An Automated Approach for Geocoding Tabular Itineraries

Rui Santos*
INESC-ID,
Instituto Superior Técnico,
University of Lisbon
Lisbon, Portugal

Patricia Murrieta-Flores
Digital Humanities Research Center,
Faculty of Humanities,
University of Chester
Chester, United Kingdom

Bruno Martins
INESC-ID,
Instituto Superior Técnico,
University of Lisbon
Lisbon, Portugal

ABSTRACT

Historical itineraries, often accessible as lists or tables describing places visited in sequence, are abundant resources and also important objects of study for humanities scholars. This article advances a novel method for automatically geocoding tabular itineraries, combining approximate string matching with a cost optimization algorithm based on dynamic programming. Experiments with a dataset of historical itineraries, with ground-truth geocoding annotations provided by domain experts and leveraging also the GeoNames gazetteer, attest to the effectiveness of the proposed method. The obtained results show that while approximate string matching can already achieve very low median errors, with many toponyms matching exactly against GeoNames entries, the combination with cost optimization can significantly improve results in terms of the average distance towards the correct disambiguations.

CCS CONCEPTS

• **Information systems → Geographic information systems**; **Specialized information retrieval**; • **Applied computing →** *Arts and humanities*;

KEYWORDS

automated geocoding; toponym matching; dynamic programming; digital humanities; geographic information retrieval

ACM Reference format:
Rui Santos, Patricia Murrieta-Flores, and Bruno Martins. 2017. An Automated Approach for Geocoding Tabular Itineraries. In *Proceedings of 11th Workshop on Geographic Information Retrieval, Heidelberg, Germany, November 30-December 1, 2017 (GIR'17),* 10 pages.
https://doi.org/10.1145/3155902.3155908

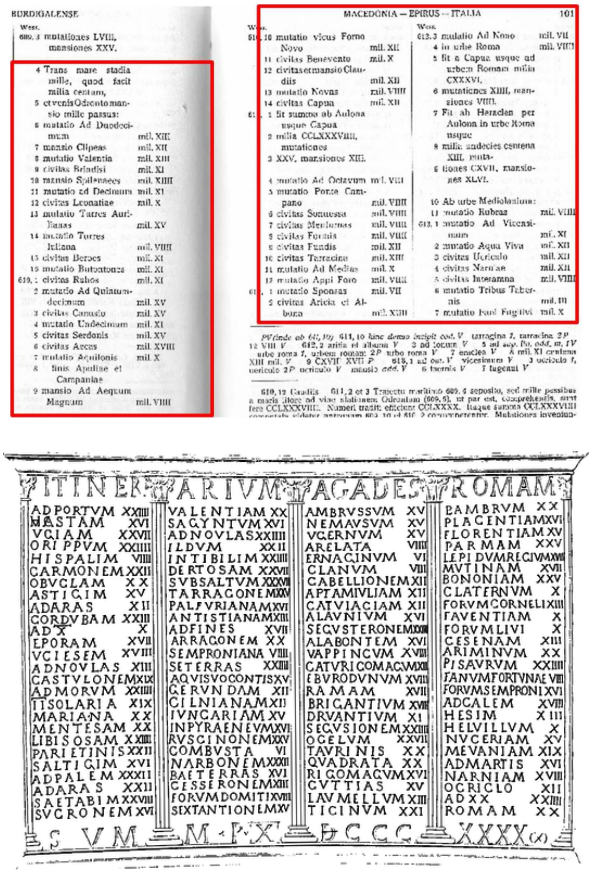

Figure 1: Two examples of historical itinerary.

1 INTRODUCTION

Historical itineraries, often accessible as tables or as sequential lists of names for the places that were visited in a particular journey, are abundant resources and also important objects of study for humanities scholars [6, 24, 31], for instance providing insights into the development of human mobility, and invaluable information related to the establishment of historical road networks. Well-known examples include the 3rd century *Itinerarium Antonini Augusti*, the 4th century *Itinerarium Alexandri*, the *Itinerarium Burdigalense* written between the 8th and 10th centuries, or the 1191 *Itinerarium Cambriae*, among many others[1]. Many historical manuscripts and/or transcriptions containing information on itineraries, dating from the Medieval period to the 20th century, are nowadays available in digital formats within repositories and initiatives such as Europeana[2], in the Internet Archive[3], or in the context of Digital Humanities projects like Pelagios[4].

