To appear in *International Journal of Digital Earth*
Vol. 00, No. 00, Month 20XX, 1–29

Learning to Combine Multiple String Similarity Metrics for Effective Toponym Matching

Rui Santos[a*] , Patricia Murrieta-Flores[b] and Bruno Martins[a]

[a]*Instituto Superior Técnico and INESC-ID, University of Lisbon, Lisbon, Portugal;*
[b]*Digital Humanities Research Center, University of Chester, Chester, United Kingdom*

(*Received 00 Month 20XX; accepted 00 Month 20XX*)

Several tasks related to geographical information retrieval and to the geographical information sciences involve toponym matching, i.e., the problem of matching place names that share a common referent. In this article, we present the results of a wide-ranging evaluation on the performance of different string similarity metrics over the toponym matching task. We also report on experiments involving the usage of supervised machine learning for combining multiple similarity metrics, which has the natural advantage of avoiding the manual tuning of similarity thresholds. Experiments with a very large dataset show that the performance differences for the individual similarity metrics are relatively small, and that carefully tuning the similarity threshold is important for achieving good results. The methods based on supervised learning, particularly when considering ensembles of decision trees, can achieve good results on this task, significantly outperforming the individual similarity metrics.

Keywords: toponym matching; supervised learning; string similarity metrics; duplicate detection; ensemble learning; geographic information retrieval

1. Introduction

Several tasks related to geographical information retrieval and to the geographical information sciences in general involve toponym matching, i.e., the problem of matching place names that share a common referent (e.g., the names *Lisboa* and *Olissipóna* can both be used to refer to Lisbon, the capital city of Portugal). Examples include the conflation of digital gazetteers and point-of-interest datasets (Samal, Seth, and Cueto 2004; Sehgal, Getoor, and Viechnicki 2006; Manguinhas, Martins, and Borbinha 2008; Hastings 2008; Cheng et al. 2010; Smart, Jones, and Twaroch 2010; Zheng et al. 2010; Martins 2011; McKenzie, Janowicz, and Adams 2014; Dalvi et al. 2014; Li et al. 2016; Berman, Mostern, and Southall 2016), where place names and other descriptive meta-data attributes need to be compared in order to detect potentially duplicate records, or address parsing in the context of geocoding and map search services (Christen, Willmore, and Churches 2006; Sengar et al. 2007; Joshi et al. 2008; Zhang et al. 2013; Berkhin et al. 2015), where names provided by the users need to be matched against a reference database. Other interesting examples include the disambiguation of place names referenced over textual con-

[*]Corresponding author. Email: ruipdsantos@tecnico.ulisboa.pt

tents (Anastácio, Martins, and Calado 2009; Grover et al. 2010; Rupp et al. 2013; Gelernter and Zhang 2013; Santos, Anastácio, and Martins 2015; Monteiro, Davis, and Fonseca 2016; Ardanuy and Sporleder 2017), digitized maps (Weinman 2013; Simon et al. 2014), or digital library resources (Smith and Crane 2001; Freire et al. 2011). In all the aforementioned tasks, toponym matching is used to address the ambiguity problems related to the fact that alternative names can be used to refer to the same place, although other techniques are often also explored in order to deal with the fact that exactly the same names can also be used to refer to different places. While toponym matching focuses on detecting alternative names that share a common referent, additional (contextual) information is often also explored to disambiguate different occurrences of the same place name.

A standard approach for toponym matching involves computing a similarity metric between the names that are to be matched, e.g. an edit distance (Levenshtein 1966; Damerau 1964) or an heuristic such as the Jaro-Winkler metric (Winkler 1990), and then taking a decision with basis on a threshold over the similarity value. While relatively effective, these methods require tuning the similarity threshold for optimal performance. Previous research, either focusing on toponym matching (Recchia and Louwerse 2013) or on the related problem of person name matching (Cohen, Ravikumar, and Fienberg 2003; Christen 2006; Moreau, Yvon, and Cappé 2008; Varol and Bayrak 2012), suggests that the performance of different string similarity algorithms is task-dependent, and that there is no single best technique.

In this article, we present the results of a wide-ranging evaluation on the performance of different string similarity metrics over the toponym matching task. Notice that we do not address the contextual disambiguation of toponym occurrences, instead focusing on the detection of variant names that share a common referent.

Specifically, we compared thirteen different algorithms, based on character-level analysis (e.g., the Jaro-Winkler metric (Jaro 1989; Winkler 1990)), vector-space operations (i.e., cosine similarity between character n-grams), hybrid approaches (e.g., the Monge-Elkan 2-level algorithm (Monge and Elkan 1996)), or even methods specifically proposed for toponym matching (e.g., the method from Davis and De Salles (2007)). We used a very large dataset in the main experiments of our comparative evaluation, consisting of millions of toponym pairs that were obtained from lists of alternative place names associated to records in the GeoNames (2017) gazetteer. The obtained results confirm that the differences in terms of performance are relatively small between the different types of string similarity metrics, and that carefully tuning the similarity threshold is important for achieving good results. Moreover, given the easy access to a very large dataset, we also experimented with the usage of supervised machine learning for combining multiple similarity metrics, avoiding the manual tuning of the similarity threshold. Experiments with classifiers for discriminating pairs of matching versus non-matching toponyms showed that ensembles of decision trees can achieve good results on this task, significantly outperforming individual similarity metrics. However, our analysis of the results also revealed that all methods under study, including supervised models combining multiple metrics, still have problems in handling cases that involve complex transliterations and highly dissimilar toponyms, motivating additional research in the area (e.g., in methods capable of matching toponyms involving different alphabets).

The rest of this article is organized as follows: Section 2 presents fundamental concepts and previous research in the area, starting with a presentation of the most popular string similarity metrics, and then describing previous work that specifically focused on the problem of toponym matching. Section 3 summarizes the individual similarity metrics that were considered in our study, and it also describes the supervised machine learning approach. Section 4 presents the experimental evaluation, detailing the general protocol, dataset and evaluation metrics, and also presenting and discussing the obtained results. Finally, Section 5 concludes the article with a summary of the most interesting findings and with possible paths for future work.

2. Concepts and Related Work

This section starts by presenting a review on string similarity metrics, previously proposed for name matching in general, afterwards discussing previous studies that specifically focused on the problem of toponym matching.

2.1. *Classic String Similarity Metrics*

The literature on string comparison metrics is abundant – e.g., see Cohen, Ravikumar, and Fienberg (2003) for a comprehensive review. Traditional methods for calculating string similarity can roughly be separated into three classes, namely character-based, vector-space based, and hybrid approaches. Character-based methods rely on character edition operations, such as deletions, insertions, substitutions and sub-sequence comparisons, while vector-space methods transform strings into vector representations, over which similarity computations are then performed. Hybrid approaches combine both these ideas, in order to improve effectiveness when matching names composed of multiple tokens.

For instance, the Levenshtein edit distance metric $d_{a,b}$ is a character-based approach corresponding to the minimum number of insertions, deletions or substitutions needed to transform a string a into another string b (e.g., the edit distance between the toponyms *Lisboa* and *Lisbonne* is three, corresponding to two insertions and one substitution, namely $a \mapsto n$, $\epsilon \mapsto n$ and $\epsilon \mapsto e$). The distance metric $d_{a,b}$ can be computed through a dynamic programming algorithm, and a corresponding normalized similarity measure $s_{a,b}$ can be defined as $s_{a,b} = 1 - \frac{d_{a,b}}{\max(|a|,|b|)}$. Many variants and/or improvements have been proposed (Navarro 2001), including the approach by Damerau (1964) in which one basic edit operation is added, namely the transposition of two characters. The Damerau-Levenshtein metric is defined through the following recursive function, where $1_{(a_i \neq b_j)}$ is the indicator function, equal to zero when $a_i = b_j$, and equal to one otherwise.

$$d_{a,b}(i,j) = \begin{cases} \max(i,j) & \text{if} \min(i,j) = 0, \\ \min \begin{cases} d_{a,b}(i-1,j)+1 \\ d_{a,b}(i,j-1)+1 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \\ d_{a,b}(i-2,j-2)+1 \end{cases} & \text{if } i,j > 1 \text{ and } a_i = b_{j-1} \text{ and } a_{i-1} = b_j \\ \min \begin{cases} d_{a,b}(i-1,j)+1 \\ d_{a,b}(i,j-1)+1 \\ d_{a,b}(i-1,j-1)+1_{(a_i \neq b_j)} \end{cases} & \text{otherwise.} \end{cases}$$

The recursive function $d_{a,b}(i,j)$ represents a distance between an i-symbol prefix (i.e., an initial sub-string) of string a and a j-symbol prefix of string b. The Damerau–Levenshtein distance between two strings a and b is thus given by $d_{a,b}(|a|,|b|)$, where $i = |a|$ denotes the length of string a and $j = |b|$ is the length of b.

More sophisticated extensions of the Levenshtein edit distance procedure have also been previously proposed, including approaches that enhance the model to allow for contiguous sequences of mismatched characters (i.e., affine gaps) in the alignment of two strings (Needleman and Wunsch 1970), or even adaptable approaches that learn appropriate weights for the different edit operations (Ristad and Yianilos 1998; Brill and Moore 2000; Bilenko and Mooney 2003a).

The Jaro metric is instead an heuristic character-based approach based on the number and on the order of common characters, specifically designed for matching strings such as person names (Jaro 1989). The Jaro similarity metric $s_{a,b}$ between two strings a and b is defined as follows, where m is the number of matching characters, and where t is half the number of character transpositions in the strings.

$$s_{a,b} = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3} \times \left(\frac{m}{|a|} + \frac{m}{|b|} + \frac{m-t}{m} \right) & \text{otherwise.} \end{cases}$$

Two characters from strings a and b are considered to match if they are equal and if they are not farther than $\left\lfloor \frac{\max(|a|,|b|)}{2} \right\rfloor - 1$ characters apart. The number of matching characters that are in a different sequence order, divided by two, defines the parameter t that encodes character transpositions.

A refined version of the Jaro metric, latter proposed by Winkler (1990), uses a prefix scale p which gives higher scores to strings that match from the beginning for a preset prefix length ℓ. Given two strings a and b, their Jaro-Winkler similarity is defined as follows, where $s'_{a,b}$ is the Jaro similarity between strings a and b, where ℓ is the length of common prefix at the start of the strings, up to a maximum of four characters, and where $p = 0.1$ is a constant scaling factor, expressing how much the score is adjusted upwards in the case of strings having a common prefix.

$$s_{a,b} = s'_{a,b} + (\ell \times p \times (1 - s'_{a,b})).$$

Christen (2006) noted that the Jaro-Winkler approach can be problematic if the strings to be matched contain multiple words that are differently ordered (e.g., in the case of the toponyms *Madeira Island* and *Island of Madeira*). To address this issue, he introduced two variants named (i) sorted Winkler and (ii) permuted Winkler. The former algorithm sorts the tokens that compose both strings before calculating their Jaro-Winkler similarity, while the later calculates the similarity over all possible token permutations and returns the maximum value.

In terms of vector-space approaches, the cosine similarity between representations based on character n-grams (i.e., based on sequences of n consecutive characters, typically with $n = 2$ and/or with $n = 3$) is a common approach. Let $\mathbf{A} =< a_1, \ldots, a_{|\Sigma|} >$ and $\mathbf{B} =< b_1, \ldots, b_{|\Sigma|} >$ be vector representations for the strings a and b, where each dimension (i.e., each a_i or b_i, with $1 <= i <= |\Sigma|$) corresponds to one of the Σ individual n-grams occurring in either string, with a value of one if the n-gram occurs in the string and zero otherwise (i.e., a binary encoding for n-gram occurrence). The cosine similarity metric is defined as follows:

$$s_{a,b} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \times \|\mathbf{B}\|} = \frac{\sum_{i=1}^{|\Sigma|} a_i b_i}{\sqrt{\sum_{i=1}^{|\Sigma|} a_i^2} \times \sqrt{\sum_{i=1}^{|\Sigma|} b_i^2}}.$$

Different variants of the cosine metric have been used in practice (Cohen, Ravikumar, and Fienberg 2003; Moreau, Yvon, and Cappé 2008), for instance leveraging the common information retrieval term weighting heuristic known as Term Frequency times Inverse Document Frequency (TF-IDF), instead of binary representations.

