Utilizing Phonetic Similarity for Cross-source and Cross-language Toponym Matching - a Benchmark and Prototype

Tomer Sagi
tsagi@cs.aau.dk

Aalborg University

Moran Zaga
University of Haifa

Sinai Rusinek
University of Haifa

Marcell Richard Fekete
Aalborg University

Johannes Bjerva
Aalborg University

Katja Hose
TU Wien

Research Article

Keywords: Toponym Matching, Multi-lingual, Grapheme to Phoneme, Transliteration

Posted Date: March 22nd, 2024

DOI: https://doi.org/10.21203/rs.3.rs-4136375/v1

Additional Declarations: No competing interests reported.

Utilizing Phonetic Similarity for Cross-source and Cross-language Toponym Matching - a Benchmark and Prototype

Tomer Sagi[1*], Moran Zaga[2], Sinai Rusinek[2], Marcell R. Fekete[1], Johannes Bjerva[1], Katja Hose[1,3]

[1*]Department of Computer Science, Aalborg University, Denmark.
[2]e-Lijah Lab, University of Haifa, Israel.
[3]Department of Informatics, TU Wien, Austria.

*Corresponding author(s). E-mail(s): tsagi@cs.aau.dk;
Contributing authors: mzaga@staff.haifa.ac.il; sinai.rusinek@gmail.com; mrfe@cs.aau.dk; jbjerva@cs.aau.dk; khose@cs.aau.dk;

Abstract

The writings of one ancient civilization often overlap in time and space with others. Many of these sources comprise unstructured text in ancient languages, causing scholars studying these civilizations to be siloed, often relying on sources in specific languages. Most recent efforts to extract structured information from historical scripts into place (toponym) and people databases (prospographies) have followed this pattern, focusing on one civilization and selected sources. The path to creating a common database runs through aligning names or toponyms between sources from disparate languages utilizing different scripts. Existing multi-lingual orthographic (string-based) comparison often relies on transliteration to a common script (Latin/English). Transliteration often creates multiple options and even more confusion. However, when integrating sources that overlap in space and time, the languages often share a common phonetic background. This commonality may prove beneficial. In this work, we present a benchmark for comparing toponyms from two linguistically and culturally related languages, namely Hebrew and Arabic. We provide a benchmark comprised of a set of dataset pairs created from historical sources written in Medieval variants of these languages, later historical Gazetteers and a modern dataset curated from Wikidata. We empirically evaluate several toponym comparison approaches over the benchmark: transliteration to a common script, direct transliteration, and phonetic comparison using a common phonetic representation. We discuss the results and the limitations of the various methods and outline future work.

Keywords: Toponym Matching, Multi-lingual, Grapheme to Phoneme, Transliteration

1 Introduction

Studying ancient civilizations requires specialization in the language, cultural background, and even specific scripts used in the period and geography under study. Scholars must learn how to read texts in various ancient, often derelict languages, written in scripts that are not only diverse but also archaic. This linguistic diversity poses a significant challenge, as it leads to a compartmentalization of academic efforts, with researchers tending to specialize in sources from specific languages or scripts. This specialization often results in silos of knowledge that are difficult to cross-reference and integrate, thus overlooking the interconnected nature of the studied civilizations, especially those that existed contemporaneously and geographically proximate to each other. Thus, the ability of scholars from disparate languages and cultural focuses to collaborate over shared data is severely limited. A scholar in Arabic literature of the Middle Ages will be severely challenged when attempting to incorporate information and even raw data from repositories in Hebrew, even though they are from the same period and may substantially overlap in the places mentioned and events referenced. Recently, there have been several cross-lingual reconciliation efforts to create shared repositories for toponyms (place names) and names of people, such as the Pleiades project [1], the World Historical Gazzetteer [2], and the Historical Index of the Medieval Middle East [3]. However, many of these attempts are hampered by the lack of robust automated tools for aligning place names across languages and benchmarks to test these over. Moreover, machine-learning-based approaches are gaining much traction and impressive results over modern languages with a wide base for training and testing resources available to them. However, no such capabilities are available for ancient languages, such as Ancient Hebrew and ancient Arabic.

A critical step in overcoming these challenges lies in aligning and comparing names and places (toponyms) across different sources and languages. This task is particularly challenging when dealing with languages that utilize distinct scripts. The conventional method of multi-lingual orthographic comparison often relies on transliterating these texts into a common script, typically Latin or English. However, this approach has its drawbacks, as transliteration can lead to multiple interpretations and further confusion, especially when dealing with ancient variants of languages.

In light of these challenges, our research focuses on a novel approach to comparing toponyms from two linguistically and culturally related languages: Hebrew and Arabic. Both languages, with their Semitic roots, share a common phonetic background despite their different scripts and historical evolutions. We posit that this phonetic commonality can be leveraged to develop more effective methods for toponym comparison.

In this work, we present our initial foray into the complex problem of cross-lingual toponym matching. We present a new benchmark comprised of place pairs from two different geographical sources, originally in two different ancient languages (Hebrew and Arabic), overlapping in space and time. We investigate the viability of two approaches towards cross-lingual toponym matching. The first utilizes transliteration rules between the two languages to perform a direct orthographic comparison in one or the other language. The second approach maps both scripts into a phonetic representation, specifically the International Phonetic Alphabet (IPA), which allows for a comparison based on phonology (speech sounds) rather than orthography.

We empirically evaluate these approaches both on our proposed benchmark and on a synthetic benchmark generated by mining the labels of toponyms in modern-day versions of these languages, i.e., modern Hebrew and modern standard Arabic from wikidata. We thereby make the following contributions: (1) Direct Transliteration method between Arabic and Hebrew; (2) Rule-enhanced grapheme to phoneme model for Hebrew and Arabic Toponyms; (3) Wikidata-based curated toponym comparison benchmark; (4) Set of matched toponym datasets from historical sources in related Semitic languages; (5) Phonetic-based toponym comparison method; and (6) Empirical evaluation of three basic approaches to toponym matches, namely - direct transliteration, transliteration to a common script, and phonetic comparison using IPA representations.

The rest of the paper is structured as follows. In Section 2 we review related work in the field of automated historical and multi-lingual toponym matching. In Section 3 we introduce our benchmark dataset and the synthetic dataset mined from wikidata. In Section 4 we detail the matching methods using transliteration and grapheme to phoneme conversion. Section 5 presents the experiments we performed and the results obtained and discusses them. We conclude in Section 6.

2 Related Work

Several works have examined the efficacy of different string-based distance metrics for toponym matching. The most relevant of these is the work by Recchia and Louwerse [4], which performed a comprehensive comparison between 21 different metrics on datasets extracted from the Geographic Names Server [5]. The datasets contain romanized toponyms from 11 countries, and the toponyms were compared with their variants used in the same country. To the best of our knowledge, our work is the first to present a cross-language benchmark of *Semitic* languages and the first attempt to compare phoneme-based to transcription-based methods using direct comparison rather than through romanization.

Joshi et al. [6] utilize transliteration methods to power a cross-lingual toponym search engine. However, they do not attempt to match the toponyms to each other as we do here, nor do they handle historical data.

Aligning toponyms across large geographical databases, a task also known as *entity alignment*, has been attempted using various methods (see review [7]). However, none of these methods have used a matched cross-language benchmark to allow direct matching rather than through the romanized labels. Furthermore, to our knowledge, this work is the first to present such a benchmark for historical sources in the Middle East rather than modern place names. Geographical entity alignment algorithms use various types of methods, including textual, semantic, structural, spatial, and recently AI-based (e.g., [8]). In this work, we focus on textual methods, often employed as part of larger solutions [9]. Specifically, we suggest two methods for direct comparison between related languages, showing them superior to methods relying on Romanization in this setting.

Several entity alignment benchmarks have been published in the past, mostly in the business (e.g., [10]) and web (e.g., [11]) domains. However, toponym alignment benchmarks are few and far between. General cross-lingual entity alignment (e.g., [12]) benefits from the fact that most entities, such as products, organizations and events, have descriptive labels and other properties (e.g, price, weight) that can be translated or compared. Toponyms, especially ones extracted from historical sources, have sparse information, often only the name of the place, the general geographical area, and some type information (river/city). Thus, toponym matching relies heavily on transliteration and orthographic comparison [13].

