proper one. demonstrating that IDF weights are more effective than token lengths in capturing the idea of token significance.

4. FMS Approximation

This section's goal is to arrive at fmsapx, an approximate representation of FMS that may support indexed relationships. A customized version of FMS called fmsapx is created by (i) ignoring the order in which the tokens in the entry and reference dataset values are arranged, and (ii) permitting each entry token to bond with the token that is "closest" to it in the reference tuple. fmsapx is an upper limit of FMS since neglecting these two distinguishing factors while connecting tuples will only lead to an increase in similarity between tuples. For instance, fmsapx measures the dataset values [Sofroniy Vrachanski, Kotel, BG, 1741] and [Vratchanski Sophroniy, Kotel, BG, 1741] as identical since the only difference between them is how the tokens are organized in the first field. In fmsapx, the parallel between sets of token substrings, or "q-gram sets," allows for the measurement of the closeness between two tokens (instead of ED among tokens used in FMS). The cohesiveness between the tiny, probabilistically selected subsets of the two q-gram sets also contributes to a good estimation of this q-gram set parallel. This property will make it easier to establish an indexed relation for fmsapx as it is necessary to identify reference tuples for each input dataset values with tokens that have a certain quantity of specified q-grams with the entry dataset value. The approximate level of q-gram set resemblance among tokens is determined first. Then, employing an "amendment term" thatis solely reliant on the value of q, this similarity is connected to the ED between tokens.

The multitude QGq(s) comprised of q-grams of a string that is made up of a large scope of q subsets of the string for which s and q each stand for a positive integer. For example, the 3-gram set QG3("Sofroniy") consists of {sof, ofr, fro, ron, oni, and niy}. QG(s) is used to signify QGq(s) since q is set to be a constant.

Jaccard Coefficient: Between two sets, S_1 and S_2, the Jaccard coefficient sim(S_1, S_2) is

$$\frac{|S1 \cap S2|}{|S1 \cup S2|}$$

Min-hash [25] Similarity: When a component of S is initiated, the computation of the min-hash signature is finished. Consequently, the likelihood that one element is discovered in $S_1 \cap S_2$ before a distinct one from $S_1 \cup S_2$ is identical to sim(S_1 ,S_2). The token parallel is then characterized in terms of how similar their respective q-gram sets' min-hashes are to one another. Q and H are assumed to be positive integers. Tokens t_1 and t_2 have a min-hash similarity of simmh(t_1, t_2) as follows:

$$\text{sim}_{mh}(t_1, t_2) = \frac{1}{H}\sum_{i=1}^{H} I\left[mh_i\left(QG(t_1)\right) = mh_i\left(QG(t_2)\right)\right]$$

The similarity function fmsapx is first developed, followed by the observations that (i) it is expected to be more than the FMS, and (ii) the prospect of fmsapx to be bigger than FMS may be completed as randomly high by selecting an acceptable min-hash size of signature.

The meaning of fmsapx If d_q = (1-1/q) is an adjustment term, and u, v are two values from a data collection. It applies the following function:

$$fms^{apx}(u,v) = \frac{1}{w(u)}\sum_{i}\sum_{t\in \text{tok}(u[i])} w(t) \cdot \max_{r\in \text{tok}(v[i])}\left(\frac{2}{q}\text{sim}_{mh}(QG(t), QG(r)) + d_q\right)$$

Examine the tuple R2 in 1 and the dataset values I4 in Table 2 of the first. Let q be 3 and H be 2. A token with weight w is indicated by the notation t:w. Assume that the tokens in I4 have the following weights: vrachanski:0.27, sophroniy:0.6, vratsa:1.1,

and 1741:2.1, for a total of 4.07. Let's say that their respective min-hash signatures are [sop, hro], [vra, cha], [vra, tsa], [174, 741]. Sofroniy, Vrachanski, Vratsa, BG, and 1741 are the tokens in R2. Assume that they each have the following min-hash signatures: [sof, ron], [vra, cha], [vra, tsal], [bg], and [174, 741] Then, "Vrachanski" matches "Vrachanski," "Sophroniy" matches "Sophroniy," "vratsa" matches "vratsa," and "1741" matches "1741." Matching "Sofroniy" with "sophroniy" yields the following score: w(sophroniy)*(2/3*0.6 + (1-1/3)) = w (sophroniy). Contrarily, fmsapx(I4, R2) would likewise take into account the fee for resolving discrepancies in the ordered sequence amidst tokens in the exact range of I4 and R2 and the fee for inserting token 'bg'. As a result, every second representation links correctly with an indicative representation, fmsapx(I4, R2) = 4.07/4.07. fms(I4, R2) is thus smaller than fmsapx (I4, R2).