The Jaccard (1912) similarity coefficient, computed between sets of character n-grams, is also commonly employed. Let $\mathbf{A} = \{a_i\}_{1<=i<=|a|}$ and $\mathbf{B} = \{b_i\}_{1<=i<=|b|}$ correspond to the sets of n-grams that compose strings a and b, with $|a|$ and $|b|$ respectively corresponding to the number of n-grams in each string. The Jaccard similarity coefficient can be defined as follows:

$$s_{a,b} = \frac{|\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A} \cup \mathbf{B}|}.$$

Simple variations of the aforementioned vector-based procedures have also been used. For instance White[1] described in his blog a simple approach that rewards common substrings and a common ordering of those sub-strings, based on computing how many adjacent character pairs (i.e., character bi-grams) are contained in both strings. The author states that his metric has been successfully employed in the retrieval of terms from a domain-specific electronic thesaurus, and also on the retrieval of place names. Considering that $\mathbf{A} = \{a_i\}_{1<=i<=|a|}$ and $\mathbf{B} = \{b_i\}_{1<=i<=|b|}$ correspond to sets of adjacent letters taken from the strings a and b (i.e., the quantities $|a|$ and $|b|$ respectively correspond to the number of bi-grams in each string), the similarity metric is defined as shown in the next equation, which corresponds to the Dice (1945) similarity coefficient between sets of bi-grams:

$$s_{a,b} = \frac{2 \times |\mathbf{A} \cap \mathbf{B}|}{|\mathbf{A}| + |\mathbf{B}|}.$$

Besides regular n-grams, some previous studies have also suggested that skip-grams (i.e., bi-grams of non-adjacent letters, considering gaps of zero, one, or two characters) are an effective choice for representing the strings to be matched (Keskustalo et al. 2003). Consider a set of integers Γ, each representing the number of skipped characters when forming bi-grams from a string (i.e., if $\Gamma = \{0, 1\}$, a string $c_1c_2c_3c_4$ would be represented as a set of skip-grams formed by skipping both zero and one characters in the original string, thus resulting in the set $\{c_1c_2, c_1c_3, c_2c_3, c_2c_4, c_3c_4\}$). Different sets Γ can be considered simultaneously, e.g. by forming a set of skip-grams with gaps of length zero, and then a separate set of skip-grams with gaps of length one and two. Finally, a similarity metric between

[1]`http://www.catalysoft.com/articles/strikeamatch.html`

two strings can be computed on the basis of these sets, by adapting the standard Jaccard similarity coefficient:

$$s_{a,b} = \frac{\sum_{\Gamma \in \{\{0\},\{1,2\}\}} |\text{skipgrams}_\Gamma(a) \cap \text{skipgrams}_\Gamma(b)|}{\sum_{\Gamma \in \{\{0\},\{1,2\}\}} |\text{skipgrams}_\Gamma(a) \cup \text{skipgrams}_\Gamma(b)|}.$$

In the previous equation, the function $\text{skipgrams}_\Gamma(a)$ returns a set of skip-grams from the input string a, considering gap lengths within the set Γ.

Hybrid second-level measures combine the advantages of character-based and vector-spaced approaches, being flexible about word order and position, while still allowing for small differences between word tokens. These methods are based on applying a sub-measure $s'_{a,b}$ to all pairs of word tokens between the two strings, and then computing a final similarity score $s_{a,b}$ based on these values. For instance the scheme proposed by Monge and Elkan (1996) involves computing the average similarity between the most similar pairs of word tokens, according to a sub-measure such as the Jaro-Winkler similarity score:

$$s_{a,b} = \frac{1}{|a|} \sum_{i=1}^{|a|} \max_{j=1}^{|b|} (s'_{a_i,b_j}).$$

In the previous equation, s'_{a_i,b_j} represents the Jaro-Winkler similarity between a token a_i from string a, and a token b_j from string b. The quantities $|a|$ and $|b|$ represent the number of tokens in strings a and b, respectively.

Authors like Moreau, Yvon, and Cappé (2008) have argued that the Monge-Elkan scheme does not perform well in practice, given that its very simple behavior favors short strings, since averaging significantly penalizes every non-matching word. The Monge-Elkan similarity $s_{a,b}$ between two strings a and b is also not symmetric, although this can be addressed by computing the average between $s_{a,b}$ and $s_{b,a}$.

Procedures such as the cosine similarity or the Jaccard coefficient can also be used as hybrid approaches, considering sets of tokens instead of sets of n-grams, and softening the metrics by allowing for small mismatches when aligning the tokens (Cohen, Ravikumar, and Fienberg 2003; Moreau, Yvon, and Cappé 2008). For instance, by considering an inner similarity metric such as the Jaro-Winkler score, a Soft-Jaccard metric between two strings a and b can be computed as follows:

$$s_{a,b} = \frac{Z}{|a| + |b| - Z}, \text{ with}$$

$$Z = \frac{\sum_{i=1}^{|a|} \max_{j=1}^{|b|} (s'_{a_i,b_j}) + \sum_{j=1}^{|b|} \max_{i=1}^{|a|} (s'_{a_i,b_j})}{2}.$$

In the previous equation, similarly to the case of the Monge-Elkan metric, s'_{a_i,a_j} represents an inner similarity between a token a_i from string a and a token b_j from string b, while the quantities $|a|$ and $|b|$ represent the number of tokens in strings a and b, respectively. The quantity Z captures the similarity between tokens from string a matching in string b, and from string b matching in string a. Notice that if the inner similarity metric s'_{a_i,a_j} returns one when the tokens are equal and zero otherwise, then the previous equation for $s_{a,b}$ is equivalent to the standard Jaccard similarity coefficient between sets of tokens. Using a similarity metric other than the identity function (e.g., the Jaro-Winkler score) can soften the results of the Jaccard coefficient, allowing for small mismatches in the tokens.

It is interesting to notice that besides the aforementioned metrics based on character operations, vector-space representations, or hybrids, some previous studies have also proposed to leverage phonetic encoding techniques (Christen 2006; Varol and Bayrak 2012), which attempt to convert a string into a code that captures the way a name is pronounced. However, this conversion process is language-dependent, and most of the designed techniques (e.g., classic approaches like Soundex or more recent methods such as Double Metaphone (Philips 1990, 2000)) have been developed with basis on the Latin alphabet and the English phonetic structure. In our work, given that we were interested in matching toponyms considering several different languages (e.g., matching toponym pairs corresponding to transliterations in different languages), we have not explored the usage of phonetic methods.

2.2. *Previous Work on Toponym Matching*

Taking inspiration on the previous work by Christen (2006), Recchia and Louwerse (2013) reported on an experimental comparison of different string similarity measures over a toponym matching task in which Romanized toponyms from different countries, taken from the GEOnet Names Server, had to be matched against alternative names for the same places. The authors found that, in general, methods that rely on the number of shared short sub-strings (i.e., bi-grams, tri-grams, and skip-grams in particular) tend to perform well, although they also observed substantial variation in the methods that worked best on the datasets from different countries. For instance, the best-performing algorithms on toponyms from China and Japan were Jaro variants, whereas edit distance performed best on data from Taiwan, and n-gram based measures worked best on countries with toponyms in Romance and Germanic languages. As a general recommendation, the authors suggest that practitioners should test several algorithms on a country-specific or language-specific dataset, and use the best-performing algorithm for future developments involving that dataset. Similarly to Recchia and Louwerse (2013), the present study also aims to provide a comprehensive comparison of different similarity metrics, although we have used a much larger dataset, have followed a different experimental protocol (e.g., the test pairs are not limited to toponyms from a same country nor do they involve only Romanized versions of toponyms), and have also tested machine learning methods for combining different similarity metrics.

Hastings and Hill (2002) have noted that standard similarity metrics, such as the ones described in the previous section and explored in the study by Recchia and Louwerse (2013), are not particularly well suited to toponym matching because, in everyday usage, the stylistic variability of place names is simply too great. Authors like Davis and De Salles (2007), Hastings (2008), Cheng et al. (2011) or Kılınç (2016) have proposed algorithms specifically designed for matching toponyms, in most cases corresponding to variations of the procedures introduced in the previous section, and often leveraging some form of canonical representation for toponyms.

For instance Hastings (2008), taking inspiration on a previous proposal by Fu, Jones, and Abdelmoty (2005), proposed an algorithm that relies on token matching, using a language-dependent stemming procedure that attempts to reduce word tokens to canonical base forms. This algorithm essentially matches the occurrence of lower-cased tokens from one toponym, excluding place-type terms and common stop-words, against the lower-cased and possibly stemmed tokens of the other toponym, and vice-versa. Common abbreviations are expanded, and common misspellings are also corrected as part of stemming. For each token pairing between the two strings, a match score of zero, one-quarter, one-half, or one is assigned, giving quarter-weight to infix matches, half-weight to prefix/stemmed matches, and full-weight to exact matches. The accumulated bi-directional score is normalized by the total number of tokens in the two strings, to account for unmatched tokens.

The methods proposed by Davis and De Salles (2007) and by Kılınç (2016) instead correspond to hybrid approaches. For instance, in the method by Davis and De Salles (2007), the first step involves dividing the toponyms into tokens, using blanks, hyphens, and other similar symbols as delimiters. For matching individual tokens, the authors proposed a variation of the Levenshtein edit distance that incorporates a practical scheme for matching accented and special characters (i.e., characters are organized into equivalence groups, so that characters belonging to the same group are considered to match). Two tokens are considered to match if their similarity is above a pre-defined threshold of $\alpha = 0.75$. The complete matching strategy involves four distinct phases, namely (1) replacing tokens that are known abbreviations by their full spelling (e.g., *avn.* would be replaced by *avenue*), (2) capturing non-standard abbreviations in one of the strings, namely single-character capitalized tokens and tokens that end with a dot character, by checking them against tokens in the other string that share a common prefix, (3) token alignment, using a procedure similar to that of the Levenshtein edit distance in order to align the tokens from one string against those from the other, and (4) computing a final similarity score through a linear combination of three different metrics between sets of tokens, one of which accounting for possible token inversions.

It is interesting to notice that the previously described methods for toponym matching are specific to particular languages and/or alphabets, in the sense that they require resources such as stemming algorithms and/or lists of stop-words and abbreviations. From the descriptions that are provided in the corresponding publications, these methods are also particularly challenging to replicate, given the language resources and the large sets of heuristics that are involved. In the present study, we nonetheless report on experiments with a simpler variation of the method described by Davis and De Salles (2007).

Several previous studies have also proposed heuristic combinations of different similarity metrics, computed over attributes such as place names, place types, and/or geospatial footprints, in order to match gazetteer records (Samal, Seth, and Cueto 2004; Fu, Jones, and Abdelmoty 2005; Hastings 2008; Smart, Jones, and Twaroch 2010; Cheng et al. 2010; Morana et al. 2014; Dalvi et al. 2014; Li et al. 2016; Berman, Åhlfeldt, and Wick 2016). For instance Dalvi et al. (2014) presented a technique based on statistical language models that is used in production at Facebook for de-duplicating the Places database, where records contain a name and a physical location. The proposed technique combines string edit operations together with two language models, namely a name model that discriminates between the core and the background parts of a name (e.g., it identifies that in a name such as *Fresca's Peruvian Restaurant*, the token *Fresca's* is the core part and *Peruvian Restaurant* is a background description), together with a spatial context model that, given a location, computes the distribution of contextual terms in that location. Through experiments, Dalvi et al. showed that the name model alone outperforms baselines based on edit distances and cosine similarities (i.e., the obtained results confirm the importance of properly handling names when matching point-of-interest datasets), and the combination with the spatial context model further improves results.

