Generalized Mongue-Elkan Method for Approximate Text String Comparison

Sergio Jimenez[1], Claudia Becerra[1], Alexander Gelbukh[2], and Fabio Gonzalez[1]

[1] Intelligent Systems Laboratory (LISI)
Systems and Industrial Engineering Department
National University of Colombia
[2] Natural Language Laboratory
Center for Computing Research (CIC)
National Polytechnic Institute (IPN), Mexico

Abstract. The Mongue-Elkan method is a general text string comparison method based on an internal character-based similarity measure (e.g. edit distance) combined with a token level (*i.e.* word level) similarity measure. We propose a generalization of this method based on the notion of the generalized arithmetic mean instead of the simple average used in the expression to calculate the Monge-Elkan method. The experiments carried out with 12 well-known name-matching data sets show that the proposed approach outperforms the original Monge-Elkan method when character-based measures are used to compare tokens.

1 Introduction

Approximate string similarity measures are used in many natural language processing (NLP) tasks such as term identification in information extraction, information retrieval, word sense disambiguation, etc. Probably the most well known character-level measure is the *edit distance* proposed by Levenshtein in 1965 [11]. The character-based measures consider the strings to be compared merely as character sequences, which makes this approach affordable when the strings to be compared are single words having misspellings, typographical errors, OCR errors, or even some morphological variations. However, in most human languages, character sequences are split into words (tokens). This property of natural language texts is exploited by token-based measures such as the resemblance coefficients [1] (e.g., *Jaccard*, *Dice*, overlap). The token-based measures compare text strings as sequences of tokens instead of sequences of characters. Such an approach is successful when it is used to compare text strings with many tokens and with different order of the tokens or missing tokens.

All previously mentioned measures based on character or tokens are static. Static string-similarity metrics as they were defined by Bilenko *et al.* [3] are those that compare two character or token sequences in an algorithmic way using solely the information contained into the sequences. Most of the character-based measures are static, that is, the characters into two strings are compared among them with a strategy in order to return a final similarity value. Some

A. Gelbukh (Ed.): CICLing 2009, LNCS 5449, pp. 559–570, 2009.

adaptive approaches [19,4] use labeled corpus as additional information to affine the parameters of static metrics such as edit distance.

The static token-based measures compare the words between and into the strings in order to return compute similarity. However, there are a wide set of approaches that uses additional information about words in order to compare the sequences. For instance, the use of corpus statistics such as TF-IDF, or combinations with static character-based metrics as it was proposed by Cohen *et al.* [7]. Although, the similarity between tokens is mainly measured using other word-space methods [2] and information sources such as WordNet [17], the static methods have importance in pattern matching, text searching, record linkage, information integration, dictionary and name matching among others particular tasks.

Additionally to the resemblance coefficients, there are others static token-based measures that compare tokens using an internal static character-based measure [5,15,12,13,16]. Monge and Elkan [15] proposed one of those hybrid measures that has been used in many name-matching and record linkage comparative studies [4,7,6,18] obtaining a competitive performance versus other non-static measures. The Monge-Elkan method preserves the properties of the internal character-based measure (*e.g.* ability to deal with misspellings, typos, OCR errors) and deals successfully with missing or disordered tokens. In fact, the Monge-Elkan method is a general and recursive token similarity method that can combine any token comparison measure, which captures semantics, translations, etc.

In this paper, we propose a generalization to the Monge-Elkan method, based on the notion of the generalized arithmetic mean, in order to control the balance between higher and lower values of the similarity between the pairs of tokens being compared. The basic Monge-Elkan method uses a simple arithmetic average to combine internal similarities, the proposed generalization provides the ability to apply heuristics that promotes more similar token pairs. Our experiments with twelve name-matching data sets show that this heuristic leads to improvement of the results.

This paper is organized as follows. In Section 2, the original Monge-Elkan method is presented. Section 3 introduces the proposed generalization. Experimental evaluation is discussed in Section 4, and concluding remarks are given in Section 5.

