a

Journal of Computer Science

Original Research Paper

Arabic Personal Name Matching: Names Written using Latin Alphabet

[1,2]Attia Nehar, [2,3]Slimane Bellaouar, [4]Djelloul Ziadi and [5]Khaled Moulay Omar

[1]Department of Computer Science, Ziane Achour University - Djelfa, Algeria
[2]Faculty of Science Laboratoire d'Informatique et de Mathematiques (LIM), Universite de Laghouat, Laghouat, Algeria
[3]Laboratoire de Mathematiques et Sciences Appliquees (LMSA)´ Faculty of Science and Technology, Universite de Ghardaia, Ghardaia, Algeria
[4]Groupe de Recherche Rouennais en Informatique Fondamentale (GR2IF), Algeria
[5]Universite de Ghardaia, Ghardaia, Algeria

Article history
Received: 11-05-2021
Revised: 15-07-2021
Accepted: 13-08-2021

Corresponding Author:
Attia Nehar
Department of Computer Science, Ziane Achour University - Djalfa, Algeria
Email: nehar.attia@gmail.com

Abstract: In many Arab countries' public administrations, Arabic personal names are written with Latin alphabet, generally, in various ways by different writers. This has led to many problems when it comes to connecting these administrations. The aim of this study was to propose two new approaches for the pairwise matching of Arabic personal names. The first approach is based on string alignment and phonetic transcription. Appropriate scoring functions were defined to catch similarity between Arabic personal names. In the second approach, we use machine learning techniques to derive a suitable model for this problem. Precisely, we suggest using a Multi-Layer Perceptron (MLP) architecture and experiment with different configurations. Performances of the new models compare well with the best-performing similarity measures (Jaro, Jaro-Winkler, Double Metaphone and Edit Distance) in terms of precision, recall and F1. Even though the work was carried out for the (Algeria/French Alphabet) case, it can be adapted to any other (country/script) case, like (Egypt/English).

Keywords: Personal Name Matching, Phonetic Transcription, Phonetic Encoding, Sequence Alignment, Machine Learning

Introduction

An increasing amounts of data are being generated every day, especially, textual data which is at the core of usage in public administrations. Personal names are written in the Latin script in most Algerian public administrations, such as civic administration, banks and insurances. Writing the same person's personal name in different administrations by many persons has led to many problems, such as when transferring money between banks without verifying transcribed personal names. Everyone is using his own cultural knowledge to map the original listen or written personal name from Arabic to Latin script, without relying on transliteration rules; leading to different spellings for the same person's name. People from diverse cultural contexts may spell the same Arabic personal name differently in the Latin language. For example, the Arabic name (عبدالرحمن) could be spelled differently as: (Abderrahmane, Abderrahman, Abdourrahmane, Abd al-rahman, . . .). This situation makes searching, retrieving and matching Arabic names very difficult when they are written in Latin script. It is worth noting that this problem is not limited to Algerian personal names, it is touching all countries influenced by French colonization, such as the North African countries.

Different techniques have been developed to solve English name matching cases. An early work by Van Berkel and De Smedt (1988) aimed to do typographical and orthographic corrections (Van Berkel and De Smedt, 1988). Christen (2006) gave a detailed discussion of personal name characteristics and presented a comprehensive number of commonly used name matching techniques. Even though the author claims that there is no clear best technique to choose, he provides series of recommendations that help to select a name matching technique. However, matching Arabic names written in Latin script is even more complicated, since "there are no rules for the translation of proper names" from Arabic script to Latin one (Halimah, 2016; Dweik and Al-Sayyed, 2016).

In this study, we propose two new approaches for pairwise matching of Arabic personal names written in Latin script (French case). The first approach is based on the use of phonetic transcription and sequence alignment. First, we start by applying phonetic rules (a function h) in order to bring together two different writings (u = Mustapha, v = Mustafa) of the same personal name (مصطفى) by: $h(u)$ = [Mustapha] and $h(v)$ = [Mustafa]. Then, we introduce a new similarity measure (score function) based on sequence alignment. The second approach relies on the use of machine learning techniques, precisely, a Multi-Layer Perceptron (MLP) architecture is proposed. A set of configurations is experimented with to determine the best performing model. To the best of our knowledge, no study has focused on this problem.

Source and Target Systems

Public administrations, in many Arabic countries, use the Latin alphabet to write personal names. As said above, this may lead to many problems, since writers do not use a consistent way to transcript these personal names. One source of inconsistency is due to the variation in writing the personal name in the source script itself. The Arabic name (فاطمة, Fatima) may be written in many different ways like: (فاطنة ،فطيمة، فاطيمة، فاطمة). Another source of variation is related to the lack of consistency when writing Arabic names in the Latin alphabet. Writers don't use transliteration rules or don't use the same rules if any. In addition, the peculiarities of source and target languages make things even worst. Both Arabic and French lack some of each other's sounds and letters. For instance, there is not a match for (ض ط ظ خ ع غ ق ه) in French and "P,G" in Arabic.

To cope with this, most of the developed approaches are based on phonetic encoding, pattern matching, or a combination of these two approaches (Christen, 2006). Phonetics is a science that studies the characteristics of human speech. It provides methods for the description, classification and transcription of speech sounds (O'Grady, 2012). The use of sequences of phonetic symbols to represent speech is known as transcription. The production of speech looks at the interaction of different vocal organs, for example, the lips, tongue and teeth, to produce particular sounds. By classification of speech, the focus is on the sorting of speech sounds into categories which can be seen in what is called the International Phonetic Alphabet (IPA), which is a framework that uses a single symbol to describe each distinct sound in the language. It is based primarily on the Latin alphabet. The IPA is maintained by the International Phonetic Association (also IPA) which provides the academic community worldwide with a notational standard for the phonetic representation of all languages (IPAIPAS, 1999).

Phonetic encoding methods are used to convert the original name string into a code based on its phonetic transcription or by the way this name is pronounced (Christen, 2006). One of the widely known phonetic encodings is the Soundex algorithm (Biot, 1956), which encodes names based on the way they sound rather than the way they are spelled so that names like ('Ahmad') and ('Ahmed') will have the same code. The generated code for a name consists of a letter and three numbers, such as A530 for the name string ('Ahmed'). The letter is always the first character of the name. Numbers are assigned to the remaining letters of the name according to Soundex rules. Zeroes are added at the end if necessary to produce a four-character code[1].

A more advanced phonetic encoding algorithm was created by Lawrence Philips called metaphone (Philips, 1990). Like the Soundex algorithm, it tries to produce an encoding of a string name based on how it is pronounced. But it uses a sequence of letters rather than just one letter to assign values. Besides, it uses the entire string name and does not truncate it after considering only some initial part. The main drawback with this system and other Soundex-derived phonetic encoding algorithms, is that they rely only on the English pronunciation of the name. To cope with this, Lawrence Philips introduced another enhanced version called double metaphone, which accounts for other foreign language pronunciations (Philips, 2000).