Taking inspiration and advancing on some of these studies, other authors have proposed to use supervised machine learning for gazetteer record conflation, using classifiers as a principled approach to combine multiple similarity metrics, computed over different types of attributes (Sehgal, Getoor, and Viechnicki 2006; Zheng et al. 2010; Martins 2011; McKenzie, Janowicz, and Adams 2014). It is interesting to notice that previous studies leveraging supervised machine learning have reported very good results on the task of conflating gazetteer records, often exceeding the value of 90% in terms of precision and recall. Although these values are above the ones reported on the present study – see Section 4 – we have that the experiments reported on the aforementioned studies have relied on very small datasets, often considering only place names in Romance and Germanic languages, or Romanized toponyms. It should also be noted that although these previous studies have combined similarity metrics computed over different types of attributes, the authors have concluded that similarity scores between place names are among the most useful features for gazetteer record conflation, together with geo-spatial distance and overlap between geo-spatial footprints. For instance Martins (2011) found that string similarity was the most informative type of information, and that n-gram overlap metrics, Jaro-Winkler, and variations of edit distance were particularly useful. Sehgal, Getoor, and Viechnicki (2006) found that edit distance outperformed both a Jaccard n-gram coefficient and Jaro-Winkler, when mapping between two sets of Romanized place names from Afghanistan. In a problem of matching points-of-interest described in two different location-based social networks, McKenzie, Janowicz, and Adams (2014) showed that the Levenshtein edit distance between names performed the best of a set of independent methods, that also included geographic distance and matches in categories and in long textual descriptions. Still, these previous studies have not systematically compared the performance of different name similarity metrics, instead focusing on the combination of multiple attributes. The present study indeed resembles recent publications addressing the problem of gazetteer conflation, although we focus on systematically evaluating the matching of place names alone. Since no other attributes besides place names are used, we do not address the problem of disambiguating equal toponyms that may refer to different places, instead focusing on finding alternative place names matching a common referent. On what regards advancements over the current state-of-the-art in toponym matching, we report on a more extensive set of experiments (i) with a very large dataset covering the entire world, (ii) with many different similarity metrics, representative of different classes, and with (iii) machine learning methods from the current state-of-the-art (e.g., ensembles of decision trees (Kotsiantis 2013; Banfield et al. 2007)).

3. Exploring Multiple Similarity Metrics for Toponym Matching

The present study has the objective of providing a comprehensive evaluation on the performance of different string similarity metrics over the task of matching toponyms. With basis on our review of previous related work, the similarity metrics that were considered for our comparative study are as follows:

(1) Normalized similarity computed from the Damerau-Levenstein edit distance;
(2) Jaro similarity;
(3) Jaro-Winkler similarity, with a scaling factor of $p = 0.1$;

(4) Jaro-Winkler similarity, computed from reversed versions of the original strings. The idea of giving higher scores to strings that match in a prefix is adequate in the case of comparing person names but, in the case of toponyms, it is often the case that similar sub-strings occur in suffixes;
(5) Sorted Jaro-Winkler similarity;
(6) Permuted Jaro-Winkler similarity. Given the very high computational costs associated to the computation of this metric, in the case of strings composed of many different word tokens, our experiments used an adapted version where the tokenization process generates a maximum of five different tokens for each string being compared, prior to the computation of the token order permutations (i.e., the last tokens from each string may actually be processed as a single token, in the case of strings with more than five word tokens);
(7) Cosine similarity between character bi-grams and tri-grams;
(8) Jaccard similarity coefficient between character bi-grams and tri-grams;
(9) Dice similarity coefficient between character bi-grams;
(10) Adapted Jaccard similarity between character skip-grams;
(11) Monge-Elkan similarity, leveraging Jaro-Winkler as an internal measure;
(12) Soft-Jaccard similarity, leveraging Jaro-Winkler as an internal measure;
(13) Simplified version of the procedure from Davis and De Salles (2007).

All the similarity metrics from the previous enumeration, except the toponym-specific metric from Item (13), were already formally described in Section 2.1. The metric from Item (13) was also covered in the analysis of related work, but we instead use a simpler variant that does not require language specific resources.

In our simplified version of the method from Davis and De Salles (2007) that is used in Item (13), each string is first segmented into individual tokens, using white-spaces and hyphens as delimiters. The comparison of individual tokens is made through a normalized similarity computed from the Damerau-Levenstein edit distance, after diacritics are removed from the original strings (i.e., removing the diacritics is equivalent to creating character groups with basis on accented characters). We do not use a list of common abbreviations, although we do attempt to match single-character capitalized tokens and tokens that end with a dot character against other tokens with a common prefix (i.e., abbreviated tokens discovered through this procedure are replaced by the corresponding complete spelling). The final similarity score is computed from the transformed strings through an average between the Soft-Jaccard coefficient, using the Damerau-Levenstein similarity as an inner metric, and a Sorted Jaro-Winkler similarity.

The different similarity metrics from the previous enumeration produce real values bounded in the interval between zero and one, with the value of one corresponding to maximal similarity. When comparing the different approaches, the toponym matching decisions were made with basis on a threshold value over the results of each similarity metric (i.e., two toponyms, taken from a test set composed of multiple pairs of toponyms, are said to match when their similarity is above a given fixed threshold), and we experimented with different values for the similarity threshold, in order to tune for performance – see the discussion in Section 4. More formally, the problem of toponym matching can thus be stated as shown in Definition 1.

Definition 1. *Let a and b be strings of characters, i.e. sequences of symbols from a finite alphabet Σ, denoting two different place names. We say that a matches b when $s_{a,b} >= \sigma$, where $s_{a,b} : \Sigma^\star \times \Sigma^\star \rightarrow [0,1]$ quantifies the similarity between a and b, and where σ is a similarity threshold.*

Given that we only considered symmetric similarity measures, the fact that a string a is considered to match a string b implies that b also matches a. Notice also that no additional information besides the strings corresponding to the toponyms is being used. Our work is therefore just addressing the discovery of alternative names that share a common referent, and not the complete task of toponym disambiguation (i.e., the disambiguation of equal toponyms that refer to different locations would require additional sources of contextual information, so as to detect cases in which the same name is being used to refer to different places (Santos, Anastácio, and Martins 2015; Monteiro, Davis, and Fonseca 2016; Ardanuy and Sporleder 2017)).

Besides comparing individual similarity metrics, we also experimented with the usage of supervised machine learning as a way of combining the multiple similarity scores. In this approach, a classification model is first inferred with basis on a training set of toponym pairs, for which we known the correct decision that should be made regarding their matching. Each instance in the set of training pairs is represented as a feature vector, in which each dimension corresponds to one of the thirteen similarity metrics introduced in the previous enumeration. After training, the induced model can be used to make a decision regarding the possible matching of new toponym pairs, again represented as vectors of similarity scores. This general procedure has the advantage of not requiring the manual tuning of the similarity threshold associated to a matching decision. It also offers a principled approach for combining the benefits of different types of similarity metrics. However, besides requiring training data, this procedure is associated to a much higher computational complexity, given that multiple similarity metrics need to be computed in order to generate the feature-based representations. The actual process of inferring the classification model is also computationally demanding but, since this can made only once and in an offline stage, we argue that the most important cost is associated to the computation of the similarity scores that constitute the representations. Past studies have argued that some of the individual similarity metrics, that were listed in the previous enumeration, are particularly expensive in terms of computational requirements. Still, as further discussed in Section 4, we believe that with modern hardware it is fairly easy to process very large datasets with these methods.

When leveraging supervised machine learning, the problem of toponym matching can be formalized as shown in Definition 2.

Definition 2. *Let a and b be strings of characters, i.e. sequences of symbols from a finite alphabet Σ, denoting two different place names. Let also $f_{a,b}$ correspond to a feature extraction procedure, which takes as input a pair of strings a and b and computes a vector $f_{a,b} =< s^1_{a,b}, \ldots, s^k_{a,b} >$ of characteristics, where each of the k dimensions is a similarity score computed between the strings a and b, such that each $s^i_{a,b} : \Sigma^\star \times \Sigma^\star \rightarrow [0, 1]$, with $1 <= i <= k$. We say that string a matches b when a classifier $c_{f_{a,b}} = \text{TRUE}$. In the previous expression, $c_{f_{a,b}} : [0, 1]^k \rightarrow \{\text{FALSE}, \text{TRUE}\}$ is a classifier, inferred through supervised machine learning from a set of training instances $\{(f_{a,b}, c) : a \in \Sigma^\star \wedge b \in \Sigma^\star \wedge c \in \{\text{FALSE}, \text{TRUE}\}\}$.*

In our experiments, we compared the performance of different types of supervised machine learning methods, relying on the implementations provided by the Scikit-learn (2017) and XGBoost (2017) Python packages. In particular, we experimented with models based on the formalism of linear support vector machines, or based on different types of state-of-the-art approaches leveraging ensembles of decision trees (i.e., random forests (Breiman 2001), extremely randomized trees (Geurts, Ernst, and Wehenkel 2006), or gradient boosted decision trees (Friedman 2001; Chen and

Guestrin 2016)).

In brief, we have that linear Support Vector Machines (SVMs) are discriminative classifiers formally defined by a separating hyperplane between two classes. The equation for the hyperplane corresponds to a linear combination of the features that represent the instances to classify. Training an SVM model involves finding the hyperplane that represents the largest separation, or margin, between the two classes, since in general the larger the margin the lower the generalization error of the classifier. Decision tree classifiers, on the other hand, are non-linear procedures based on inferring a flow-chart-like structure, where each internal node denotes a test on an attribute, each branch represents the outcome of a test, and each leaf node holds a class label (Kotsiantis 2013). Decision trees can be learned by splitting the source set of training instances into subsets, based on finding an attribute value test that optimizes the homogeneity of the target variable within the resulting subsets (e.g., by optimizing an information gain metric). This process is repeated on each derived subset in a recursive manner. Recently, ensemble strategies for combining multiple decision trees have become popular in machine learning and data mining competitions, often achieving state-of-the-art results (Banfield et al. 2007).

Random forests operate by independently inferring a multitude of decision trees at training time, afterwards outputting the class that is the mode of the classes returned by the individual trees (Breiman 2001). Each tree in the ensemble is trained over a random sample, taken with replacements, of instances from the training set. The process of inferring the trees also adds variability to each tree in the ensemble by selecting, at each candidate split in the learning process, a random subset of the features. The procedure known as extremely randomized trees adds one further step of randomization (Geurts, Ernst, and Wehenkel 2006). Instead of computing the locally optimal splits with basis on criteria such as the information gain, at each node of the tree and for each feature under consideration, this method instead selects a random value for the split. Finally, the gradient boosting approach operates in a stage-wise fashion, sequentially learning decision tree models that focus on correcting the decisions of predecessor models. The prediction scores of each individual tree are summed up to get the final score. Chen and Guestrin (2016) introduced a scalable tree boosting library called XGBoost, which has been used widely by data scientists to achieve state-of-the-art results on many machine learning challenges.

All the aforementioned methods are relatively standard in the machine learning literature. More information about these procedures can be found on the documentation for the scikit-learn package, or in popular machine learning textbooks (Bishop 2006; James, Witten, and Hastie 2014; Murphy 2013; Daumé 2015).

4. Experimental Evaluation

This section details the experimental methodology and the results that were obtained. Section 4.1 starts by describing the main dataset used in our tests, presenting a statistical characterization. This section also describes the evaluation protocol and the metrics that were considered for assessing result quality. Section 4.2 presents and discusses the obtained results, also introducing several illustrative examples.

Table 1. Statistical characterization of the main dataset used in our experiments.