2 The Monge-Elkan Method

Monge and Elkan [15] proposed a simple but effective method to measure the similarity between two text strings that contains several tokens, using an internal similarity function $sim'(a, b)$ able to measure the similarity between two individual tokens a and b. Given two texts A, B, with $|A|$ and $|B|$ being their respective number of tokens, and an external inter-token similarity measure sim', the Monge-Elkan measure is computed as follows:

$$sim_{MongeElkan}(A,B) = \frac{1}{|A|}\sum_{i=1}^{|A|}\max\left\{sim'(a_i,b_j)\right\}_{j=1}^{|B|} \quad (1)$$

Informally, this measure is the average of the similarity values between the more similar token pairs in both strings. In fact, the Monge-Elkan measure approximates the solution to the *optimal assignment problem* in combinatorial optimization [10] with time complexity of $O(|A| \times |B|)$. This approximation is a reasonable tradeoff between accuracy and complexity, because known exact solutions to this combinatorial problem have the time complexity of $O(min(|A|,|B|)^3)$ and require much more sophisticated algorithms [10].

Consider the following example using as internal measure sim' a normalized edit distance converted to a similarity measure (subscripts denote tokens within the strings):

$A =$"*Lenovo inc.*"; $a_1 =$"*Lenovo*"; $a_2 =$"*inc.*"
$B =$"*Lenovo corp.*"; $b_1 =$"*Lenovo*"; $b_2 =$"*corp.*"
$sim'(a_1,b_1) = 1 - \frac{0}{6} = 1$; $sim'(a_1,b_2) = 1 - \frac{5}{6} = 0.1666$
$sim'(a_2,b_1) = 1 - \frac{5}{6} = 0.1666$; $sim'(a_2,b_2) = 1 - \frac{4}{4} = 0$
$sim_{MongeElkan}(A,B) = \frac{1}{2}(\max(sim'(a_1,b_1), sim'(a_1,b_2)) +$
$\max(sim'(a_2,b_1), sim'(a_2,b_2)))$
$sim_{MongeElkan}(A,B) = \frac{1}{2}(1 + 0.1666) = 0.5833$

The Monge-Elkan measure it is not symmetrical, consider the example shown in Figure 1-1. The method iterates the tokens in string A looking for the most similar token in string B in order to make pairs (linked with arrows in Figure 1), then their similarities are averaged. Clearly, the averaged similarity values are different when the content of the strings A and B are swapped. This asymmetry is also evident when the number of tokens is different in both strings (see Figure 1-2). However, this behavior captures the redundancy of the string *"aaaa xaaa yaaa"* at the first case in Figure 1-2. Additionally, at the second case in the same figure, the tokens *"xaaa"* and *"yaaa"* are ignored because of the high similarity between the two tokens *"aaaa"*. Both cases unintentionally implements heuristics convenient for matching. Monge noticed [14] that symmetry may be

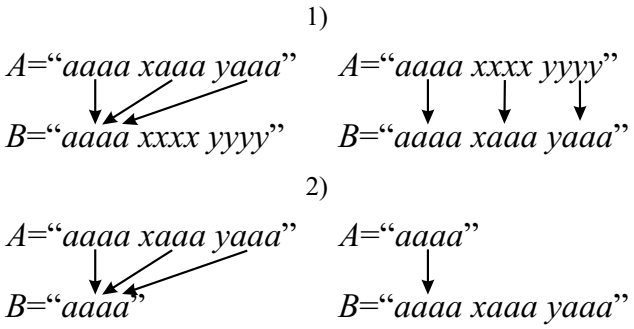

Fig. 1. Asymmetry examples of the Monge-Elkan measure

a natural requirement for many matching applications but not for all. For instance, the name "Alvaro E. Monge" matched "A. E. Monge" while the reverse is not necessarily true.

In the following subsections, for the sake of completeness some character-based string measures that we used as the internal measure sim' in our experimental evaluation are briefly described.

