An Improved N-gram Distance for Names Matching

Conference Paper · December 2019

DOI: 10.1109/ICOICE48418.2019.9035154

4 authors:

Salah Al-Hagree
Sana'a University
24 PUBLICATIONS **112** CITATIONS
SEE PROFILE

M. Sanabani
Thamar university
22 PUBLICATIONS **115** CITATIONS
SEE PROFILE

Mohammed Hadwan
Qassim University
63 PUBLICATIONS **771** CITATIONS
SEE PROFILE

Mohammed Abdullah Al-Hagery
Qassim University
62 PUBLICATIONS **385** CITATIONS
SEE PROFILE

An Improved N-gram Distance for Names Matching

Salah AL-Hagree
Department of Computer Sciences &Information Technology
Ibb University, Yemen
Email:s.alhagree@gmail.com

Maher Al-Sanabani
Faculty of Computer Science and Information Systems
Thamar University, Yemen
Email: M.sanabani@gmail.com

Mohammed Hadwan
IT department, college of Computer.
Qassim University
Email: M.hadwan@qu.edu.sa

Mohammed A. Al-Hagery
Computer Science Department, College of Computer, Qassim University, KSA,
Email:hajry@qu.edu.sa

***Abstract* — *N*-grame distance (N-DIST) was originally developed by Kondrak's lately to measure the distance between two strings. It was found that this distance can be computed by an elegant dynamic programming procedure. The N-DIST has played important roles in a wide array of applications due to its representational efficacy and computational efficiency. To effect a more reasonable distance measure, the normalized edit distance was proposed. Many algorithms and studies have been dedicated along this line with impressive performances in recent years. There is, however, a fundamental problem with the original definition of N-DIST that has remained elusive: its context-free nature. In determining the possible actions, i.e., deletion, insertion, transposition and substitution, consider action was given to the local behaviors of the string question that indeed encompass great amount of useful information regarding its content. This framework, two operations are developed. The original N-DIST algorithm does not consider the transposition operations, and the algorithm has fixed the cost of insertion and deletion operations. In addition, the proposed E-N-DIST algorithm computes the costs of substitution and transposition operations is dependent on 2^{n+1} -1 states while the original N-DIST algorithm has been only dependent on 2^n states. In this paper, the experiments carried out show the E-N-DIST algorithm, which gives a sort of results, which are more accurate than the algorithms under discussion.**

Keywords*:— *Name Matching, Levenshtein Distance, N-grame distance, Dynamic Programming.

I. INTRODUCTION

Name matching is a universal technique's to solve problems in different areas such as Text Mining[ ], natural language processing (NLP)[ ], computer vision[ ], image processing[ ], pattern recognition [40], Computational biology "DNA" [8, 9], Spelling correction [10, 11, 12, 13], Text retrieval [14, 15] Handwriting recognition [16] and Linking database [17, 18] and Name Recognition [4], etc... like, from database deduplication to vetting names against watch lists for fraud prevention, to identifying a person in a database by name alone. As a result, deciding whether two names represent the same identity is a substantial technical challenge.

However, name variations due to errors, data-entry errors such as transpositions, misspellings, or name variants introduced due to added or missing characters, hyphenation, accents and spaces of a great significance.

Therefore, experts in this field of study developed and used a number of name matching algorithms. Where measures are classified as similarity (as closer to zero as more related are the names) or distance (as greater is the value as more related are the names). A normalized distance / similarity measure keeps a scale between different distance values. Distance and Similarity measures detect the particular sound-alike (phonetic case) and look-alike (orthographic cause) issue. It is of great importance to measure the similarity or dissimilarity between two strings for the name matching.

Edit distance (ED). Given the two names X and Y as sequences of size n and m, respectively, ED (also called Levenshtein distance (LD)) refers to the minimum cost of editing operations (deletion, substitution and insertion) to convert the sequence X into Y [3, 4–5]. For example, pair cycloserine and cyclosporine have a distance of two because there are needed at least two edit operations (a substitution p→e and an elimination of letter o) to transform cycloserine in cyclosporine.

Damerau–Levenshtein distance (DLD) in [13, 17] is quite similar to Levenshtein distance. The only difference is that Damerau–Levenshtein distance allows one more edit operation: the transposition of two adjacent characters.

Modified Damerau-Levenshtein Distance (MDLD) Algorithm Purpose: Perform Modified Damerau-Levenshtein Distance test on two input strings, supporting block transpositions of multiple characters have been applied on Oracle database and time complex O(N3), (is available on [14].

The pioneering work is due to Kondrak's [1] who defined the N-DIST D(X, Y) between two string X; Y as the minimum total number of changes C, insertions I, and deletions Required to change Y to X, i.e.

The idea introduced by Kondrak [1-2] is to use character n-grams instead of single letters when measuring edit distance which is called N-DIST. This algorithm has used a new family of word similarity measures based on n-grams, which are intended to combine the advantages of the unigram and the n-gram measures [20, 21-22, 24]. This measurement has been evaluated on three different word-comparison tasks: the identification of confusable drug names, translational cognates, and genetic cognates. The results of these experiments show that the new measurement of n-gram distance outperforms their unigram equivalents. The compared algorithm has been used the n-gram distance measurement for measuring gradual similarity of n-grams in BI, Bigram (where n = 2) and TRI, Trigram (where n=3). Since they measure distances rather than similarity.

In this paper, we propose a new algorithm based on N-DIST that increase the accuracy to name matching.

II. DEFINITIONS

The mentioned measures are classified as orthographic (distance and similarity) or phonetic in relation to the used approach to name matching. In this paper, we focus on orthographic distance measures.