There are a few examples of same-language historical toponym-matching datasets. For example, Hastings et al. [14] align place names in English from recent USA history. The few examples we are aware of for cross-lingual historical toponym matching efforts are for places extracted from Dutch-German shipping maps ([15]) and the comprehensive World Historical Gazetteer (WHG) [2], which collects toponym data from many different scholars. With respect to entity alignment, WHG allows the user to perform exact match orthographic comparisons between a dataset's toponym labels and external sources such as Wikidata and GeoNames. However, datasets in the WHG are not matched with each other, and thus do not enable scholars to enrich each other with information from sources of other languages and cultural backgrounds. Here, we present a benchmark enabling cross-cultural toponym matching and evaluation of direct cross-lingual methods.

3 A Cross-lingual Semitic Toponym Matching Benchmark

In this section, we present our benchmark, comprising multiple datasets matched against each other in pairs. We shortly describe each dataset and comment on the prospect of finding matches between them.

We utilized four distinct datasets, two in Arabic and two in Hebrew. The initial Arabic dataset is a structured adaptation of Kitāb Mu'jam al-Buldān (The Countries Dictionary Book)[16]. This lexical masterpiece, authored in the early 13th century by the Muslim geographer Yāqūt al-Ḥamawī, provides a comprehensive account of various locales, predominantly within the Muslim world. The content encompasses diverse elements such as types of places, administrative hierarchies, geographical representations, and historical events. In a previous study, we employed a rule-based extraction approach to construct a systematically organized database containing many descriptive entities related to these places. Given the extensive nature of the lexicon, we opted to categorize the places based on geographical regions for more manageable analysis and presentation. The second Arabic dataset is sourced from the al-Ṯurayyā Project [17]. Extracted from the "Atlas Du Monde Arabo-Islamique a I'Epoque Classique: IXe-Xe siècles" (The Arab-Islamic World and Classic Europe Atlas: 9th-10th

Centuries) by Georgette Cornu[18], this database comprises over 2,000 toponyms. The dataset includes place names presented in both Arabic and Latin letters, accompanied by information on place types and, at times, a textual description of the location. The authors of the dataset employed a distinctive transliteration system characterized by a "one-to-one letter representation, with every Arabic letter transcribed distinctively." [17]. In keeping with our methodology applied to Yaqut's gazetteer, we parsed the al-Ṯurayyā dataset based on regional categorizations that align with our existing datasets. The first of the two Hebrew datasets is a list of Hebrew place names from the Medieval book of Travels of Benjamin of Tudela, composed in the 1170s. The text was digitized and annotated, and the places were geo-referenced for the building of the TraveLab tri-lingual digital edition `TraveLab` project [19]. As the main Hebrew text, the 1840 Hebrew print edition by Adoph Asher[20] was used, but variants were added from the aligned editions of both Arabic and English Translations and other Hebrew manuscript and print witnesses, which present a dynamically changing tradition of place naming. Though small (306 place entities), this dataset is closer in time to the Medieval modern sources, and though it does entail places from a part of Benjamin's route in areas outside the Muslim world, there is still much overlap with the rest of his route. The second, much larger Hebrew collection is `Kima` project[1], a gazetteer for historical and contemporary place names in languages written in the Hebrew script. The gazetteer provides a stable and shared reference for linking place names in digitized resources and documents and, where possible, attestation for the uses in the toponyms. Being mainly based on Hebrew print, its sources are predominantly early modern and modern. From Kima, we selected the toponyms from modern Spain, the countries of North Africa, and the Middle East that would be relevant to match the Arabic datasets.

Table 1 Dataset Pairings and Matches

Pairing	Dataset 1 (size)	Dataset 2 (size)	#Matches
1	Yaqut - Andalus/Magreb (484)	Kima - Andalus/Magreb (559)	28
2	Yaqut Al-Sham (687)	Kima - Al-Sham (1899)	30
3	Kima - Al-Sham (1899)	Thuraya - Al-Sham (291)	21
4	Tudela (306)	Althurayya (2241)	18
5	Damast (447)	Tudela (306)	32

Table 2 The five dataset pairs used in the provided benchmark and the number of matches found between each pair.

Table 2 summarizes the pairs of datasets compared in this benchmark. For each dataset, the number of places is mentioned in parentheses. for example, the Tudela dataset contains 306 places. Some of the datasets are regional subsets of the original dataset. For example, when comparing Yāqūt and Kima in dataset pair number two, we do this within the region of Al-Sham (The Levant, greater Syria), where these datasets contain 687 and 1899 places, respectively. We identified 30 exactly matched places out of 1,304,613 possible matches. We also compared Yāqūt and Kima datasets

[1] https://data.geo-kima.org/

within the regions of Andalusia and Maghreb, resulting in 28 matched locations. We compiled the exact matches by running our tool using various combinations of its matching algorithms (see Section 4) and different thresholds and manually reviewing the identified matches. Verification was done by scholars familiar with the respective datasets. However, we do acknowledge that some of the possible matches may have been overlooked as it is not feasible to manually review over a million possible combinations of places. We invite future users of this benchmark to submit matched pairs that we may have overlooked.

4 Toponym Matching Methods

In this section, we detail our toponym matching methods, developed for direct comparison between closely related languages. We first detail our transliteration-based method (Section 4.1 followed by our phoneme-based method (Section 4.2.

4.1 Transliteration

It is common practice, in toponym matching, to transliterate or transcribe both toponym labels to Latin/English letters. The resulting labels are often referred to as *Romanized* labels (e.g., [4]). Some of these transliteration/transcription methods may supplant standard Latin script with diacritics to signify letters and sounds that do not exist in English. For example, many methods (e.g., [21], [22]) use an accented ī to signify the Arabic letter ي (Ya). This may cause issues for edit-distance metrics such as Levenshtein [23] which consider this letter to be completely different from the regular *i*, which is used to transliterate the comparable Hebrew letter י (Yud). Therefore, we created a direct transliteration library that transcribes Arabic and Hebrew scripts using a rule-based linguistic-aware algorithm. The library is available as open-source code [24] and as a python package [25]. The library employs simple one-to-one translation tables where no extra rules are needed and more elaborate rule-based transcriptions where the transliteration differs according to letter position and other contexts. For example, the Hebrew letter ה (Heh) would normally be transliterated to ه (Ta), but if it is at the end of the word, to ة (Ta Marbuta). The library also generates multiple variants when the transcription has unsolvable ambiguity. For example, a ד (daled) letter in Hebrew may be transliterated into both د (dāl) and ض (ḍād). The user may subsequently choose which variant to use. We arbitrarily chose the first generated variant in this work for consistent experimental results. In the experiments performed, we employ the Jaro [26] and Levenshtein [23] distance metrics, defined as follows.

Definition 1 (Jaro Distance Metric)**.** *Given two strings s_1 and s_2, the Jaro similarity score, $J(s_1, s_2)$, is calculated using the following formula.*

$$J(s_1, s_2) = \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right)$$

Where m is the number of matching characters (two characters from s_1 and s_2 are considered matching if they are the same and not farther apart than

$\max(len(s_1), len(s_2))/2 - 1$*), t is half the number of transpositions (a transposition occurs when the same character appears in both strings but in different orders), $|s_1|$ and $|s_2|$ are the lengths of the strings s_1 and s_2, respectively. The Jaro distance is then defined as $1 - J(s_1, s_2)$.*

Definition 2 (Levenshtein Distance Metric)**.** *The Levenshtein distance between two strings, a and b, is defined as the minimum number of single-character edits (insertions, deletions, or substitutions) required to change one string into the other. Formally, the distance $L(a, b)$ can be defined recursively as follows:*

$$L(a,b) = \begin{cases} |a| & \text{if } |b| = 0, \\ |b| & \text{if } |a| = 0, \\ L(tail(a), tail(b)) & \text{if } head(a) = head(b), \\ 1 + \min \begin{cases} L(tail(a), b), \\ L(a, tail(b)), \\ L(tail(a), tail(b)) \end{cases} & \text{otherwise.} \end{cases}$$

Here, $|a|$ and $|b|$ denote the lengths of strings a and b, respectively; $tail(\mathcal{S})$ denotes the string $\mathcal{S}$ without its first character, and $head(\mathcal{S})$ denotes the first character of string $\mathcal{S}$.

