An Alignment Comparator for Entity Resolution with Multi-valued Attributes

Pablo N. Mazzucchi-Augel and Hector G. Ceballos

Tecnológico de Monterrey, Campus Monterrey, Mexico
{A00814519,ceballos}@itesm.mx

Abstract. Entity matching is a problem that concerns many data management processes. If we consider matching between entities represented by RDF individuals we might find attributes values lists with variable-length for some properties, which will lead us to the problem of comparing multi-valued attributes, e.g. comparing author names lists for determining publication matching. This matching technique would be more complex than comparing fixed-length records, but less complex than comparing XML documents. Instead of comparing a single string, representing the concatenation of these values, each value of one vector should be compared against all values of the other vector. We propose a set of heuristics to address the alignment and comparison process of multi-valued attributes and evaluate them in the context of bibliographic databases. Our first results show that it is possible to reduce the comparisons amount and provide an aggregated similarity metric that outperforms the average similarity of cross product comparisons.

Keywords: Entity Resolution, Author Matching, Multi-Valued Attributes, Bibliographic databases.

1 Introduction

Entity matching (also referred to as duplicate identification, record linkage, entity resolution or reference reconciliation) is a crucial task for data integration and data cleaning. It is the task of identifying entities (objects, data instances) referring to the same real world entity [10]. Nowadays, organizations maintain records referring to the same entity in distributed databases. Some attributes are redundant, but others complement the information. Problems arise when information must be put together in order to extract knowledge.

For instance, in the integration of bibliographic databases publications have multi-valued attributes like authors and keywords that traditionally have been represented as a single string with a concatenation of items, but in RDF these values can be found disaggregated. Similarity between multi-valued attributes must be summarized in a single value that could be pondered in the overall similarity score.

For Linked Data, data integration consists of adding links between equivalent entities (e.g. publications) by accessing the entity description in distributed repositories. Frameworks like Silk [17] provide facilities for this purpose, but treat similarity between multi-valued attributes as the average of the cross product comparisons. In this

paper, we propose a novel heuristic which increases the efficiency when comparing multi-valued attributes, as long as it reduces the amount of comparisons and improves the distinction between equivalent lists.

The paper is structured as follows: Section 2 present some of the string matching and aggregation techniques find in literature. Section 3 gives an overview of the proposed approach. In Section 4 we show results of the different heuristics proposed and discuss them in Section 5, in order to identify the most effective one to address this problematic.

2 Background

Next we describe some approaches proposed for approximate string matching (particularly on author names) and for aggregating individual comparisons.

2.1 String Matching Techniques

The entity resolution problem has been known for more than five decades, not just in the statistics area, but also in the database community as well as in the AI community. Entity resolution or duplicate detection relies on string comparison techniques, which deal with the typographical variation of string data. Multiple methods have been developed for this task, and each method works well for particular types of strings. It is not the objective of this paper to develop an exhaustive and detailed explanation of string matching techniques, but to mention which are the most useful and used the bibliographic database domain. For a comprehensive review refer to [1], [19], [6].

So, to deal with typographical errors, the most suitable approaches are the character-based similarity metrics. The *edit distance* calculate the minimum amount of edit operations (insert, delete or replace) of single characters needed to transform a string S_1 into S_2. The version of edit distance where each operation cost 1, is known as *Levenshtein* distance. If those strings are truncated or shortened, a better metric is the *Smith-Waterman* distance. Winkler modified the metric introduced by Jaro (a string comparison algorithm that was mainly used for the comparison of last and first names) to give higher weight to prefix matches since prefix matches are more important for surname matching [13]. It does not just find common characters, but focus also in their order.

Character-based similarity metrics work well for typographical errors. However, it is often the case that typographical conventions lead to rearrangement of words (e.g., John Smith versus Smith, John). In such cases, character-level metrics fail to capture the similarity of the entities. Token-based metrics try to compensate this problem. In this group, could be categorized the *Monge-Elkan* distance, the *cosine* or *tf.idf* similarities, the *Jaccard* similarity or the hybrid approach *Soft TF-IDF* that combines the token based with the string-based methods [6].