4.1. Conclusion Fuzzy Similarity Function

ETI's main goal is to enable the competent invocation of an applicant set S of indicative data set values for each input tuple u which has resemblance to u higher than the nominal parallel criterion c. As stated in the description of fmsapx, fmsapx (u, v) is calculated by associating the tokens' min-hash mark in tok(v) and tok (u). Consequently, in order to identify the applicant set, we must expertly identify a group of reference dataset values that share min-hash q-grams with each token t in tok(u). Study the sample entry dataset from Figure 1 (Sofronius of Vratsa, Vratsa, BG, 1739). In the picture, the top row shows the representations in the entry value set, the bottom series order sets (S_1 to S_8) of tids of reference dataset values that have corresponding tokens with min-hash signatures included the appropriate q-gram, and the peak row displays the tokens in the input tuple. For instance, the set S_1US_2 consists of reference tuples with tokens in the Name column that include the word "sofronius" as part of their min-hash q-gram. The blending of all Si's includes the applicant set S, and this behavior extends to the q-gram signatures of all tokens. Every q-gram is deposited in ETI together with an ordered set of all the reference tuple tids that have tokens with min-hash signatures that include s, in order to detect such groupings of tids.

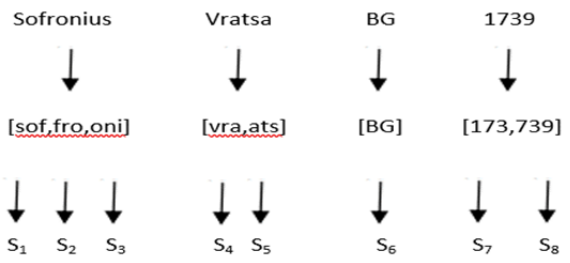

Figure 1. Generation of candidate set

It might be assumed that R is the link to the reference and H signifies the magnitude of the min-hash mark if the building of the ETI is being properly stated.

Table 3. Sample Correlation of ETI

Q-Gram	Coordinate	Column	Frequency	Tid-list
oph	0,5	0,5	0,5	{R1}
oni	2,5	0,5	0,5	{R1}
vra	0,5	0,5	2,5	{R3,R1}
tsa	2,5	0,5	2,5	{R3,R1}
sof	0,5	0,5	0,5	{R2}
vra	0,5	0,5	0,5	{R2}
ius	0,5	0,5	0,5	{R2}
kot	0,5	2,5	3	{R1,R2,R3}
bg	2,5	3	3	{R1,R2,R3}
tel	0,5	3	3	{R1,R2,R3}
173	0,5	4	3	{R1,R2,R3}
739	2,5	4	0,5	{R1}
741	2,5	4	0,5	{R2}
737	2,5	4	0,5	{R3}

Table 3 displays an illustration of an ETI link for the indicative relation in Table 1 where q=3,2 and H=2,5. We presume that a token's min-hash signature is the token itself if its length is less than q. The tuple [R2 Sofroniy Vrachanski, Vratsa, BG, 1741] in Table 1 contains the tid R1 in the tid-order of very q-gram, with the relevant token min-hash signatures being {[ofr, ron], [vra, ach, han, nsk], [vra, tsa], [bg], [174, 741]}.

4.2. The Processing of Query

The algorithm for processing fuzzy matches queries in the study under consideration asks for K fuzzy equivalents of entry data set values u whose resemblances (according to FMS) to u are over a smallest comparison level of c. By employing the ETI effectively, it is aimed to decrease the searches made against the reference relation. We started by outlining the fundamental method, which retrieves tid-order by checking the ETI for every q-gram in all tokens in u's min-hash signatures. Then, to dramatically reduce the amount of ETI lookups, optimistic short circuiting is used, which takes advantage of variations in token significance and the necessity to collect just the K nearest dataset value.