4.2 Grapheme-to-Phoneme Conversion

We employ Grapheme-to-Phoneme (G2P) conversion and subsequently apply phonetic distance measures as an alternative approach to cross-lingual toponym matching. This allows to take into consideration the fact that phonemes, i.e., speech sounds (e.g., consonants and vowels) do not differ from each other to the same extent. Contemporary phonological theory characterizes phonemes based on phonological features, including phonation, airstream mechanism, and manner and place of articulation [27]. When language change distances the phonology of otherwise related languages like Arabic and Hebrew from each other, it is more common for only a subset of phonological features to diverge. G2P and phonetic distance measures allow a more accurate and fine-grained comparison between toponyms than simple transliteration.

We use the `g2ps` library[2], part of the `LanguageNet` project[3], as our G2P conversion tool. This is due to its coverage in our target languages of English, Hebrew and Levantine Arabic compared to other G2P tools, which are not compatible with all these languages or do not provide truly phonemic transcription[4]. Training lexicons provided by `g2ps` can be used to train `Phonetisaurus`[5], a weighted finite-state transducer on mappings between graphemes and corresponding phonemes [28]. We augment this transducer with a series of sensible post-processing rules defined to enable more accurate transcriptions. We examine the contribution of these rules to the matching accuracy in Section 5.

[2]https://github.com/uiuc-sst/g2ps
[3]http://www.isle.illinois.edu/sst/research/darpa2015/index.html
[4]See https://github.com/topics/grapheme-to-phoneme
[5]https://github.com/AdolfVonKleist/Phonetisaurus

We compare the resulting phonemes using the distance module of the PanPhon library [29][6]. The library converts input phonemes (represented using IPA symbols) into two possible representations: Dolgopolsky primes [30] and phonological feature vectors. It also contains various distance measures based on edit distance, i.e., the number of edit operations required to transform one string to another. Dolgopolsky primes are groupings of individual phonemes into distinct sound classes with categories representing labial obstruents (e.g. /p/ and /b/), laryngeals (e.g. /h/ and /ɦ/), and vowels (e.g. /e/ and /ə/). Once the phonemes are assigned to their corresponding prime, the resulting sequences can be compared using Levenshtein distance. The drawback of Dolgopolsky primes is that they assign phonemes into mutually exclusive categories, thus not directly comparing phonological features. There is, for instance, a separate category in Dolgopolsky primes for coronal fricatives, coronal affricates, and other coronal obstruents, sounds that are obviously similar in their manner of articulation.

On the other hand, phonological feature vectors represent phonemes directly using abstract phonological features linguists break them down to. These may refer to articulatory features, i.e., how a sound is formed, including categories such as *lateral*, *nasal*, *voiced*, or *labial* among all. Additional phonological features express the *consonantal* nature of a sound, or its *syllabicity*[7]. Features values are encoded as either positive, negative, or unspecified. All IPA symbols can be broken down into these feature vectors, and the library contains various distance measures that can operate on these vectors, including *feature edit distance*, *Hamming feature edit distance*, and *weighted feature edit distance*. These distance measures differ in the associated cost of feature edits. The first two assign fixed costs to changing any features, while weighted feature edit distance considers that certain sound changes are more common cross-linguistically than others. For instance, voicing and devoicing a consonant from /p/ to /b/ or /g/ to /k/ is extremely common, as is the lengthening and shortening of vowels. The low cost of changing these phonological feature values reflects this. On the other hand, making a continuant, like an /s/ sound, into a non-continuant, such as /t/ or vice versa, has a significantly higher cost, modeling how unlikely this sound change is.

Toponym matching across related languages, such as Hebrew and Arabic, might benefit greatly from considering how close the individual sounds to each other are phonologically. When using an appropriate similarity threshold, we hypothesize that phonologically-based toponym matching might improve on merely string-based approaches.

5 Experiments

In the following Section, we perform two experiments. In the first experiment (Section 5.1). We empirically evaluate the performance of our transliteration and phonetic methods over our benchmark. We perform toponym matching and report

[6]https://github.com/dmort27/panphon

[7]The full range of phonological features included in PanPhon can be found at https://github.com/dmort27/panphon/tree/master#ipa-character-databases-ipa__basescsv-and-ipa__allcsv.

on the performance of our transliteration-based and phonetic-based methods on this benchmark.

In a second experiment (Section 5.2], we perform a more detailed phonetic and orthographic distance analysis. We do so by comparing the transliteration-based and phonetic-based metrics with state-of-the-art string-based metrics over a large and diverse dataset, namely - Wikidata toponyms in modern variants of Arabic and Hebrew. We perform the comparison using different distance metrics.

5.1 Toponym Matching over Historical Dataset Pairs

5.1.1 Setup and Metrics

Toponym matching experiments were performed using our prototypical MEHDIE[31] toponym matching tool. Normally, MEHDIE employs both the transliteration-based and the phonetic methods. However, we used the configuration options to run experiments with only one of the methods at a time. Configuration settings were varied to examine the sensitivity of the results to different threshold settings.

The MEHDIE tool allows matching a given dataset to another dataset by performing parallelized pair-wise comparisons between toponyms distributed over instances of the matching system, which are spun up by demand by Google Cloud Run infrastructure.

The tool first performs a preprocessing step on both datasets where toponyms (for example, in Hebrew) and their variants, as supplied with the dataset, are transliterated to the opposite language (for example, Arabic) and Romanized. The tool then performs a blocking step to reduce the number of comparisons, which is common in large-scale matching problems [32]. However, general-purpose blocking methods rely on the linguistic similarity of the matched entities, which is difficult to use in our case, where the information is limited to the entity's name, which is often linguistically opaque, and the entity's location. The tool, therefore, employs a simple clustering step where every name is assigned to groups according to the existence of one of a set of similar-sounding letters in the name. For example, all names from both datasets with at least one of the letters *p,b,f* would be grouped together and eventually matched. The assumption is that if the pair of toponyms do not share any related letter in the Romanized version, they are not relevant to each other. The blocking step reduces the number of comparisons, on average, by half.

Following the blocking step, the tool performs a pairwise comparison over transliterated variants of the toponym from the same script language (Hebrew / Arabic) using the Jaro metric. Results over the given thresholds continue to the next step, which is a phonetic comparison using the Hamming feature distance metric over the IPA representation of the toponym. Finally, if location information is available for both matched toponyms, we filter out those pairs whose distance exceeds the given threshold. Phonetic threshold values are varied between *[0.85, 0.9, 0.95]*, orthographic thresholds between *[0.7, 0.8, 0.9]*, and distance thresholds between *[10, 50, 200]*.

Results are provided in terms of *Precision*, *Recall*, *F-1*, and *F-5* defined as follows. Precision is the proportion of true positives of the number of matched pairs the tool returns. Recall is the proportion of true positives of the number of matched pairs known to be in this dataset pair. F-1 is the harmonic mean of these two measures, penalizing imbalanced results (e.g., low recall and high precision) more than the standard mean. F-5 is a recall-favoring version of the F-measure. The equations of the general F-measure, F-1 and F-5, are given below with P and R designating *Precision* and *Recall*, respectively.