Characteristic	Value
Number of toponym pairs	5,000,000
Number of matching toponym pairs	2,500,000
Number of pairs with toponyms from the same country	4,999,260
Number of pairs with equal toponyms after lowercasing and removing diacritics	180,453
Number of pairs with matching equal toponyms after lowercasing and removing diacritics	167,933
Number of pairs with matching toponyms that are completely dissimilar	625,754
Number of pairs with non-matching toponyms that are completely dissimilar	993,409
Average difference in number of characters per toponym pair	3.74
Average number of characters per toponym	22.72
Average number of word tokens per toponym	3.59
Number of characters in largest toponym	188
Number of pairs with both toponyms involving only Latin characters	3,320,403
Number of pairs with at least one toponym involving CJK characters	625,007
Number of pairs with at least one toponym involving Cyrillic characters	400,860
Number of pairs with at least one toponym involving Arabic characters	390,145
Number of pairs with at least one toponym involving Thai characters	221,007
Number of pairs with at least one toponym involving Greek characters	25,941
Number of pairs with at least one toponym involving Armenian characters	16,361
Number of pairs with at least one toponym involving Hebrew characters	8,503
Number of pairs with at least one toponym involving Georgian characters	4,979
Number of pairs with at least one toponym involving Devanagari characters	2,087

4.1. *Dataset and Experimental Methodology*

Our experiments have mostly relied on a dataset of five million pairs of toponyms, half of which corresponding to alternative names for a same place. The dataset was generated from lists of alternative place names associated to records in the publicly available GeoNames gazetteer (i.e., each place that is described in GeoNames can be associated to multiple names, often corresponding to historical denominations or to transliterations in multiple alphabets/languages, and thus we can leverage this information to build a large dataset covering toponyms from all around the globe). The matching pairs of toponyms in our dataset correspond to alternative names with more that two characters that, after converting all characters into their lower-cased equivalents, do not match in every character. The non-matching pairs of toponyms correspond to names for different places, not necessarily within the same country, that also have more than two characters. In order to build a dataset that is both representative and challenging for automated classification, a significant portion of the non-matching pairs of toponyms should not be completely dissimilar. According to this intuition, we preferred toponym pairs having a Jaccard similarity above zero (i.e., when building the dataset, if the similarity between a non-matching pair of toponyms was equal to zero, we discarded the pair with a probability of 0.75).

Table 1 presents simple characterization statistics for the dataset used in our experiments. In Figure 1 we present charts with the distribution for the length (i.e., the number of characters or the number of word tokens) of the toponyms present in the dataset, whereas in Figure 2 we present a chart with the distribution for the difference in the number of characters per matching/non-matching toponym pair, a quantity that is related to the difficulty in matching toponyms. Notice that some of the toponyms collected from GeoNames do indeed correspond to very large strings (e.g., the toponyms *Cathedrale Sainte-Marie de Valence* and *Iglesia Catedral-Basílica Metropolitana de la Asunción de Nuestra Señora de Valencia* constitute one of the matching pairs in the dataset, which is misclassified by the different methods under study) and the largest of these has a total of

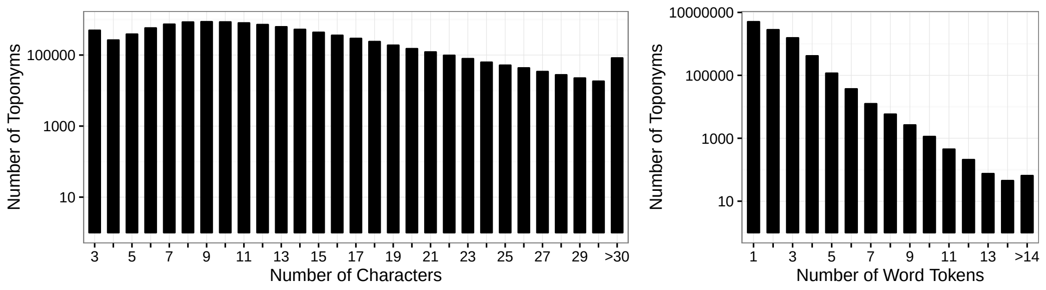

Figure 1. Distribution for the length of the toponyms present in the main dataset.

188 characters and 21 word tokens (i.e., the toponym *Krung Thep Mahanakhon Amon Rattanakosin Mahinthara Ayuthaya Mahadilok Phop Noppharat Ratchathani Burirom Udomratchaniwet Mahasathan Amon Piman Awatan Sathit Sakkathattiya Witsanukam Prasit*, corresponding to a city in Thailand). Moreover, in Figure 3, we present a chart with the distribution for the number of toponym pairs per country (i.e., the number of pairs where both toponyms belong to the same country, either referring to the same real-world place or not), focusing on the 10 countries with the most toponym pairs in our dataset. Figure 3 also presents a chart with the distribution for the number of toponym pairs per alphabet (i.e., the number of pairs where at least one of the toponyms only uses characters from a given alphabet, that either refer to the same real-world place or not). Recall that Recchia and Louwerse (2013) observed a substantial variation in the methods that worked best on the datasets from different countries, suggesting that practitioners should test different algorithms on their country-specific or language-specific data.

Also as part of our dataset characterization, we analyzed the distribution of the number of toponym pairs according to the Damerau-Levenshtein similarity between the strings, which again relates to the difficulty in performing automated classification. Figure 4 presents these results, showing that, as expected, the number and percentage of matching toponym pairs becomes higher as the similarity increases. It should nonetheless also be noted that most toponym pairs have low values for the Damerau-Levenshtein similarity, and a significant number of matching toponyms do indeed belong in these lower similarity intervals.

A variety of experimental methodologies have been used to evaluate the accuracy

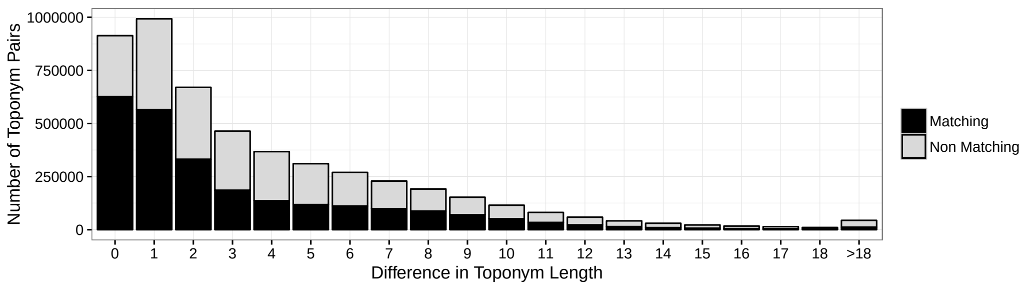

Figure 2. Distribution for the difference in the number of characters per matching/non-matching pair.

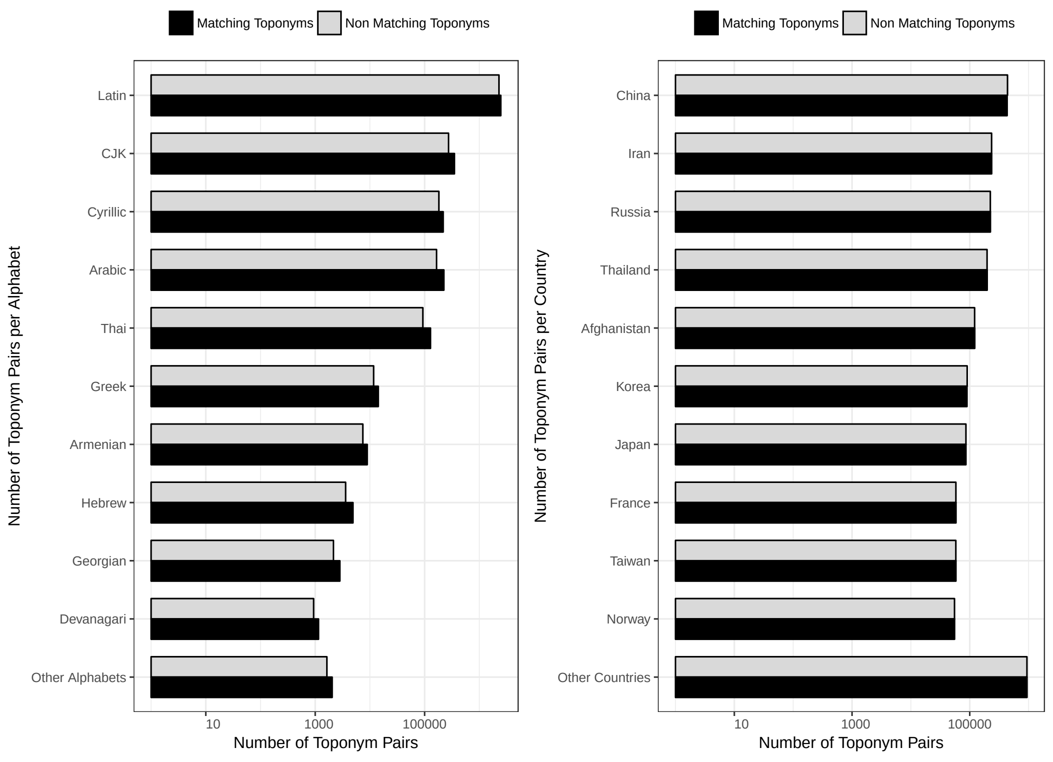

Figure 3. Distribution, over alphabets or countries, for the number of matching/non-matching pairs.

of methods for matching potential duplicates. Authors like Bilenko and Mooney (2003b) have advocated that standard information retrieval metrics such as precision and recall provide an informative evaluation methodology, and thus we have also used these metrics in the present study. Precision and recall are per-class metrics that focus on different aspects of quality for a classification method. Precision is the ratio formed by the number of items correctly assigned to a class divided by the total number of items assigned to the class (e.g., in a toponym matching problem with two possible classes, respectively *match* versus *non-match* decisions, and when focusing on the class that corresponds to the matches, precision corresponds to the number of

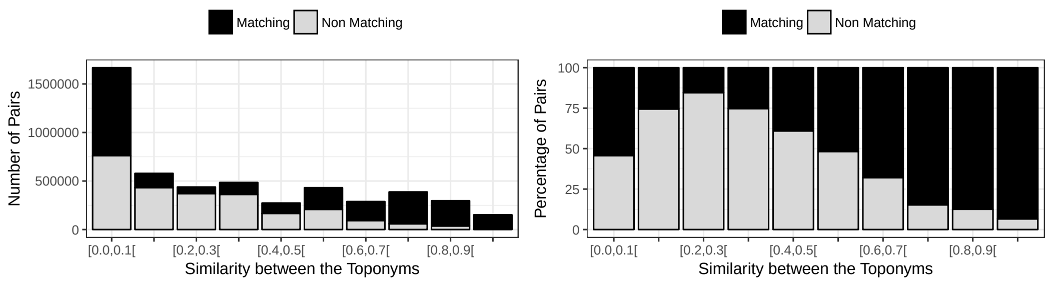

Figure 4. Distribution for the number of pairs according to the Damerau-Levenshtein similarity between the strings, both in absolute values and in terms of the percentage of toponym pairs.

matching pairs that were correctly identified, over the number of matching pairs that were produced by the classification method). Recall, on the other hand, is the ratio between the number of items correctly assigned to a class (e.g., the correct matching decisions), as compared with the total number of items in the class (i.e., the number of correctly matching toponym pairs in the evaluation dataset). Since precision can be increased at the expense of recall, we also computed the the F1-measure, which equally weights precision and recall through their harmonic mean. Finally, we also measured the quality of the results through the accuracy metric, which corresponds to the proportion of correct decisions (i.e., matches or non-matches) that were returned by the method under evaluation.

In the experiments with the GeoNames dataset involving machine learning, we relied on a two-fold cross-validation methodology, in which the available data was first split into two distinct subsets. Classification models were trained and evaluated twice over our main dataset, using one subset of the data for training and the other for evaluation. For each of the aforementioned evaluation metrics, we report averaged values over the two subsets of the data.