4.2.1. The Basic Algorithm

Following is the fundamental technique for handling the fuzzy match query based on an input data set value called u. The IDF weight w(t) for each token t in tok(u) is computed and demands the t frequency. These occurrences may be reserved in the ETI and retrieved using a separate SQL query for each token. It may be assumed that the token frequencies are currently accessible from the token-

frequency cache. The min-hash mark mh(t) of every representation t follows that. (If $|t| \leq q$, then mh(t)=[t] is defined.) Each q-gram in mh has the weights w(t)/|mh(t)| given to it (t). Using the ETI, is determined an applicant group S of reference dataset values whose parallel with the entry dataset value u (as determined by fmsapx and fms) is greater than c. To check if the dataset values' similarities to u (as per FMS) are indeed higher than c, all dataset values in S are derived based on the indicative relation. The K dataset values with the K highest resemblance scores are remitted from among those that pass the verification test.

5. Some Experiments

As the reference relation, a clean data set of values from a primary operational data warehouse is used. By including mistakes in randomly chosen subsets of dataset values, the input datasets are created. All features of actual data like differences in token extents and token occurrences, are conserved in the problematic entry dataset values. There are two primary categories of techniques for injection of error: Type I method, in which errors contained in tokens have an identical chance of occurring in any given column, and Type II method, in which the weight of errors in tokens is correlated to the frequency of those errors occurring in a dataset column. This is a typical scenario because the likelihood of false versions of something depends on how frequently it happens. Because mistakes in low significance, high occurrence tokens do not influence greatly the FMS similarity, the type II error technique of introducing errors is prone to FMS.

Table 4. Errors descriptions and type

ej	Description of Error
1	Error of spelling: change representation
2	Token substitution
3	Lack of value: u[i] = null
4	Reduction: reduce u[i] by 4 or less characters
5	Token consolidation
6	Transposition of representation

Implemented evaluation metrics include:

The time needed to process a group of entry dataset values by employing the logic for fuzzy matching, split up by the required timeline to process an entry dataset value employing the naive technique is known as the "normalized elapsed time" (comparing an input dataset value to each reference tuple). A fuzzy match algorithm is said to outperform a naive algorithm if its standardized time is lower than the total number of input tuples.

Accuracy: The proportion of the entry dataset values for which the fuzzy match algorithm correctly classifies the original dataset value that gave rise to the incorrect input tuple.

Setting of parameters: In the study's final tests, we decide on K=1 (which denotes that only the closest fuzzy match is recovered), q=3 for the size of the q-gram, c=0.0 for the minimal similarity threshold, and cins=0.4 for the token insertion factor, which is necessary for evaluating FMS.

The methods and parameters that will be examined are indicated in this section using the following representation. The following symbols are used to denote the signature calculation approach: A_H, $A \in \{Q, Q+T\}$ and $H \geq 0$. Q+T stands for a combination of token signatures and q-grams, whereas Q stands for solely q-grams. H stands for the quantity of the marked q-grams. For instance, Q+T_2 signifies a mark that contains a token together with two q-grams, but Q+T_0 denotes a signature that contains just tokens.

First, the quality of the FMS and ed is assessed in order to gauge accuracy, and then the fuzzy match algorithm's performance is gauged.

5.1. Evaluation of ED and FMS

When comparing two datasets—one with Type I error insertion methods and a second one with Type II inaccuracy insertion methods—the resulted grade of FMS is superior to ed. These two datasets each include about 1000 dataset values. As a result, the likelihood of error in each column is 0.91, 0.49, 0.49, and 0.61.

Instead of comparing the effectiveness of algorithms that discover fuzzy matches, the goal of this experiment is to determine the quality of similarity functions. To find the optimal fuzzy match for each entry value set, the naive approach is used for this purpose

The reference table shows the accuracy differences between FMS and ed for each dataset. FMS outperforms ed, especially for datasets formed with Type II errors as opposed to Type I errors. Only datasets using the Type I error technique were highlighted in the comparison in order to eliminate the bias in favour of the FMS.