2.1 Bigrams

Q-grams [20] are substrings of length q of a longer string. Depending on q, q-grams are called unigrams, bigrams (or 2-grams), trigrams, and so on. For instance, all bigrams of the string *laptop* are *la*, *ap*, *pt*, *to*, *op*. One way to compute the similarity between two strings is the Dice coefficient

$$sim'(a,b) = 2\frac{|B(a) \cap B(b)|}{|B(a)| + |B(b)|}$$

where $B(x)$ is the set of bigrams of a string x, i.e., the measure is obtained by dividing the number of bigrams common to the two strings by the average number of bigrams in each string. There are several variants of the measures in the q-grams family such as skip-grams, positional q-grams, etc.; see [6] for a comparative study. We used bigrams with one padding character at the beginning and ending of the string because empirical results [9] have shown that padding increase the matching performance.

2.2 Edit Distance

The *edit distance* was originally proposed by Levenshtein [11]. It is equal to the minimum number of editing operations required to transform one sequence into the other. The three basic editing operations are insertion, deletion, and substitution. Several modifications to the original edit distance have been proposed, varying cost schemes and adding more edit operations such as transpositions, opening, and extending gaps; see [8] for a recent survey. The solution [21] for computing the edit distance is a dynamic programming algorithm that stores in a matrix the counts of edit operations for all possible prefixes of both strings. This algorithm computes the edit distance between two strings a and b of length $|a|$ and $|b|$ with a time complexity of $O(|a| \times |b|)$ and space complexity of $O(\min(|a||b|))$.

The edit distance measure can be normalized in the range [0,1] dividing the total number of operations by the number of characters in the longer string. Once normalized, the edit distance can be converted to similarity subtracting the distance value to the number 1.

2.3 Jaro Similarity

The Jaro similarity measure [22] between two strings of length $|a|$ and $|b|$ characters, has the space and time complexity of only $O(|a| + |b|)$. It considers the

number c of characters in common in the two strings and the number of transpositions t as follows:

$$sim_{Jaro}(a,b) = \frac{1}{3}\left(\frac{c}{|a|} + \frac{c}{|b|} + \frac{c-t}{c}\right).$$

If $c = 0$ then $sim_{Jaro} = 0$ by definition. The common characters are considered only in a sliding window of size $\max(|a|,|b|)/2$. In addition, the common characters cannot be shared and are assigned with a greedy strategy. In order to compute the transpositions t between the two strings, the common characters are ordered according to their occurrences in the strings being compared. The number of non-coincident characters at the same positions in both strings is the number of transpositions t. The values returned by the Jaro similarity are in the range $[0,1]$.

3 Proposed Method

Since the Monge-Elkan method uses the arithmetic mean to average all the similarities measured between the selected token pairs, those similarities have the same weight in the final measure. However, it is possible to think that higher values in the internal sim' values may be more informative as lower ones. We believe that promoting the members with higher similarities in the average calculated according to the equation 1 leads to a more natural similarity measure, more suitable for at least some applications. One way of implementation of such promoting can be the use of a generalized mean instead of the arithmetic mean:

$$\bar{x}(m) = \left(\frac{1}{n} \cdot \sum x_i^m\right)^{\frac{1}{m}} \tag{2}$$

Different values of m result in different known "means": $m = 1$ gives the arithmetic mean; $m = 2$, quadratic mean; $m \to 0$, geometric mean; $m \to -1$, harmonic mean, $m \to \infty$, maximum, and $m \to -\infty$, minimum. The arithmetic mean assigns equal weights to all values of x_i in the mean; values of $m > 1$ promote greater values in the mean. The higher the value of m, the stronger the promotion of higher values of x_i in the mean. The generalized-mean concept is closely related to the Minkowski distance in Euclidean spaces and the values of $m = 1$ and 2 are known as City-block and Euclidean distances respectively.