Sequence Similarity

Measuring the similarity between two sequences or two words consists of evaluating to what extent these sequences are close and even identical. This task is often used in several important fields, including information retrieval, bioinformatics, language and speech processing, machine translation, etc. In this section, we will illustrate some similarity measures and briefly explain their calculation methods. We will explore a number of similarity measures and distance metrics, namely the Jaro, Jaro-Winkler and the Edit Distance metric or Levenshtein distance. Let's first start by giving some preliminary definitions.

An alphabet (denoted by Σ) is a finite set of symbols. We denote the size of alphabet Σ by $|\Sigma|$. A string $S = s_1 s_2 ... s_n$ over Σ is a finite sequence of symbols drawn from Σ with length $|S| = n$ and s_i denotes the i^{th} element of S. The

[1]Soundex System - National Archives, https://www.archives.gov/research/census/soundex

symbol Σ ^∗ denotes the set of all strings over the alphabet Σ, whereas Σ^n is the set of strings with length equals to *n*. The symbol ε denotes empty string

A string T is a sub-string of a string S if there are strings $U \in \Sigma$ ^∗ and $V \in \Sigma$ ^∗ such that $S = UTV$ (*U* and *V* can be empty strings). Let *U* and *V* be two strings. The concatenation of *U* and V is the string *UV* formed by writing symbols of U first, then writing the symbols of *V*.

Let us start with the Levenshtein distance, also referred to as edit distance, which is a string metric for measuring the difference between sequences. It allows insertions, deletions and replacements to start from one string and get to the other one. In its simplified form, each operation costs 1. So the Levenshtein distance between two sequences is the minimal number of insertions, deletions and replacements to make the two sequences equal (Levenshtein, 1966). This distance is symmetric and it holds $0 \leq d\_lev(S, T) \leq \max(|S|,|T|)$.

The Jaro metric [Jaro, 1989] is a widely used similarity measure in the community of record-linkage (Cohen *et al*., 2003). It was used mainly for duplicate name detection. For two strings *U* and *V*, let *U'* be the characters in *U* that are common with V (the meaning of common here is that the matching character must be within half the length of the shorter string) and inversely let *V'* be the characters in *V* that are common with *U*. Let *T_{UV}* measure the number of transpositions of characters in *U'* relative to *V'*. The Jaro similarity simj is given by:

$$sim_j(U,V) = \frac{1}{3}\left(\frac{|U'|}{|U|} + \frac{|V'|}{|V|} + \frac{|U'| - T_{U,V}}{2|U|}\right) \quad (1)$$

A variant of Jaro similarity is proposed by William E. Winkler which gives more favorable rating p to strings that shares a long common prefix of length l (Winkler, 1990):

$$sim_w(U,V) = sim_j(U,V) + l * p * \left(1 - sim_j(U,V)\right) \quad (2)$$

The standard value for the constant *p* is 0.1 and l is considered up to a maximum of 4 prefix characters.

String Alignment Based Approach

The edit distance is formalized as a general parametric method that is calculated with a specific set of allowed edit operations and each operation is assigned a cost. This can be further generalized by sequence alignment algorithms which make the operation's cost depends on its context. In this study, we propose a new approach for pairwise matching of Arabic personal names written in the Latin alphabet. It is based on the use of phonetic transcription and sequence alignment, which uses all the allowed edit operations with a specific cost for each one. These costs are chosen carefully to match personal names with different spellings.

Let Σ be an alphabet and *U, V* two strings over Σ. An alignment of *U* and *V* is a word w ∈ ((ΣU{−}×ΣU {−})\ {(−,−)}) ∗ with:

$$U = \Phi\left(\Pi_1(w)\right), V = \Phi\left(\Pi_2(w)\right)$$

where Π1, Π2 are, respectively, the first and second projections and Φ is a function that replaces every occurrence of '−' in a string by *ε*.

The size of an alignment w, denoted the by |w|, is the number of symbols in w,

We illustrate this by an example. Let Σ = {*A, C, G , T*} an alphabet and *U = GAT GAG, V = GTCGAAG* two strings over Σ. A possible alignment of U and V is given by:

$$w = (G,G)(A,-)(T,T)(G,C)(-,G)(-,A)(A,A)(G,G)$$

So that:

U' = $\Pi_1(w) = G\ A\ T\ G - -\ A\ G$

V' = $\Pi_2(w) = G - T\ C\ G\ A\ A\ G$

We can check that: $\Phi(U') = U$ and $\Phi(V') = V$.

Definition: Let x, y ∈ Σ. An edit operation is a symbol (*x, y*) ∈ ((ΣU {−} ×ΣU {−}) \ {(−,−)}). It is called:

- Substitution if $x \neq y \neq$ '-'
- Deletion if $x \neq$ '-' and y = '-'
- Insertion if *x* = '-' and $y \neq$ '-'
- Identity if *x = y*

To evaluate an alignment score, we first define a score function as follow:

$$sc: (\Sigma \cup \{-\} \times \Sigma \cup \{-\}) \setminus \{(-,-)\} \to \square$$

Given an alignment $w = w_1 w_2 \ldots w_n$, the score of this alignment can be defined as:

$$Sc(w) = \sum_{i=1}^{n} sc(w_i) \quad (3)$$

For two strings *U* and *V*, there may be many possible alignments. We have to find out the best one, i.e., the alignment with the optimal score. Let *W*(*U, V*) be the set of all possible alignments of two strings *U* and *V*. The optimal alignment is calculated using dynamic programming method such as:

$$w_{opt} = \max\left(Sc(w)\right) w \in W(U,V) \quad (4)$$

Considering Eq. (3) and (4), we derive two similarity functions. The first one will be used to calculate the similarity between name strings without any transcription. So we define a score matrix between different symbols of Latin

alphabet. Values will be chosen, with respect to the type of symbols in w_i (consonant/consonant, consonant/vowel or vowel/vowel), from the set {−10, −7, −5, −3, −2, −1, 0, 1, 2, 3, 4, 5} as shown in Table 1 and 2. For the second similarity function, it will be used to calculate the similarity of transcribed names. Also, we define another score matrix between different symbols of the IPA. Values will be chosen, with respect to the type of phonemes in w_i and their phonetic similarity, from the set {−10, −5, −4, −2, −1, 0, 1, 2, 3, 4, 5} as mentioned in Table 3 and 4. Optimal alignment scores (w_{opt}) are normalized to have values within [0,1]. Then, two strings are considered similar if the optimal score is greater than a fixed threshold *t*.

As mentioned before, score matrix values are chosen appropriately to account for similarity between Latin letters when they are used to write Arabic names (Tables 2 and 4). For example, the Arabic name (طارق) may be spelled differently as ('Tarik') or ('Tariq'). Hence, it is wise to have non negative scores for pairs of letters like ('j', 'g') and ('k', 'q'). Likewise, some Latin letters are sometimes used indifferently to spell Arabic names, like in (عبدالرحمن) which is written as 'Abdurrahman' or 'Abdurrahman'. Thus, a neutral score for pairs like ('a', 'e') is more convenient.

Furthermore, this approach can be applied to other country/script cases, like for Egyptian personal names written in the Latin alphabet. Indeed, having an idea of how Egyptian writers pronounce the Latin alphabet enables us to derive a scoring function adapted to catch similarity between Egyptian personal names.