- **F-β Score** provides a means to balance precision and recall in a single metric, using a weighted harmonic mean. It is expressed as:

$$F_\beta = (1+\beta^2) \cdot \frac{P \cdot R}{(\beta^2 \cdot P) + R} \tag{1}$$

  The β parameter determines the weight of recall in this balance, with values greater than 1 favoring recall, thus prioritizing the identification of all actual matches.
- **F-1 Score** is the harmonic mean of precision and recall with equal importance, suitable for scenarios where both metrics are equally valued:

$$F_1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{2}$$

- **F-5 Score** emphasizes recall five times more than precision by setting $\beta = 5$. This is particularly useful in our case, where the discovery of new matches is highly sought after, and we are willing to pay for it by reviewing more (mostly false) results:

$$F_5 = (1+5^2) \cdot \frac{P \cdot R}{(5^2 \cdot P) + R} \tag{3}$$

5.1.2 Results

The complete set of results is provided as an electronic supplement to this paper. In the following, we present some of the main findings. Figure 1 presents the different metric values for different threshold settings applied to the two methods. As expected, an increasing threshold improves precision at the expense of recall. The phonetic method seems to be severely impacted by lowering the threshold. A reduction of 0.1 (from 0.95 to 0.85) in the phonetic threshold improves recall by 0.3 and reduces precision by more than half. The transliteration-based method, by comparison, gains a 0.06-0.14 in recall and suffers only a 0.07-0.024 reduction in precision from the same reduction. Regarding F-1, the low precision values coupled with high recall rates cause the F-1 measure to be relatively low in both methods at 0.1-0.32 and to be maximized in high-threshold scenarios where the precision is maximized. Optimal F-5 values are obtained for a threshold of 0.9 in both the orthographic and the phonetic methods.

Figure 2 compares the impact of different distance thresholds on the different dataset pairs. The figure presents results in terms of the maximal F-5 result obtained across all threshold settings and comparison methods. Results indicate that applying a more restrictive distance threshold generally reduces performance. This emanates from the reduced recall, which our users value. This general trend does not hold in the case of Kima-Thurayya, where applying a restrictive threshold improves the F-5

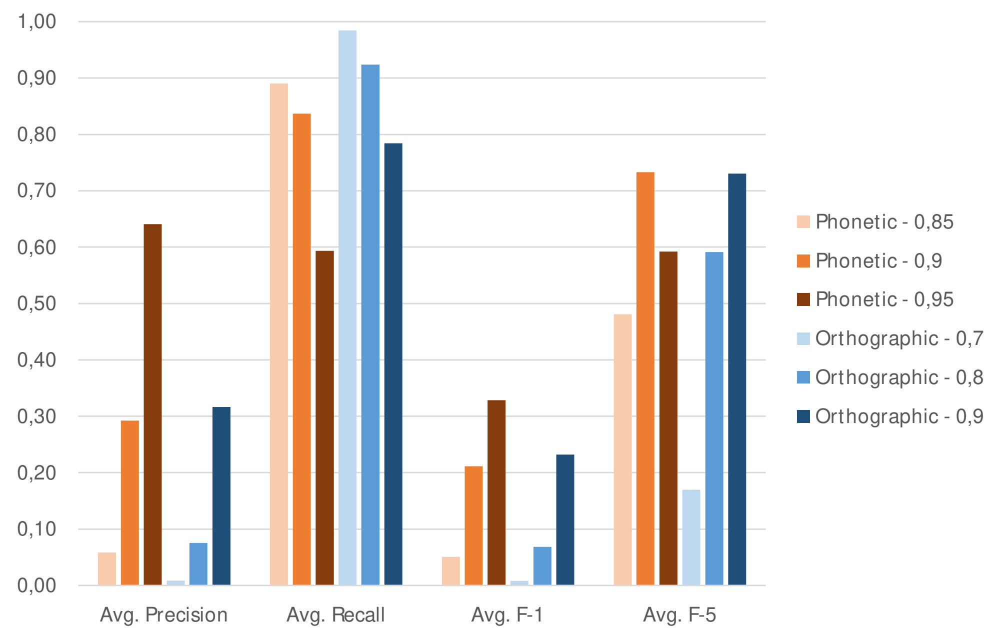

Figure 1 Comparison of different threshold values for both methods.

score somewhat, owing to the very low precision caused by the more lenient distance thresholds.

In Figure 3, we compare the two matching methods across the five dataset pairs in F-5 terms, reporting the maximal result overall threshold and distance threshold settings. Results are mixed, with dataset pairs involving the Tudela dataset showing better performance for the orthographic method, while in the other dataset pairs, the phonetic method is superior.

5.1.3 Discussion

As F-5 better represents users' preferences in our scenario, it is interesting to see that the best F-5 is obtained at a threshold of 0.9 for both the orthographic and phonetic methods. Combined with the inconclusive comparison results over different dataset pairs, this result indicates that users should opt to use both methods in combination to maximize the number of matches found. The fact that restrictive distance thresholds harm performance can partially be attributed to our choice of the F-5 metric owing to our users' preference for high recall and relative tolerance of low precision. However, it can also be traced to the fact that location data for historical sources is inaccurate. In some cases, the historical place has a location that is accurately mapped to research-based evidence of settlement. Still, in others, it may be approximated using reported travel times from other locations or using inaccurate historical maps [33]. Although it is common for place names in the ancient Middle

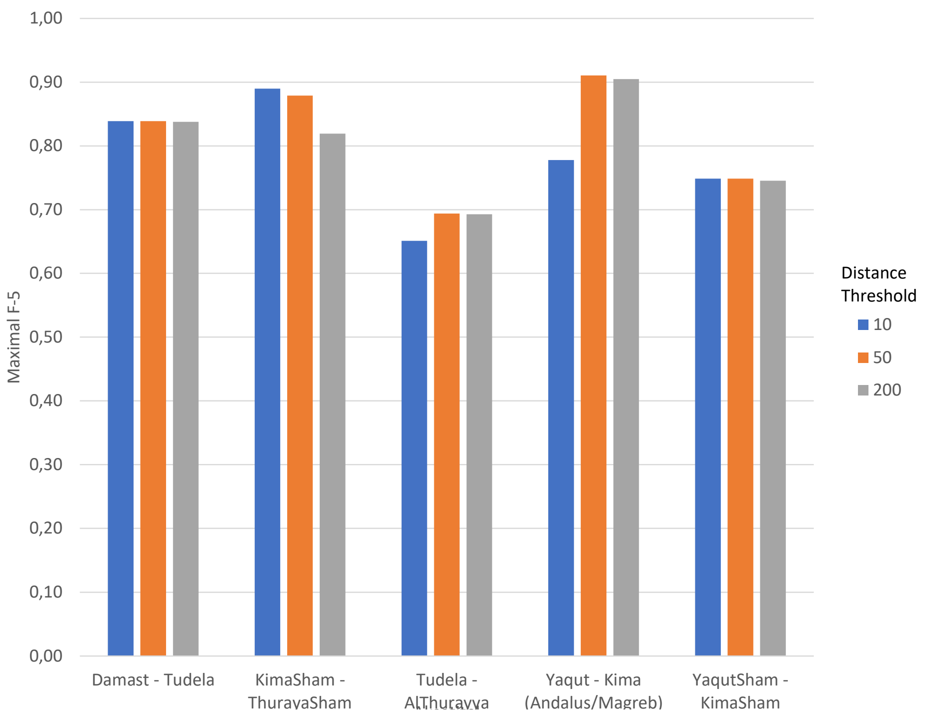

Figure 2 Best result in terms of F-5 by matched dataset pair and distance threshold.

East to be repeated (e.g., الطيرة in Israel[8], the Palestinian Authority[9], and Syria[10]), it seems the benefits of using a more lenient distance threshold outweigh this risk. The phonetic similarity tool between Semitic languages proved more efficient in places like Tiberias in the Yāqūt and Kima in al-Sham that are phonetically closer in Hebrew (טבריה-Tveria) and Arabic (طبرية-Tabaria) than in an orthographic comparison of the Romanized versions. In another example Gaza (غزة-Ghaza in Arabic and עזה-'Aza in Hebrew) were matched based on two criteria: phonetic title similarity by confidence 0.97222 and exact geographic similarity [34.45, 31.516]. The phonetic tool adeptly disregards grammatical nuances such as the Arabic definite article "ال", leading to enhanced statistical accuracy. Consequently, it yields high matches in locations where "ال" appears in Arabic but not in Hebrew, as exemplified by Zarqa (الزرقاء-זרקא) and Ramla (الرملة-רמלה). Nevertheless, the transliteration-based matching tool was also able to identify pairs that the phonetic tool missed. For example, the city of Zaragoza in Andalusia (Spain) is spelled by Yāqūt (سرقسطة-Saraqustah) and by Kima's gazetteer (סרגוסה-Saragosa). Here, the phonetic similarity is rather distant. However, since the orthographic comparison is much more computationally efficient, we can use it over multiple name variants supplied by the sources and take the maximal similarity score.

