Short Paper

Identifying Bias in Name Matching Tasks

Alexandros Karakasidis
Department of Applied Informatics
University of Macedonia
Thessaloniki, Greece
a.karakasidis@uom.edu.gr

Evaggelia Pitoura
Department of Computer Science & Engineering
University of Ioannina
Ioannina, Greece
pitoura@cs.uoi.gr

ABSTRACT

One of the applications of string similarity measures regards identifying the same entity within one or more corpora of data, also known as the problem of entity resolution. While a lot of methods have been introduced for assessing the similarity of two strings, there has not been yet a study to examine whether these methods are appropriate, in terms of effectiveness and fairness when applied to names of specific groups of individuals. In this paper, we provide extensive experimental results over a number of popular string measures which indicate that string comparison methods fall short when applied to specific groups, a fact leading to algorithmic bias against these groups. We also share some thoughts and guidelines on the way we envision, database practitioners should address such cases.

1 INTRODUCTION

Recently, the problem of algorithmic bias and fairness has drawn significant attention [8]. There is an increasing concern about the potential risks of data-driven approaches in decision making algorithms [2, 8, 15, 16, 18], raising a call for equal opportunities by design [9]. Biases can be introduced at different stages of the design, implementation, training and deployment of machine learning algorithms.

The problem has mainly been studied in the context of fairness in classification tasks, where individuals are classified in a positive or negative class. Example applications include, among others, hiring, school admission, crime risk factor estimation, and advertisement selection. There are two general approaches to defining fairness, namely *group* and *individual fairness* [6]. A common example of group fairness is *statistical parity*, where the proportion of members in a protected class that receive positive classification is asked to be identical to the proportion in the general population. Individual fairness requires that similar people are treated similarly.

In this paper, we draw attention to issues of bias in the task of entity resolution and, in particular, to string matching for name matching tasks. Approximate string matching forms the basis of a variety of algorithms in many applications where the ability to identify a word, despite of misspellings or typographical errors, is of crucial importance.

As such, there is a significant body of work on string comparison methods for name matching tasks [3, 7]. There has also been work on assessing the performance of these string matching algorithms [4, 17]. Amongst others, Lange and Naumann [11] consider different similarity measures for different frequencies of names in a database. Nevertheless, to the best of our knowledge, it is the first time that the problem of bias occurring in name matching is examined.

In this paper, we first provide a definition of bias in name matching tasks. Intuitively, we consider a matching algorithm to be negatively biased against a specific group of names, if the mismatches occurring for names in this group exceed those occurring on the average. Then, we examine a number of widely used string similarity measures in terms of bias.

We provide extensive experimental evidence that there is bias against certain race groups when applying string comparison algorithms for name matching tasks. This kind of biased behavior appears mainly against people with Asian origin, not intentionally, but as a consequence of the basic characteristic these names have, which is short length, compared to the names of people of other races. We then discuss approaches towards making methods for approximately matching names fair.

In brief, the contributions of this paper are outlined as follows:

- We identify the problem of bias in string comparison methods for name matching tasks.
- We propose a definition of bias for name matching tasks.
- We provide empirical evidence (in the form of extensive experimentation) of bias in a number of string similarity measures.
- We discuss how this problem may be addressed.

The rest of this paper is structured as follows. Section 2 introduces our measure for quantifying bias in name matching tasks, followed by a short description of the methods used in our assessment. Next, in Section 3, we present our experimental evaluation. Finally, we conclude in Section 4, summing up our findings, providing our vision on how bias cases for name matching tasks should be addressed by database practitioners and provide some thoughts for future work.

2 METHODOLOGY

In this section, we present our definition for bias in name matching tasks and a brief description of the string matching methods used in our evaluation.

2.1 Defining Bias

We need to quantitatively define the notion of bias in the context of record matching. In general, we want to capture the fact that for specific groups in the user population, there is unfair behavior. Let us use U to represent the general set of users. We assume that users are divided into m groups, G_i, $\cup_i G_i = U$, and $G_i \cap G_j = \emptyset$, for $i \neq j$, based on the value of some protected attribute, such as, for example, race, or gender.

Intuitively, we assume that there is bias against some specific user group G, if there are more errors in the results of matching when it comes to users in G than in the general population. There are two cases of errors, or *mismatches*. The first case is when two records are falsely reported as a match. This corresponds to a False Positive (FP). The second case is when two records that correspond to the same entity are not matched. This corresponds to a False Negative (FN). This is formalized in Definition 1.