Table 1: Alphabetical score principle

Pair of symbols	Score
(Letter, −)	-5
(Vowel, consonant)	-10
(Vowel, vowel)	-3, -2, -1, 0, 1, 2, 3, 4
(Consonant, consonant)	-10, -7, -5, -3, -2, 0, 2, 3
Similar letters	5

Table 2: Examples of alphabetical scores

Pair of IPA symbols	Score
('b', 'b')	5
('a', 'e')	0
('j', 'g')	2
('i', 'q')	-10
('b', 'p')	-2
('o', 'i')	-1
('q', 'k')	3

Table 3: Phonetic score principle

Pair of IPA symbols	Score
(Vowel, consonant)	-10
(Vowel, vowel)	-3, -2, -1, 0, 1, 2, 3, 4
(Consonant, consonant)	-10, -5, -4, -2, 0, 2, 3
Similar symbols	5

Table 4: Examples of phonetic scores

Pair of symbols	Score
('b', 'b')	5
('œ', 'y')	0
('u', 'œ')	2
('E', 'E')	5
('Z', 'Ã')	5
('K', 'g')	1
('j', 'e')	-2

Machine Learning Based Approach

In the first approach, score matrix values are chosen by a human expert to account for similarity between Latin letters when used to write Arabic names. These matrix values will reflect a point of view that may differ from expert to expert. To alleviate this dependence on human expertise, we can derive these values from data by learning.

The problem of Arabic Personal Names Matching can be formalized as a machine learning problem as follows. Let Σ be the set of French language alphabet letters with a supplementary symbol '−' and *U*, *V* two Arabic string names written using Σ. We set m to be the maximal size of Arabic string names. *U'* and *V'* are two strings over Σ derived, respectively, from *U* and *V* by a lowercase of all symbols and right-padding each string with the symbol '−' to have $|U'| = |V'| = m$. Lets now derive a string $W \in \Sigma *$ as the concatenation of *U'* and *V'* (obviously $|W| = m' = 2\times m$). We define now a function $f(W)$ as follows:

$$f(W) = \begin{cases} 1, \text{ if } U' \Diamond V'. \\ 0, \text{ otherwise.} \end{cases} \quad (5)$$

Such that $U' \Diamond V'$ means that *U'* and *V'* are two equal strings or they represent the same person's name spelled differently.

The function f can be learned using an annotated dataset of name pairs and an adequate learning model. Indeed, each instance of the dataset represents a pair of string names which will be hand-marked as a positive instance (class 1), hence, representing the same person's name (eventually spelled differently), or marked as a negative instance (class 0) when string names refer to different persons.

Unlike when dealing with a typical learning problem, where similarity is calculated between instances, in the problem of Personal Names Matching as formulated above, similarity accounts for the pairs of string names within the same instance. A neural network model is well suited for this situation. We opted for a feed-forward neural network architecture (a Multi-Layer Perceptron) with an input layer with 2×m neurons fed by the characters of W (Fig. 1). This architecture has n (with $n \geq 1$) hidden layers and a single output neuron with a sigmoid transfer function.

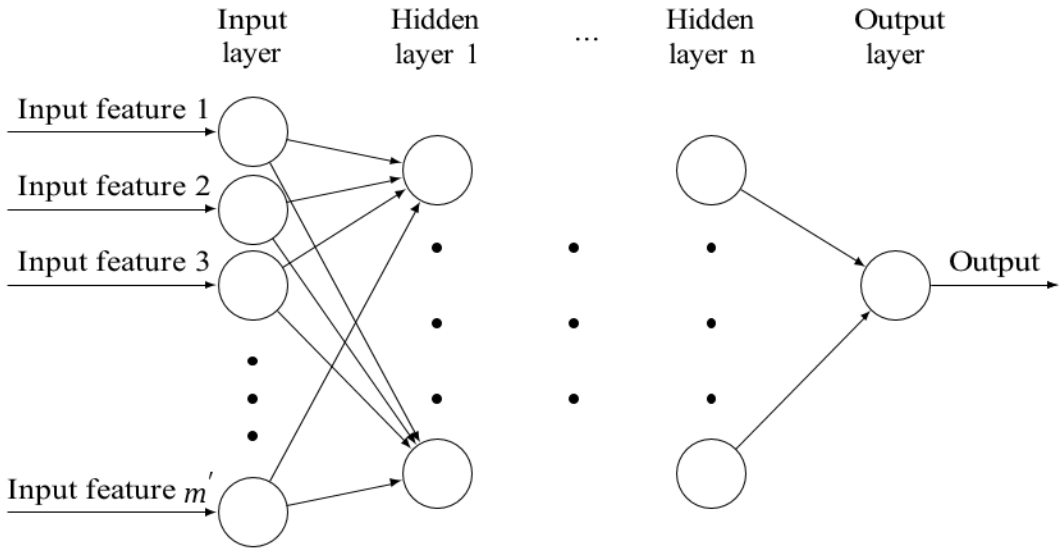

Fig. 1: General architecture of our feed-forward neural network

Results and Discussion

The first set of experiments is devoted to the first approach. It aims at showing the effect of using an appropriate scoring model, which ought to catch similarity between identical Arabic person names written in Latin alphabet by many writers, hence, spelled differently.

Performances of proposed similarity measures (abbreviated hereafter: Alpha and phone), are assessed against four other similarity measures, namely: Edit Distance (Levenshtein distance), Jaro, Jaro-Winkler and Double Metaphone, abbreviated as: Edit dist., Jaro, Jaro wink and dmeta respectively. These measures were calculated on the original string names, except for dmeta which is calculated using an edit distance on codes generated by the Double Metaphone algorithm.

In the second set of experiments, performances of the proposed neural architecture are assessed with different settings. First, we consider a Multi-Layer Perceptron with only one hidden layer and we show the effect of varying its size, i.e., the number of neurons. Then, we consider an MPL with two hidden layers and a grid search is performed over the size of these hidden layers. All configurations are run for two activation functions; relu and logistic sigmoid.

Dataset and Experimental Configurations

A large dataset was collected from many lists of personal names taken from Algerian civic administrations, banks and insurances. This dataset contains 20868 records representing more than 5000 unique first and last names. A pre-processing phase consists of cleaning string names by removing non-alphabetical symbols and numbers, then lowercase characters. Because it is infeasible to do a matching of the entire dataset, consisting of 20868 names, against itself, a subset was selected by a stratified random sampling method. Indeed, we choose a size of 1000 entries (approximately 5% of total dataset size) by dividing the alphabetically ordered dataset into 10 equal subsets, then 100 entries were randomly drawn from each subset. The resulting list is carefully hand-matched against itself to have (1000 × 1000) annotated matrix. A given entry equals 1 if corresponding names are identical or represent the same name with different spellings, otherwise, entry equals 0. To meet the requirements of equation 5, we have generated from these entries another dataset where each entry consists of a string W (the concatenation of *U'* and *V*') and the corresponding class value (0 or 1). The resulting dataset consists of 551 775 neatly annotated pairs of name strings and their corresponding classes. Table 5 shows the dataset details.

Performances are evaluated using four metrics; accuracy, precision, recall and F1, to account for all usage contexts (Table 6).