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

[1] http://www.peterrobins.co.uk/itineraries/
[2] http://www.europeana.eu
[3] http://archive.org
[4] http://commons.pelagios.org

Figure 1 provides illustrative examples. The top part of Figure 1 presents a page from a book with a transcription of the *Itinerarium Burdigalense*, which is the earliest known Christian *itinerarium*. It was written by an anonymous pilgrim, recounting his journey made between 333 A.D and 334 A.D from Burdigala (present-day Bordeaux, France) to Jerusalem and back. The bottom part of Figure 1 shows the itinerary inscribed in one of the Vicarello Cups, i.e. four silver cups discovered in 1852 near the baths of Aquae Apollinares at Vicarello, Italy, having on their outside an itinerary that goes from Gades (modern Cadiz) over land to Rome, including 104 stopping points along the way.

Few historical tabular itineraries are nonetheless directly associated with map-based representations and, in many cases, there is little information on the actual routes taken in between locales. As such, there are many interesting questions related to early travelling routes. We believe that the analysis of historical itineraries (e.g., for consistency checking, or enabling new inquires and inferences about the routes) can be facilitated through the analytical tools of Geographical Information Systems (GIS) and/or through map-based representations for these data. The research reported on this article concerns with automatically geocoding historical itineraries, leveraging innovative methods that explore the idea that travelers tend to choose the most efficient routes (e.g., itineraries will likely minimize the distance between locations [2, 5, 6, 21, 38]).

In brief, the proposed method is based on a sequence of four stages, combining string similarity search and well-known optimization procedures, in order to find the most likely route. On the first stage, we use string similarity [25, 26] to look for candidate disambiguations in a large-coverage gazetteer. A state-of-the-art string matching method [29], leveraging supervised machine learning for combining multiple similarity metrics, can then optionally be used to further filter/restrict the set of candidates associated to each place in the itinerary. A least-cost path between pairs of candidates, visited in sequence over the itinerary, is afterwards estimated on the third stage. For now we are only considering a geodesic path over the Earth's surface (i.e., distance measured *as the crow flies*), although future developments can consider alternative path computation methods at this stage, e.g. leveraging terrain slope or historical land-coverage for estimating movement costs [3, 8, 9, 23]. Finally, Step 4 leverages the distance associated to each of the paths between pairs of candidates, which were computed in Stage 3, to find an overall best path for the entire itinerary, also disambiguating each of the toponyms to the most likely candidate. A dynamic programming algorithm, similar to Viterbi decoding in the context of hidden Markov models [10, 34], is used at this stage to efficiently compute the global path that minimizes the traveled distance.

The proposed method was evaluated through tests with manually geocoded itineraries (e.g., measuring the distance between the estimated disambiguation and ground-truth geo-spatial coordinates for the places in each itinerary). We relied on a dataset originally provided by Peter Robins[5], containing a total of 24 sequences sub-divided into segments of varied lengths, corresponding to well-known historical itineraries. We also used the GeoNames[6] gazetteer for supporting the disambiguation of toponyms into geo-spatial coordinates, i.e. a resource which focuses on modern administrative geography but that nonetheless lists many historical variants as alternative place names. Our experiments showed that while approximate string matching can already achieve very low median errors (e.g., many of the toponyms in historical itineraries match exactly with entries in GeoNames, and thus the median distance towards the correct disambiguations is quite low), the combination with cost optimization can significantly improve results in terms of the average distance towards the correct disambiguations. Methods leveraging the intuition that travelers tend to choose the least costly routes, in combination with approximate string matching for finding gazetteer entries that are likely to correspond to the historical toponyms in the itineraries, are indeed effective for automatically geocoding these resources. The best results, when simultaneously looking at the mean and median errors, were achieved with the combination of dynamic programming, for minimizing distances, with the string matching method leveraging supervised learning.

The rest of this paper is organized as follows: Section 2 describes related work, focusing on previous methods for geocoding itineraries. Section 3 presents the proposed method, outlining the main stages and detailing the optimization procedure based on dynamic programming. Section 4 presents the experimental evaluation of the proposed method, detailing the evaluation methodology and discussing the obtained results. Finally, Section 5 summarizes our main conclusions, and presents possible paths for future work.