The Edit distance (ED gives the two names X and Y as sequences of size n and m, respectively, ED (also called Levenshtein distance (LD)) refers to the minimum cost of editing operations (deletion, substitution and insertion) to convert the sequence X into Y [3, 4–5]. For example, the edit distance between Zantac and Xanax is 3 because the minimum transformation involves two substitutions (Z → X and c → x) and one deletion (letter t). In this paper, editing operations (deletion and insertion) have a cost of one is w_1 and w_2, respectively. In this case, the edit distance between X and Y is given by Lev(i, j) computed by the following recurrence in Equation (1) [23]:

$$\mathbf{Lev}_{s,t}(i,j) = \begin{cases} \mathbf{Max}(i,j) & (i = 0 \text{ or } j = 0) \\ \boldsymbol{Min} \begin{cases} \mathbf{Lev}_{s,t}(i,j-1) + \mathbf{w}_1 \text{ , is 1.} \\ \mathbf{Lev}_{s,t}(i-1,j) + \mathbf{w}_2 \text{ , is 1.} \\ \mathbf{Lev}_{s,t}(i-1,j-1) + \mathbf{w}_3 \end{cases} & \end{cases} \quad (1)$$

This computes the cost of substitution (replacement, Cr) the cost can be noted Cr. the computed Cr can take value within an interval [0.0, 1.0].

Cr =W_3, if ($S_i \neq T_j$, is 1.else $S_i = T_j$, is 0.) {**Case** 1}

Damerau–Levenshtein distance (DLD)[13, 17] is quite similar to Levenshtein distance. The only difference is that Damerau–Levenshtein distance allows one more edit operation: the transposition of two adjacent characters. The DLD algorithm between two strings s and t is described by the recursive relation in Equation. (2).

$$\mathbf{DLev}_{s,t}(i,j) = \begin{cases} \mathbf{Max}(i,j) & (i = 0 \text{ or } j = 0) \\ \boldsymbol{Min} \begin{cases} \mathbf{DLev}_{s,t}(i,j-1) + \mathbf{w}_1 \text{ , is 1.} \\ \mathbf{DLev}_{s,t}(i-1,j) + \mathbf{w}_2 \text{ , is 1.} \\ \mathbf{DLev}_{s,t}(i-1,j-1) + \mathbf{w}_3 \\ \mathbf{DLev}_{s,t}(i-2,j-2) + \mathbf{w}_4 \end{cases} & \end{cases} \quad (2)$$

This computes the cost of substitution and transposition, the cost can be denoted as C_{rt}, it can take a value within an interval [0.0, 1.0].

- $\mathbf{C_{rt}} = W_3$, if ($S_i \neq T_j$, is 1.else $S_i = T_j$, is 0.) {**Case** 1}
- $\mathbf{C_{rt}} = W_4$, if ($S_i \neq T_{j-1}$ and $S_{i-1} = T_j$, is 0. Else, is 1) {**Case** 2}

In [14, 16], A Modified Damerau-Levenshtein Distance (MDLD) algorithm purposed and tested on two input strings, supporting block transpositions of multiple characters have been applied on Oracle database and time complex $O(N^3)$, (is available on [14]. This computes the cost of substitution and transposition. The cost can be denoted by C_{rtm}. It can take value within an interval [0.0, 1.0].

- $\mathbf{C_{rtm}} = W_3$, if ($S_i \neq T_j$, is 1.else $S_i = T_j$, is 0.) {**Case** 1}
- $\mathbf{C_{rmt}} = W_4$, ($S_i \neq T_{j-1}$ and $S_{i-1} = T_j$, is 0. Else, is 1){**Case** 2}
- $\mathbf{C_{rmt}} = W_5$, = d[(j - block * 2) * sl + i - block * 2] + cost + block - 1; if (pure_levenshtein == 0 and i >= 2 * block and j >= 2 * block and swap1 == 1 and swap 2 == 1) **{Case** 3}.

(block transpositions of multiple characters, when bloke is two and three).

The N-gram distance proposed by Kondrak [6, 23, 24] and it combines features implemented by grams of size b, non-crossing-links constraints and the first letter it is repeated at the begging of the drug name. To clarify this measurement in the compared algorithm, the two strings are given: $X=x_1 \ldots x_k$ and $Y=y_1 \ldots y_k$. Let $T_{i;j}=(x_1 \ldots x_k, y_1 \ldots y_k)$ and $T^n_{i;j}==(x_{i+1} \ldots x_{i+n}, y_{j+1} \ldots y_{j+n})$. These strings are divided into all possible sub-strings of N consecutive letters then aligned and compared. Equation (3) presents the score of n-gram distance measure.

$$\mathbf{d_n}(\mathbf{T^n_{i,j}}) = \frac{\mathbf{1}}{\mathbf{n}} \sum_{\mathbf{u=1}}^{\mathbf{n}} \mathbf{d_1}(\mathbf{x_{i+u}}, \mathbf{y_{j+u}}), \quad (3)$$

The N-DIST $\mathbf{NDIST}_{s,t}$ between two strings s and t is described by the recursive relation shown in Equation. (4).

$$\mathbf{NDIST}_{s,t}(i,j) = \text{Min} \begin{cases} \text{NDIST}_{s,t}(i-1,j) + 1. \\ \text{NDIST}_{s,t}(i,j-1) + 1. \\ \text{NDIST}_{s,t}(i-1,j-1) + \mathbf{d_n}(\mathbf{T^n_{i,j}}). \end{cases} \quad (4)$$

Fig 1. shows the distance matrix of the 2 strings "Zantac" and "Zyrtec" with (BI) Bigram (n=2) in the compared algorithm.