Results and Discussion of the String Alignment Based Approach

In order to evaluate quality of different measures on the dataset, first, we show the effect of varying similarity threshold values. For alpha, phone, jaro and jaro wink measures, thresholds were taken from the set of values [0.8, 1] with 0.01 step. Results are reported for each measure in Fig. 2, 3, 4 and 5. For the edit dist and dmeta, thresholds are {0, 1, 2, 3}. Results are reported in Fig. 6 and 7. Then, we report results of each similarity measure with its best performing threshold based on the F1 metric (Fig. 8).

From Fig. 2, 3, 4 and 5, we show that, with an appropriate threshold, the alpha, phone, jaro and jaro wink measures achieved their best performances in terms of F1 metric. In Fig. 8, these best performances are compared.

As expected, alpha (with F1 = 94.16%) and phone (with F1 = 95.06%) gave very competitive results with the best performing measure (Jaro with F1 = 93.96%). Moreover, we can notice the significant gap between recall and precision for each similarity measure except for alpha and phon. A possible interpretation of this finding is that alpha and phone, with their appropriate scoring model, are more able to account for Arabic name strings in which more than one Latin character may refer to the same Arabic character (the letter "¼" may be spelled as "k" or "q" in Latin alphabet).

As shown in Fig. 6, it is clear that the Double Metaphone similarity measure, which is based on phonetic encoding, did not perform well for the Arabic personal names matching problem. It failed to achieve 70% precision with its best performing threshold (equals to 0). This is not a surprising result since only the first letter and consonants are kept in the generated code by this method. The Edit Distance (Fig. 7) gave its better results with a threshold equals to 1 (F1 = 92.99%). With more than one difference between string names, Edit Distance will keep catching more true positives, hence, enhancing recall at the expense of precision. This can be explained by the fact that increasing threshold will account for both identical string names spelled differently and non-identical string names with near spelling. This irreconcilable situation indicates the inability of the Edit Distance measure for the Arabic personal names matching problem.

To have a good understanding of these results, a deeper analysis of errors is required. We give comparative ratios of False Positives (FP) and False Negatives (FN) for different measures over those of the best performing phone measure (Eq. 6 and 7). This may provide us with more knowledge on where each measure is failing to catch similarities between Arabic names written in the Latin alphabet.

$$FP\_ratio_{measure} = \frac{FP_{measure}}{FP_{phon}} \quad (6)$$

$$FN\_ratio_{measure} = \frac{FP_{measure}}{FP_{phon}} \quad (7)$$

From the second column of Table 7, we can notice that Double Metaphone is more effective in avoiding false negatives (ratio equals 33.64%). This could be explained by the fact that Double Metaphone tries to produce an encoding of a string name based on how it is pronounced and Arabic is a highly phonemic language, that is why two different spellings of the same persons' name share a high phonemic similarity. For the context of our application, we know that false positives are more dangerous than false negatives. It may be bearable to have a warning indicating that two string names are different, although they are referring to the same person than to miss two really different string names. Thus, false positives need more attention. It can be inferred from the third column of Table 7 that jaro-wink measure, with 100% ratio, was as efficient as the phone measure at avoiding false positives. Analysis of erroneous decisions taken by alpha and phone could reveal more facts.

Indeed, in Table 8 we give examples of miss-classified pairs of string names. Starting by FPs of the alpha measure, we can infer that these errors are due to Arabic string names with slight writing differences, mostly at the end of names, like in (عماري، عمار) and (عمراني، عمران) which are written as ("AMMAR" and "AMMARI") and ("AMRAN", "AMRANI") respectively. It is worth noting that these errors are well addressed by the phone measure. Another source of FPs is due to Arabic letters ('ع', 'ا') which are transcribed equally by writers as an ('A'). This leads to confusion when it comes to writing names where these two Arabic letters are adjacent but with inverse order, like in (باعمارة and بعامرة) which are transcribed nearly the same as "BAAMARA", "BAAMERA"). The examination of FNs reveals that there are many sources of errors, which can be summarized as follows: First, the same Arabic name may be pronounced very differently among writers, like in (دخينيسة) which is transcribed as ("DKINISSA" or "EDKHAINISSA"). The second writer focuses on vowels at the beginning and middle of the name, so that the transcribed name became (ادخينيسة). Second, many Arabic letters are transcribed inconsistently by writers, like the letter ('غ') in second position of the name (جغاب), which is transcribed as ('GH') in ("DJEGHAB") and by ('R') in ("DJERAB"). The second writer is influenced by the French language pronunciation of the letter ('R') which is very close to the Arabic pronunciation of the letter ('غ'). Likewise, the second occurrence of letter ('س') in (بن ساسيى) is transcribed as ('C') in ("BEN SACI") and by ('SS') in ("BEN SESSI").

For the phon measure, analysis of FPs and FNs will allow us to uncover many types of errors. Starting with FPs, the first example ("BELABBASE", "BELABBACI") is showing that different Arabic names with nearly the same pronunciation will be missed by the phone measure. This can be explained by the fact that in the phone measure procedure, names are first transformed to their phonetic transcription. The second example ("BENCHOUIHA", بن شويحة) highlights a problem related to the letter ('ح') which is transcribed by most writers as ('H'). But, when it comes to the phone measure, this letter will be silent, leading to confusion with ("BENCHOUIA", بن شوية). The third example tackles

another issue with transcribing Arabic letters. The letter (‘ض’) is transcribed inconsistently to (‘D’) or (‘DH’) by different writers. When it is transcribed to (‘D’), this will lead to confusion with the letter (‘X’) which is also transcribed to (‘D’). That is why names like (“BOUDERISSA”, بود ريسة) and (“BOUDERSSA”, بود ريسة) are considered identical by the phone measure. For the FNs, errors shown in the last three examples suggest that phone measure is negatively sensitive to the introduction of new sounds in string names. The injection of the letter (‘D’) in (“BEDJAJ”) by the first writer makes it quite different from (“BEJAJE”), written by another writer. Likewise, introducing the letter (‘L’) in the names: (“ABO KACEM”,“ABOULKACEM”) and (“ABDE AZIZ”,“ABDELAZIZE”) make them quite different names.

In light of the above discussion of the results and analysis of different error types, we emphasize the fact that dealing with Arabic personal name matching is still a very challenging task.

Results and Discussion of the Machine Learning Based Approach

Two sets of experiments are performed to assess the performances of our second approach to the Arabic personal names matching, which is implemented using an MLP classifier. The training/testing configuration is compiled using a stratified k folding (with k = 4). In the first configuration, we consider an MLP with one hidden layer, with size l, which is trained using different values for the size l (taken from the set of values [25,400] with 25 step) and two activation functions, namely: Relu and logistic sigmoid functions. Results are reported in Fig. 9 and 10.