2.2 Aggregation

In this section we list the techniques found in the literature for generating an aggregated score, using the numerous comparisons performed. The linear sum assignment problem (LSAP) is one of the most famous problems in linear programming and in

combinatorial optimization. Informally speaking, given an *NxN* cost matrix $C = (c_{ij})$ the objective is to match each row to a different column in such a way that the sum of the corresponding entries is minimized. In other words, to select *N* elements of *C* so that there is exactly one element in each row and one in each column, the sum of the corresponding costs should be the minimum [2]. Based on that approach, [18] introduced a linear sum assignment procedure to force 1-1 matching because he observed that greedy algorithms often made erroneous assignments. A greedy algorithm is one in which a record is always associated with the corresponding available record having the highest agreement weight. Subsequent records are only compared with available remaining records that have not been assigned. In [14] and [9] is used the Euclidean distance metric to combine distance similarity values, generated by the comparisons of multiples attributes of a tuple.

As the standard edit distance comparators did not work as well as the Jaro-Winkler formula in their experiments, [20] combined string distance's techniques (such as *Levenshtein* and *LCS*), averaging them, which seemed to produce better results. The objective was to use more information from the strings, and take advantage of each algorithm strength. The combined *root mean square (RMS)* has been also used by [8] to improve linkage accuracy over any single comparator.

In [3], a learning scheme is used to combine several of the distance functions detailed above. A binary SVM (support vector machine) classifier was trained, using as feature vector the numeric scores of those functions, and their confidence in the match task as the result of the comparison. It slightly outperforms the individual metrics. SVM do not generate an aggregated value like the other presented techniques, but combines partial results in order to improve the classification process.

3 The Matching Process

In this section we give an overview of the process that heuristics, presented in detail in Section 3.4, follow to increase the effectiveness in the alignment and comparison process of multi-valued attributes. Figure 1 illustrates the complete process, consisting of three stages: alignment, comparison and similarity aggregation.

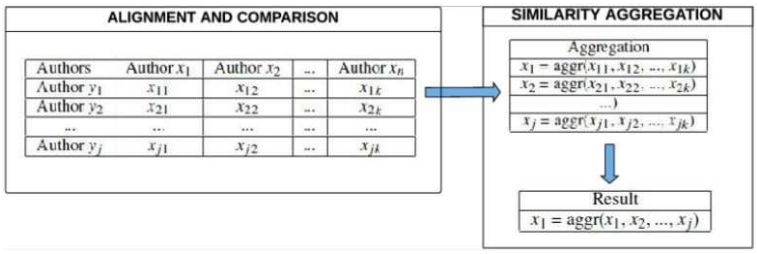

Fig. 1. Multi-valued attributes Alignment and Comparison Process.

3.1 Alignment

The alignment phase consists on arranging elements of both lists for having a comparison matrix where the diagonal $\{(1,1),...,(N,N)\}$ denotes the right comparisons to make. In some cases, multi-valued attributes are already ordered hence this step might not be necessary, e.g. publication author lists.

According to the clustering and blocking methods used in the Google Refine project [7], Fingerprint [12] resulted the most useful and promising approach to align multiple attribute values in such a way that the comparison process performance is increased. Basically, Fingerprint splits a string in tokens and order them alphabetically.

For some heuristics we applied Fingerprint for sorting the item lists. In these cases, after executing FingerPrint over each element of the lists, each array was arranged in ascending order. The string obtained from FingerPrint was just used for ordering purpose. For comparison, the real name of authors was used.

3.2 Comparison Strategies

The proposed comparison strategies try to minimize the number of author name comparisons. The combination of comparison strategies determine which elements of the comparison matrix $\mathcal{M}$ are filled: the less author-name comparisons, the better. Consider the lists of authors $A_1[N]$ and $A_2[M]$, being $N \geq M$. The comparison strategies will determine which element of $A_1[i]$ (0 < i < N) and $A_2[j]$ (0 < j < M) will be compared. The result of the comparison (the individual similarity) is stored in $\mathcal{M}[i,j]$. The comparison strategies used were:

- **Full-Matrix**: compare all elements in $A_1[i]$ with all elements in $A_2[j]$.
- **Diagonal**: compare elements in $A_1[i]$ with elements in $A_2[j]$, only when i=j.
- **Row-Col-Deletion**: if $\mathcal{M}[i,j] \geq$ min-similarity, comparisons $\mathcal{M}[a,b]$ are skipped for $i < a < N,\ j < b < M$.
- **Row-Similarity-Threshold**: if $\mathcal{M}[i,j] \geq$ min-similarity, continue comparing in the next row of the matrix ($\mathcal{M}[i+1,j]$), i.e. $\mathcal{M}[i,b]$ is left empty for $j < b < M$.