1	2	3	4	5	6
- **Z**	**Z** a	a n	n **t**	**t** a	a **c**
| |	|		|	|	|
- **Z**	**Z** y	y r	r **t**	**t** e	e **c**
0	0.5	1	0.5	0.5	0.5

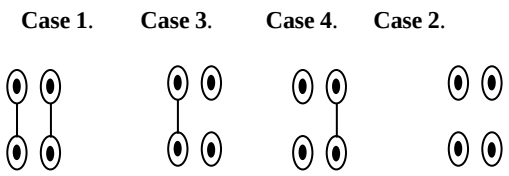
Fig 1.The distance matrix of strings " Zantac " and " Zyrtec ".

The distance between "Zantac" and "Zyrtec" is equal to 3. The compared algorithm incorporates both normalization and affixing to measure the distance. The affixing method aims at emphasizing the initial segments, which tend to be much more important than final segments in determining the similarity of word. A unique special symbol is defined for each letter of the

original alphabet. Each word is augmented with a prefix composed of n − 1 copies of the special symbol that corresponds to the initial letter of the word. For example, if n = 3, Zantac is transformed into -- Zantac. The compared algorithm gives the same cost for (insert, delete and replace) edit operations as shown in Figure 2. With BI (n=2) and it does not give any attention to letter's similarity. Therefore, the compared algorithm does not give accurate results when applied to the English language and other languages.

The N-DIST algorithm gives different costs for substitution and transposition of two final letters of the word strings last, operations as shown in Figure 2 and equation (4).

$$\mathbf{d_n(T^n_{i,j})} = \begin{cases} \mathbf{w_1} = 0, \text{if}(S_{i-1} = T_{j-1}) \text{ and } (S_i = T_j) \text{ Case 1} \\ \mathbf{w_2} = 1, \text{if}(S_{i-1} = T_{j-1}) \text{and}(S_i = T_j) \text{ Case 2} \\ \quad \text{and}(S_{i-1} \neq T_j) \text{ and } (S_i \neq T_{j-1}) \\ \mathbf{w_3} = 0.5, \text{if}(S_{i-1} = T_j) \text{ and } (S_i = T_{j-1}) \text{ Case 3} \\ \mathbf{w_4} = 0.5, (S_{i-1} \neq T_{j-1}) \text{ and } (S_i = T_j) \text{ Case 4} \end{cases} \quad (4)$$



cost= $\mathbf{w_1}$ is 0., cost= $\mathbf{w_3}$ is **0.5**., cost= $\mathbf{w_4}$ is **0.5**., cost= $\mathbf{w_2}$ is 1.

Fig 2. The substitution costs with BI in the N-DIST algorithm.

In this paper, we used the similarity measures for algorithms (LD, DlD, MDLD and N-DIST) in Equation (5). Where $\mathbf{distance}_{s,t}(i,j)$ denotes the between the strings Si and Tj and max (|si|, |tj|) denotes the maximum value of the characters contained in the strings Si and Tj.

$$\boldsymbol{Sim\_Func}_{s,t}(i,j) = 1 - \frac{\mathbf{Distance}_{s,t}(i,j)}{\mathbf{Max}(|s|,|t|)} \quad (5)$$

III. Related Work

Kondrak [1] proposed the orthographic N-DIST distance and the phonetic Aline similarity [6, 7] where the recall metric is used to evaluate the results of 12 measures. Kondrak [1] concludes that N-DIST is the best orthographic measure. Furthermore, the average of Bisim, Aline, Prefix, and NED measures outperform to N-DIST [7]. The pseudo code of the N-DIST algorithm is in (is available on [8]. In [9, 10] have been used N-DIST algorithm in different applications based on the English language while [11, 12] enhanced N-DIST algorithm for matching Arabic names.

The main contribution of this paper is maximizing the efficiency of name matching algorithm by doing the following: enhancing the N-DIST algorithm in Latin based language by taking into account multiple states of transposition operation to handle different errors.

The enhanced algorithm has been called E-N-DIST algorithm.

IV. The Proposed Enhancement N-DIST Algorithm

This section presents the proposed enhancement for N-DIST algorithm that is called E-N-DIST algorithm.

The E-N-DIST algorithm is unlike the original N-DIST in following aspects:

- The E-N-DIST algorithm computes the costs of substitution and transposition operations dependent the on $\mathbf{2^{n+1}-1}$ states in equation (6), while the original N-DIST algorithm has been only dependent on $\mathbf{2^n}$ states in equation (7). The original N-DIST algorithm does not consider the transposition operations.
- The E-N-DIST algorithm computes the estimated cost of deletion and insertion operations depending on the number of states divided by N while, the original N-DIST algorithm has fixed the cost of deletion and insertion operations. Therefore, the cost value is one for any insertion and deletion operations.

$$\text{Number of multiple states} = 2^{n+1} - \mathbf{1} \quad (6)$$

$$\text{Number of Stats} = 2^n \quad (7)$$

The main goal of the E-N-DIST algorithm is tailored to get a high accurate name matching algorithm by handling a different type of spelling errors for Latin based language specially English language.

A) Definition of E-N-DIST Distance

The E-N-DIST algorithm has the same structure and component as the N-DIST algorithm but the E-N-DIST algorithm is founded on for increasing accuracy for name matching it is needed to find the set of weights W= {w_1;w_2;...; w_n} of the scale of distance for **E-N-DIST** by equation (6). If n=2, find the set of weights W= {w_1;w_2;... ;w_7} or if n=3, find the set of weights W= {w_1;w_2;... ;w_{15}} or if n=4, , find the set of weights W= {w_1;w_2;... ;w_{31}}.., etc. While the original **N-DIST** algorithm for name matching it is needed to find the set of weights W= {w_1; w_2; ...; w_n}, by equation (7). If n=2, find the set of weights W= {w_1;w_2;... ;w_4} or if n=3, find the set of weights W= {w_1;w_2;... ;w_8}or if n=4, find the set of weights W= {w_1;w_2;... ;w_{16}}.., etc.