4.2. *Experimental Results*

In a first set of experiments, we attempted to evaluate the performance of the individual similarity metrics that were listed in Section 3, varying the similarity threshold between zero and one, with a step size of 0.05. The charts in Figure 5 illustrate the results obtained for the different similarity metrics, in terms of precision, recall, F1, and accuracy. Table 2 summarizes the results for each metric, considering the threshold value that corresponded to the highest accuracy, and presenting also the average processing time involved in computing each similarity metric for sets of fifty thousand records. Although we have relied on Python implementations for the different similarity metrics, which indeed could be further optimized, the values that are reported already provide a good indication for the processing time that one can expect with modern commodity hardware (i.e., in this case, a PC with an Intel Core I7 6700 CPU running at 3.4 GHZ, and with 16GB DDR4 RAM).

The obtained results show that, when properly tuned, the different similarity metrics achieve very similar results in terms of matching quality. Although there are significant differences in terms of computational efficiency, it is fairly easy to perform experiments with very large datasets, using any of the similarity metrics that were considered in the present study. The best results in terms of accuracy, with individual similarity metrics, were achieved with the Jaro-Winkler score computed over reversed strings, closely followed by the Damerau-Levenshtein metric. Both these approaches, as well as the other methods under study, also achieved very high scores in terms of precision (e.g., the Damerau-Levenshtein method achieved a precision of 78.65, which corresponds to the best precision value for all methods that were considered in our study), although recall was relatively low.

Table 2 also presents results for the different approaches based on supervised machine learning, leveraging two-fold cross-validation as stated on the previous section. With this evaluation approach, the entire dataset is used in the computation of the considered evaluation measures, and thus the obtained results are directly comparable to those that were reported for the individual similarity metrics.

It is important to notice that the supervised learning algorithms that were used in our experiments all involve hyper-parameters that can impact their performance. In our tests, we fixed the values for these hyper-parameters (e.g., we often relied on

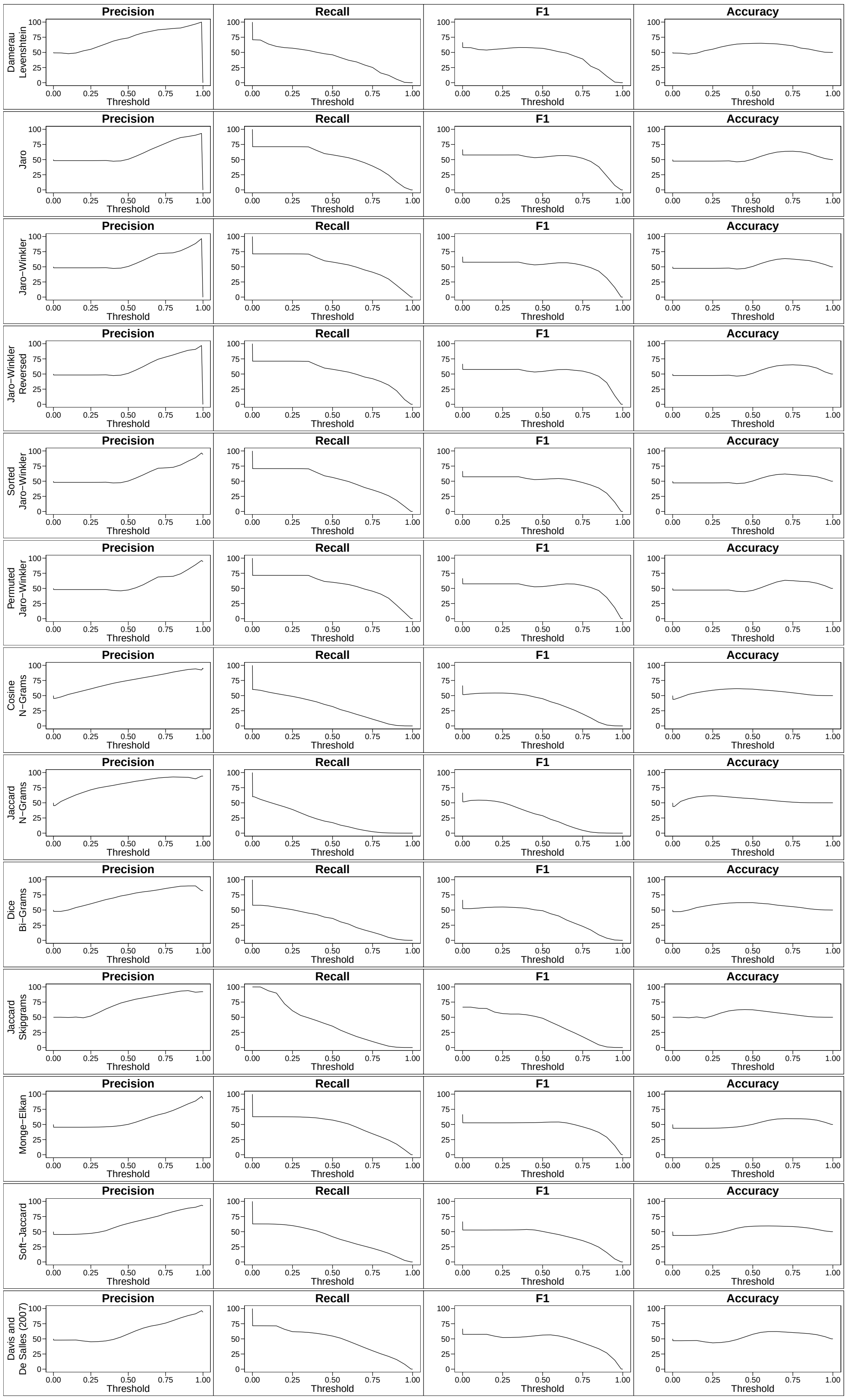

Figure 5. Experimental results obtained by the different similarity metrics.

Table 2. Experimental results obtained with the different methods under consideration.

Method	Accuracy	Precision	Recall	F1-Score	Time (50K Pairs)
Damerau-Levenstein ($\alpha = 0.55$)	65.07	**78.65**	41.36	54.21	0.27 sec.
Jaro ($\alpha = 0.75$)	63.78	76.95	39.34	52.06	0.25 sec.
Jaro-Winkler ($\alpha = 0.70$)	63.59	71.74	44.84	55.19	0.25 sec.
Jaro-Winkler Reversed ($\alpha = 0.75$)	**65.17**	78.00	42.26	54.82	0.27 sec.
Sorted Jaro-Winkler ($\alpha = 0.70$)	61.89	71.44	39.62	50.97	0.34 sec.
Permuted Jaro-Winkler ($\alpha = 0.70$)	63.42	68.90	**48.91**	**57.21**	87.18 sec.
Cosine N-Grams ($\alpha = 0.40$)	61.50	70.37	39.75	50.80	3.03 sec.
Jaccard N-Grams ($\alpha = 0.25$)	61.72	71.50	38.97	50.44	0.81 sec.
Dice Bi-Grams ($\alpha = 0.50$)	62.18	75.36	36.19	48.90	0.61 sec.
Jaccard Skipgrams ($\alpha = 0.45$)	62.69	73.44	39.76	51.59	2.02 sec.
Monge-Elkan ($\alpha = 0.70$)	59.57	65.83	39.79	49.60	0.54 sec.
Soft-Jaccard ($\alpha = 0.60$)	59.43	69.65	33.43	45.18	0.56 sec.
Davis and De Salles (2007) ($\alpha = 0.65$)	62.10	71.03	40.86	51.88	1.27 sec.
Support Vector Machines	72.38	69.17	**80.76**	74.52	101.52 sec.
Random Forests	**78.67**	**78.03**	79.80	78.91	131.60 sec.
Extremely Randomized Trees	78.37	78.00	79.04	78.52	169.83 sec.
Gradient Boosted Trees	78.54	77.51	80.42	**78.94**	99.05 sec.

the default values considered in the implementations that were used) and did not attempt to fine-tune the different models. For instance, the regularization constant in the case of the linear SVM models was kept at the default value of one. Random forests and ensembles of extremely randomized trees involved a total of 600 trees in each model, while ensembles based on gradient boosting involved a total of 3000 trees. Usually, increasing the number of trees in these ensemble approaches results in better estimates, at the expense of increased computational demands.

The results on Table 2 show that all the considered supervised learning algorithms can significantly outperform the individual similarity metrics in terms of matching quality. Only the Damerau-Levenstein method outperformed the learning approaches in terms of precision, but with much lower values in terms of accuracy (i.e., a difference of 13.6 points, in comparison with the best method), F1-score (i.e., a difference of 24.7 points towards the best method), and recall (i.e., almost half the points from those of the best method). The methods based on ensembles of decision trees were particularly effective, with very high values in terms of precision and at the same time almost doubling the recall values obtained by the individual similarity metrics. The best results, in terms of accuracy, were achieved with the method based on random forests, corresponding to a precision of 78.03 and a recall of 79.80. The computation times that are reported on Table 2, for the case of the methods based on supervised machine learning, includes the time involved in computing the different similarity metrics that are used as features, plus the time involved in applying the classification algorithm, ignoring the time spent on model training (i.e., model training can be performed only once and in an offline stage, thus not impacting the performance of these methods when matching previously unseen toponyms). As expected, the methods based on supervised machine learning are computationally more demanding, although with modern hardware it is still fairly easy to process very large datasets. Notice nonetheless that a significant proportion of the time involved in the application of these models relates to the computation of the permuted Jaro-Winkler similarity metric, which is indeed quite demanding, even if we limit the number of tokens to permute to five.

In order to assess the contribution of this last particular feature on the overall quality of the results, we performed an additional set of experiments in which the same supervised learning methods were used without the feature corresponding to the permuted Jaro-Winkler similarity. Table 3 presents the obtained results, showing

Table 3. Experimental results obtained with the different machine learning methods under consideration, without using the permuted Jaro-Winkler similarity metric as a feature.

Method	Accuracy	Precision	Recall	F1-Score	Time (50K Pairs)
Support Vector Machines	72.32	69.30	80.16	74.34	8.78 sec.
Random Forests	**78.63**	**77.99**	79.78	78.88	45.82 sec.
Extremely Randomized Trees	78.35	77.96	79.04	78.50	94.50 sec.
Gradient Boosted Trees	78.49	77.45	**80.38**	**78.89**	9.47 sec.

that the permuted Jaro-Winkler metric only introduces a marginal improvement, at a high computational cost. In these tests, the model based on random forests is still the most effective in terms of accuracy, performing almost as well as the model that leverages all the features (i.e., 78.63 versus 78.67, in terms of accuracy), and only requiring approximately one third of the time for processing 50K toponym pairs. The computational effort involved in the usage of learned models is indeed significant, when compared to some of the simpler string similarity metrics. However, when attempting to match toponyms against entries in a worldwide gazetteer like GeoNames, simple approaches based on a single similarity metric (e.g., the Jaccard coefficient over n-grams, with a permissive threshold in order to get a high recall, and noting that this method can easily be used for searching over millions of strings through implementations based on an inverted index) can be used as filters, limiting the number of pairs that are latter analyzed with machine learning methods.

In classification models based on decision trees, the relative rank (i.e., the depth) of a feature used as a decision node in a tree can be used to assess the relative importance of that feature with respect to the predictability of the output variable. Features used at the top of the tree contribute to the final decision of a larger fraction of the training instances. The expected fraction of the instances they contribute to can thus be used as an estimate of the relative importance of the features. By averaging those estimates, over several randomized trees, one can reduce the variance and effectively use the resulting values for evaluating the importance of the different features. The charts in Figure 6 plot the relative feature importance, as computed from the models based on random forests, extremely randomized trees, and gradient boosting, when using all the features. The numbers in the Y axis corresponding to each feature (i.e., the values for the similarity metrics) follow the

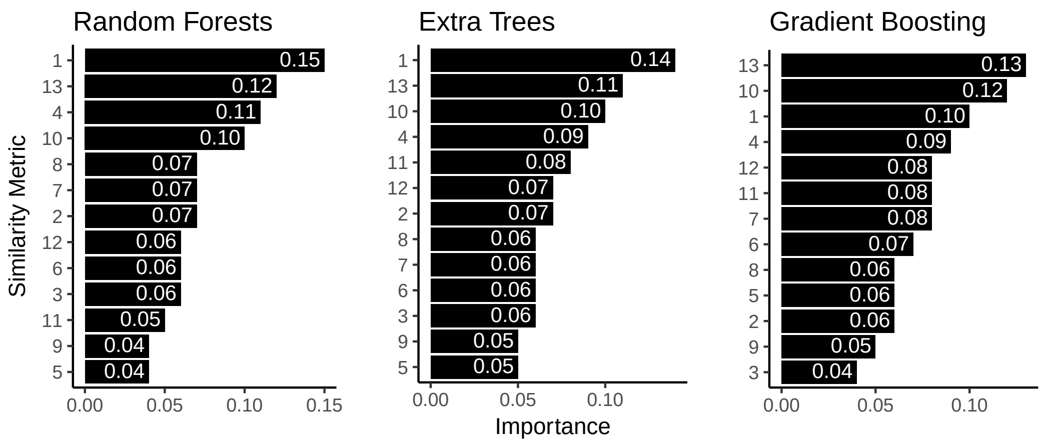

Figure 6. Importance estimates for the different similarity metrics.