We additionally defined a stop criterion for avoiding unnecessary comparisons:

- **Partial-Average-Threshold**: compare elements in $A_1[i]$ and $A_2[j]$ until reaching 80% of $\mathcal{M}$. If Max-Average (described in Section 3.3) is above min-similarity, continue with the comparison process; otherwise, stop.

Given the nature of the element comparison problem and based on the taxonomy described by [5] we used *Content-based* matching approaches to determine the similarity of two entities (author names in our case). We decided to combine two techniques that compare atomic values: one for character-to-character comparison (*Jaro Winkler*), and another for token-to-token comparison (*Longest Common Substring (LCS)*). For combining the results of both of them, we used the *root mean square (RMS)*, to improve the similarity grade over a single comparator [11, 8]. Those author names which similarity score was above a defined threshold were considered similar.

3.3 Similarity Aggregation

Aggregation strategies try to determine the final similarity grade between the processed lists of authors. The strategies used are:

- **Average**: $\frac{\sum_{i=0}^{N}\sum_{j=0}^{M}\mathcal{M}[i,j]}{(N\times M)}$
- **Max-Average**: Max(Avg (x), Avg(y)), where $x = \sum_{i=0}^{N} Max(\mathcal{M}[i,*])$ and $y = \sum_{j=0}^{M} Max(\mathcal{M}[*,j])$.

In plain words Max-Average chooses between the maximum average of the sum of maximum values obtained from column or rows.

3.4 Evaluated Heuristics

We considered a heuristic as a selection of an alignment choice (original or Figerprint), a combination of comparison strategies, and an aggregation style. A summary and a brief description of the evaluated heuristics (labeled $H_0, H_1, H_2, H_4, ..., H_7$), can be found in Table 1. Comparison strategies are decomposed in two criteria: one for choosing which elements to compare next, and another for determining when to stop comparing.

Table 1. Evaluated Heuristics.

Heuristic	Alignment	Comparisons		Aggregation
H_0 / H_1	Original Order	Full Matrix (NxM)		H_0 Average H_1 Max-Average
		Who's Next	**Stop Criteria**	
		if (j < M) then j=j+1 else j=0, i=i+1	i = N and j = M	
H_2	FingerPrint	1. Diagonal. 2. If sim < min-similarity then H_4.		Max-Average
		Who's Next	**Stop Criteria**	
		if (i<N) and (j<M) j=j+1, i=i+1	i = N and j = M	
H_4/ H_5	Original Order	Row-Col-Deletion + Partial-Average-Threshold		Max-Average
		Who's Next	**Stop Criteria**	
		if ($A_1 \wedge A_2$ are visible) then Row-Col-Deletion Row-Similarity-Threshold else if (j < M) then continue next column (j=j+1) else continue next row (i=i+1)	H_4 Partial-Average-Threshold or (i=N and j=M) H_5 i=N and j=M	
H_6/ H_7	H_6 Original Order H_7 FingerPrint	Row-Col-Deletion		Max-Average
		Who's Next	**Stop Criteria**	

		if ($A_1 \wedge A_2$ are visible) then
		H_6: Row-Col-Deletion
		H_7: Row-Col-Deletion, Row-Similarity-Threshold
		if (j < M) then continue next column (j=j+1)
		else continue next row (i=i+1)
			H_6
			i=N and j=M
			H_7
			Partial-Average-Threshold
			or (i=N and j=M)	

The heuristic H_0 which averages all possible author names comparisons, is the traditional process followed to establish a similarity grade between two lists of authors, so we select it as our baseline, in order to compare if the other heuristics generate any

improvement in the process of establishing a similarity grade when comparing multi-valued attributes.

Algorithm 1 shows an implementation of heuristic H_4, which implements comparison strategies Row-Col-Deletion and Row-Similarity-Threshold, whereas it uses the Partial-Average-Threshold criterion for quit comparing if the comparison is not good enough.