The following sub-sections illustrate how the E-N-DIST algorithm calculates the cost of edit operations, which are:

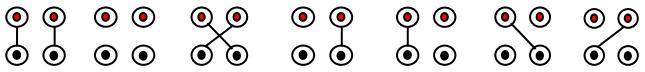
1. Transposition: transposition of the character ‘e’ with a ‘d’ in ‘preceed’
2. Replacement: replacement of the character ‘e’ with ‘s’ in ‘promiss’.
3. Insertion: Add a character ‘l ’ in ‘allmost ’.
4. Deletion: Omission of the character ‘m’ from ‘dilema’.

A. *The Cost of substitution and transposition Operations*

The proposed E-N-DIST algorithm introduces new states as weights cases for the transposition cost and substitution cost operations. These weights depend on N-gram, for example, the number of weights is determined depend on N-grams by equation (7) when n=2. These states are weighted as symbols w_1, w_2, w_3, w_4, w_5, w_6 and w_7 are used to adapt to operational environment and get results that are more accurate on different situations. The cost of transposition and substitution in proposed E-N-DIST algorithm is computed dependent on the weights of states that have have shown in Figure 3 and in equation (8). While the original N-DIST algorithm (cf. Eqs. 4)

$$\mathbf{d_n}(\mathbf{T^n_{i,j}}) = \begin{cases} \mathbf{w_1} = 0, \text{if}(S_{i-1} = T_{j-1}) \text{ and } (S_i = T_j) \text{ Case 1} \\ \mathbf{w_2} = 1, \text{if}(S_{i-1} = T_{j-1}) \text{and } (S_i \neq T_j) \text{ Case 2} \\ \quad \text{and}(S_{i-1} \neq T_j) \text{ and } (S_i \neq T_{j-1}) \\ \mathbf{w_3} = 0.5, \text{if}(S_{i-1} = T_j) \text{ and } (S_i = T_{j-1}) \text{ Case 3} \\ \mathbf{w_4} = 0.5, \text{if}(S_{i-1} \neq T_{j-1}) \text{ and } (S_i = T_j) \text{ Case 4} \\ \mathbf{w_5} = 0.5, \text{if}(S_{i-1} = T_{j-1}) \text{ and } (S_i \neq T_j) \text{ Case 5} \\ \mathbf{w_6} = 0.5, \text{if}(S_{i-1} = T_j) \text{ and } (S_i \neq T_{j-1}) \text{ Case 6} \\ \mathbf{w_7} = 0.5, \text{if}(S_{i-1} \neq T_j) \text{ and } (S_i = T_{j-1}) \text{ Case 7} \end{cases} \quad (8)$$

Case 1. Case 2. Case 3. Case 4. Case 5. Case 6. Case 7.



$C_{st} = w_1$. $C_{st} = w_2$. $C_{st} = w_3$. $C_{st} = w_4$. $C_{st} = w_5$. $C_{st} = w_6$. $C_{st} = w_7$.

Fig 4. Weighted cases substitution and transposition

The function initialied to N after that depend on the condition of the states the cost will be decreased by 1. Such as shown in Figure 5.

The weights of different states are computed depending on the function that is shown in Figure 5. These weights are as an example when N=2. To clarify the value of weights as shown in presented in (cf. Eqs. 8). These values are computed in the following:

- The initial value of cost=N when N=2.
- For each similarity between letters in different positions will be decreased the value by one. The weight =cost / N for example in Case 1 w_1= 0 / 2=0, w_2= 2 / 2=1 in Case 2 and w_4= 1 / 2=0.5 in Case 4, Case 5, Case 6 and Case 7.

Function. To Compute the Cost of Substitution and Transposition of N-gram Letters

Input: **N-gram** Letters (Letters1 and Letters2)
Output: Cost c n-gram Distance (**cost_s_t**)

Decimal Substitution and Transposition Op (so_name [ ], tar_name[ ])
if (ni< n - 1)
{
if (so_name[i - 1 + ni] == tar_name[ni])
{
cost_s_t --;
}
else if (so_name[i - 1 + ni] == tar_name[ni + 1])
{
cost_s_t --;
}
else if (so_name[i - 1 + ni+1] == tar_name[ni])
{
cost_s_t --;
}
if (so_name [i - 1 + ni] == 0)
{ //discount matches on prefix
tn--;
} }
else if (ni>= n - 1)
{
if (so_name[i - 1 + ni] == tar_name[ni])
{
cost_s_t --;
}
else if (so_name i - 1 + ni] == tar_name[ni - 1])
{
cost--;
}
else if (so_name [i - 1 + ni-1] == tar_name [ni])
{
cost_s_t --;
}}
Return Cost

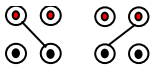
Fig 5. The function of substitution and transposition operations of N-gram Letters.

B. *The Cost of Deletion and Insertion Operations*

In the E-N-DIST algorithm, the cost of insertion and deletion is splitting to two cases such that shown in Figure 6. Each case takes half weights (0.5) for more detecting errors for insertion and deletion operations. Unlike the original N-DIST that give (1) for the cost of deletion and insertion as a fixed value. The function of deletion and insertion operations return the cost of deleting and inserting the letter (identical to primary axis's) depending on the conditions which are for example N=2 as shown in Figure 6. The inserting and deleting returns the value w_8 and w_9 respectively as the cost of deleting and inserting the letter (identical secondary axis's). These weights handle words, instead of a duplicate letter or deleting the duplicated.