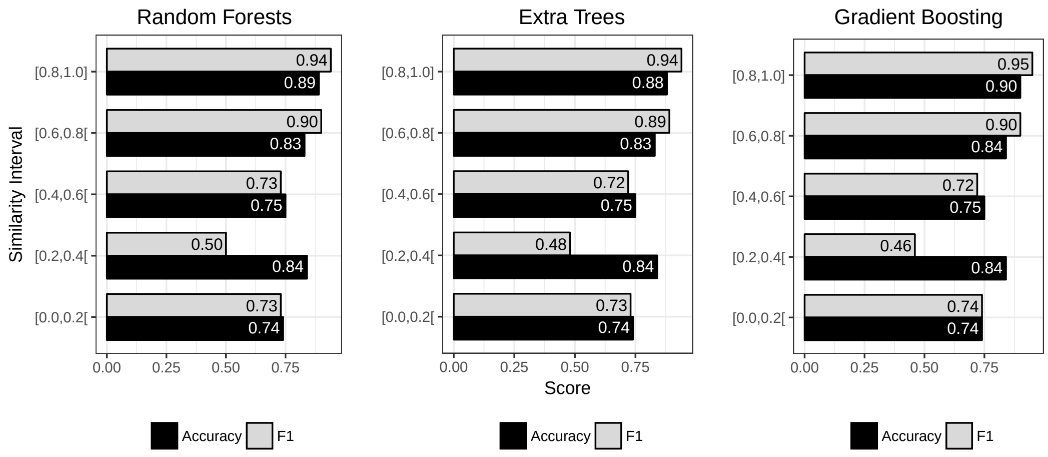

Figure 7. Accuracy and the F1-measure for different Damerau-Levenshtein similarity intervals.

order in the enumeration from Section 3, and also the order by which features are presented in Table 2 and Figure 5. These plots suggest that all the considered features are somewhat informative, with approaches such as the Damerau-Levenshtein similarity (i.e., the top-performing method in terms of precision) appearing to contribute significantly to the final results. It is interesting to notice that, in all three types of models, the importance ranking that is attributed to the features remains relatively consistent. The top-contributing features always included the Damerau-Levenshtein metric and the metric from Davis and De Salles (2007), which itself combines different types of heuristics. The permuted Jaro-Winkler similarity metric, that is computationally the most expensive, always appears in the lower half of the features, when sorting them according to the estimated importance.

In Figure 7, we plot the results obtained with different classification methods based on ensembles of trees (i.e., the best performing methods in terms of accuracy, and also the methods considered on Figure 6), in terms of their effectiveness as a function of the Damerau-Levenshtein similarity between the toponyms. Better results are achieved in the case of highly similar toponym pairs, which should naturally be expected given that the large majority of the pairs within those intervals indeed correspond to matching toponyms – see the plots in Figure 4. Worse results in terms of the F1-measure are achieved for the interval $[0.2, 0.4[$, although the accuracy in this interval is quite high. The low score in terms of the F1-measure relates to a poor performance in terms of recall for the matching pairs, whereas the fact that there are few matching pairs within this interval explains the high accuracy. These results also point to the fact that the proposed techniques still have difficulties in correctly matching toponym pairs that are highly dissimilar.

We also analyzed the results according to the alphabets and countries associated to the toponyms. In the case of the alphabets, we report only on accuracy for the random forest model, as computed over the subset of pairs where (i) at least one of the toponyms uses characters from a given alphabet, or (ii) both toponyms are consistent in their alphabet. In the case of countries, we report on accuracy with the random forest model for pairs where both toponyms belong to the same country. Figure 8 presents these results, and we can see that a worse accuracy is achieved for the case of pairs where different alphabets are being used in each of

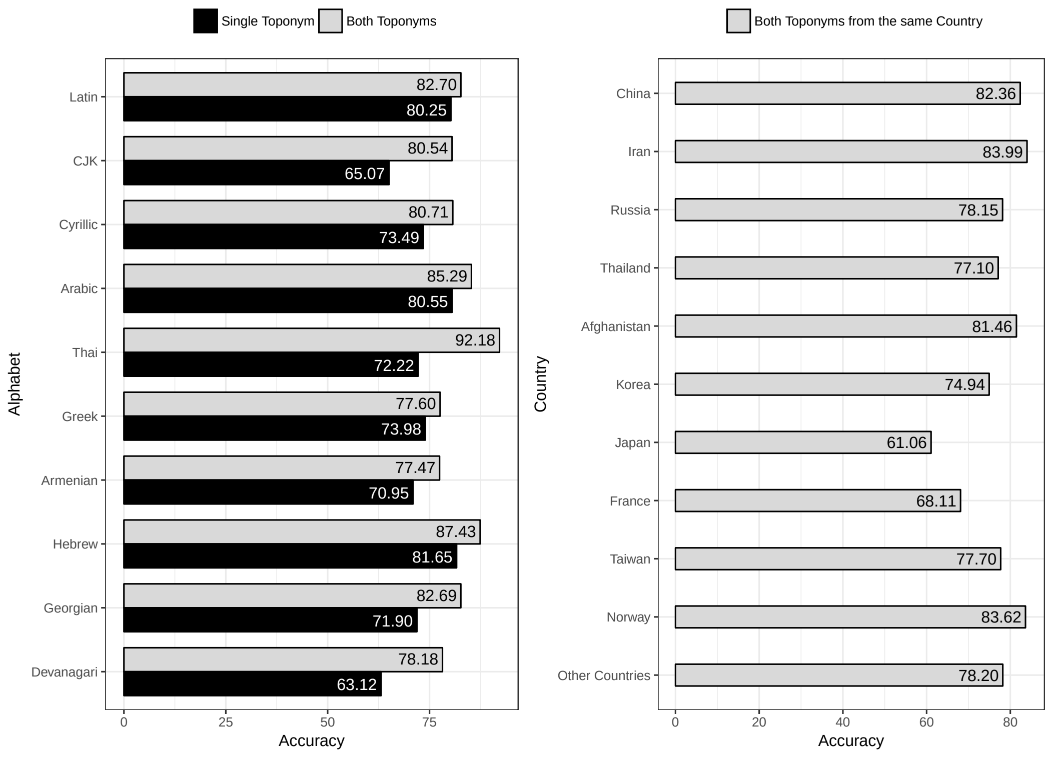

Figure 8. Accuracy results for toponym pairs from different alphabets or countries.

the involved toponyms. Matching strings across different alphabets is indeed an important limitation in the methods studied in this article, given that they always depend on being able to match characters. Figure 8 also shows that results can vary significantly across countries. The highest accuracy was measured for Nepal (i.e., an accuracy of 88.00, for a total of 3492 toponym pairs), and the lowest for Vatican City (i.e., an accuracy of 35.71, for a total of 14 toponym pairs). The top 3 countries with more toponyms in the dataset all show accuracy values above 86.5.

In complement to the quantitative analysis of the results obtained over the GeoNames dataset, we performed a more detailed error analysis in order to better understand the capabilities of the methods under study, e.g. manually inspecting the obtained results, and checking if the proposed machine learning methods could also generalize to datasets from domains other than GeoNames. For instance, we tried to see if the considered methods were indeed effective at matching city names against historical variants, or when matching exonyms (e.g., English names for German, French, or Italian toponyms), using a smaller dataset with pairs collected from Wikipedia pages containing toponym lists.

The smaller dataset involved a total of 3,000 pairs, half of which corresponding to matching toponyms. The matching examples were collected directly from Wikipedia, whereas the non-matching instances were obtained by taking one of the toponyms from the matching pairs, searching for the five most similar toponyms associated to different places in the data collected from Wikipedia, and randomly choosing one of these most similar examples. The different methods from Table 2 were also applied to the smaller dataset, using the pre-established thresholds for the similarity

Table 4. Experimental results obtained with the smaller dataset collected from Wikipedia lists.

Method	Accuracy	Precision	Recall	F1-Score
Damerau-Levenstein ($\alpha = 0.55$)	49.23	41.86	12.39	19.12
Jaro ($\alpha = 0.75$)	49.00	41.15	12.32	18.96
Jaro-Winkler ($\alpha = 0.70$)	41.70	32.63	19.13	24.12
Jaro-Winkler Reversed ($\alpha = 0.75$)	43.00	28.62	11.84	16.75
Sorted Jaro-Winkler ($\alpha = 0.70$)	42.03	32.81	18.79	23.89
Permuted Jaro-Winkler ($\alpha = 0.70$)	41.67	32.87	**19.61**	**24.57**
Cosine N-Grams ($\alpha = 0.40$)	49.60	40.26	8.40	13.90
Jaccard N-Grams ($\alpha = 0.25$)	49.77	40.88	8.33	13.84
Dice Bi-Grams ($\alpha = 0.50$)	**51.17**	**47.58**	8.12	13.87
Jaccard Skipgrams ($\alpha = 0.45$)	51.07	47.08	8.33	14.15
Monge-Elkan ($\alpha = 0.70$)	41.90	32.14	17.96	23.05
Soft-Jaccard ($\alpha = 0.60$)	47.40	38.40	14.25	20.78
Davis and De Salles (2007) ($\alpha = 0.65$)	45.63	37.10	17.62	23.89
Support Vector Machines	51.87	50.52	20.03	28.69
Random Forests	53.80	51.89	34.89	41.73
Extremely Randomized Trees	**61.53**	**65.49**	**43.50**	**52.27**
Gradient Boosted Trees	54.33	53.86	39.85	45.81

measures, and using the classification models trained with GeoNames data (i.e., we made our classification decisions with basis on the average of the confidence scores produced by the two models that were trained from GeoNames data, for each classification method in the cross-validation tests). Table 4 presents the obtained results, showing significantly worse scores that those reported for the GeoNames dataset. Using the same similarity thresholds was not particularly effective (i.e., we measured an accuracy below 50% for most of the different string similarity metrics, and recall values below 20% for the matching class), confirming the need for properly tuning these values. The Dice similarity coefficient between bi-grams achieved the best results in terms of accuracy for the individual metrics. The different machine learning methods outperformed the individual similarity metrics, particularly in terms of the recall metric, although the obtained results are inferior to those reported for the GeoNames dataset. Several matching pairs of toponyms (e.g., *Burdigala* and *Bordeaux*) with similarity values below the considered thresholds were correctly classified by learned models, which at the same time also correctly handled highly similar non-matching pairs (e.g., *Berolinum* and *Verodunum*).

Table 5 illustrates the results obtained from the experiments with the smaller dataset, in the case of matching toponym pairs. First, we show results for the ten largest cities in the United Kingdom, and then for ten other examples corresponding to exonyms. These examples again show good results for models based on ensembles of decision trees, particularly in the case of exonyms, and they confirm the difficulty in manually setting an appropriate similarity threshold (i.e., some of the matching pairs are indeed highly dissimilar, and we therefore have that considering a low similarity threshold would likely lead to many false positives).