[8]https://www.wikidata.org/wiki/Q167594
[9]https://www.wikidata.org/wiki/Q12189847
[10]https://www.wikidata.org/wiki/Q12189848

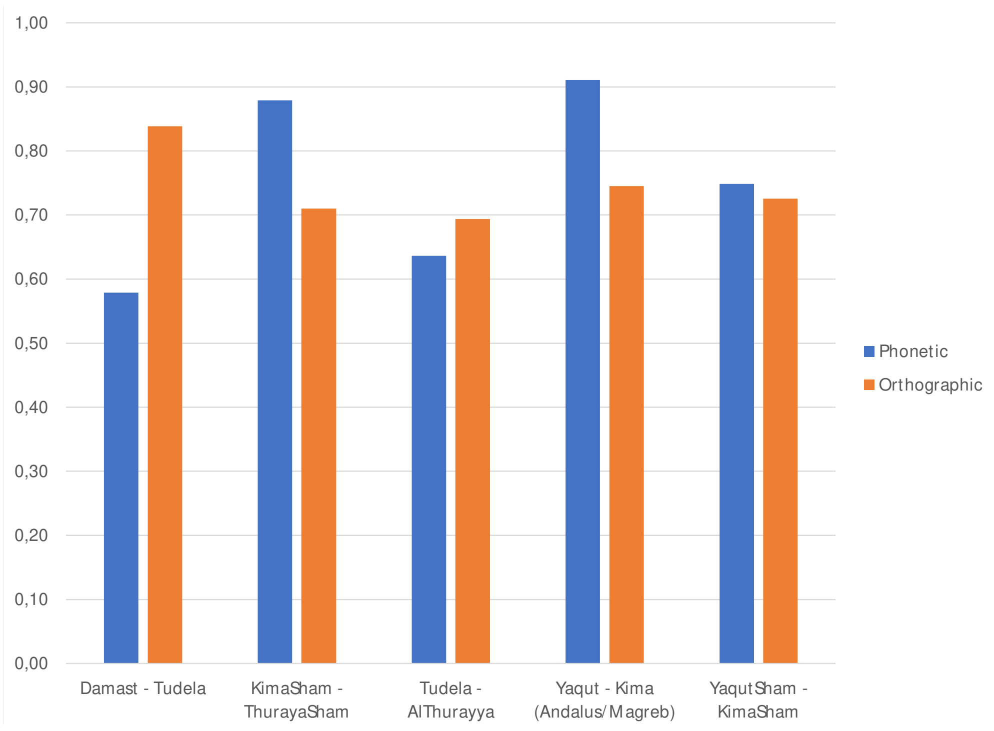

Figure 3 Best result in terms of F-5 by matched dataset pair and method.

In this case, the algorithm found in one of the variants that existed in Kima originally סרקסטה (Sarakastah) whose direct transliteration to Arabic orthographically matches that of Yāqūt.

5.2 Comparing Orthographic Distances between modern Toponym names in Hebrew and Arabic

5.2.1 Dataset Description

We queried wikidata using the SPARQL query depicted in Listing 1 to construct our test case. The query uses the *wdt:p625* location property to identify concepts that may be toponyms. These are then filtered for those concepts that have a label in both Arabic and Hebrew. The query was run over Wikidata using the qlever query engine [34] and resulted in 41560 toponym label pairs[11].

Listing 1 "SPARQL Query to fetch Hebrew and Arabic labels"

```
SELECT ?place ?placeLabel_he ?placeLabel_ar WHERE {
  ?place wdt:P625 ?location . # P625 is the property for coordinate
      location
```

[11] https://qlever.cs.uni-freiburg.de/wikidata, retrieved November 9th 2023

ID	Hebrew	HE IPA	HE Translit AR	Arabic	AR IPA	AR Translit HE
1015647	קלרקסדורפ	klrksdoʁf	قلرقسدورف	كليركسدورب	kljrksdu:rb	כּלירכּסדורבּ
1015654	סנטו דומינגו	snto dominŋv	سنطو دومينچو	سانتو دومينغو	sa:ntu: dwme:nɣw	סאנתו דומינע'ו
1015672	וואנגנוי	vʔŋnvi	وانچنوي	وانجانوي	awa:ndʒa:nwj	ואנג'אנוי
1015699	לירה	liʁa	ليرة	ليرا	ljra:	לירא
1015727	מבאלה	mvʔla	مبالة	مبالي	mba:lj	מבּאלי
1015773	פורירואה	foriroʔa	فوريرواة	بوريريوا	bo:ri:re:awa:	בּוריריוא
1015796	ויקטור הארבור	wiqtor hʔrbor	ويقطور هاربور	فيكتور هاربور	fi:ktu:r ha:rbwr	פיכּתור הארבּור
1015805	צ'אנגשו	tsʔŋʃo	ضانچشو	تشانغشو	tʃa:nɣʃaw	תשאנע'שו
101583	תבאי	tbʔi	تباي	طيبة	tˁe:ba	טיבּה
1015916	הדיבו	hdivo	هديبو	حديبو	ħdajbu:	חדיבּו

Figure 4 Result of preprocessed Wikipedia dataset

```
  ?place rdfs:label ?placeLabel_he FILTER(LANG(?placeLabel_he) = "he
      ") .
  ?place rdfs:label ?placeLabel_ar FILTER(LANG(?placeLabel_ar) = "ar
      ") .
}
```

We then perform several pre-processing steps. The processing code, as well as both the raw and processed files, are available as an electronic appendix to this paper. We begin by removing all places for which either the Hebrew or the Arabic label comprises more than two words. This is done to filter out locations that are not real places but locations of events or businesses. Such as the *Academy of Social Sciences of the Central Committee of CPSU*[12] or *Air France Flight 296*[13]. Although this method may sound restrictive, one should remember that prepositions in Hebrew and determiners in both languages are prefixed to the word. For example, *the Jordan river* is in Hebrew (transliterated to Latin characters) *Nahar **Ha**yarden* where *Ha* is the equivalent to *the* and in Arabic it is *Naher **Al**urdun* where *Al* is the determiner. Thus, in both Hebrew and Arabic, this step will not remove many actual Toponyms but is very helpful to avoid skewing the results by measuring distances between long sentences describing events and businesses that are not common in our benchmark datasets that contain names of places. Furthermore, these sentences are better compared with translation as they contain common nouns and adjectives rather than proper nouns.

We remove language tags (e.g., '@ar') from the toponyms and then perform transliteration of each toponym to its opposite language using the specialized middle-east language transliteration package *translit-me*[14]. The original Arabic and Hebrew labels are then converted into an IPA phonetic representation. Figure 4 shows an excerpt of the result.

[12] https://www.wikidata.org/wiki/Q4059245
[13] https://www.wikidata.org/wiki/Q406486
[14] https://pypi.org/project/translit-me/, version 1.0.4, retrieved October 30th, 2023

5.2.2 Metrics

As a baseline comparison, We use string comparison metrics for toponym matching from those compared by Recchia and Louwerse [4]. The authors compared 21 string comparison methods over datasets containing Romanized toponyms from 11 countries. We chose the top three best-performing metrics the authors identified for Arabic Toponyms (Saudi Arabia datasets) as they are the closest linguistic and geographic to our datasets. We compare the performance of *Skip-grams*, *Editex*, and *Syllable* over the Romanized versions of both Hebrew and Arabic labels. Using standard Jaro and Levenshtein distance metrics, we also compare strings in the same character set, i.e., Hebrew or Arabic, where one is the original label and the other is transliterated. The following are the definitions of these metrics based on [4]. We employ the methods as implemented by a python port[15] of the FEBRL package [35].

Skip-grams

We use the method first proposed by Keskustalo et al. [36] where common bigrams of skip size one and two are counted and divided by the total number of these bigrams in the shorter place name.