2 RELATED WORK

Previous research with significant similarities towards the work that is presented in this article has been previously reported [5, 6], e.g. in the form of preliminary studies with methods for disambiguating place references in the context of tabular descriptions for historical itineraries (i.e., methods for linking each toponym, presented in a tabular itinerary, to the correct geo-spatial coordinates). These previous studies proposed to minimize the sum of all string distances between itinerary toponyms and candidate disambiguations, whereas our approach instead leverages the idea of minimizing geo-spatial distances between candidate disambiguations. Our particular way of looking at the problem can be naturally formulated as optimizing the cost of traversing a graph structure corresponding to a trellis, in which the edges encode distances (e.g., geodesic distances or even terrain traversal costs estimated through some other procedure) between consecutive locations.

In a first study, Blank and Henrich [5] leveraged string similarity together with the fact that tabular itineraries often contain an approximate geo-spatial distance between each toponym, presented in sequence, and the former toponym in the itinerary. A graph is first created for representing the tabular itinerary. The graph contains two special nodes encoding the beginning and the termination of the itinerary, and it also contains nodes $n_{i,j}$ representing the possible disambiguations t_j for each toponym h_i in the itinerary. The Jaro-Winkler [36] string distance function $s(h_i, t_j)$ is applied to each toponym h_i of the itinerary table and all toponyms t_j available in a gazetteer (i.e., the GeoNames gazetteer). If the distance $s(h_i, t_j) \leq \arg\min_{t_j} s(h_i, t_j) + \delta$, with δ representing a predefined string distance threshold, then a node $n_{i,j}$ is considered in the graph

[5] http://www.peterrobins.co.uk/itineraries/list.html
[6] http://www.geonames.org/

for representing a possible toponym disambiguation. The edges in the graph encode the possible sequences of the itinerary.

Besides the string distance function, the authors also considered filters for further restricting the sets of possible disambiguations for each toponym. A spatial filter is for instance applied to each candidate's geo-spatial location, so as to ensure that candidates are contained within a given area. This area depends on the distance value that is registered on the input table towards the previous itinerary entry. Another filter checks the azimuth change in the trajectory that can correspond to the disambiguated itinerary, after the inclusion of the candidate disambiguation. If the change is greater than a threshold α, then the candidate disambiguation is rejected. This second filter is based on the fact that itinerary trajectories are usually as direct as possible to get from one location to the next. A candidate from the gazetteer that, according to string similarity, might be a good match for a toponym in the itinerary can be promptly rejected by either the spatial filter or the azimuth change filter. To perform the actual disambiguation, the shortest string distance path that connects the start to the end nodes is calculated, and the nodes involved in the shortest path are returned.

Blank and Henrich [5] evaluated the proposed method with an itinerary containing German place names, concluding that 40% of the toponyms were correctly identified. The authors also reported on some examples for the toponyms that were evaluated, illustrating that the proposed filters do indeed work properly.

In a subsequent study [6], Blank and Henrich tested different string similarity metrics, and they also advanced a depth-first branch and bound algorithm for performing the actual disambiguation, thus improving the formalization of their procedure. In addition to the Jaro string distance, the authors experimented with (i) the Levenshtein distance, (ii) a string distance method based on n-gram overlap, considering bi-grams and tri-grams, (iii) a method based on character skip-grams, (iv) the DAS distance metric, specifically proposed for toponym matching [16], and (v) different phonetic encodings, namely (a) the Cologne phonetics, (b) Soundex, (c) Phonet, and (d) the New York Identification and Intelligence System Phonetic Code. A graph is created through the same procedure that was advanced in the original study, but the actual toponym disambiguation is now made through a depth-first branch and bound algorithm that iteratively expands the disambiguation candidates, e.g. in decreasing order from the best to the worst.

The evaluation was made with 15 itineraries that, when combined, contained 218 stopping points. The toponyms contained in the itineraries were all in German and the best performing string distance metric was the Cologne phonetic distance, reaching an accuracy of 54.1% when leveraging string similarity alone (i.e., without the geo-spatial filters for distance or azimuth angle). The authors also tested different node expansion orders (i.e., in stopping order, in reverse stopping order, and selectively picking the best candidate), concluding that expanding nodes in reverse order usually grants a better accuracy, while expanding first the best candidate grants, in most of the cases, an inferior distance between the disambiguations and the true locations (i.e., when wrong, the method chooses candidates that are closer to the real locations).