In Table 6, we further illustrate the results obtained with the method based on random forests, showing examples of both matching and non-matching pairs of toponyms from the GeoNames dataset, that were either correctly or incorrectly classified. Our manual analysis of the results indicates that although the methods based on supervised learning can already detect some of the difficult matching cases (e.g., cases with strings that have a high Levenshtein distance between them), problems remain in terms of detecting some of the more complex transliterations.

Table 5. Illustrative examples for the results with the smaller dataset, when matching names for cities in the United Kingdom against historical variants, and when matching exonyms collected from Wikipedia lists.

Modern Name	Historical Name	Damerau-Levenshtein Similarity	Matching Decision SVM	RF	ERT	GBT
London	Londinium	**0.56**	✗	✓	✗	✗
Birmingham	Bromwicham	0.50	✗	✓	✗	✓
Leeds	Ledes	**0.80**	✓	✓	✓	✓
Glasgow	Glas Cau	0.50	✗	✗	✗	✗
Sheffield	Shaffeld	**0.78**	✓	✓	✓	✓
Bradford	Bradeford	**0.89**	✓	✓	✓	✓
Liverpool	Lerpwl	**0.56**	✗	✓	✗	✓
Edinburgh	Edenesburg	**0.60**	✓	✓	✓	✓
Manchester	Mameceaster	**0.64**	✓	✓	✓	✓
Bristol	Bricstow	**0.75**	✓	✓	✓	✓
Constance	Konstanz	**0.67**	✓	✓	✓	✓
Vienna	Wien	0.50	✗	✓	✓	✓
Swabia	Schwaben	0.50	✗	✓	✓	✗
Horsemarket	Rossmarck	0.45	✓	✓	✓	✓
Pruce	Preußen	**0.57**	✗	✓	✗	✓
Cologne	Köln	0.29	✗	✓	✓	✓
Wirtemberg	Württemberg	**0.75**	✓	✓	✓	✓
Berne	Bern	**0.80**	✓	✓	✓	✓
Leghorn	Livorno	0.50	✗	✓	✓	✗
Frisia	Friesland	**0.56**	✗	✓	✓	✗

5. Conclusions and Future Work

This article presented the results of a wide-ranging evaluation on the performance of thirteen different string similarity metrics, representative of different classes of methods, over the task of matching toponyms. Using a very large dataset with five million pairs of toponyms, collected from lists of alternative place names taken from the GeoNames gazetteer, we showed that the differences in performance for the considered similarity metrics are relatively small, although carefully tuning the similarity threshold involved in matching decisions is important for achieving good results. We then experimented with the usage of supervised machine learning for combining multiple similarity metrics, which has the natural advantage of avoiding the manual tuning of similarity thresholds. Our experiments show that the methods based on supervised learning, particularly when considering ensembles of decision trees, can achieve good results on the toponym matching task, significantly outperforming the individual similarity metrics. However, the obtained results also showed problems in terms of handling matching toponyms that are highly dissimilar.

Despite the interesting results, there are indeed many open challenges for future work in toponym matching. For instance, the machine learning algorithms that were employed in the present study have hyper-parameters (e.g., the regularization

Table 6. Illustrative examples for the results obtained with the method based on random forests.

	Correctly Classified	Incorrectly Classified
Matching	*Ozernoje ; Ozernoye*	*Lingzhithang ; Lingzi Tāng*
	Broadstairs ; Brodsters	*Cawdor Castle ; Chateau de Cawdor*
	Orange County ; Comté d'Orange	長間瀬 ; *Nagamase*
	Saut Majami ; South Miami	*Klejmont ; Claymont*
	Corno Gries ; Grieshorn	*St. Simons ; Saint Simons Island*
	Siedenbrünzow ; Zidenbrincov	*Pontypool ; Pont-y-pŵl*
	Sodelhas ; Soudeilles	*Corsica settentrionale ; Alta Córcega*
Non-Matching	*Buenavista ; Buenache*	*ban pa kha ; Ban Pak Dong*
	Monterej ; Quận Montgomery	*Observatorio Griffith ; Griffith Park*
	Hayy ad Duhayni ; Hayy ad Dihliz	*Jaypāsa ; Jaypara*
	Frankfort ; Frederick	*Cavalier ; Okrug Kavalir*
	Cleveland ; Clermont	*Ben Bhuidhe Mhor ; Beinn Bhuidhe*
	Génissieux ; Geissan	*Bliskastel' ; Bliesdorf*
	Lichtenstein ; Likhtenrade	*San Joaninho ; Sao Joanico*

constant in the case of SVM models, or the maximum tree depth in the case of gradient boosted ensembles of decision trees) that, when properly tuned, can lead to slight improvements on the results. It should nonetheless be noted that, although for future work we can consider additional experiments related to the tuning of these parameters, the results that are reported here already confirm the advantages of using supervised machine learning for combining multiple similarity metrics.

More interestingly, our currently ongoing experiments are focused on evaluating supervised learning approaches that, instead of using the values of similarity metrics as features, attempt to leverage the input strings directly. A previous publication by Bilenko and Mooney (2003a) has already described an approach, based on SVM models, which produces better similarity estimates than standard approaches across a range of difficult examples. In their method, each string is first converted to a vector-space representation. An instance vector is then created from the pair of vector-space representations, each component of which corresponding to the product of weights for the elements of each vector. The pair instance is finally classified by an SVM model. Taking inspiration on more recent studies that addressed the natural language processing problem of detecting paraphrases (i.e., phrases that express the same meaning using different words), we are currently experimenting with the usage of deep learning methods based on recurrent and/or convolutional neural networks (Hu et al. 2014; Yin and Schütze 2015; Bowman et al. 2015; Rocktäschel et al. 2016; Yin et al. 2016; Wan et al. 2016; Sun et al. 2017; Parikh et al. 2016). In general, these methods are based on modeling a pair of phrases (i.e., two sequences of characters or sequences of tokens) through recurrent and/or convolutional neural network nodes, afterwards producing a representation for the pair of strings under comparison by aggregating interactions between the outputs of the recurrent/convolutional parts of the model. A matching score is finally produced through a transformation of the nodes that model the interactions. As happens in the case of paraphrase identification, these fully-differentiable deep neural models have the potential to significantly outperform the feature-based supervised learning methods used in the present study, by capturing both common and complex character transliterations in the matching pairs of toponyms from the training data (i.e., matching Asian or Arabic toponyms against their Western transliterations, which would be almost impossible for the similarity metrics considered in the present study), and in general improving results in the case of highly dissimilar pairs.

Currently ongoing work is also addressing the application and extension of the methods presented in this article to different problems within the broad field of digital humanities, particularly problems that involve the analysis of toponyms in context (e.g., disambiguating place name references in textual documents (Santos, Anastácio, and Martins 2015; Wing 2016; Ardanuy and Sporleder 2017)). Recent work within the geospatial humanities, particularly within the context of historical geographic information systems, has highlighted the importance of automated methods for handling toponyms (Smith and Crane 2001; Grover et al. 2010; Rupp et al. 2013; Simon et al. 2014; Gregory et al. 2015; Murrieta-Flores et al. 2015; Blank and Henrich 2016; Wing 2016; Murrieta-Flores, Donaldson, and Gregory 2017; Clifford et al. 2016; Butler et al. 2017). Approaches looking to map and analyze the geographies mentioned in large collections of historical materials (e.g., historical newspapers, letters, governmental reports, epistolaries, travel guides, tabular itineraries, and many other types of records) are often confronted with the issue of correctly matching alternative forms of place names (e.g., involving spelling variations, historical changes, OCR errors, etc.) against gazetteer entries. The methods presented in this article, and the aforementioned extensions based on deep learning, are of significant interest to these particular application domains. Currently ongoing experiments are, for instance, addressing the usage of toponym matching through supervised machine learning (i.e., the method based on random forests that was evaluated in this article) within the context of automatically geocoding historical itineraries, taking inspiration on the previous work by Blank and Henrich (2016).

Acknowledgements

We would particularly like to thank our colleagues Pável Calado and Mário J. Silva, for their comments on preliminary versions of this work.

Funding

This research was supported by the Trans-Atlantic Platform for the Social Sciences and Humanities, through the Digging into Data project with reference HJ-253525, and also through the Reassembling the Republic of Letters networking programme (EU COST Action IS1310). The researchers from INESC-ID also had financial support from Fundação para a Ciência e Tecnologia (FCT), through project grants with references PTDC/EEI-SCR/1743/2014 (Saturn) and CMUPERI/TIC/0046/2014 (GoLocal), as well as through the INESC-ID multi-annual funding from the PIDDAC programme (UID/CEC/50021/2013).

References

Anastácio, Ivo, Bruno Martins, and Pável Calado. 2009. "Classifying documents according to locational relevance." In *Proceedings of the Portuguese Conference on Artificial Intelligence,* Springer.

Ardanuy, Mariona Coll, and Caroline Sporleder. 2017. "Toponym Disambiguation in Historical Documents Using Semantic and Geographic Features." In *Proceedings of the International Conference on Digital Access to Textual Cultural Heritage,* ACM.

Banfield, Robert E., Lawrence O. Hall, Kevin W. Bowyer, and W. Philip Kegelmeyer. 2007. "A comparison of decision tree ensemble creation techniques." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (1).

Berkhin, Pavel, Michael R. Evans, Florin Teodorescu, Wei Wu, and Dragomir Yankov. 2015. "A New Approach to Geocoding: BingGC." In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* ACM.

Berman, Merrick Lex, Johan Åhlfeldt, and Marc Wick. 2016. "Historical Gazetteer System Integration: CHGIS, Regnum Francorum, and GeoNames." In *Placing Names : Enriching and Integrating Gazetteers,* edited by Merrick Lex Berman, Ruth Mostern, and Humphrey Southall. Indiana University Press.

Berman, Merrick Lex, Ruth Mostern, and Humphrey Southall, eds . 2016. *Placing Names : Enriching and Integrating Gazetteers.* Indiana University Press.

Bilenko, Mikhail, and Raymond J Mooney. 2003a. "Adaptive duplicate detection using learnable string similarity measures." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* ACM.

Bilenko, M., and R. J. Mooney. 2003b. "On Evaluation and Training-Set Construction for Duplicate Detection." In *Proceedings of the KDD Workshop on Data Cleaning, Record Linkage, and Object Consolidation,* ACM.

Bishop, Christopher M. 2006. *Pattern Recognition and Machine Learning.* Springer.

Blank, Daniel, and Andreas Henrich. 2016. "A depth-first branch-and-bound algorithm for geocoding historic itinerary tables." In *Proceedings of the ACM Workshop on Geographic Information Retrieval,* ACM.

Bowman, Samuel R., Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. "A large annotated corpus for learning natural language inference." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* ACL.

Breiman, Leo. 2001. "Random forests." *Machine learning* 45 (1).

Brill, Eric, and Robert C. Moore. 2000. "An Improved Error Model for Noisy Channel Spelling Correction." In *Proceedings of the Annual Meeting on Association for Computational Linguistics,* ACL.

Butler, James O., Christopher E. Donaldson, Joanna E. Taylor, and Ian N. Gregory. 2017. "Alts, Abbreviations, and AKAs: historical onomastic variation and automated named entity recognition." *Journal of Map & Geography Libraries* 13 (1).

Chen, Tianqi, and Carlos Guestrin. 2016. "XGBoost: A Scalable Tree Boosting System." In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* ACM.

Cheng, Gang, Xiaoping Lu, Xiaosan Ge, Haiyang Yu, Yupeng Wang, and Xiaotian Ge. 2010. "Data fusion method for digital gazetteer." In *Proceedings of the International Conference on Geoinformatics,* IEEE.

Cheng, Gang, Fei Wang, Haiyang Lv, and Yinling Zhang. 2011. "A new matching algorithm for chinese place names." In *Proceedings of the International Conference on Geoinformatics,* IEEE.

Christen, Peter. 2006. "A Comparison of Personal Name Matching: Techniques and Practical Issues." In *Proceedings of the Workshops at the IEEE International Conference on Data Mining,* IEEE.