Smith-Waterman

This method [37] employs a variation of Levenshtein distance where letters of the same sound group, e.g. (aeiou, bpv, dt, etc.) have a lower edit distance than if they belong to different letter groups. This method resembles our phonetic method but is more crude as it groups together many unrelated letters and is based on the English language pronunciation of these characters.

Syllable

The syllable alignment algorithm [38] analyzes syllables instead of individual characters. It transforms characters in place names into groups, forming sequences that represent syllables. The algorithm then calculates the least costly way to change one sequence into another, using a set of weighted operations on both character groups and syllables.

Phonological distance measures

In order to carry out toponym matching based on phonological similarity, we convert candidate toponyms to phonological vectors, where each element is a phonological feature of the sound (e.g., labial). We then use various distance measures in the `PanPhon` library [29] to compare them as described in Section 4.2. All of these distance measures are based on the concept of edit distance, i.e., the number of substitutions required to transform one toponym into another. In the following, we describe the fundamental categories.

For the *feature edit distance*, every substitution has an associated *cost* of 1/24: i.e., normalized by the total number of phonetic features. The same cost is associated with any insertion or deletion. In cases where the edit goes from an unspecified to a

[15] https://github.com/thomaswyrick/febrl, retrieved November 12th, 2023

specified phonological feature, the edit cost is half of the above. This is necessary since certain features only make sense in the context of other features. Obstruents like /t/, for instance, cannot be specified for the feature strident, but once they change into a fricative such as /θ/, they have to become specified. Since this involves additional feature changes anyway – from -continuant to +continuant –, this prevents inflating edit costs.

$$d_{feature}(\text{ins, del, sub}) = \frac{1}{||\mathbf{f}||}(\sum_i \sum_j \frac{1}{2}||\mathbf{s}_i - \mathbf{t}_j||) \tag{4}$$

The cost of all edit operations between two phonemes is captured in Equation 4. Vectors $\mathbf{s}_i$ and $\mathbf{t}_j$ stand for the i-th and j-th members of the phonological vectors $\mathbf{s}$ source phoneme and $\mathbf{t}$ target phoneme. The vector $\mathbf{f}$ represents all possible feature values, and its magnitude is 24. Since 1, -1, and 0 represent positive, negative, and unspecified values for each feature, the normalizer term $\frac{1}{2}$ ensures that the cost of changing from specified to unspecified and vice versa is less than changes between specified values. Again, this prevents the inflation of edit costs when certain feature changes are necessarily accompanied by other feature changes.

In the case of *Hamming feature edit distance*, insertions and deletions cost 1 without normalization, while feature edits and edit from unspecified to specified all cost 1/24 (see Equation 5).

$$\text{Difference} = \begin{cases} 1 & \text{if } \mathbf{s}_i \neq \mathbf{t}_i \\ 0 & \text{if } \mathbf{s}_i = \mathbf{t}_i \end{cases}$$
$$d_{hamming}(\text{sub}) = \frac{1}{||\mathbf{f}||}(\sum_i \text{Difference}(\mathbf{s}_i, \mathbf{t}_i)) \tag{5}$$

Finally, *weighted feature edit distance* takes into account the probability of the kinds of feature edits determined by how likely these are cross-linguistically, but does not carry out normalization. Cross-linguistically frequent feature changes, such as changes in voicing or vowel length, have an associated cost of 0.125, less common changes come with a cost of 0.25 or 0.5, whereas rarer ones such as change in syllabicity have a cost of 1.

See Table 3 for a comparison between the categories.

Edit distance	**Substitution (specified)**	**Substitution (unspecified)**	**Insertion/deletion**
Feature	1/24	1/48	1/24
Hamming feature	1/24	1/24	1
Weighted feature	1/8, 1/4, 1/2 or 1*	1/8, 1/4, 1/2 or 1*	1/8, 1/4, 1/2 or 1*

Table 3 Comparison of the costs associated with the edit distances defined in the `PanPhon` library. The asterisk marks that the exact edit cost depends on the associated features.

5.2.3 Results

Figure 5 and Figure 6 present the box plots and density plots of the different metrics over the wikidata datasets. *ar*-prefixed metrics are measured between the original

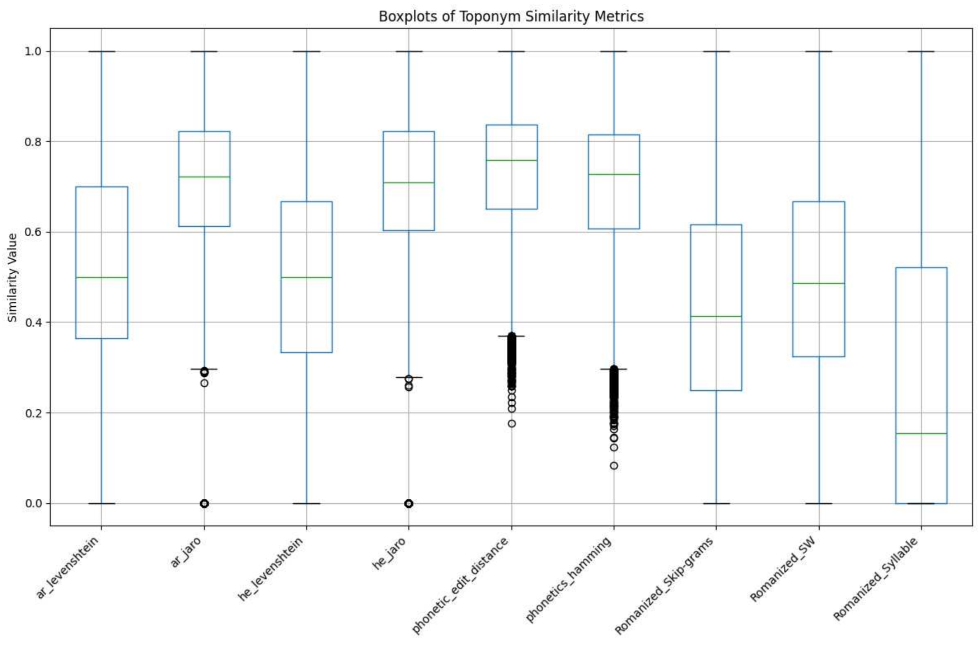

Figure 5 Box plots of string similarity and phonetic similarity Metrics over Wikidata toponyms.

Arabic place name and the Hebrew place name that was transliterated into Arabic. Similarly, *he*-prefixed metrics compare between the original Hebrew toponym and the one translitered into Hebrew. *Romanized*-prefixed metrics compare between the romanized versions of both labels (i.e., romanized Hebrew and romanized Arabic). Metrics prefixed with *phonetic* are calculated over the IPA representations of the toponyms. In general, a result closer to one is more desirable. Thus the perfect distribution in Figure 5 would be a dot on the 1.0 horizontal line, and in Figure 6, it would be a vertical line at 1.0. Table 4 summarizes the information shown in the box plots in numerical form. In bold, we https://www.overleaf.com/project/643e50c261715e99c1f76c9bhighlight the highest value for the quartile, mean, and minimum values which is consistently that of the phonetic feature edit distance metric. A close second and third best are the two Jaro metrics directly comparing Hebrew and Arabic labels with their transliterated counterparts with very similar performance.