Definition 1 (group mismatch). *The mismatch M for a user group G of size N is defined as* $M(G) = \frac{\sum_{u \in G} FP(u)+FN(u)}{N}$, *where* $FP(u)$ *and* $FN(u)$ *are the false positives and false negatives when matching names referring to user u in G.*

We determine whether a name matching method is biased towards a particular group by comparing its mismatch performance for this group with its mismatch performance for all groups of the general population U. To this end, we define

$$GM(U) = \frac{\sum_{G_i \in U} M(G_i)}{m} \quad (1)$$

Definition 2 (name matching bias). *The bias B of a string comparison method for a group population G is defined as:*

$$B(G) = \frac{GM(U) - M(G)}{GM(U)} \quad (2)$$

Let us now discuss the values of bias and their meanings. If G's mismatches are equal to the average number of mismatches of all groups, then its bias is equal to zero. On the other hand, if the mismatches of G are more than the average, then bias gets a negative value, and, in this case, we consider that the method examined is negatively biased against G. Finally, when the average number of mismatches is greater than that of G, then we assume that the method is biased in favor of G. Values close to 0 indicate low bias, while as the absolute value of bias increases, this is an indication of increased bias, either positive or negative.

In this paper, we consider string matching algorithms applied to names and race as the protected attribute.

2.2 Methods for Name Matching

Let us now briefly present some popular string comparison methods for name matching tasks that we use in our empirical evaluation. Let us consider two strings, which, in our case, represent names, S_1 and S_2.

Edit Distance. The edit distance between S_1 and S_2 is the minimum number of edit operations required to transform S_1 to S_2. We use the Levenshtein edit distance [12] where an edit operation is a character insertion, deletion or replacement, and each operation has cost equal to 1.

Jaro & Jaro-Winkler Similarities. The Jaro similarity [10] between two strings S_1 and S_2 is defined as:

$$JS(S_1, S_2) = \begin{cases} 0, & \text{if m=0} \\ \frac{1}{3}\left(\frac{m}{|S_1|} + \frac{m}{|S_2|} + \frac{m-t}{m}\right), & \text{otherwise} \end{cases}, \text{where } |S_1|, |S_2|$$

are respectively the lengths of S_1 and S_2, m is the number of matchings, and t is the number of transpositions defined as follows. Two characters are considered matching, if they are the same and within distance less than $\left\lfloor \frac{max(|S_1|,|S_2|)}{2} \right\rfloor - 1$. The number of transpositions is calculated as follows: The i-th common character in S_1 is compared with the i-th character in S_2. Each nonmatching character is a transposition.

Jaro - Winkler similarity [19] is based on Jaro similarity, favoring prefix matches, as they are considered of higher importance for name matching. It is defined as: $JWS(S_1, S_2) = JS(S_1, S_2) + l\,p(1 - JS(S_1, S_2))$, where l is the length of common prefixes of S_1, S_2, topping at four common characters and p a scaling parameter defaulting to 0.1.

Q-gram based methods. Q-grams are successive substrings of a string each of length Q. Each string is divided into a set of distinct Q-grams. Let N_1 and N_2 be the sets of Q-grams of strings S_1 and S_2 respectively. There are various set comparison measures [13]. In this paper, we use the Dice coefficient and Jaccard index, where the Dice coefficient is defined as $DC(S_1, S_2) = \frac{2|N_1 \cap N_2|}{|N_1|+|N_2|}$, and the Jaccard index as to $JI(S_1, S_2) = \frac{|N_1 \cap N_2|}{|N_1 \cup N_2|}$.

Table 1: Soundex mapping table.

a, e, h, i, o, u, w, y	→	0	*l*	→	4
b, f, p, v	→	1	*m, n*	→	5
c, g, j, k, q, s, x, z	→	2	*r*	→	6
d, t	→	3			

Soundex. Soundex, based on English language pronunciation, is the oldest (patented in 1918 [14]) and best known phonetic encoding algorithm. It keeps the first letter of a string and converts the rest into numbers according to Table 1. All zeros (vowels and 'h', 'w' and 'y') are then removed and sequences of the same number are merged to a single one. The final code is the original first letter and three numbers (longer codes are stripped off, and shorter codes are padded with zeros).

3 EXPERIMENTAL EVALUATION

In this section, we present the findings of our empirical assessment.