The function initializes the value of N after that depends on the condition of the states the cost will be decreased by 1. Such as shown in Figure 7.

Case 8. Case 9.



$C_{st} = w_8$. $C_{st} = w_9$.

Fig 6. Weighted cases inserting and deleting.

Function. To Compute the Cost of Deleting and Inserting of N-gram Letters

Input: **N-gram** Letters (Letter)
Output: Cost Deleting and Inserting Distance (**cost**)
Decimal Deleting and Inserting Op (so_name$_{[\ ]}$, tar_name$_{[\ ]}$)
if (ni< n - 1)
{
if (so_name $_{[i - 1 + ni]}$ == tar_name[ni + 1])
{
costid--;
}
}
else if (ni>= n - 1)
{
if (so_name$_{[i - 1 + ni]}$ == tar_name[ni-1])
{
costid--;
}} **end if**
Return costid

Fig 7. The function of inserting and deleting operations of N-gram letters.

The pseudo code of the E-N-DIST algorithm is in (is available on [31].

V. The Experiment and Results

This section shows the experiments that have been carried in this paper to illustrate the proposed E-N-DIST algorithm and compare it against different algorithms.

The following sub-sections A. presents the data preparation of datasets which are used in this paper while the sub-sections B. and C. will elaborate the comparison study which is carried out to evaluate the performance of the proposed algorithm for name matching.

A. Data Preparation of Datasets

This part is devoted to explaining the names of Multilanguage used to test the proposed algorithms for name matching. A collection of datasets were used in these experiments for further exploration to test E-N-DIST algorithm and compare it against different algorithms.

This is due to the absent of the standard collection of names exists. Therefore, eleven datasets extracted manually form that name, which is shown in Table 2. All the datasets have multi language such English, Portuguese and Arabic names. Each dataset contains some names with different possible variation of errors like spelling errors and typographical of same names. A collection of all kind of variation is considered in the variation of datasets

TABLE 2. NAME MATCHING DATASETS.

No	Database Name	Description	Source
1	**Dataset 4**	These found in the employee's database of the Education ministry in Yemen., 600 pairs and Arabic	[11]
2	Dataset 5	Spell error, (14 and 60 pairs, **Subset**) in 4013 pairs and English	[1][28]
3	Dataset 6		
4	Dataset 7		
5	Dataset 8	Spelling correction, word corrections in Portuguese. 120 pairs and Portuguese	[26]
6	Dataset 9	Scientific Name 'CAAB'	[27]
7	Dataset 10	Scientific Name 'Dalcin name pairs', English and 171 pairs	
8	Dataset 11	Scientific Name 'CAABWEB', English and 2047 pairs	
9	Dataset 12	Scientific Name 'GRIN genera', English and 189 pairs	
10	Dataset 13	Scientific Name 'CAAB Genera', English and 115 pairs	
11	Dataset 14	Scientific Name 'CAABWEB Genera', English and 853 pairs	

B. Experimental Results for E-N-DIST Algorithm

In the section, a comparative study is carried out to evaluate the performance of the proposed E-N-DIST algorithm. The first experiment has been carried for the proposed E-N-DIST algorithm and original N-DIST algorithm as compared algorithm with N equal to Bi and Tri (Bi=2, Tri=3) respectively. This experiment conducted using Dataset 5 that includes 14 pairs of names. The result of this experiment is shown in Table 2. The E-N-DIST Algorithm gives better results than the LD, DLD, MDLD and N-DIST algorithms especially when comparing names transposition as shown in Table 2 for example, the names in 1 and 4 rows. Unlike the LD, DLD, MDLD and N-DIST algorithms, the E-N-DIST algorithm is sensitive to replacement as shown in 2, 3, 5 and 6 rows. The repeated letters are handled by E-N-DIST Algorithm, deletion and dictation errors more efficiently than the LD, DLD, MDLD and N-DIST algorithms as shown in 7, 8, 9, 10, 11, 12, 13 and 14 rows. The E-N-DIST algorithm

[1] https://dl.dropboxusercontent.com/u/58181220/spellchecker_testcase.rar

shows many advantages over the LD, DLD, MDLD and N-DIST algorithms as aforementioned.

Therefore, the E-N-DIST algorithm gives more accurate results than the LD, DLD, MDLD and N-DIST algorithms with BI and TRI for all pairs in dataset 5 as presented in Table 2 and Figure 8.

TABLE 2. COMPARISON BETWEEN ALGORITHMS

	Dataset		Compared Algorithms			Proposed Algorithm	
			LD	DLD	MDLD	N-DIST	**E-N-DIST**
1	precede	preceed	0.71	0.86	0.86	0.79	1.00
2	promise	promiss	0.86	0.86	0.86	0.93	1.00
3	absence	absense	0.86	0.86	0.86	0.86	0.86
4	achieve	acheive	0.71	0.86	0.86	0.71	0.86
5	algorithm	algorythm	0.89	0.89	0.89	0.89	0.89
6	similar	similer	0.86	0.86	0.86	0.86	0.86
7	accidentally	accidenaly	0.92	0.92	0.92	0.92	0.96
8	almost	allmost	0.86	0.86	0.86	0.86	0.93
9	amend	ammend	0.83	0.83	0.83	0.83	0.92
10	dilemma	dilema	0.86	0.86	0.86	0.86	0.93
11	embarrass	embarass	0.89	0.89	0.89	0.89	1.00
12	harass	harrass	0.86	0.86	0.86	0.86	1.00
13	occurred	occured	0.88	0.88	0.88	0.88	0.94
14	really	realy	0.83	0.83	0.83	0.83	0.92
	AVERAGE (PERCENTAGE SIMILARITY)		0.84	0.86	0.86	0.85	**0.93**