Algorithm 1 Heuristic H_4 (Row-Col-Deletion + Row-Similarity-Threshold + Partial-Average-Threshold)

```
Input: alist_1 / alist_2
Result: Matrix M

1  for (i, i < alist_1.length , i++) {
2    if (alist_1[i] is available)
3      for(j, j < alist_2.length, j++) {
4        if (alist_2[j] is available)
5          M[i,j] = calculate_similarity(alist_1[i], alist_2[j])
6          if (M[i,j] >= threshold)
7            Turn off availability of alist_1[i], alist_2[j]
8            i = i+1 // Continue on next row
9          end-if
10       end-if
11       if (checkpoint)
12         if (Max-Average(M) < min-similarity)
13           Stop process
14     }
15   end-if
16 }
```

4 Experimental Results

In this Section we describe the datasets used in our experiments and the execution time for each heuristic. Finally, we present the results obtained in similarity classification and efficiency experiments.

4.1 Datasets

We use two data-sets, one extracted from an internal repository and the other one from ISI Web of Knowledge. Both contain 548 records, the first one with information about papers published by our University researchers and the second one with the equivalent papers in the ISI Web database. From both datasets we used the publication ID and the author list (parsed for comparing individual author names).

Each author list of one file was compared against all the other author lists of the second file, performing a total of 300,304 comparisons. Of this total, 1,052 author-lists were equivalent (all the items in one list were in the other and vice versa). This is because the same group of authors could appear in multiple publications.

It is important to mention the difference between the author names comparison and the author lists comparison. The first one refers to the comparison performed between two authors, for example *Gutierrez-Vega, JC* against *Munoz-Rodriguez, D*, while the second one implies the aggregated similarity between two lists of authors, for example $\{$*Gutierrez-Vega, JC; Chavez-Cerda, S; Rodriguez-Dagnino, RM*$\}$ versus $\{$*Aleman-Llanes, E; Munoz-Rodriguez, D; Molina, C*$\}$.

4.2 Running Time

To evaluate the computational cost of the algorithms, we measured the time each of them required to process the 300,304 author list comparisons. We used the average time of five trials for each heuristic. Previous to each trial, the warm-up stage last 1200 ms. A subset of the records were used multiple times during that period. The experiments had been run on an *Intel(R) Core(TM) i7 CPU Q 720 @ 1.60 GHz* machine, running a *64-bit Operating System (Windows 8 Pro)* with *4GB RAM*. The algorithms were implemented in Java. The average run time of each heuristic is shown in Table 2. It can be seen that heuristic H_4 run faster than H_0 and the rest of the evaluated heuristics.

Table 2. Total average processing time for 300,304 records.

Heuristic ID	Run Time (in sec)	Heuristic ID	Run Time (in sec)
H_0	241	H_5	258
H_1	261	H_6	260
H_2	293	H_7	237
H_4	204		

4.3 Author Names Alignment

In the first place we evaluated if the proposed heuristics were capable of identifying one to one matches, and based on that determining how similar two author lists were. We used 0.82 as the minimum similarity threshold for our experiments. Author name comparisons with similarity above this threshold were considered as similar. Higher thresholds were tested, but resulted in a higher quantity of false-negatives. We also used the same threshold (0.82) to decide if two author lists were similar or not. Considering the author-name comparisons made, each pair of author lists was classified in the following groups:

- **No match**: None of the authors matched.
- **Some matches**: At least one author matched (this means that at least one author had a similarity value above or equal to the threshold), but not all of them.

– **All match**: All authors matched (both author lists have the same length ($N = M$) and N author name comparisons are above the min-similarity threshold).

In Figure 2 is shown the result of the classification. Note that in this experiment the baseline is not H_0. The baseline indicates how many authors actually shared the compared author-lists. It is convenient to mention the reason those heuristics which perform all possible comparisons, exceed the classified amount of authors-lists in the *Some matches* group. Cases like *Acevedo-Mascarúa, J.* and *Aceves, J* produce similarity values (0.84) above threshold (0.82), when in fact, those authors were not the same person. Then, when compared the following lists: {*Acevedo-Mascarúa, J.; Salguerio, M.*} and {*Arcos, D; Sierra, A; Nunez, A; Flores, G; Aceves, J; Arias-Montano, JA*}, the heuristics determined that one similarity existed, when actually was not the case as none of the authors were similar.

The small number of author list pairs classified in the *All match* group (in this case all the heuristics had a similar behavior) was due to a similar problem. We found for example the author lists {*Elías-Zúñiga, A.*; *Millard, B.*} and {*Zuniga, AE*; *Beatty, MF*} which are equivalent lists, but author name variants avoided to determine individual matches above the similarity threshold. Those are situations where it becomes difficult to identify that they refer to the same authors, even for a human expert. The heuristics classified this comparison into the *No match* group, when in fact it should be classified into the *All match* one. In spite of these particular situations, H_4 and H_7 were the heuristics which best classified the authors-lists into the groups *No matches* and *Some matches*.