3.1 Experimental Setup

For our evaluation, we use data from the US Census Bureau[1], list of names. This list contains 162253 distinct names and the frequency of appearance of each of these names in several race groups in the agency's database. We selected three of the most popular of these race groups. These are the groups of names from people characterized as White, Afro-americans and those with Asian or Pacific origin. Since people of different races may have the same name, we keep only those names for which at least 90% of the people having these names belong to a single group. Then, we extracted the 50 most common names from each of these groups. The statistics of the dataset used in this study are depicted in Table 2.

Since our aim is to assess the effectiveness of string similarity methods for name matching tasks, we need alternative or erroneous versions of the names that we have extracted, for matching them with their original versions. For this purpose, we have employed the data corrupter from the German Record Linkage center [1]. This was configured to perform one random edit operation (insertion, deletion or replacement) in each of the names we have extracted, so as to produce 1 error for each of them, as it is rather unusual for a word to have more than one or two typographical errors [5].

Then, we perform record linkage between the original and the corrupted version of the datasets using each of the string comparison methods presented earlier. For each of these methods, we perform the same experiment 10 times. The matching thresholds used for each method are illustrated in Table 3.

For the Levenshtein distance, the minimum difference between two strings is equal to 1, meaning one modification. Jaro, Jaro-Winkler, Dice are Jaccard are similarity measures returning values between 0, for total dissimilarity, and 1 for absolute similarity. Finally, we desire codes produced by Soundex to totally coincide, since its representation contains approximation characteristics.

[1] Available at https://www.census.gov/topics/population/genealogy/data/2010_surnames.html

Table 2: Statistics of the dataset.

Race Group	Average occurrences per name in top-50	Average occurrences per name in group	Number of distinct names in group
Asian	68861.44	964.1	2897
Black	3027.06	272.52	2364
White	91959.86	681.72	100688

We built a testbed using Anaconda Python 3.5 and conducted our experiments on Ubuntu 18.04 LTS, powered by an Intel i5 processor with 8 GBs of RAM. The results of our assessment are

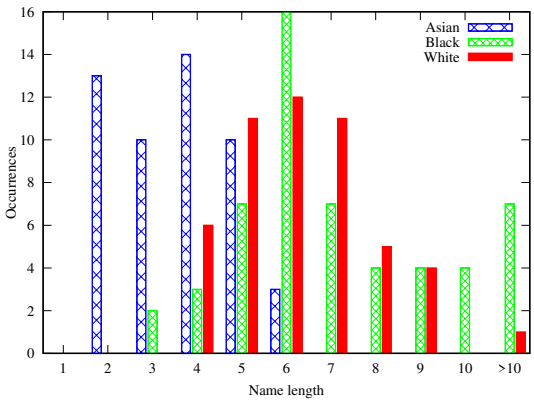


Figure 1: Length distributions for the top-50 names.

illustrated in Figure 2. In all plots, the horizontal axis is used to display the various thresholds used in our evaluation, while the vertical axis represents bias. In each plot, the results for all races are illustrated using error bars, displaying the minimum, average and maximum measurement.

3.2 Discussion of the Results

Figure 2 indicates that all measures treat names from different groups differently, that is, all measures exhibit some form of bias. As a first step to interpret this bias, we seek to identify similar characteristics in each of the group of names. As such, we measure the lengths of the names in each group. These results are illustrated in Fig. 1. As we can see, the shortest names belong to people whose race is characterized as Asian.

We will now attempt to associate the length of these race groups with the behavior of the string comparison methods. As we can observe from Fig.2(a), the Levenshtein distance method performs worse for names with Asian origin than for names associated with people of White or African race. For the Asian group, the Levenshtein distance incurs negative bias lower than -1.67 when considering as matching names featuring an edit distance equal to 2, which is the lowest score in our evaluation. We speculate that, the reason for this behavior is the shorter length of the Asian names, as opposed to the other two groups, causing more ambiguous matches. The method seems to be slightly biased in favor of both the other two groups, with the White group receiving higher positive bias between the two.

As shown in Fig 2(b), Jaro also exhibits a continuous negative bias for the Asian group and a positive bias towards the African group. For the White names dataset, however, the situation is somewhat mixed. For matching threshold equal to 0.7, the bias is negative, then it becomes almost zero for threshold equal to 0.8 and it turns positive for a threshold equal to 0.9. This occurs because as we increase the threshold, the number of false positive matches drops, while the number of false negative matches remains merely unchanged. For Jaro-Winkler (Fig. 2(c)) the negative bias against Asian names is evident, again, as the positive bias in favor of African names. In this case, however, the method seems to also be negatively biased against White names as well.