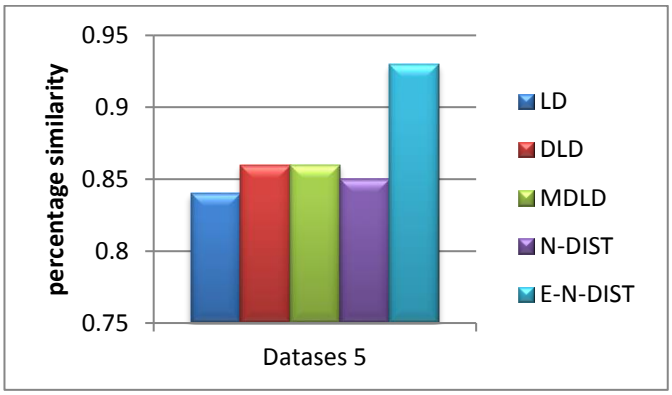


Fig 8. The percentage similarity of LD, DLD, MDLD, N-DIST and E-N-DIST algorithms for Dataset

In order to understand how the editing operations in the E-N-DIST algorithm works with variation of names, it will be elaborate of in detail in the following examples and as shown in Figure 9. Figure 9 shows how the E-N-DIST algorithm measures the distance between the name1"accidentally" and name2 "accidentally" as the first step. The distance is 0.50; therefore, the similarity is 96%. It is apparent that the proposed algorithm provides a lower cost for replacing "a" with "e" from name2 into name1 due to their form similarity.

Furthermore, more experiments have been carried out with a diversity of datasets to get the ensure about the powerful of E-N-DIST algorithm. Ten datasets were chosen and tested on the LD, DLD, MDLD, N-DIST and E-N-DIST algorithms with N=BI as presented in Table 3. That clearly shows the ability of E-N-DIST algorithm in name matching.

Figure 10. shows the accuracy of the percentage similarity as a mean for all datasets. In this figure, the E-N-DIST algorithm gets 90% while LD, DLD, MDLD and N-DIST algorithms get 88%, 88%, 88% and 86%, respectively. Therefore, the E-N-DIST algorithm gives more accurate results than the LD, DLD, MDLD and N-DIST algorithms for all datasets, because LD, DLD, MDLD and N-DIST algorithms have not considered the transposition operations of Latin based language speedily English language.

0		-a	ac	cc	ci	id	de	en	nt	ta	al	ly
		1	2	3	4	5	6	7	8	9	10	11
-a	1	0.00	1.00	3.00	5.00	6.00	7.00	8.00	9.00	9.00	9.00	11.00
ac	2	0.50	0.00	0.50	1.00	2.00	3.00	4.00	5.00	5.50	6.50	7.50
cc	3	1.50	0.50	0.00	0.50	1.50	2.50	3.50	4.50	5.50	6.50	7.50
ci	4	2.00	1.00	0.50	0.00	0.50	1.50	2.50	3.50	4.50	5.50	6.50
id	5	3.00	2.00	1.50	0.50	0.00	0.50	1.50	2.50	3.50	4.50	5.50
de	6	4.00	3.00	2.50	1.50	0.50	0.00	0.50	1.50	2.50	3.50	4.50
en	7	5.00	4.00	3.50	2.50	1.50	0.50	0.00	0.50	1.50	2.50	3.50
nt	8	6.00	5.00	4.50	3.50	2.50	1.50	0.50	0.00	0.50	1.50	2.50
ta	9	6.50	5.50	5.50	4.50	3.50	2.50	1.50	0.50	0.00	0.50	1.50
al	10	7.00	6.00	6.50	5.50	4.50	3.50	2.50	1.50	0.50	0.00	0.50
ll	11	8.00	7.00	7.50	6.50	5.50	4.50	3.50	2.50	1.50	0.50	0.50
ly	12	9.00	8.00	8.50	7.50	6.50	5.50	4.50	3.50	2.50	1.00	**0.50**

Fig 9. The distance between accidentally → accidentally in the E-N-DIST algorithm.

A. The Estimate Measure.

This estimate measure using the f-measure which is also called f-score, the name matching quality has proven to be effective [15, 18, 29] which is based on precision and recall. For classification tasks, these metrics are applied. The predicted class of an item with the actual class are compared refer to table 4. Based on Table 4, precision and recall are defined as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (9)$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (10)$$

Moreover, the F-measure is defined as the weighted combination of precision and recall. The F-measure is defined by:

$$\text{F} - \text{Measure} = \frac{2.\text{Precision}.\text{Recall}}{\text{Precision} + \text{Recall}} \quad (11)$$

Since f-measure is an accuracy measure between 0 and 1, the higher the values, the better and more accurate are the results. The experiment as can be seen in Table 5., the mean of f-measures achieved by the proposed E-N-DIST algorithm on all dataset and threshold, is 0.91 which outperforms the other algorithms.

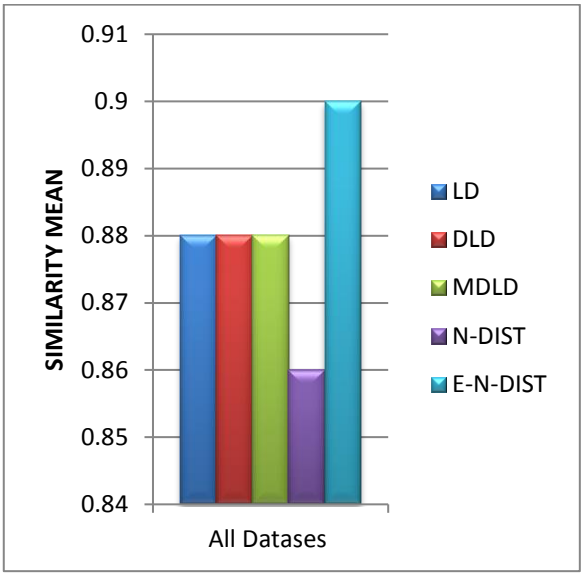


Fig 10. The similarity mean of LD, DLD, MDLD, N-DIST and E-N-DIST algorithms for all datasets.