Adelfio and Samet [2] addressed the related problem of identifying and extracting itinerary tables from Web pages. Instead of focusing on the disambiguation of the toponyms that are present on the itineraries, these authors have instead focused on discriminating between Web tables that describe a true itinerary, and other resources with a geographical context (e.g., demographic tables, associating place names to the corresponding number of inhabitants). A pipeline with three steps was considered to extract itineraries from the Web, involving (i) a table crawler that scans a large portion of the Web to build a dataset containing tables with a geographic context, (ii) a geo-tagger that identifies geographic references in the tables and disambiguates them through simple heuristics, and (iii) an itinerary identifier that uses supervised machine learning to decide if a table indeed represents an itinerary or not.

In Step (ii) of the proposed pipeline, Adelfio and Samet used a geo-tagging method based on spatial coherence, inspired on a previous publication [1]. The main contributions are in Step (iii) of the proposed pipeline, where the authors proposed to use a combination of multiple features as input to a classifier, under the general assumption that itineraries are efficient in terms of how the stopping points are ordered. The authors explored the idea of generating small variations on the order of the original stopping points, to determine if the variation is more efficient than the order presented in the itinerary table being considered. Two efficiency measures that leveraged this scheme (i.e. a local efficiency and a general efficiency metric) were presented by the authors.

Let $L = \{l_1 l_2 \dots l_n\}$ represent the ordered set of locations, and consider that $\delta_{i,j}(L)$ is a function that returns the value of one if the locations l_i and l_{i+1} have a shorter path length than the length obtained by the order given in L, or returns the value of zero otherwise. The local efficiency metric expresses how the listed order could be more efficient by considering a variation on consecutive stopping points, and is defined as $\epsilon_1(L)$ in Equation 1. The general efficiency metric instead expresses how the order could be more efficient by considering coarse variations of non consecutive stopping points, as is defined by $\epsilon_2(L)$ in Equation 2.

$$\epsilon_1(L) = \frac{1}{n-3} \sum_{i=1}^{n-3} \delta_{i,i+2}(L) \tag{1}$$

$$\epsilon_2(L) = \frac{1}{\binom{n-2}{2}} \sum_{i=1}^{n-3} \sum_{j=i+2}^{n-1} \delta_{i,j}(L) \tag{2}$$

Besides the two efficiency measures, other features were also considered. These include (i) a binary feature indicating if the table describes a round trip, with the same location featured in the first and last positions, (ii) the number of ordered date/time columns, (iii) the number of ordered numeric columns, (iv) the number of text columns that are sorted alphabetically, and (v) the occurrence of words that are indicative of an itinerary (e.g., words such as *itinerary*, *trip* or *travel*, among others). The aforementioned features are provided to a binary classifier that decides if the table is indeed an itinerary. Given the variety of feature types that were considered, the authors experimented with different types of classification methods, namely with (i) a naïve Bayes model, (ii) a decision tree, and (iii) a support vector machine. Through experiments, the authors concluded that the decision tree classifier was more precise (i.e., they measured a precision of 0.72 when retrieving true

itinerary tables, and an F1 score of 0.73), although the support vector machine had a highest recall.

Moncla et al. [21] described a process for annotating spatial entities in text that, taking inspiration on other studies addressing toponym resolution [11, 18, 22, 27, 30, 35], can geocode a natural language description for a route. Natural language processing is first employed to recognize the toponyms referenced in the input text, combining a cascade of transducers with the use of gazetteers. These toponyms are then disambiguated through an approach that combines clustering based on spatial density, with semantic matching of geographical feature types. A graph-based method is finally used to find the sequence of disambiguated toponyms corresponding to the order by which they are visited in the itinerary.

The graph-based method starts by building a complete graph from the text, using vertices to represent the mentioned locations, and edges to represent segments between pairs of locations. A multi-criteria analysis method is used to assign a weight to each edge of the complete graph, combining local information extracted from the text with physical features obtained from external datasets (e.g., from gazetteers or terrain elevation models). The set of considered criteria includes proximity in the source text, geo-spatial distance, terrain slope and motion orientation, or temporal relations extracted from the text. From the weighted graph, the authors compute a minimum spanning tree, in order to get an undirected acyclic graph connecting all vertices. The spanning tree is finally transformed into a partially directed acyclic graph (i.e., when available, verbs expressing motion relations, such as *goes to* or *reach*, are used to assign a direction to some the corresponding edges), and the longest path on the spanning tree is used to identify the starting and ending points in the sequence that represents the itinerary.