Christen, Peter, Alan Willmore, and Tim Churches. 2006. "A probabilistic geocoding system utilising a parcel based address file." In *Data Mining : Theory, Methodology, Techniques, and Applications,* edited by Graham J. Williams and Simeon J. Simoff. Springer.

Clifford, Jim, Beatrice Alex, Colin M Coates, Ewan Klein, and Andrew Watson. 2016. "Geoparsing history: Locating commodities in ten million pages of nineteenth-century sources." *Historical Methods: A Journal of Quantitative and Interdisciplinary History* 49 (3).

Cohen, William, Pradeep Ravikumar, and Stephen Fienberg. 2003. "A Comparison of String Distance Metrics for Name-Matching Tasks." In *Proceedings of KDD Workshop on Data Cleaning and Object Consolidation,* AAAI.

Dalvi, Nilesh, Marian Olteanu, Manish Raghavan, and Philip Bohannon. 2014. "Deduplicating a Places Database." In *Proceedings of the International Conference on World Wide Web,* ACM.

Damerau, Fred J. 1964. "A technique for computer detection and correction of spelling errors." *Communications of the ACM* 7 (3).

Daumé, Hal. 2015. *A Course in Machine Learning.*

Davis, Clodoveu A, and Emerson De Salles. 2007. "Approximate String Matching for Geographic Names and Personal Names." In *Proceedings of the Brazilian Symposium on GeoInformatics,* Springer.

Dice, Lee R. 1945. "Measures of the amount of ecologic association between species." *Ecology* 26 (3).

Freire, Nuno, José Borbinha, Pável Calado, and Bruno Martins. 2011. "A metadata geoparsing system for place name recognition and resolution in metadata records." In *Proceedings of the Annual International ACM/IEEE Joint Conference on Digital Libraries,* ACM.

Friedman, Jerome H. 2001. "Greedy function approximation: a gradient boosting machine." *Annals of Statistics* 29 (5).

Fu, Gaihua, Christopher B. Jones, and Alia I. Abdelmoty. 2005. "Building a Geographical Ontology for Intelligent Spatial Search on the Web." In *Proceedings of the IASTED International Conference on Databases and Applications,* Springer.

Gelernter, Judith, and Wei Zhang. 2013. "Cross-lingual Geo-parsing for Non-structured Data." In *Proceedings of the ACM Workshop on Geographic Information Retrieval,* ACM.

GeoNames. 2017. `http://www.geonames.org/`.

Geurts, Pierre, Damien Ernst, and Louis Wehenkel. 2006. "Extremely randomized trees." *Machine learning* 63 (1).

Gregory, Ian, Christopher Donaldson, Patricia Murrieta-Flores, and Paul Rayson. 2015. "Geoparsing, GIS, and textual analysis: current developments in spatial humanities research." *International Journal of Humanities and Arts Computing* 9 (1).

Grover, Claire, Richard Tobin, Kate Byrne, Matthew Woollard, James Reid, Stuart Dunn, and Julian Ball. 2010. "Use of the Edinburgh geoparser for georeferencing digitized historical collections." *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 368 (1925).

Hastings, Jordan. 2008. "Automated conflation of digital gazetteer data." *International Journal of Geographical Information Science* 22 (10).

Hastings, Jordan, and Linda Hill. 2002. "Treatment of duplicates in the alexandria digital library gazetteer." In *Proceedings of the International Conference on Geographic Information Science,* Springer.

Hu, Baotian, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. "Convolutional neural network architectures for matching natural language sentences." In *Proceedings of the International Conference on Neural Information Processing Systems,* MIT Press.

Jaccard, Paul. 1912. "The distribution of the flora in the alpine zone.." *New phytologist* 11 (2).

James, Gareth, Daniela Witten, and Trevor Hastie. 2014. *An Introduction to Statistical Learning: With Applications in R.* Springer.

Jaro, Matthew 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).

Joshi, Tanuja, Joseph Joy, Tobias Kellner, Udayan Khurana, A Kumaran, and Vibhuti Sengar. 2008. "Crosslingual Location Search." In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval,* ACM.

Keskustalo, Heikki, Ari Pirkola, Kari Visala, Erkka Leppänen, and Kalervo Järvelin. 2003. "Non-adjacent digrams improve matching of cross-lingual spelling variants." In *Proceedings of the International Symposium on String Processing and Information Retrieval,* Springer.

Kılınç, Deniz. 2016. "An accurate toponym-matching measure based on approximate string matching." *Journal of Information Science* 42 (2).

Kotsiantis, Sotiris B. 2013. "Decision trees: a recent overview." *Artificial Intelligence Review* 39 (4).

Levenshtein, Vladimir I. 1966. "Binary codes capable of correcting deletions, insertions and reversals." *Soviet Physics Doklady* 10 (1).

Li, Lin, Xiaoyu Xing, Hui Xia, and Xiaoying Huang. 2016. "Entropy-Weighted Instance Matching Between Different Sourcing Points of Interest." *Entropy* 18 (2): 45.

Manguinhas, Hugo, Bruno Martins, and José Borbinha. 2008. "A geo-temporal web gazetteer integrating data from multiple sources." In *Proceedings of the International Conference on Digital Information Management,* IEEE.

Martins, Bruno. 2011. "A supervised machine learning approach for duplicate detection over gazetteer records." In *Proceedings of the International Conference on GeoSpatial Sematics,* Springer.

McKenzie, Grant, Krzysztof Janowicz, and Benjamin Adams. 2014. "A weighted multi-attribute method for matching user-generated points of interest." *Cartography and Geographic Information Science* 41 (2).

Monge, Alvaro E., and Charles P. Elkan. 1996. "The Field Matching Problem: Algorithms and Applications." In *Proceedings of the International Conference on Knowledge Discovery and Data Mining,* AAAI.

Monteiro, Bruno R., Clodoveu A. Davis, and Fred Fonseca. 2016. "A survey on the geographic scope of textual documents." *Computers & Geosciences* 96 (C).

Morana, Anthony, Thomas Morel, Bilal Berjawi, and Fabien Duchateau. 2014. "GeoBench: A Geospatial Integration Tool for Building a Spatial Entity Matching Benchmark." In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* ACM.

Moreau, Erwan, François Yvon, and Olivier Cappé. 2008. "Robust similarity measures for named entities matching." In *Proceedings of the International Conference on Computational Linguistics,* ACL.

Murphy, Kevin Patrick. 2013. *Machine Learning: a Probabilistic Perspective.* MIT Press.

Murrieta-Flores, Patricia, Alistair Baron, Ian Gregory, Andrew Hardie, and Paul Rayson. 2015. "Automatically analyzing large texts in a GIS environment: The Registrar General's reports and cholera in the 19th Century." *Transactions in GIS* 19 (2).

Murrieta-Flores, Patricia, Christopher Elliott Donaldson, and Ian Norman Gregory. 2017. "GIS and literary history: advancing digital humanities research through the spatial analysis of historical travel writing and topographical literature." *Digital Humanities Quarterly* 11 (1).

Navarro, Gonzalo. 2001. "A Guided Tour to Approximate String Matching." *ACM Computing Surveys* 33 (1).

Needleman, Saul B, and Christian D Wunsch. 1970. "A general method applicable to the search for similarities in the amino acid sequence of two proteins." *Journal of molecular biology* 48 (3).

Parikh, Ankur P, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. "A Decomposable Attention Model for Natural Language Inference." In *Proceedings of the Conference on Empirical Methods in Natural Language Processing,* ACL.

Philips, Lawrence. 1990. "Hanging on the metaphone." *Computer Language* 7 (12).

Philips, Lawrence. 2000. "The double metaphone search algorithm." *C/C++ Users Journal* 18 (6).

Recchia, Gabriel, and Max Louwerse. 2013. "A Comparison of String Similarity Measures for Toponym Matching." In *Proceedings of The ACM SIGSPATIAL International Workshop on Computational Models of Place,* ACM.

Ristad, Eric Sven, and Peter N. Yianilos. 1998. "Learning string-edit distance." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20 (5).

Rocktäschel, Tim, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočiskỳ, and Phil Blunsom. 2016. "Reasoning about entailment with neural attention." In *Proceedings of the International Conference on Learning Representations,* (also published as arXiv preprint arXiv:1509.06664).

Rupp, C. J., Paul Rayson, Alistair Baron, Christopher Donaldson, Ian Gregory, Andrew Hardie, and Patricia Murrieta-Flores. 2013. "Customising geoparsing and georeferencing for historical texts." In *Proceedings of the IEEE International Conference on Big Data,* IEEE.

Samal, Ashok, Sharad Seth, and Kevin Cueto. 2004. "A feature-based approach to conflation of geospatial sources." *International Journal of Geographical Information Science* 18 (5).

Santos, João, Ivo Anastácio, and Bruno Martins. 2015. "Using machine learning methods for disambiguating place references in textual documents." *GeoJournal* 80 (3).

Scikit-learn. 2017. `http://scikit-learn.org/`.

Sehgal, Vivek, Lise Getoor, and Peter D Viechnicki. 2006. "Entity resolution in geospatial data integration." In *Proceedings of the Annual ACM International Symposium on Advances in Geographic Information Systems,* ACM.

Sengar, Vibhuti, Tanuja Joshi, Joseph Joy, Samarth Prakash, and Kentaro Toyama. 2007. "Robust Location Search from Text Queries." In *Proceedings of the ACM International Symposium on Advances in Geographic Information Systems,* ACM.

Simon, Rainer, Peter Pilgerstorfer, Leif Isaksen, and Elton Barker. 2014. "Towards semi-automatic annotation of toponyms on old maps." *e-Perimetron* 9 (3).

Smart, Philip D., Christopher B. Jones, and Florian A. Twaroch. 2010. "Multi-source toponym data integration and mediation for a meta-gazetteer service." In *Proceedings of the International Conference on Geographic Information Science,* Springer.

Smith, David A., and Gregory Crane. 2001. "Disambiguating Geographic Names in a Historical Digital Library." In *Proceedings of the European Conference on Research and Advanced Technology for Digital Libraries,* Springer.

Sun, Chengjie, Yang Liu, Chang'e Jia, Bingquan Liu, and Lei Lin. 2017. "Recognizing Text Entailment via Bidirectional LSTM Model with Inner-Attention." In *Proceedings of the International Conference on Intelligent Computing,* Springer.

Varol, Cihan, and Coskun Bayrak. 2012. "Hybrid Matching Algorithm for Personal Names." *Journal of Data and Information Quality* 3 (4).

Wan, Shengxian, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. "A Deep Architecture for Semantic Matching with Multiple Positional Sentence Representations." In *Proceedings of the AAAI Conference on Artificial Intelligence,* AAAI.

Weinman, Jerod. 2013. "Toponym Recognition in Historical Maps by Gazetteer Alignment." In *Proceedings of the International Conference on Document Analysis and Recognition,* IEEE.

Wing, Benjamin Patai. 2016. "Text-based document geolocation and its application to the digital humanities." Ph.D. thesis. University of Texas.

Winkler, William E. 1990. "String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage." In *Proceedings of the Annual Meeting of the American Statistical Association - Section on Survey Research Methods,* AMS.

XGBoost. 2017. `http://xgboost.readthedocs.io/`.

Yin, Wenpeng, and Hinrich Schütze. 2015. "Convolutional neural network for paraphrase identification." In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics,* ACL.

Yin, Wenpeng, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. "ABCNN: Attention-based convolutional neural network for modeling sentence pairs." *Transactions of the Association for Computational Linguistics* 4 (1).

Zhang, Qi, Jihua Kang, Yeyun Gong, Huan Chen, Yaqian Zhou, and Xuanjing Huang. 2013. "Map Search via a Factor Graph Model." In *Proceedings of the ACM International Conference on Information & Knowledge Management,* ACM.

Zheng, Yu, Xixuan Fen, Xing Xie, Shuang Peng, and James Fu. 2010. "Detecting nearly duplicated records in location datasets." In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems,* ACM.