Table 5: Dataset details

Dataset size	Class 0 size	Class 1 size
551775	550498	1277

Table 6: Performance metrics

Accuracy	Precision	Recall	F1
$\frac{TP+TN}{TP+TN+FP+FN}$	$\frac{TP}{TP+FP}$	$\frac{TP}{TP+FN}$	$\frac{2\times TP}{2\times TP+FP+FN}$

Table 7: Ratios of FP and FN of different measures to the phone measure

Measures	Ratio FN	Ratio FP
Edit distance	93.49%	390.00%
Double Metaphone	33.64%	3290.00%
Alpha	112.15%	130.00%
Jaro	113.08%	140.00%
Jaro-winkler	124.30%	100.00%

Table 8: Examples of erroneous decisions taken by alpha and phon measures

	Errors	
Measure	----------	----------
Alpha measure	(“AMMAR”, عمار)	(“AMMARI”, عماريى)
FP	(“AMRAN”, عمرانى)	(“AMRANI”,عمرانيى)
	(“BAAMARA”,باعمارة)	(“BAAMERA”,بعامرة)
FN	(دخينيسة)	(“DKINISSA”,“EDKHAINISSA”)
	(جغاب)	(“DJEGHAB”,“DJERAB”)
	(بن ساسيى)	(“BEN SACI”,“BEN SESSI”)
Phone measure	(“BELABBASE”,	(“BELABBACI”, بلعباسى)
FP	(“BENCHOUIHA”,	(“BENCHOUIA”,بن شويية)
	(“BOUDERISSA”,	(“BOUDERSSA”, بود ريسة)
FN	(بلعباس)	(“BEDJAJ”,“BEJAJE”)
	(بن شويحة)	(“ABO KACEM”,“ABOULKACEM”)
	(بودريسة)	(“ABDE AZIZ”,“ABDELAZIZE”)

Table 9: Best performing models from our two approaches

Model/measure	F1	Precision	Recall
MLP_1_relu	91.08	95.91	87.17
MLP_1_logistic	92.99	98.85	89.02
Alpha	94.16	96.00	92.40
MLP_2_relu	94.37	94.10	94.65
MLP_2_logistic	94.82	97.84	92.15
phone	95.06	97.31	92.92

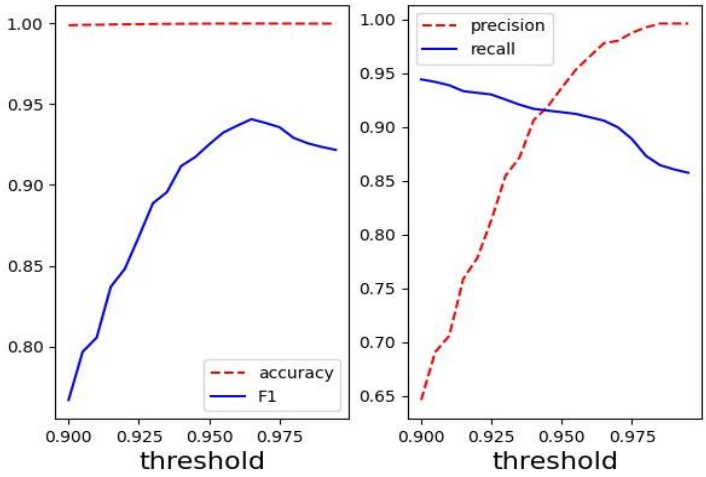

Fig. 2: Performances of Alphabetic scoring measure (alpha) in terms of accuracy, F1, precision and recall

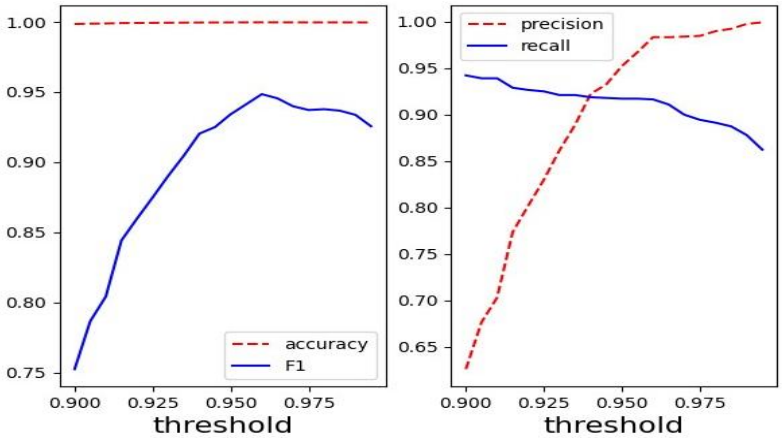

Fig. 3: Performances of phonetic scoring measure (phone) in terms of accuracy, F1, precision and recall

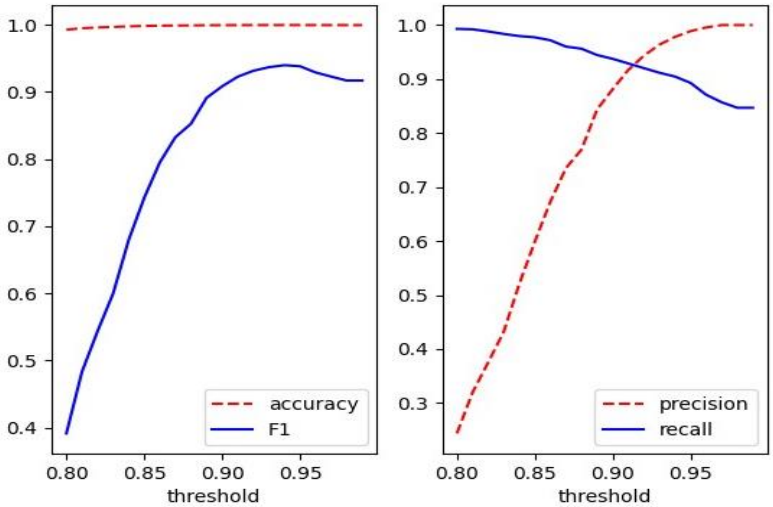

Fig. 4: Performances of Jiro measure (jaro) in terms of accuracy, F1, precision and recall

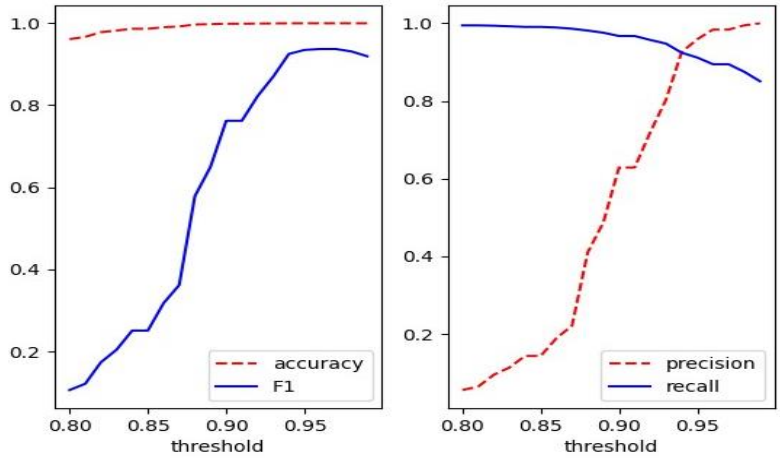

Fig. 5: Performances of Jaro-Winkler measure (jaro wink) in terms of accuracy, F1, precision and recall