Table 5. Accuracy of FMS and ED

	FMS	ED
Accuracy on Type I	67%	61%
Accuracy on Type II	94%	68%

Table 6. Probabilities of errors for data set initiation

Dataset	Error Probabilities: [Name, City, Country, Post. Code]
D1	[0.92, 0.92, 0.92, 0.92]
D2	[0.81, 0.6, 0.6, 0.7]
D3	[0.71, 0.6, 0.6, 0.28]

5.2. Algorithms Precision

Based on datasets D1, D2, and D3, which were produced by employing the type I error approach for insertion, several methods' accuracy is evaluated. Table 6 shows the error probabilities for each column for these three datasets. For the purposes of this experiment, D3 is more pristine than D2, which is more spotless than D1. D1, D2, and D3 each contain 1541 tuples. There are around 100,000 tuples in the customer relation that is used as the foundation for the reference relation in all tests. Figure 2 displays the outcomes of which we infer the following.

Figure 2. Accuracy

(i) Min-hash signatures meaningfully enhances precision: Q_H is more accurate (for H>0) (6% to 27%) than Q+T_0 (with representations only).

(ii) Based on the observation that H > 0, Q+T_H is equally precise as Q_H, the addition of tokens to the signature has no effect on the accuracy.

(iii) Even little signatures boost accuracy more than larger ones: Although Q_2 is more precise than Q_1, there is little to no difference between Q_2 and Q_3.

5.3. Efficiency

This experiment measures the standardized required time for completing the execution of the procedure for fuzzy matching, the number of potential indicative dataset values retrieved for every entry dataset value, and the tids handled for the entry dataset values in order to demonstrate the productivity of the implemented algorithms. The normalized elapsed durations are validated by Figure 3, from which the following may be deduced:

(i) The fuzzy match algorithms used in this research process all 1541 input tuples in less than 2.5 seconds on average, making them 2 to 3 times faster than the naïve technique.

(ii) As the size of the signature increases, the query's processing time decreases. The existence of more q-grams supports improved identification of variances among the similarity scores of tids, even though the ETI must be looked up for more q-grams. Accordingly, signature size decreases the approximate sum of indicative value sets, retrieved for every entry value set.

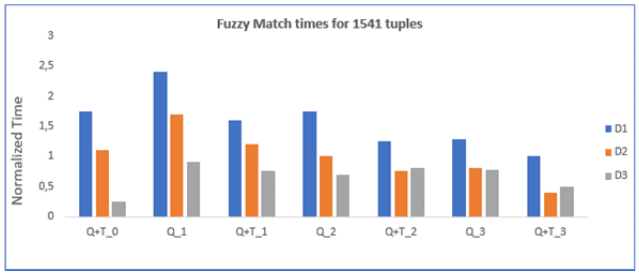

Figure 3. Normalized Elapsed Times

6. Conclusion

This study advances a precise FMS function for matching compromised entry dataset values with flawless value sets from an indicative relation by including the ED similarity into the idea of tokens and their importance. Following that, the ETI and a powerful algorithm are used to categorize the closest fuzzy matching reference tuples with a high degree of probability. By employing genuine datasets is confirmed the efficiency of the implemented similarity function and the precision of the respective algorithms in the context of optimizing the recall while meeting the required precision target.

Acknowledgements

This work is funded in part by CLaDA-BG, the Bulgarian National Interdisciplinary Research e-Infrastructure for Resources and Technologies in favor of the Bulgarian Language and Cultural Heritage, part of the EU infrastructures CLARIN and DARIAH, Grant number DO01-301/17.12.2021.

References

[1]. Manghi, P., & Mikulicic, M. (2011, October). PACE: A general-purpose tool for authority control. In *Research Conference on Metadata and Semantic Research* (pp. 80-92). Springer, Berlin, Heidelberg.

[2]. Ziad, Z. (2022). How Best In Class Fuzzy Matching Solutions Work: In Combining Established and Proprietary Algorithms. *Suffield.* Retrieved from: https://dataladder.com/fuzzy-matching-software/ [accessed: 10 June 2022].

[3]. Martelli, A., Ravenscroft, A., & Holden, S. (2017). Python in a Nutshell. (3rd, Ed.) *O'Reilly Media, Inc.*

[4]. Rad, R., & Etaati, L. (2021). In The Definitive Guide to Power Query in Power BI and Excel (pp. 130-147). *RADACAD Systems Limited.*

[5]. Pang, C., Gu, L., Hansen, D., & Maeder, A. (2009). Privacy-preserving fuzzy matching using a public reference table. In *Intelligent Patient Management* (pp. 71-89). Springer, Berlin, Heidelberg.