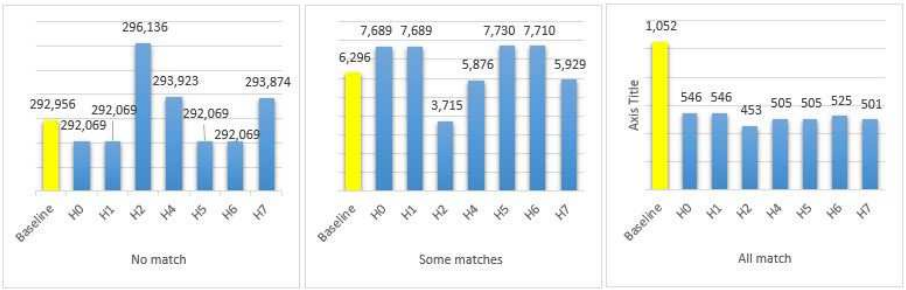

Fig. 2. Author Lists comparisons and average similarity percentage of each block.

4.4 Author Lists Similarity

Next we evaluated if the similarity between author lists is better approximated by aggregating the individual author similarities chosen by each heuristic. Figure 3 shows the amount of author lists having an aggregated similarity above 0.82. As expected H_1 found out the biggest quantity of similar record pairs, but at cost of performing all versus all author name comparisons. The low amount of similarity found by H_0 is due to the

aggregation strategy used, since it negatively impacts on the general average, allowing just a few authors-lists having an average similarity above the threshold.

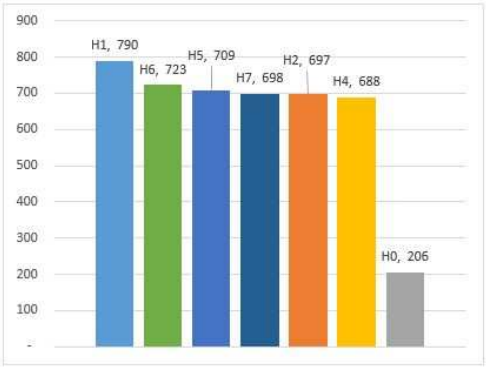

Fig. 3. Author Lists with aggregated similarity above threshold.

Figure 4 shows both the amount of author name comparisons performed and the average aggregated similarity in each group for each heuristic. The order in which each heuristic is showed, intents to point out the reduction of the comparisons achieved by H_4 and H_7. As expected, H_0 and H_1 performed the biggest quantity of comparisons due to their *Full-Matrix* strategy. Despite H_1executes all the comparisons, like H_0, it is important to highlight the different aggregation process they are following. The difference could be noticed, particularly, in the increment of the average for the *Some matches* group between H_0 and H_1. As *Max-Average* is just considering from all the comparisons performed, the ones that maximizes the average (it means it is considering and giving importance to author-names linkage), the average similarity grows up. The same occurs for the group *All match*. On the contrary this rise is not so evident in the average similarity of the group *No match* because there is not any author-name comparison value above the threshold (0.82) which impacts positively in the final result.

This reduction in the amount of comparisons achieved by H_4 and H_7 could be noticed also in the reduction of the average similarity in the groups *some matches* and *no matches*. Since these heuristics, with the objective of increasing the performance, do not perform all the comparisons, it makes possible that even when a common author exists, it is not noticed as the comparison never takes place because of their logic. In this way, that similarity value (which is above the established threshold, 0.82) is prevented of impacting positively in the final average. This is more evident if heuristics H_4 and H_7 are compared with H_5 and H_6 respectively, as their only difference is that H_4 and H_7 make use of the *Partial-Average-Threshold* strategy, performing fewer comparisons as it evaluates if the Max-Average is above the threshold, once the 80% of the author-names have been compared. If it is not above the threshold, the comparison process finishes, as it is not expected that the remaining 20% of the authors impact on the final result, which is to determine if the authors-lists are similar or not.