The proposed generalized Monge-Elkan measure is then expressed as follows:

$$sim_{MongeElkan_m}(a,b) = \left(\frac{1}{|a|}\sum_{i=1}^{|a|}\left(\max\left\{sim'(a_i,b_j)\right\}_{j=1}^{|b|}\right)^m\right)^{\frac{1}{m}} \tag{3}$$

Other approaches implement similar heuristics such as soft-TF-IDF [7,16] and the combination of the Dice coefficient with the Jaro-Winkler measure proposed by Michelson and Knoblock [12]. Those approaches uses also an internal measure

sim' in order to determine the "soft" intersection $A \cap B$ between two sets of tokens A and B, selecting pairs of tokens $[a_i, b_i]$ having $sim'(a_i, b_i) > \theta$. The proposed generalization to the Monge-Elkan method avoids the need to establish an *a priori* threshold for the internal measure sim', implementing the promoting heuristic without assumptions about the range of values of sim'.

Consider again the proposed example in Section 2. Whereas the final measure of 0.5833 obtained with the original Monge-Elkan method seems to be very low taking into account that $sim'(a_1, b_1)$ is equal to 1. Consider using $m = 2$ (*i.e.* quadratic mean or Euclidean distance) in the same pair of strings:

$$sim_{MongeElkan_2}(A, B) = \left(\tfrac{1}{2}\left(1^2 + 0.1666^2\right)\right)^{\frac{1}{2}} = 0.7168$$

This quadratic mean gives greater importance to the number 1 than to 0.1666. The intuition behind the proposed approach is that values of m greater than 1 can improve the performance of the matching measure giving greater importance to those pairs of tokens $[a_i, b_i]$ that are more similar.

4 Experimental Evaluation

The aim of the experiments is to determine which values of m improve the performance of the basic Monge-Elkan measure in a specific string-matching task. For this, we take several name-matching data sets, establish a performance metric for the matching task, select character-based measures, experiment with different m, and evaluate the statistical evidence to assess the possible improvement in the matching performance.

4.1 Experimental Setup

The name-matching task consists of comparing two strings that contain names, addresses, telephone numbers, etc., in order to decide whether the two strings refer to the same entity. A data set used for testing the name matching techniques is usually represented as two sets of strings and a subset of their Cartesian product that defines valid matches. Table 1 describes twelve data sets we used. For each set, we give the total number of different strings in it, the size of the two sets of strings (set_1 and set_2), the size of the Cartesian product of the two sets, the number of pairs that are valid matches, and the total number of tokens.[1] The *Animal* data set was not originally separated into two relations, so the relations were obtained using the 327 valid matches and the 689 strings involved in those matches.

The strings in the data sets were not explicitly separated into tokens, therefore a tokenizing process was performed. We considered as separators of tokens (words) the following characters: blank space, parentheses, equality sign, dash,

[1] The data sets are from http://www.cs.cmu.edu/~wcohen/match.tar.gz, except for the two last ones, which are from http://www.isi.edu/~michelso/data/bft_data.zip and https://sourceforge.net/projects/febrl, respectively.

Table 1. Data sets used for our experiments.

Name	# records	$\|set_1\|$	$\|set_2\|$	$\|set_1\| \times \|set_2\|$	# matches	# tokens
Birds-Scott1	38	15	23	345	15	122
Birds-Scott2	719	155	564	87,420	155	3,215
Birds-Kunkel	336	19	315	5,985	19	1,302
Birds-Nybird	985	67	918	61,506	54	1,957
Business	2,139	976	1,163	1,135,088	296	6,275
Game-Demos	911	113	768	86,784	49	3,263
Parks	654	258	396	102,168	250	2,119
Restaurants	863	331	532	176,062	112	9,125
UCD-people	90	45	45	2,025	45	275
Animals	1,378	689	689	474,721	327	3,436
Hotels	1,257	1,125	132	148,500	1,028	9,094
Census	841	449	392	176,008	327	4,083

slash, coma, colon, and semicolon, as well as their sequences; leading and trailing blank spaces were removed. The number of tokens obtained with this tokenizing procedure in each data set is reported in Table 1. Finally, all letters were converted to uppercase, and French diacritics were removed .