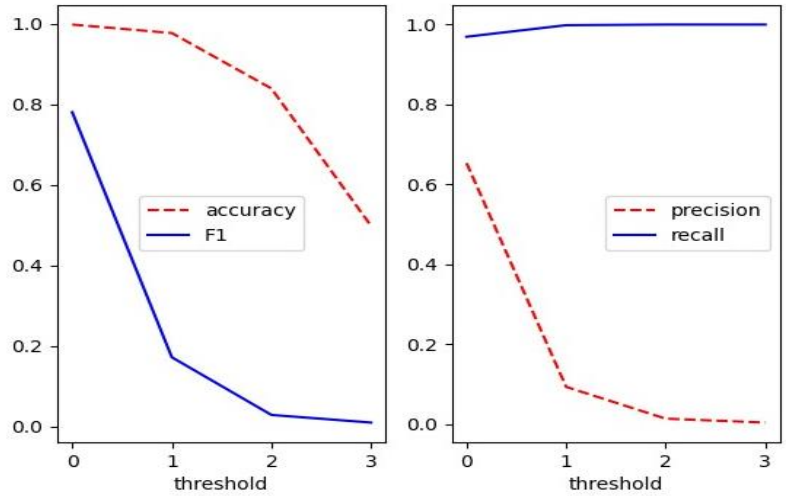

Fig. 6: Performances of Double Metaphone measure (dmeta) in terms of accuracy, F1, precision and recall

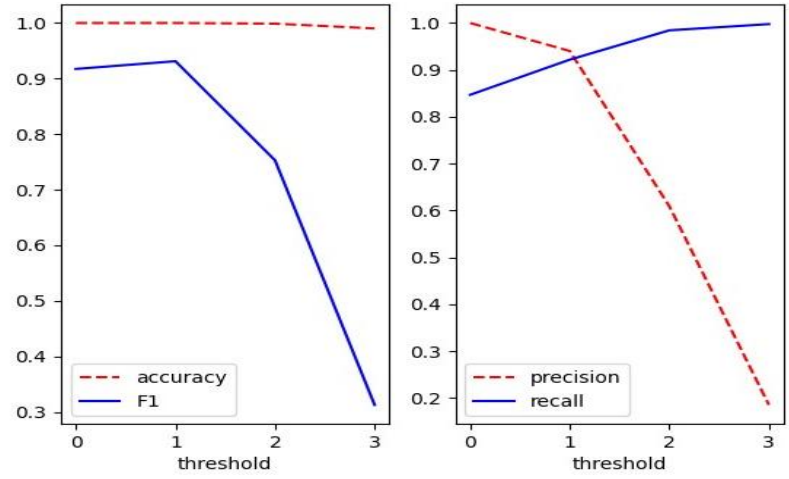

Fig. 7: Performances of edit distance measure (edit dist) in terms of accuracy, F1, precision and recall

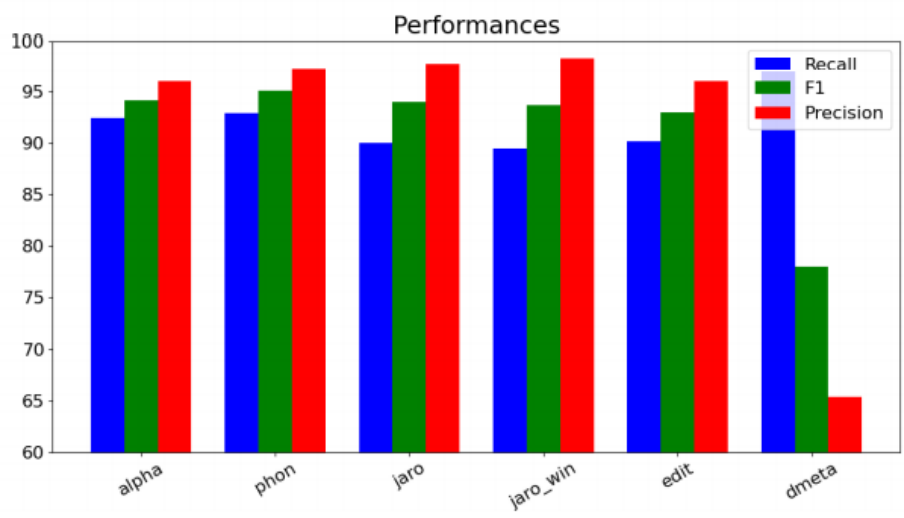

Fig. 8. Performances of alpha, phone, jaro and jaro-winkler metrics in terms of F1, recall and precision

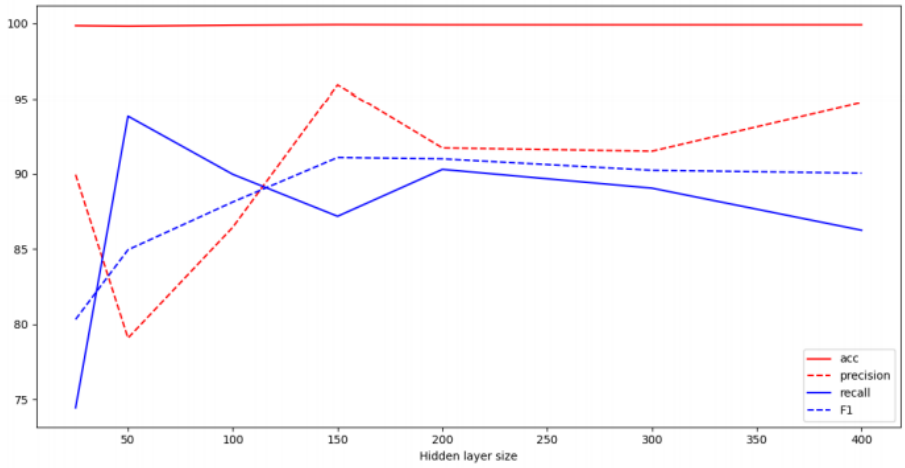

Fig. 9: Performances of the MLP with one hidden layer and relu activation function

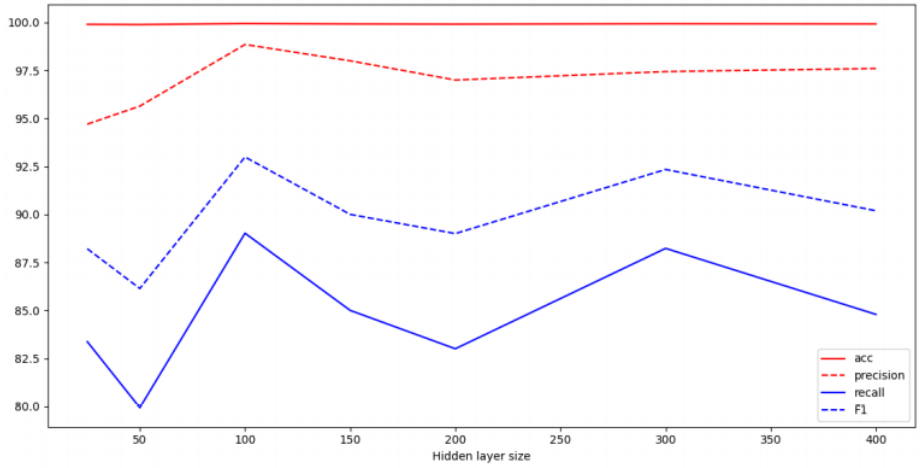

Fig. 10: Performances of the MLP with one hidden layer and logistic sigmoid activation function

As shown in Fig. 9, the MLP with one hidden layer and a relu activation function gave its best result with 150 neurons (F1 = 91.08%, precision = 95.91% and recall = 87.17%). With logistic sigmoid activation function (Fig. 10), best performance was reached with 100 neurons (F1 = 92.99%, precision = 98.85% and recall = 89.02%). A logistic sigmoid activation function seems to be more appropriate with this architecture.