TABLE 3. RESULT OF PROPOSED AND COMPARED ALGORITHMS

	Dataset	Compared Algorithms		Proposed Algorithm		
		LD	DLD	MDLD	N-DIST	E-N-DIST
1	Dataset 6 (English 60 pairs)	0.83	0.86	0.86	0.81	0.89
2	Dataset 7 (English 4013 pairs)	0.84	0.85	0.85	0.82	**0.89**
3	Dataset 8 (Portuguese 120 pairs)	0.84	0.84	0.84	0.82	0.84
4	Dataset 9 'CAAB' (641 pairs)	0.95	0.94	0.94	0.93	0.95
5	Dataset 10 'Dalcin name pairs' (171 pairs)	0.94	0.95	0.95	0.93	0.97
6	Dataset 11 'CAABWEB' (2047 pairs)	0.93	0.93	0.93	0.92	**0.95**
7	Dataset 12 'GRIN genera' (189 pairs)	0.88	0.89	0.89	0.87	0.90
8	Dataset 13 'CAAB Genera' (115 pairs)	0.88	0.90	0.90	0.85	0.91
9	Dataset 14 'CAABWEB Genera' (853 pairs)	0.88	0.88	0.88	0.87	0.90
10	Dataset 4 '**Arabic** name (600 pairs)	0.79	0.80	0.80	0.73	0.77
	SIMILARITY MEAN	0.88	0.88	0.88	0.86	**0.90**

TABLE 4. CORRESPONDENCE BETWEEN THE ACTUAL AND THE PREDICTED CLASSES.

Algorithm		Predicted	
		Relevant	Irrelevant
Actual (Truth)	Relevant	True Positive (TP)	False Negative (FN)
	Irrelevant	False Positive (FP)	True Negative (TN)

Table 6 presents the F1-scores for different scenarios. For the dataset 8 (Portuguese 120 pairs), using different Edit Distance. The best results were retrieved with the threshold values for a correct match of 0.65, 0.70, 0.75, 0.80, 0.85and 0.90 for LD, DLD, N-DIST, MDLD and E-N-DIST respectively [9-18].

.

TABLE 5. AVERAGE F-MEASURE VALUES (BEST RESULTS SHOWN BOLDFACE AND WORST RESULTS UNDERLINED) WITH THRESHOLD 0.90, 0.85, 0.80, , 0.75, 0.70 AND 0.65, OF ALL DATASETS TESTED (3 FOR ENGLISH, 1 FOR PORTUGUESE, 3 FOR SPECIES, 3 FOR GENERA, 1 FOR ARABIC).

		Compared Algorithms				Proposed Algorithm
	Dataset	LD	DLD	MDLD	N-DIST	E-N-DIST
1	Dataset 6 (English 60 pairs)	0.77	0.85	0.74	0.85	0.91
2	Dataset 7 (English 4013 pairs)	0.75	0.76	0.73	0.89	0.90
3	Dataset 8 (Portuguese 120 pairs)	*0.80*	*0.80*	*0.77*	*0.80*	*0.82*
4	Dataset 9 'CAAB' (641 pairs)	0.99	0.99	0.99	0.99	**1.00**
5	Dataset 10 'Dalcin name pairs' (171 pairs)	1.00	1.00	1.00	1.00	**1.00**
6	Dataset 11 'CAABWEB' (2047 pairs)	0.96	0.97	0.94	0.98	0.98
7	Dataset 12 'GRIN genera' (189 pairs)	0.93	0.94	0.88	0.94	**0.95**
8	Dataset 13 'CAAB Genera' (115 pairs)	0.95	0.96	0.88	0.96	**0.97**
9	Dataset 14 'CAABWEB Genera' (853 pairs)	0.91	0.93	0.84	0.79	0.91
10	Dataset 4 'Arabic name (600 pairs)	0.66	0.68	0.53	0.68	0.70
	F-MEASURE MEAN	0.87	0.89	0.83	0.89	0.91

TABLE 6. F1-SCORES OF DIFFERENT ALGORITHMS, THRESHOLDS AND SIMILARITY CALCULATION

	Algorithms / thresholds	65	70	75	80	85	90
1	(LD)	0.987	0.961	0.938	0.889	0.750	0.273
2	(DLD)	0.987	0.961	0.938	0.894	0.750	0.273
3	(N-DIST)	0.966	0.952	0.924	0.863	0.710	0.222
4	MDLD	0.987	0.961	0.938	0.894	0.750	0.273
5	**(E-N-DIST)**	**0.974**	**0.961**	**0.947**	**0.894**	**0.776**	**0.378**

VI. CONCLUSION AND FUTURE WORK:

We have formulated a new concept of n-gram distance, which generalizes the standard unigram string distance.

In this research, E-N-DIST is proposed and presented as a new orthographic measure for recognizing dataset pairs based on N-DIST distance with an extension to soften the scale of similarity between the bi-grams that conforms a name matching. Precisely, on 2^{n+1} -1 states calculate. The experiments showed that the new E-N-DIST measures overcome the LD, DLD, MDLD and N-DIST measures.

In the future, this algorithm can be extended to be able to handle the token name in Arabic or any other language.