We carried out 21 experiments for each data set, combining the three measures explained in Section 2 (bigrams, edit distance, and Jaro similarity) with seven fixed values for he exponent m in equation 3: m =0.00001, 0.5, 1 (*i.e.* the standard Monge-Elkan measure), 1.5, 2, 5, and 10.

4.2 Performance Metrics

The problem of matching between two sets of strings can be viewed as a classification problem over the Cartesian product of the sets. The training or testing data set (a *gold standard*) provides a subset of the Cartesian product of the two sets of strings judged by human annotators as valid matches (positive*s*); its complement is the set of non-valid matches (negatives). In each experiment, a similarity measure value in the range $[0, 1]$ is computed for each possible string pair in the data set. In order to decide whether a pair is a valid match, it is necessary to establish a threshold θ; the pairs of strings with the similarity value greater than or equal to θ are labeled as positive and the rest as negative. For a specific value of θ and a data set, four result sets are defined:

True positives: String pairs marked as valid matches in the gold standard and labeled by the algorithm as positive.

True negatives: String pairs not marked as valid matches in the gold standard and labeled by the algorithm as negative.

False positives: String pairs not marked in the gold standard as valid matches but labeled by the algorithm as positive.

False negatives: String pairs marked as valid matches in the gold standard but labeled by the algorithm as negative.

We measure the performance of a classification task is in terms of *precision*, *recall*, and *F-measure* (the harmonic mean between precision and recall) given by the following expressions:

$$Precision = \frac{|True\,positives|}{|True\,positives| + |False\,positives|}$$

$$Recall = \frac{|True\,positives|}{|True\,positives| + |False\,negatives|}$$

$$F - measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

For each experiment (a combination of a string matching method and a dataset) the number of true positives, false positives, and false negatives were counted ranging the threshold θ form 0 to 1 with 0.01 increment, thus obtaining 101 values of precision, recall, and F-measure. Figure 2 plots the three measures obtained in a typical experiment, showing the trade-off between precision and recall: recall has a decreasing tendency, while precision is generally increasing. The maximum value reached by the F-measure is known as *F1 score*; it is obtained at the threshold θ with the best balance between precision and recall. F1 score can also be seen as a general performance measure of the experiment. F1 score is close to the intersection point of the precision and recall curves, which is another general performance measure for the experiment; however, the F1 score measure is more commonly used. Finally, the F1 score metric reflects the maximum balanced performance that a string measure can reach in a specific matching task.

The same data can be used to plot a precision vs. recall curve, as shown for a typical experiment in Figure 3. From this curve, it is possible to obtain