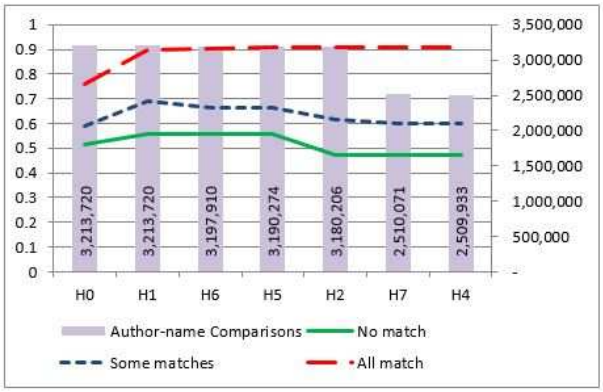

Fig. 4. Author Name comparisons with similarity average per group.

5 Discussion

With the aim of comparing the results obtained by the heuristics, different statistical tests, available in the SPSS tool, were performed (ANOVA, Turkey HSD). The classification groups (*No match*, *Some matches*, *All matches*) as well as their respective similarity averages were used. We choose to evaluate the heuristics H_0, H_1, H_4 and H_7, because the first one makes use of the traditional approach, H_1 is an improvement of H_0, and H_4, H_7 obtained promising results during the experiments. Through those analyses could be observed, in the first place, that the highest difference between the medians was obtained by H_4 and H_7 in the *No match* and *All matches* groups (see Fig. 5). This difference is significantly higher if we compare the heuristics H_4 and H_7 with H_0. This implies that the proposed aggregation approach throws better results than a simple average. The *Max-Average* aggregation is causing that the H_1 mean similarity be higher than the H_0 one, in all the groups (because it choose the higher similarity every time). Nevertheless the heuristics H_4 and H_7 correct this bias, separating the *No match* and *Some match* groups from the *All match* one. In Figure 5 can be appreciated this separations with more detail.

The reason of why H_4 and H_7 find a small amount of similar authors-lists pairs (Figure 3) is because of the previously mentioned difficult-to-resolve cases, just using the author name and surname attributes. Another justification could be that they perform less comparisons than the rest of the heuristics. It is important to mention that, Anyhow, the amount of similarity found by each heuristic does not differ much, except for H_0 and H_1, which perform much more comparisons.

Continuing with the authors-list comparisons analysis, in those cases where author lists are equivalent (all match), heuristics H_4 and H_7 reduce the number of comparisons due to the alignment and comparison process, managing to find, mostly, those similar authors. This is shown in Figure 4, in the (almost) constant line of the *All match* group. The fact of performing less comparisons also affect if those lists are quite similar (*Some match* group) or non similar at all (*No match* group). As the process could be stopped if at least 80% of the authors do not match, then the heuristics which perform more

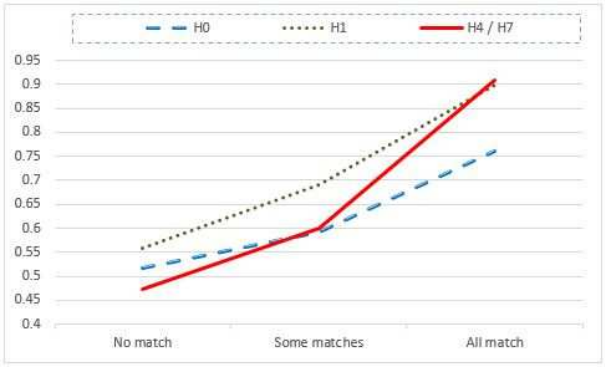

Fig. 5. Comparison of the rice of the similarity average between different groups.

comparisons could increase the average similarity with the remaining 20%. But the final objective is not to link authors but to determine if two authors-lists are similar or not. Then, the heuristics H_4 and H_7, besides improving the difference between medians, reduce the number of comparisons.

Other problem which affects the heuristics H_0 and H_1 is when they process authors-lists which are very similar, but are not the same. In those situations, these heuristics would find higher amount of similarities than the ones that really exist. For example, consider the following lists: *Khan, M.A.*;*Maroof, S.A.*; *Khan, M.Y.* and *Uzair, M.*;*Khan, M.A.*; *Khan, M.Y.*. In this context H_0 and H_1 would found 4 similarities, when the maximum number of authors is 3. It is not the case for the heuristics which force the 1-1 matching, as H_4 and H_7. The difference with the approaches presented by [2] and [18], is that H_4 and H_7 do not need to perform all the comparisons to find the best alignment.