In the second configuration, we consider an MLP with two hidden layers. A grid search was performed over the size of these hidden layers (the size of each layer is drawn from the set of values [25,200] with 25 step) and with the relu and the logistic sigmoid activation functions. Results are reported in Fig. 11 and 12.

For the relu activation function, the best-performing model (F1 = 94.37%, precision = 94.10% and recall = 94.65%) was achieved by (l1, l2) = (25,200). However, for the logistic sigmoid activation function, the best-performing model (F1 = 94.82%, precision = 97.84 and recall = 92.15%) was reached by (l1, l2) = (150,175). Here also, using logistic sigmoid activation function has yielded a better results than relu.

Comparing our two approaches (Table 9), we can confirm that alpha outperformed all configurations of the MLP with one hidden layer. The phone similarity measure has outweighed all configurations of the MLP with two hidden layers.

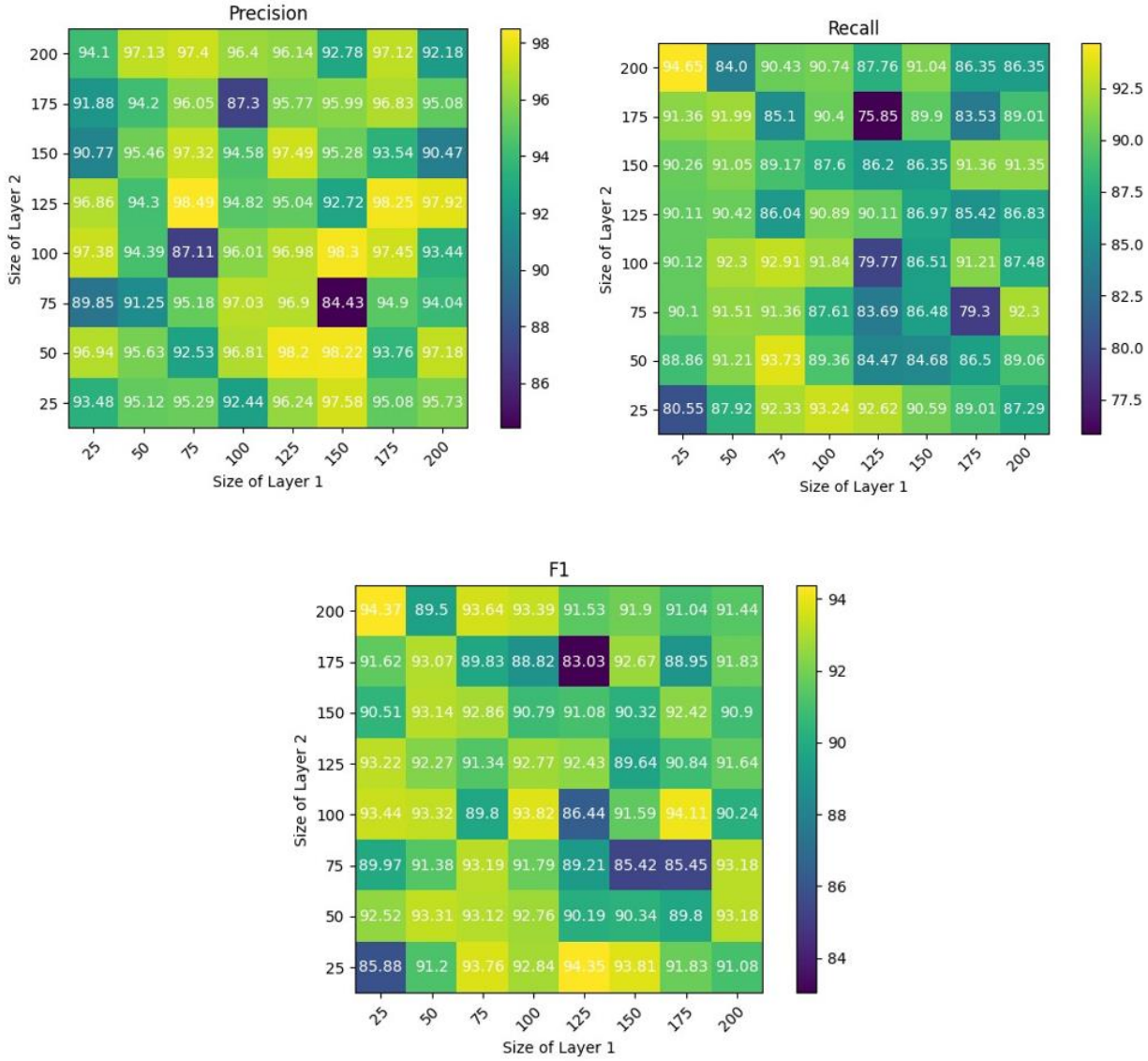

Fig. 11: Performance of the MLP with two hidden layers and relu activation function, in terms of precision, recall and F1