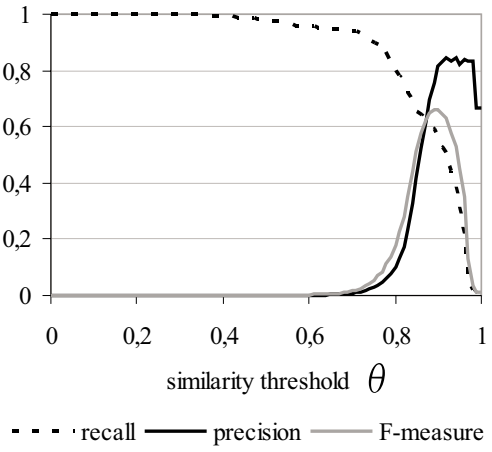

Fig. 2. Precision/recall/F-measure vs. threshold curves for a typical experiment

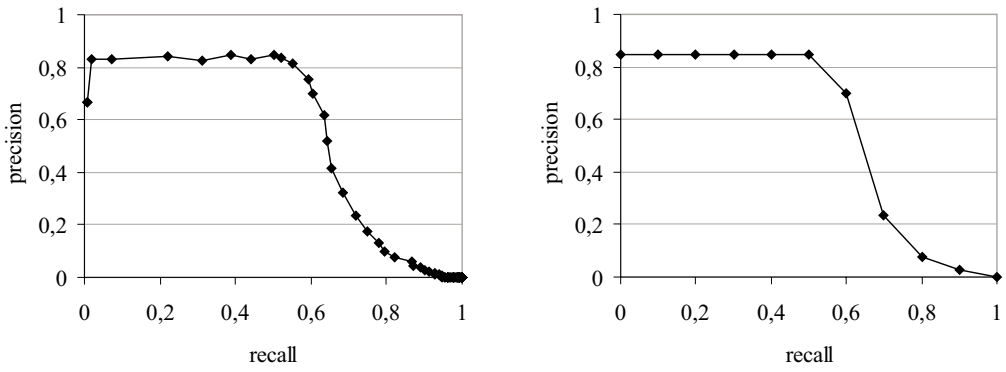

Fig. 3. Example of precision-recall curve and its interpolated version

another general performance measure called *interpolated average precision* (IAP) commonly used in information retrieval [2]. The interpolated precision at recall point r is the maximum precision obtained at points with recall greater than or equal to r. One method for computing IAP is to interpolate the precision vs. recall curve at 11 evenly distributed recall points (*i.e.,* 0, 0.1, 0.2, ...,1); the area under the interpolated precision vs. recall curve is the IAP.

IAP is a good performance measure. Indeed, consider a classifier that obtains precision and recall values of 1 at some θ. The interpolated precision-recall curve becomes a step (*i.e.* 1×1 square) with its elbow at the point $(1, 1)$; the area under that curve is 1. Such a result is the best possible performance, which is achieved by a perfect classifier.

IAP reflects the general performance of the string matching technique considering all possible values for the θ threshold. Although, in practical applications only one value of the θ threshold is used, the performance metrics obtained considering the whole range of values of θ give an assessment of the convenience of the measure for a specific matching problem. In summary, the F1 score metric assesses the maximum possible performance at a fixed threshold and the IAP metric assesses the ability of the classifier to separate the valid matches from non-valid matches regardless the θ threshold.

In order to report a single value for both performance metrics (F1 score and IAP) the results obtained from each dataset are averaged with weights according with the total number of records on each dataset.

4.3 Results

The IAP and F1 score for each internal character-level similarity measure obtained matching each one of the 12 datasets are plotted in Figure 4 and Figure 5, respectively. The natural baseline for the proposed generalization for the Monge-Elkan method is the original method, obtained when $m = 1$ (highlighted in the plots with a dotted vertical line). The results show clearly that values of m below 1 obtain lower performance than the baseline in both metrics. All curves reach

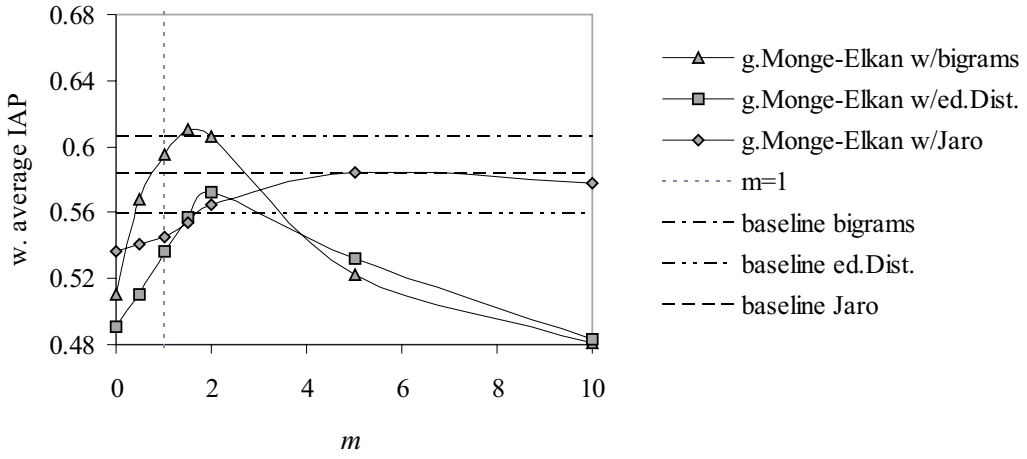

Fig. 4. Weighted Average IAP behavior of the exponent m in extended Monge-Elkan method for each internal character-level string similarity measure