The FingerPrint approach has not made big contributions in these experiments because the lists were already ordered (in most cases). It would be of great help if we would have been comparing keywords or if those authors-lists were not ordered at all, as it happens with the Dublin Core [4] or SWRC [15] ontologies, where the authorship relation is not following any order.

6 Conclusions

In this paper we presented the first results of an approach to align, compare and aggregate the similarity of multi-valued attributes. It was compared against the traditional approaches, using different heuristics, and the results obtained were promising. It reduced in 22% the amount of comparisons performed, making the process more efficient. It also increased the classification quality, showing a considerable difference between the No match and All matches medians.

Despite these results seems promising we still need to improve the validation of its efficacy. We will divide the *Some matches* group in two smaller blocks: those that have a similarity value above 50% (for example) and those that are below that threshold. This would help to improve the quality assessment of the proposed heuristics.

Regarding those hard-to-determine similarities, we would add new different attributes (affiliation or common coauthors) to help the algorithms to determine if two authors represent the same entity or not. This would help the heuristics to increase the amount of similarities of the All matches group. Adding this approach to entity resolution frameworks like Silk [17] or OYSTER [16] would help us to verify if it is useful in the entity resolution process and if it improves the linkage between similar entities.

References

1. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intelligent Systems 18(5), 16–23 (2003)
2. Burkard, R., Dell'Amico, M., Martello, S.: Assignment Problems. Siam, Philadelphia (2009)
3. Cohen, W.W., Fienberg, S.E.: A Comparison of String Distance Metrics for Name-Matching Tasks. In: Proceedings of the ACM Workshop on Data Cleaning, Record Linkage and Object Identification (2003)
4. DCMI: Dublin Core Ontology (2012), http://dublincore.org/documents/dces/
5. Dorneles, C.F., Gonçalves, R., Santos Mello, R.: Approximate data instance matching: a survey. Knowledge and Information Systems 27(1), 1–21 (2010)
6. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. Knowledge and Data Engineering, IEEE Transactions on 19(1), 1–16 (2007)
7. Google: Google Refine Project (2012), http://code.google.com/p/google-refine/
8. Grannis, S.J., Overhage, J.M., McDonald, C.: Real world performance of approximate string comparators for use in patient matching. Studies in health technology and informatics 107(Pt 1), 43–7 (2004)
9. Guha, S., Koudas, N., Marathe, A., Srivastava, D.: Merging the Results of Approximate Match Operations. In: Proceedings of The Thirtieth international conference on Very large data bases. pp. 636–647 (2004)
10. Köpcke, H., Rahm, E.: Frameworks for entity matching: A comparison. Data & Knowledge Engineering 69(2), 197–210 (2010)
11. Köpcke, H., Thor, A., Rahm, E.: Comparative evaluation of entity resolution approaches with FEVER. In: Proceedings of 35th Intl. Conference on Very Large Databases (VLDB) (2009)
12. Morris, T., Huynh, D.: FingerPrint Method (2010), https://github.com/OpenRefine/OpenRefine/wiki/Clustering-In-Depth
13. Porter, E.H., Winkler, W.E.: Approximate String Comparison and its Effect on an Advanced Record Linkage System. Tech. rep. (1997)
14. Ravikumar, P., Cohen, W.W., Fienberg, S.E.: A secure protocol for computing string distance metrics. In: Proceedings of the Workshop on Privacy and Security Aspects of Data Mining at the Int. Conf. on Data Mining. pp. 40–46 (2004)
15. Sure, Y., Bloehdorn, S., Haase, P., Hartmann, J., Oberle, D.: The swrc ontology - semantic web for research communities. In: Proceedings of the 12th Portuguese Conference on Artificial Intelligence - Progress in Artificial Intelligence (EPIA 2005). Springer (2005)
16. Talburt, J.R.: Entity resolution and information quality. Elsevier (2011)
17. Volz, J., Bizer, C., Gaedke, M., Kobilarov, G.: Silk a link discovery framework for the web of data. In: Proceedings of the 2nd Workshop on Linked Data on the Web (2009)
18. Winkler, W.E.: Advanced Methods For Record Linkage. Section on Survey Research Methods (American Statistical Association) (1994)
19. Winkler, W.E.: Overview of record linkage and current research directions. In: Proceedings of Bureau of the Census. Citeseer (2006)
20. Yancey, W.E.: Evaluating string comparator performance for record linkage. Statistical Research Division Research Report (2005)