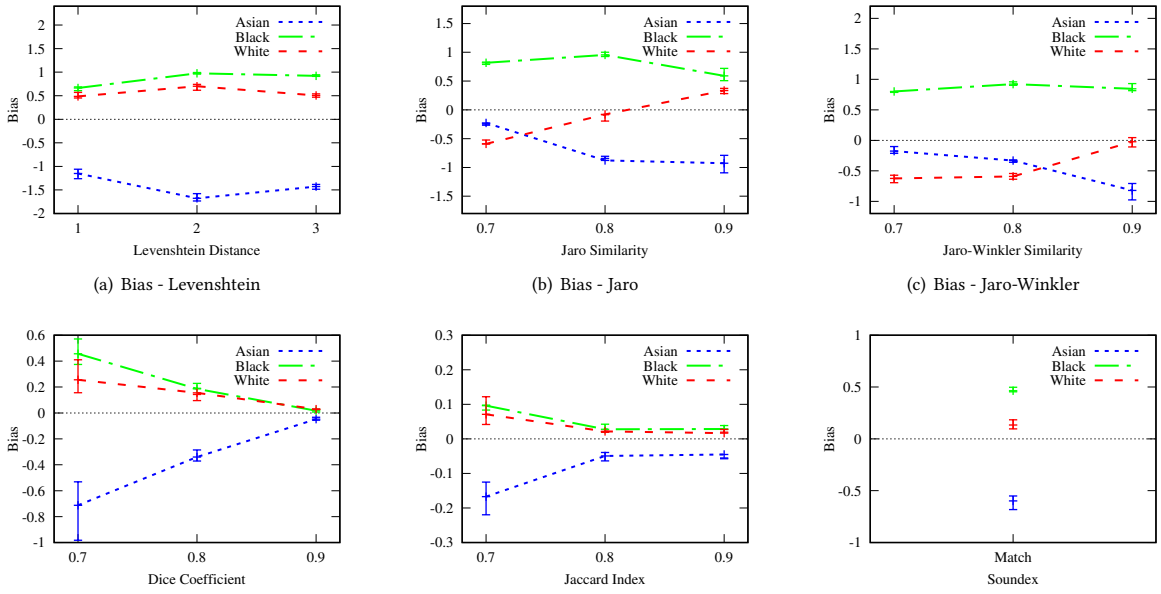
Let us now consider the case of breaking names into sets of Q-grams. For the Dice coefficient 2(d), we observe that the bias for all groups tends to converge to zero, when we use a high matching thresholds equal to 0.9. This is because, there are zero matches in most cases. As such, the number of false negatives is almost the same for each case and the number of false positives is the factor affecting the bias for each method. Examining each group individually now, it is evident that, again, there is negative bias towards the Asian group. The other two groups exhibit similar behavior, with the African one receiving more positive bias. The use of the Jaccard index (Fig. 2(e)) results in a similar behavior. Again, the Asian dataset is far below average, while the other two are very close and slightly positive. In both cases, however, the incurred bias is much lower than in the previous methods.

Table 3: Comparison methods and threshold values used.

Method	Threshold values
Levenshtein	1, 2, 3
Jaro, Jaro-Winkler, Dice, Jaccard	0.7, 0.8, 0.9
Soundex	1

Finally, we consider the use of Soundex, which has been specifically designed for name matching tasks. In this case, a threshold is not used for matching, and there is a match when two phonetic codes coincide. The results for this case are illustrated in Figure 2(f). As we may see, again, the results are similar with the previous cases. The Asian names exhibit negative bias, exceeding -0.5, while African names score very close to +0.5. White names are very close to the average case, slightly positively biased, nevertheless. A reason for this behavior is that, Soundex does not consider vowels. As such, since Asian names are usually short having similar vowels, it is very easy for the algorithm to be led to false positives.

Summarizing, regardless of the string similarity measure used, there is significant evidence that there is negative bias when matching names of people belonging to the Asian race, while there is positive bias when matching names of people belonging to the African race. Depending on the method used, White names may be either positively or negatively biased, falling, in most situations, within the average case. Besides this general behavior, some of the string measures are better than others in terms of the volume of bias, with the Q-gram Jaccard method outperforming the others by exhibiting the smaller bias.