[6]. West, R., Zacharias, M., Assaf, W., Aelterman, S., Davidson, L., & D'Antoni, J. (2020). *SQL Server 2019 Administration Inside Out,* (pp. 342-360). Microsoft Press.

[7]. Verborgh, R., & De Wilde, M. (2013). In Using OpenRefine (pp. 121-127). *Packt.*

[8]. Yu, M., Li, G., Deng, D., & Feng, J. (2016). String similarity search and join: a survey. *Frontiers of Computer Science*, *10*(3), 399-417.

[9]. Bruck, A., & Tilahun, T. (2015). Enhancing amharic information retrieval system based on statistical co-occurrence technique. *Journal of Computer and Communications*, *3*(12), 67.

[10]. Sayers, A., Ben-Shlomo, Y., Blom, A. W., & Steele, F. (2016). Probabilistic record linkage. *International journal of epidemiology*, *45*(3), 954-964.

[11]. Jia, L., Li, M., Miao, D., & Xi, J. (2012). ETI: an efficient index for set similarity queries. Frontiers of Computer Science, 700-712.

[12]. Navarro, G., Sutinen, E., Tanninen, J., & Tarhio, J. (2000, June). Indexing text with approximate q-grams. In *Annual Symposium on Combinatorial Pattern Matching* (pp. 350-363). Springer, Berlin, Heidelberg.

[13]. Christen, D. M. (2012). Concepts and Techniques for Record Linkage. *Entity Resolution, and Duplicate Detection,© Springer-Verlag Berlin Heidelberg*.

[14]. Sun, D., & Wang, X. (2018, June). MLS-Join: An Efficient MapReduce-Based Algorithm for String Similarity Self-joins with Edit Distance Constraint. In *International Conference on Cloud Computing and Security* (pp. 662-674). Springer, Cham.

[15]. Li, P., Cheng, X., Chu, X., He, Y., & Chaudhuri, S. (2021, June). Auto-FuzzyJoin: Auto-Program Fuzzy Similarity Joins Without Labeled Examples. In *Proceedings of the 2021 International Conference on Management of Data* (pp. 1064-1076).

[16]. Nguyen, T. P. Q., & Kuo, R. J. (2019). Partition-and-merge based fuzzy genetic clustering algorithm for categorical data. *Applied Soft Computing*, *75*, 254-264.

[17]. Dong, X., Halevy, A., & Madhavan, J. (2005, June). Reference reconciliation in complex information spaces. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data* (pp. 85-96).

[18]. Efthymiou, V., Papadakis, G., Papastefanatos, G., Stefanidis, K., & Palpanas, T. (2017). Parallel meta-blocking for scaling entity resolution over big heterogeneous data. *Information Systems*, *65*, 137-157.

[19]. Cohen, W. W. (2000). Data integration using similarity joins and a word-based information representation language. *ACM Transactions on Information Systems (TOIS)*, *18*(3), 288-321.

[20]. Sarawagi, S., & Bhamidipaty, A. (2002, July). Interactive deduplication using active learning. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 269-278).

[21]. Ananthakrishna, R., Chaudhuri, S., & Ganti, V. (2002, January). Eliminating fuzzy duplicates in data warehouses. In *VLDB'02: Proceedings of the 28th International Conference on Very Large Databases* (pp. 586-597). Morgan Kaufmann.

[22]. Herzog, T. N., Scheuren, F. J., & Winkler, W. E. (2007). *Data quality and record linkage techniques* (Vol. 1). New York: Springer.

[23]. Li, Y., Li, J., Suhara, Y., Doan, A., & Tan, W. C. (2020). Deep entity matching with pre-trained language models. *Proceedings of the VLDB Endowment*, *14*(1), 50-60.

[24]. Zhao, C., & He, Y. (2019, May). Auto-em: End-to-end fuzzy entity-matching using pre-trained deep models and transfer learning. In *The World Wide Web Conference* (pp. 2413-2424).

[25]. Sohrabi, M. K., & Azgomi, H. (2017). Parallel set similarity join on big data based on locality-sensitive hashing. *Science of computer programming*, *145*, 1-12.