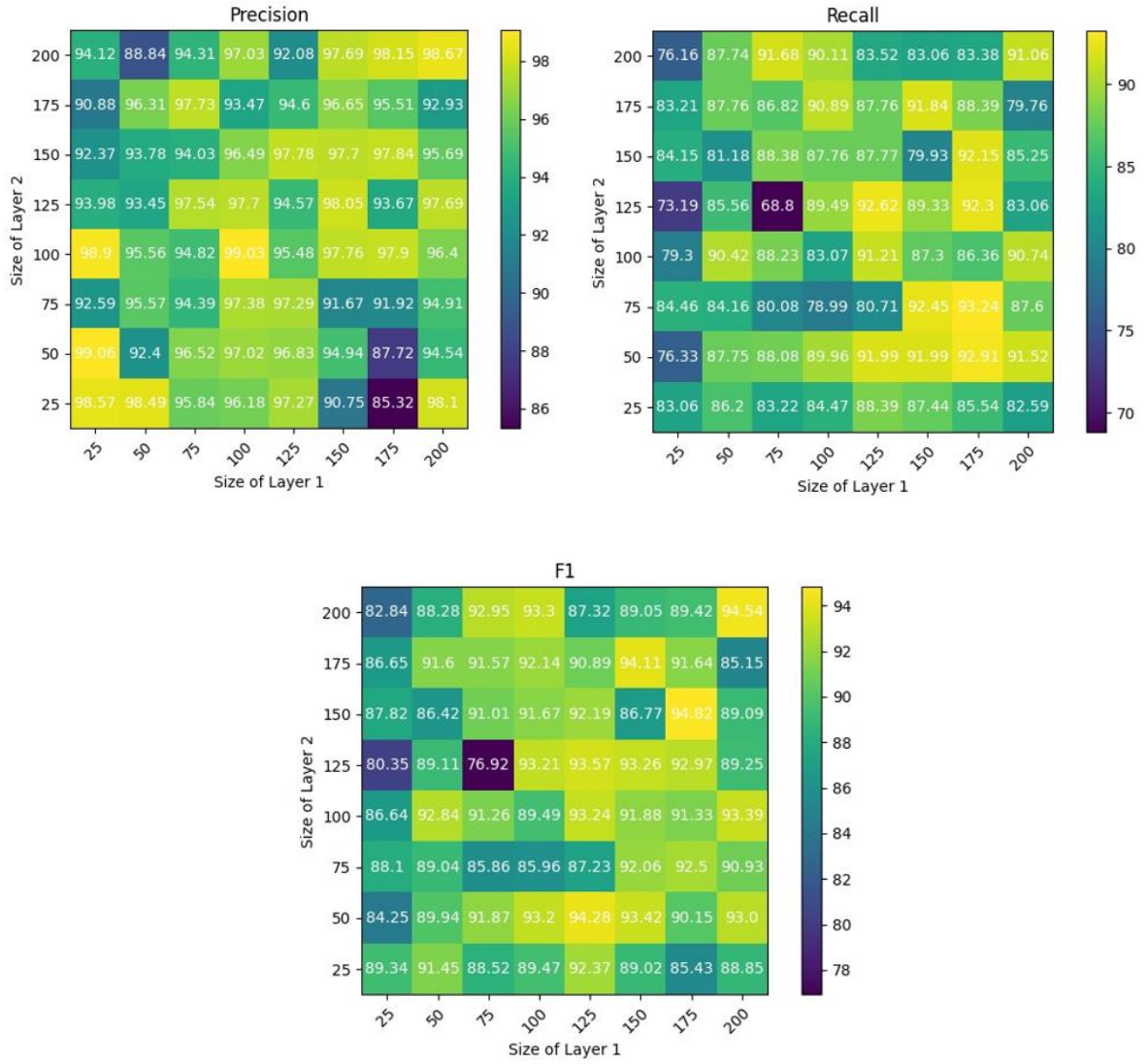

Fig. 12: Performance of the MLP with two hidden layers and logistic activation function, in terms of precision, recall and F1

Conclusion and Future Directions

In this study, we introduced two new approaches for pairwise matching of Arabic personal names, written with the Latin alphabet. The first approach is based on the use of string names alignment with an appropriate scoring model and the phonetic transcription. The first derived method operates on source names written with Latin alphabet without any transcription. An appropriate scoring model is defined based on human expertise that gave our alpha similarity measure. In the second method, string names are first converted to their phonetic transcription, then a scoring model for the IPA alphabet was defined, which resulted in our phone similarity measure.

Implementation of this approach and analysis of experimental results against four other similarity measures, namely: Edit Distance, Double Metaphone, Jaro and Jaro-Winkler showed the appropriateness of our derived measures. We found that alpha and phone gave a reasonable precision-recall trade-off. Most notably, this is the first study, to our knowledge, to address the Arabic personal names matching problem as an alignment of strings written in Latin alphabet and mapped to phonetic transcription.

In the second approach, we proposed a simple yet effective neural architecture to learn a classifier that maps a pair of string names to a binary class. Experiments showed that using a deep neural network architecture (two hidden layers) and by means of an appropriate size and activation function, the MLP succeeded to reach very

good performances. Though, using more data and deeper architectures can result in a more powerful classifier.

However, some limitations are worth noting. The deep analysis of bad decisions taken by alpha and phone similarities appeals for more efforts on dealing with peculiarities of both Arabic phonetics and Latin script. In future work, we will focus on annotating more large dataset and experimenting with more elaborated scoring models.

Acknowledgment

The authors are grateful to general direction of scientific research and technological development (DGRSDT) – algeria, for supporting this research.

Author's Contributions

Attia Nehar: Designed the research plan, developed the theory and performed the computations and verified methods. Contributed to the presentation, analysis and interpretation of the results.

Slimane Bellaouar: Made considerable contributions to this research by critically reviewing the literature review and the manuscript for significant intellectual content.

Djelloul Ziadi: Supervised the study and made considerable contributions to this research by critically reviewing the manuscript for significant intellectual content. provided critical feedback in manuscript.

Khaled Moulay Omar: Presented idea. Contributed to the collect of data and annotation.

Ethics

This article is original and contains unpublished material. The corresponding author confirms that all of the other authors have read and approved the manuscript and no ethical issues involved.

References

Biot, M. A. (1956). Theory of deformation of a porous viscoelastic anisotropic solid. Journal of Applied physics, 27(5), 459-467. https://aip.scitation.org/doi/abs/10.1063/1.1722402

Christen, P. (2006, December). A comparison of personal name matching: Techniques and practical issues. In Sixth IEEE International Conference on Data Mining-Workshops (ICDMW'06) (pp. 290-294). IEEE. https://ieeexplore.ieee.org/abstract/document/4063641

Cohen, W. W., Ravikumar, P., & Fienberg, S. E. (2003, August). A Comparison of String Distance Metrics for Name-Matching Tasks. In II Web (Vol. 3, pp. 73-78). http://dc-pubs.dbs.uni-leipzig.de/files/Cohen2003Acomparisonofstringdistance.pdf

Dweik, B., & Al-Sayyed, S. I. (2016). Translating Proper Nouns from Arabic into English: Barriers and Procedures. Arab World English Journal (AWEJ) Special Issue on Translation, (5). https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2795878

Halimah, A. (2016). Translating Arabic Proper Names: A Foreignizing Approach. International Journal of English Language and Linguistics Research, 4(2), 1-16.

IPAIPAS. (1999). Handbook of the International Phonetic Association: A guide to the use of the International Phonetic Alphabet. International Phonetic Association and International Phonetic Association Staff. Cambridge University Press. ISBN-10: 0521637511.

Jaro, M. A. (1989). Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. Journal of the American Statistical Association, 84(406), 414-420. https://www.tandfonline.com/doi/abs/10.1080/01621459.1989.10478785

Levenshtein, V. I. (1966, February). Binary codes capable of correcting deletions, insertions and reversals. In Soviet physics doklady (Vol. 10, No. 8, pp. 707-710). https://nymity.ch/sybilhunting/pdf/Levenshtein1966a.pdf

O'Grady, G. (2012). Key concepts in phonetics and phonology. Macmillan International Higher Education. ISBN-10: 1137292725.

Philips, L. (1990). Hanging on the metaphone. Computer Language, 7(12), 39-43.

Philips, L. (2000). The double metaphone search algorithm. C/C++ users journal, 18(6), 38-43. https://dl.acm.org/doi/abs/10.5555/349124.349132

Van Berkel, B., & De Smedt, K. (1988, February). Triphone Analysis: A Combined Method for the Correction of Orthographical and Typographical Errors. In Second Conference on Applied Natural Language Processing (pp. 77-83). https://www.aclweb.org/anthology/A88-1011.pdf

Winkler, W. E. (1990). String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. https://eric.ed.gov/?id=ED325505