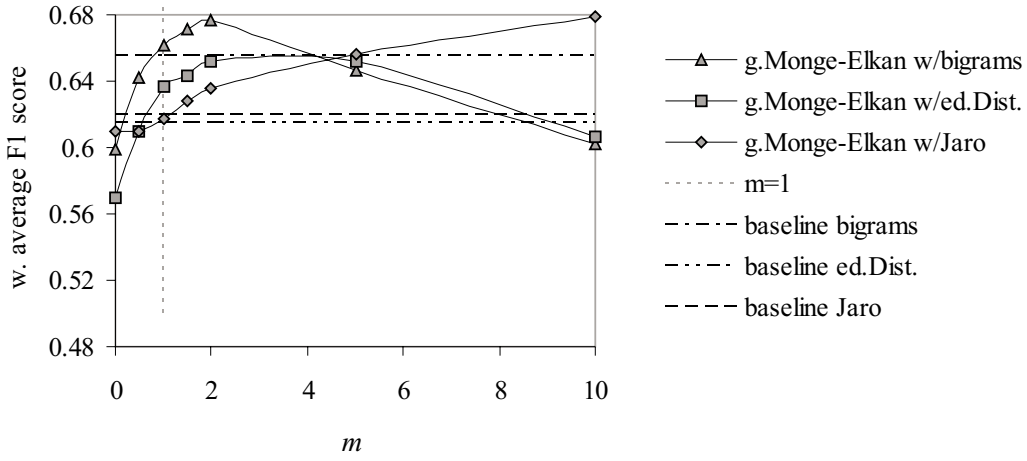

Fig. 5. Weighted average F1 score by the exponent m in generalized Monge-Elkan method for different internal character-level string similarity measures

their maximum performance at values of m above 1, showing that the original method can be improved with the proposed generalization.

Additional baselines are provided in Figures 4 and 5 using directly (without the Monge-Elkan method) the string measures edit distance, bigrams and Jaro similarity for comparing the whole records disregarding any token splitting. Those baselines allow comparing the used character-based measures against the proposed method using the same measure as internal sim' measure. The results obtained with the IAP metric show that whereas the performance of the original Monge-Elkan method is below those baselines, their generalized versions reach the three baselines. Regarding F1 score, those baselines are clearly outperformed.

5 Conclusions

We proposed a generalization of the Monge-Elkan method integrating the arithmetic generalized mean concept to the original expression. That generalization implements a heuristic of giving greater importance in the combined measure to the pairs of tokens whose similarity is higher in comparison with the similarity of other pairs. The proposed method was tested on 12 name-matching data sets with three representative character-based string measures: bigrams, edit distance, and the Jaro similarity. The results showed that the performance of the original Monge-Elkan method can be improved for values of the generalized mean exponent (m) above 1. Particularly, the best results were obtained with $m = 2$ (*i.e.* Euclidean distance) for the measures bigrams and edit distance, and $m = 5$ for the Jaro similarity when they are used as internal similarity measure in the proposed method.

Additionally, the proposed method reached (and clearly outperformed in some cases) the matching performance of the three character-based measures used independently. The original Monge-Elkan method fails to reach that baseline but provides robustness against disordered and missing tokens. The proposed generalization keeps that robustness without compromising the matching performance.

In future work, we plan to incorporate another non-static internal similarity measures to the proposed method and evaluate its application to other tasks such as word sense disambiguation, information retrieval and textual entailment.