(a) Bias - Levenshtein (b) Bias - Jaro (c) Bias - Jaro-Winkler

(d) Bias - Q-gram Dice (e) Bias - Q-gram Jaccard (f) Bias - Soundex

Figure 2: Bias of the string comparison methods.

4 CONCLUSIONS, THOUGHTS AND FUTURE WORK

Bias of this or some other form may be present in different steps of entity resolution and a lot of work is needed to formalize, detect and remedy it in the many different algorithms and techniques around this important task.

Our preliminary studies have revealed that, even in the basic task of name matching, there may be bias, in the sense that the quality of the results varies based on the group an individual belongs to. In this case, bias is not malicious, but rather a side effect of the data values (names) of the attributes of individuals belonging to specific groups. For name matching, we believe that we provide some useful insight to database practitioners for handling such situations. To this end, considering a single threshold for matching names from all origins, does not yield correct results for the Asian names, therefore this is a factor of concern. A first measure to alleviate this issue would be to consider the lengths of the names, or race information, if available, in cases of name matching tasks and maybe perform this task separately for this class of names.

For the future, there are many additional steps to be taken, as our study focuses on a very specific task, namely string matching based on names. We aim at performing further experiments considering data from more distinct groups which are going to be addressed as hidden parameters and examining deeper the structures of the names of each case, in order to reveal more factors that cause algorithmic bias in name matching tasks.

REFERENCES

[1] Tobias Bachteler and Jarg Reiher. 2012. *A Test Data Generator for Evaluating Record Linkage Methods.* Technical Report. German RLC Work. WP-GRLC-2012-01.

[2] Solon Barocas and Andrew D. Selbst. 2014. Big Data's Disparate Impact. *SSRN eLibrary* (2014).

[3] Peter Christen. 2012. *Data Matching: Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer Publishing Company, Incorporated.

[4] William Cohen, Pradeep Ravikumar, and Stephen Fienberg. 2003. A comparison of string metrics for matching names and records. In *Kdd workshop on data cleaning and object consolidation*, Vol. 3. 73–78.

[5] Fred J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *CACM* 7, 3 (1964).

[6] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference.* ACM, 214–226.

[7] Ahmed K. Elmagarmid, Panagiotis G. Ipeirotis, and Vassilios S. Verykios. 2007. Duplicate Record Detection: A Survey. *IEEE Transactions on Knowledge and Data Engineering* 19, 1 (Jan. 2007), 1–16.

[8] Sara Hajian, Francesco Bonchi, and Carlos Castillo. 2016. Algorithmic bias: From discrimination discovery to fairness-aware data mining. In *SIGKDD.* ACM, 2125–2126.

[9] White House. 2016. Big Data: A Report on Algorithmic Systems, Opportunity, and Civil Rights. *Washington, DC: Executive Office of the President, White House* (2016).

[10] Matthew A. Jaro. 1978. *UNIMATCH : a record linkage system : users manual.* Bureau of the Census Washington.

[11] Dustin Lange and Felix Naumann. 2011. Frequency-aware Similarity Measures: Why Arnold Schwarzenegger is Always a Duplicate. In *CIKM.* ACM.

[12] Vladimir I Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. 707–710.

[13] Chen Li, Jiaheng Lu, and Yiming Lu. 2008. Efficient merging and filtering algorithms for approximate string searches. In *ICDE.* IEEE, 257–266.

[14] Margaret. Odell and Robert Russell. 1918. The Soundex coding system. *US Patents* 1261167 (1918).

[15] Evaggelia Pitoura, Panayiotis Tsaparas, Giorgos Flouris, Irini Fundulaki, Panagiotis Papadakos, Serge Abiteboul, and Gerhard Weikum. 2018. On Measuring Bias in Online Information. *SIGMOD Rec.* 46, 4 (Feb. 2018), 16–21.

[16] Andrea Romei and Salvatore Ruggieri. 2014. A multidisciplinary survey on discrimination analysis. *The Knowledge Engineering Review* 29, 05 (2014), 582–638.

[17] Chakkrit Snae. 2007. A comparison and analysis of name matching algorithms. *International Journal of Applied Science. Engineering and Technology* 4, 1 (2007), 252–257.

[18] Julia Stoyanovich, Serge Abiteboul, and Gerome Miklau. 2016. Data, Responsibly: Fairness, Neutrality and Transparency in Data Analysis. In *EDBT.*

[19] William E Winkler. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. (1990).