REFERENCES

[1] Kondrak, G.: N-Gram similarity and distance. In: Consens, M., Navarro, G. (eds.) SPIRE 2005. LNCS, vol. 3772, pp. 115–126. Springer, Heidelberg (2005). https://doi.org/10.1007/11575832_13

[2] Christian Eduardo Millán-Hernández, René Arnulfo García-Hernández(&), Yulia Ledeneva, and Ángel Hernández-Castañeda, 2019, Soft Bigram Similarity to Identify Confusable Drug Names, J. A. Carrasco-Ochoa et al. (Eds.): MCPR 2019, LNCS 11524, pp. 433–442

[3] Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. J. ACM 21, 168–173 (1974).

[4] Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, pp. 707–710 (1966

[5] Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: a survey. IEEE Trans. Knowl. Data Eng. 19, 1–16 (2007)

[6] Kondrak, G.: Phonetic alignment and similarity. Comput. Hum. 37, 273–291 (2003)

[7] Kondrak, G.: Algorithms for language reconstruction (2002)

[8] http://alvinalexander.com/java/jwarehouse/lucene/contrib/spellchecker/src/java/org/apache/lucene/search/spell/NGramDistance.java.shtml)

[9] Abdulhayoglu M.A., Thijs B., Jeuris W, "Using character n-grams to match a list of publications to references in bibliographic databases", DOI 10.1007/s11192-016-2066-3, 2016.

[10] Abdulhayoglu, M. A. and Thijs, B. "Matching bibliometric data from publication lists with large databases using n-grams". In Proceedings of 14th international society of scien to metrics and informetrics conference (ISSI), Vienna, Austria, Vol. 2, pp. 1151–1158, 2011

[11] Al-Sanabani, M., Al-Hagree, S., "Improved An Algorithm For Arabic Name Matching". Open Transactions On Information Processing ISSN(Print): 2374-3786 ISSN(Online): 2374-3778.2015.

[12] Alsurori, M., Al-Sanabani, M., & Salah, A. H. (2018). Design an Accurate Algorithm for Alias Detection, ISSN: 2074-9023 (Print), ISSN: 2074-9031 (Online).

[13] Mark P.J. van der Loo, "The stringdist package for approximate string matching". The RJournal 6(1) 111-122, 2014.

[14] Boehmer B (2002) Levenshtein Distance algorithm: Oracle PL/SQL implementation.Available:http://forums.oracle.com/forums/thread.jspa?messageID=202783#356435, also archived copy at http://web.archive.org/web/20120526084237/http://www.merriampark.com/ldplsql.htm. Accessed 2014.

[15] Christen, P., "A Comparison of Personal Name Matching Techniques and Practical Issues", Technical Report TR-CS-06-02, Joint Computer Science Technical Report Series, Department of Computer Science, 2006.

[16] Boehmer and Ton, is available on https://confluence.csiro.au/public/taxamatch/the-mdld-modified-damerau-levenshtein-distance-algorithm, 2008..

[17] Damerau, F., “A Technique for Computer Detection and Correction of Spelling Errors”, Communications of the ACM, 7(3), pp.171-176, 1964.

[18] Delgado, J., Galárraga, F., Fuertes, W., Theofilos Toulkeridis, Cesar Villacís, Fidel Castro, “A Proposal of an Entity Name Recognition Algorithm to Integrate Governmental Databases”, 2016...

[19] Medhat, D., Hassan, A., Salama, C. “A Hybrid Cross-Language Name Matching Technique using Novel Modified Levenshtein Distance”, 978-1-4673-9971-5/15, IEEE, 2015.

[20] Mustafa, S. H., and Al-Radaideh, “Using N-Grams for Arabic Text Searching, ” Journal of the American Society for Information Science and Technology, vol. 55, no. 11, pp. 1002–1007, 2004.

[21] Mustafa, S. H., and Al-Radaideh, “Character contiguity in N-gram-based word matching: the case for Arabic text searching, ” Information Processing & Management, vol. 41, no. 4, pp. 819–827, 2005.

[22] Ullmann, J.R.: “A binary n-gram technique for automatic correction of substitution deletion, insertion and reversal errors in words”. The Computer Journal 20(2), 141– 147 (1977).

[23] Beernaerts, Jasper., Debever, E., Lenoir, M., De Baets, B., & Van de Weghe, N. (2019). A method based on the Levenshtein distance metric for the comparison of multiple movement patterns described by matrix sequences of different length. Expert Systems with Applications, 115, 373-385.

[24] Ahmed, F. and Nürnberger, A. “N-grams Conflation Approach for Arabic, ” in ACM SIGIR Conference, Amsterdam, 2007.

[25] Al-Ssulami, A., M.., “Hybrid string matching algorithm with a pivot”, Journal of Information Science, Vol. 41(1) 82–88, 2015.

[26] Ahmed, K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. “Duplicate Record Detection: A Survey”. IEEE Trans. on Knowl. and Data Eng., 19(1):1–16, January 2007.

[27] Rees, T., “Taxamatch, an Algorithm for Near (‘Fuzzy’) Matching of Scientific Names in Taxonomic Databases”, PLoS ONE 9(9): e107510. doi:10.1371/ journal.pone.0107510, 2014.

[28] https://dl.dropboxusercontent.com/u/58181220/spellchecker_testcase.rar

[29] Aqeel S.U, Beitzel S.M, Jensen E.C, Grossman D. & Frieder O., “ On the Development of Name Search Techniques for Arabic”. Journal of the American Society for Information Science and Technology (JASIST) 57(6): hlm. 728-739. 2006.

[30] http://www.codemiles.com/java-examples/find-the-similarity-between-two-strings-in-java-t11127.html