References

1. De Baets, B., De Meyer, H.: Transitivity-preserving fuzzification schemes for cardinality-based similarity measures. European Journal of Operational Research 160, 726–740 (2005)
2. Baeza-Yates, R., Ribero-Neto, B.: Modern Information Retrieval. Addison Wesley / ACM Press (1999)
3. Bilenko, M., Mooney, R., Cohen, W., Ravikumar, P., Fienberg, S.: Adaptive name matching in information integration. IEEE Intelligent Systems 18 (5), 16–23 (2003)
4. Bilenko, M., Mooney, R.J.: Adaptive duplicate detection using learnable string similarity measures. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2003)
5. Chaudhuri, S., Ganjam, K., Ganti, V., Motwani, R.: Robust and efficient fuzzy match for online data cleaning. In: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data (2003)
6. Christen, P.: A comparison of personal name matching: Techniques and practical issues. Technical report, The Australian National University, Department of Computer Science, Faculty of Engineering and Information Technology (2006)
7. Cohen, W.W., Ravikumar, P., Fienberg, S.E.: A comparison of string distance metrics for name-matching tasks. In: Proceedings of the IJCAI 2003 Workshop on Information Integration on the Web (2003)
8. Elmagarmid, A.K., Ipeirotis, P.G., Verykios, V.S.: Duplicate record detection: A survey. IEEE Transactions on Knowledge and Data Engineering 19(1), 1–16 (2007)

9. Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E., Järvelin, K.: Non-adjacent digrams improve matching of cross-lingual spelling variants. In: Nascimento, M.A., de Moura, E.S., Oliveira, A.L. (eds.) SPIRE 2003. LNCS, vol. 2857, pp. 252–265. Springer, Heidelberg (2003)
10. Kuhn, H.W.: The hungarian method for the assignment problem. Naval Research Logistics Quarterly 2 2, 83–97 (1955)
11. Levenshtein, V.: Bynary codes capable of correcting deletions, insertions and reversals. Doklady Akademii Nauk SSSR 163(4), 845–848 (1965)
12. Michelson, M., Knoblock, C.A.: Unsupervised information extraction from unstructured, ungrammatical data sources on the world wide web. International Journal on Document Analysis and Recognition 10(3), 211–226 (2007)
13. Minton, S.N., Nanjo, C., Knoblock, C.A., Michalowski, M., Michelson, M.: A heterogeneous field matching method for record linkage. In: Proceedings of the Fifth IEEE International Conference on Data Mining (2005)
14. Monge, A.: An adaptive and efficient algorithm for detecting approximately duplicate database records. International Journal on Information Systems Special Issue on Data Extraction, Cleaning, and Reconciliation (2001)
15. Monge, A., Elkan, C.: The field matching problem: Algorithms and applications. In: Proceedings of The Second International Conference on Knowledge Discovery and Data Mining, (KDD) (1996)
16. Moreau, E., Yvon, F., Cappé, O.: Robust similarity measures for named entities matching. In: Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008) (2008)
17. Pedersen, T., Pakhomov, S.V.S., Patwardhan, S., Chute, C.G.: Measures of semantic similarity and relatedness in the biomedical domain. Journal of Biomedical Informatics 40(3), 288–299 (2007)
18. Piskorski, J., Sydow, M.: Usability of string distance metrics for name matching tasks in polish. In: Proceedings of the 3rd Language and Technology Conference, Poznan (2007)
19. Ristad, E.S., Yianilos, P.N.: Learning string edit distance. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(5) (1998)
20. Ullmann, J.R.: A binary n-gram technique for automatic correction of substitution deletion, insertion and reversal errors in words. The Computer Journal 20(2), 141–147 (1977)
21. Wagner, R.A., Fischer, M.J.: The string-to-string correction problem. Journal of the Association for Computing Machinery 21(1), 168–173 (1974)
22. Winkler, W., Thibaudeau, Y.: An application fo the fellegi-sunter model of record linkage to the 1990 us decenial census. Technical report, Bureau of the Census, Washington, D.C. (1991)