5.2.4 Discussion

Both the direct transliteration (Hebrew ↔ Arabic) with Jaro and the IPA phonetic algorithms perform considerably better than the distance metrics applied over the Romanized toponyms. Jaro is somewhat hindered by the limitations of the Jaro algorithm, which handles very short strings, and the limitations of the dataset, which contains many non-Middle-eastern names that require various apostrophes and forced

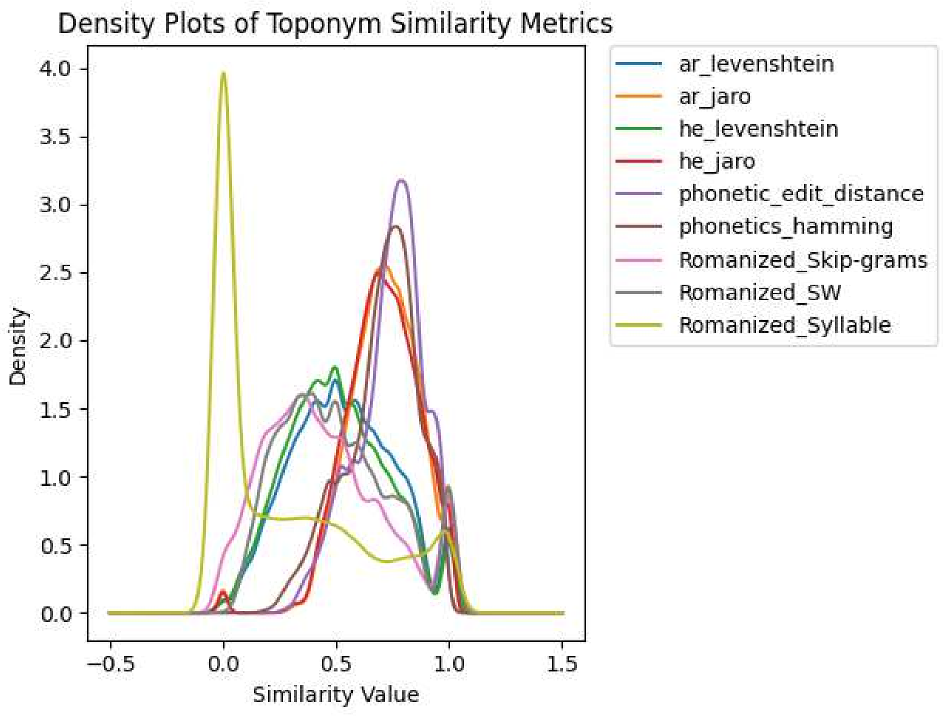

Figure 6 Density plots of string similarity and phonetic similarity Metrics over Wikidata toponyms.

Metric	AR Ln	HE Ln	AR Jaro	HE Jaro	Skip-G	SW	Syllable	ϕ ED	ϕ Hg
mean	0.53	0.51	0.71	0.71	0.45	0.50	0.29	**0.74**	0.70
std	0.23	0.22	0.16	0.16	0.26	0.24	0.33	0.15	0.16
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	**0.18**	0.08
25%	0.36	0.33	0.61	0.60	0.25	0.33	0.00	**0.65**	0.61
50%	0.50	0.50	0.72	0.71	0.41	0.49	0.15	**0.76**	0.73
75%	0.70	0.67	0.82	0.82	0.62	0.67	0.52	**0.84**	0.81
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 4 Descriptive statistics of Similarity metrics over the wikidata dataset. Abbreviations used - Ln: Levenstein Distance, Skip-G: skip Grams, SW: Smith-Waterman, ϕ: Phonetic, ED: Edit Distance, Hg: Hamming Distance

letter pairs. For example, the Chinese city of Changshu[16] (Figure 4 third to last row) whose first sound is transliterated by the editors of wikidata using the available letters in Arabic and Hebrew to a T+Sh construct. These letter combinations cause the match distance to grow artificially in replacement-based algorithms such as Jaro. Interestingly, the Levenstein distance metric performs poorly in both languages.

[16] https://www.wikidata.org/wiki/Q1015805

We inspected the cases with the largest gaps between the phonetic and the Jaro distance metric. All of the results could be explained by one or more of the following reasons.

Several cases reveal actual errors in Wikidata (as, for example, the case of "Bethlehm, Georgia"(Q3261160) which was erroneously named 'Georgia' in Hebrew (we corrected it on Dec.19, 2023) or "Jaffa port"(Q2911475) which erroneously contained the Hebrew word for Israel in the field of the Arabic variant (corrected Dec.29 2023). The distance criterion even tracked cases of Wikidata vandalism, as in the case of the town Yeruham (Q1687708), that in early November 2023 was renamed (Palestine), a renaming that was reverted since.

In many other cases, the names given in the different languages are in fact different names for the same place. Two examples are revealing: the island of Imbros (wikidata entity Q658437) bears the same name (إمبروس) in Arabic as in English and other Wikipedia languages, which chose to follow the older Greek name Ἴμβρος. The Hebrew name גקצ'אדה, however, follows the official current Turkish name of the island Gökçeada since 1970. Another example is Wikidata entity Q581478: The Arabic name is derived from the Czech name, Břeclav, which in turn is named after the founder of the local castle, Duke Bretislav I. The Hebrew form preserves the former German name which was probably derived from the name of a Slavic tribe which lived in the area. The example of Q826132 shows another known phenomenon in toponym history, in which a hyponym (a name that refers to a part of a certain region) like (Lagat/Laja,اللجاة) became synonymous with its hypernym (the name of the entire region), as in this case Trachon / טרכון). The Arabic and Hebrew choices of the name each corresponds with one of the options. Behind most of these cases lies the cultural and political history of place naming, which is occasionally reflected in the choice of Wikidata editors.

Another common reason for large reported differences between the Arabic and Hebrew names occurs when the root name is basically the same, but in one of the languages the name includes a type word (such as Wadi, Bridge, Mountain, Mound, River, cave) and not in the other. Thus while the 'Fengtai District' of Beijing (Q393831) is named simply فنغتاي in Arabic, the Hebrew name includes also the Hebrew word for 'district': רובע פנגטאי. The river Soča, on the other hand, is simply named סוצ׳ה in Hebrew while the Arabic includes the term for 'river', نهر سوكا.

While the choice to include the type word in the name seems random and occurs in both languages, a similar reason for large distance between the variants stems from linguistic reasons: often, the Arabic name of a place will include the determiner ال as an inherent part of the toponym. Thus the Iraqi city of Najaf (Q168193) is named simply נג׳ף) in Hebrew but النجف - "the Najaf" in the Arabic, and the determiner letters prefixed to the Arabic version cause the distance between matching letters to be too great for the Jaro algorithm to take into account. Here, we encounter a peculiar property of the Jaro metric where the allowable distance between matching letters is proportional to the length of the shortest comparison term. This may emanate from the fact that Jaro was developed for comparing paragraphs of text rather than single words. Thus, for a three-letter word like the Hebrew version of this toponym, the two-letter determiner prefix distances the matching Arabic letters too far away to be included in the Jaro calculation, resulting in a Jaro score of zero. For the phonetic versions of these two toponyms, this problem does not exist since we have implemented, as one of the post-processing rules, a rule that removes the determiner from the phoneme. Applying a similar rule for type names (river, county, etc.) could improve both metrics' performance and be justified in a toponym-focused matching system such as ours.

A third type of difference between the names in the two languages stems from the inclusion of Matres Lectionis. While both the Arabic and Hebrew scripts lack vowel letters in principle, they may occasionally use specific letters to indicate vowels. In Modern Hebrew, the adoption of full spelling means that the Hebrew name is more likely to include more letters, as the case of Kutum (Q3317346), for which the Hebrew name includes the five letters parallel to the English name, and the Arabic name only the three consonants k,t,m. While for string similarity metrics the distance of two letters, the phonetic distance metrics accords smaller weights to differences in vowel letters, and thus is more likely to match the two names.

The most frequent explanation for the differences between the phonetic and Jaro similarity metrics also exhibits the strength of the phonetic approach: the case of Pest (Q210205), the eastern part of Budapest, is a clear example: in both Hebrew and Arabic the name is written with three letters for the consonants, but though the names are in principle the same, neither letter can, however, be unequivocally transliterated to its parallel letter in the other language. The phonetic approach is sensitive to the closeness between the bilabial plosives p (פ) and b (ب), between the sibilant second letters and the Voiceless alveolar plosive third letters.

6 Conclusion

In this work, we have presented a first-of-its-kind benchmark for cross-language toponym matching between Hebrew and Arabic historical toponyms. The benchmark comprises five dataset pairs from four different gazetteers of the Middle East, Magreb, and Andalus areas. We have further presented and evaluated two novel approaches to this problem. Our transliteration-based approach and our phonetic-based approach were explained and evaluated, both on our benchmark task and on a large-scale synthetic task created using Wikidata, and compared to traditional methods relying on the romanization of toponyms and the use of edit-distance and n-gram-based string comparison methods. Results indicate that both approaches perform well on this task, especially when considering our potential users' preference for high recall and tolerance for low precision. In future research, we hope to explore the use of multi-lingual embeddings and other machine-learning-based methods on this benchmark and construct similar benchmarks for other language pairs that share a common historical and linguistic background. The benchmark and full results are provided as an electronic supplement to this work.

References

[1] Pleiades Team: Pleiades. https://pleiades.stoa.org/. Accessed: 01/01/2024

[2] Grossner, K., Mostern, R.: Linked places in world historical gazetteer. In: Proceedings of the 5th ACM SIGSPATIAL International Workshop on Geospatial Humanities. GeoHumanities '21, pp. 40–43. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3486187.3490203

[3] Thomas A. Carlson: HIMME. https://medievalmideast.org/index.html. Accessed: 01/01/2024

[4] Recchia, G., Louwerse, M.: A comparison of string similarity measures for toponym matching. In: Proceedings of The First ACM SIGSPATIAL International Workshop on Computational Models of Place. COMP '13, pp. 54–61. Association for Computing Machinery, New York, NY, USA (2013). https://doi.org/10.1145/2534848.2534850

[5] National Geospatial-Intelligence Agency: Geographic Names Server. https://geonames.nga.mil/geonames/GNSHome/index.html. Accessed: 01/01/2024

[6] Joshi, T., Joy, J., Kellner, T., Khurana, U., Kumaran, A., Sengar, V.: Crosslingual location search. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. SIGIR '08, pp. 211–218. ACM, New York, NY, USA (2008). https://doi.org/10.1145/1390334.1390372

[7] Sun, K., Zhu, Y., Song, J.: Progress and challenges on entity alignment of geographic knowledge bases. ISPRS International Journal of Geo-Information **8**(2) (2019) https://doi.org/10.3390/ijgi8020077

[8] Santos, R., Murrieta-Flores, P., Calado, P., Martins, B.: Toponym matching through deep neural networks. International Journal of Geographical Information Science **32**(2), 324–348 (2018)

[9] Santos, R., Murrieta-Flores, P., Martins, B.: Learning to combine multiple string similarity metrics for effective toponym matching. International journal of digital earth **11**(9), 913–938 (2018)

[10] Zhang, Z., Liu, H., Chen, J., Chen, X., Liu, B., Xiang, Y., Zheng, Y.: An industry evaluation of embedding-based entity alignment. In: Proceedings of the 28th International Conference on Computational Linguistics: Industry Track, pp. 179–189 (2020)

[11] Zhang, R., Trisedya, B.D., Li, M., Jiang, Y., Qi, J.: A benchmark and comprehensive survey on knowledge graph entity alignment via representation learning. The VLDB Journal **31**(5), 1143–1168 (2022)

[12] Sun, Z., Hu, W., Li, C.: Cross-lingual entity alignment via joint attribute-preserving embedding. In: d'Amato, C., Fernandez, M., Tamma, V., Lecue, F., Cudré-Mauroux, P., Sequeda, J., Lange, C., Heflin, J. (eds.) The Semantic Web

– ISWC 2017, pp. 628–644. Springer, Cham (2017)

[13] Martins, B.: A supervised machine learning approach for duplicate detection over gazetteer records. In: International Conference on GeoSpatial Sematics, pp. 34–51 (2011). Springer

[14] Hastings, J.: Automated conflation of digital gazetteer data. International Journal of Geographical Information Science **22**(10), 1109–1127 (2008)

[15] Ardanuy, M.C., Sporleder, C.: Toponym disambiguation in historical documents using semantic and geographic features. In: Proceedings of the 2nd International Conference on Digital Access to Textual Cultural Heritage. DATeCH2017, pp. 175–180. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3078081.3078099

[16] Yāqūt, al-Rūmī al-Hamawī: Kitāb Mu'jam al-Buldān (The Countries Dictionary Book). Dār Ṣādir, Beirut (1977). Original work published in the 13th Century. 5 vols.

[17] Maxim Romanov: al-Ṯurayyā Project. https://althurayya.github.io/. Accessed: 01/01/2024

[18] Cornu, G.: Atlas Du Monde Arabo-Islamique a I'Epoque Classique: IXe-Xe Siècles (The Arab-Islamic World and Classic Europe Atlas: 9th-10th Centuries). Brill, Leiden (1983)

[19] TravelLab: Benjamin of Tudela. Accessed: March 1st, 2024 (n.d.). https://teipublisher.info/exist/apps/TraveLab/Benjamin%20of%20Tudela.xml

[20] Benjamin, Asher, A., Zunz, L., Lebrecht, F.: The Itinerary of Rabbi Benjamin of Tudela. A. Asher & co

[21] Gibb, H.A.R.: The Encyclopaedia of Islam. Brill Archive, ??? (1998)

[22] Wehr, H.: A Dictionary of Modern Written Arabic. Otto Harrassowitz Verlag, Wiesbaden (1979)

[23] Levenshtein, V.I., *et al.*: Binary codes capable of correcting deletions, insertions, and reversals. In: Soviet Physics Doklady, vol. 10, pp. 707–710 (1966). Soviet Union

[24] MEHDIE Project: The translit-me source code repository. https://gitlab.com/m8417/hebrew-transliteration-service. Accessed: 01/01/2024

[25] MEHDIE Project: The translit-me Python Package. https://pypi.org/project/translit-me/. Accessed: 01/01/2024

[26] Jaro, M.A.: Advances in record-linkage methodology as applied to matching the

1985 census of tampa, florida. Journal of the American Statistical association, 414–420 (1989)

[27] Catford, J.C., Esling, J.H.: Articulatory phonetics. Encyclopedia of Language and Linguistics **9**, 425–442 (2006)

[28] Novak, J.R., Minematsu, N., Hirose, K.: Phonetisaurus: Exploring grapheme-to-phoneme conversion with joint n-gram models in the wfst framework. Natural Language Engineering **22**(6), 907–938 (2016)

[29] Mortensen, D.R., Littell, P., Bharadwaj, A., Goyal, K., Dyer, C., Levin, L.: Panphon: A resource for mapping ipa segments to articulatory feature vectors. In: Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pp. 3475–3484 (2016)

[30] Dolgopolsky, A.B.: A probabilistic hypothesis concerning the oldest relationships among the language families of northern eurasia. Typology, relationship and time: a collection of papers on language change and relationship by soviet linguists, 27–50 (1986)

[31] MEHDIE: The MEHDIE toponym matching tool. https://tool.mehdie.org/. Accessed: 01/01/2024

[32] Papadakis, G., Skoutas, D., Thanos, E., Palpanas, T.: Blocking and filtering techniques for entity resolution: A survey. ACM Computing Surveys (CSUR) **53**(2), 1–42 (2020)

[33] Tucci, M., Giordano, A.: Positional accuracy, positional uncertainty, and feature change detection in historical maps: Results of an experiment. Computers, Environment and Urban Systems **35**(6), 452–463 (2011) https://doi.org/10.1016/j.compenvurbsys.2011.05.004

[34] Bast, H., Buchhold, B.: Qlever: A query engine for efficient sparql+text search. In: Proceedings of the 2017 ACM on Conference on Information and Knowledge Management. CIKM '17, pp. 647–656. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3132847.3132921

[35] Christen, P.: Febrl- an open source data cleaning, deduplication and record linkage system with a graphical user interface. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1065–1068 (2008)

[36] Keskustalo, H., Pirkola, A., Visala, K., Leppänen, E., Järvelin, K.: Non-adjacent digrams improve matching of cross-lingual spelling variants. In: String Processing and Information Retrieval: 10th International Symposium, SPIRE 2003, Manaus, Brazil, October 8-10, 2003. Proceedings 10, pp. 252–265 (2003). Springer

[37] Smith, T.F., Waterman, M.S., *et al.*: Identification of common molecular subsequences. Journal of molecular biology **147**(1), 195–197 (1981)

[38] Gong, R., Chan, T.K.: Syllable alignment: A novel model for phonetic string search. IEICE transactions on information and systems **89**(1), 332–339 (2006)