Moncla et al. evaluated their method with a set of 90 itineraries in different idioms, that had been previously annotated. The obtained results were compared against manually produced trees, and also against the real trajectories associated to the itineraries. Approximately 71.4% of the inferred itineraries were within a buffer surrounding the geodesic path between each pair of true locations, with a width of 15% of the corresponding path length.

3 THE PROPOSED METHOD

Taking inspiration on the previous studies described in Section 2, we propose a method for geocoding tabular itineraries that is sound, effective, conceptually simpler (i.e., we avoid the use of filters for geo-spatial distance or azimuth angles, and the worse-case computational complexity associated to the cost optimization procedure is now $O(T \times N^2)$ instead of N^T, where N is the number of itinerary stops and T is the number of candidate disambiguations per itinerary stop), and also easily extendable (e.g., new heuristics for matching strings or for computing least-cost paths between pairs of locations can easily be integrated).

The proposed method can be seen as a sequence of four steps, combining string matching with a cost optimization procedure based on dynamic programming. The four steps are as follows:

(1) For each toponym in the itinerary, we retrieve a list of candidate disambiguations by searching for similar strings in a database containing records from the GeoNames gazetteer. Each candidate is a tuple containing a place name, a unique identifier for the corresponding record in GeoNames, the modern population count, and the geo-spatial coordinates of latitude and longitude. The search for similar strings is made through the Whoosh[7] Python library, which efficiently retrieves candidates sorted according to a metric derived from the overlap between sets of character n-grams in both strings. For increased computational efficiency, we restrict the retrieved list to the top 30 most similar candidates.
(2) The list of candidate disambiguations is optionally re-ranked through a state-of-the-art string matching procedure, which leverages supervised machine learning for combining multiple similarity metrics, and which is detailed in a separate publication [29]. The string matching method essentially corresponds to an ensemble of decision trees (i.e., a random forest classifier) that verifies if a pair of toponyms does match or not (i.e., the model checks if both toponyms correspond to the same real-world place), leveraging a combination of 13 different string similarity metrics as the descriptive features that are passed as input to the classifier. The model was trained with a large dataset of 5 million toponym pairs collected from the GeoNames gazetteer and, on 2-fold cross-validation experiments, it achieved an accuracy of 78.67 – see the previous publication by Santos et al. for additional details [29]. The candidates are re-ranked according to the confidence of the classifier on making assignments to the positive class and, for increased computational efficiency, we restrict the list of candidates after re-ranking to the top k most similar candidates (i.e., in our experiments, we considered k equal to 15). Note that result re-ranking is an optional step and, in some of our experiments, we directly used the top 15 candidates from Step 1. The results from this step are modeled as a trellis, i.e. a graph where the nodes correspond to the disambiguation candidates for the sequence of toponyms in the itinerary – see Figure 2.
(3) For each pair of candidates (c_i, c_{i+1}) that appear associated to consecutive places visited in the itinerary (i.e., for each pair (c_i, c_{i+1}) such that c_i is a candidate disambiguation for place s_i and c_{i+1} is a candidate disambiguation for a place s_{i+1} visited immediately after s_i), we estimate the most likely path for traveling between the two locations, as well as the geo-spatial distance associated to the path. The edges shown in the trellis from Figure 2 are weighted with basis on these geo-spatial distances. In the experiments that are reported on this paper, we relied on a naive approach based on the shortest surface-path (i.e., geodesic) between the pairs of locations, using Vincenty's formulae for calculating the corresponding distance [33]. For future work, we also plan to experiment with an approach based on least-cost path analysis, using a cost surface built from raster datasets encoding terrain slope and/or historical land-coverage, and using a procedure such as the A* search algorithm to find the most likely route between two locations in a raster grid [3, 8, 23, 37].
(4) Compute the most likely candidate for each place in the itinerary (i.e., globally disambiguate the toponyms in the

[7] http://pypi.python.org/pypi/Whoosh

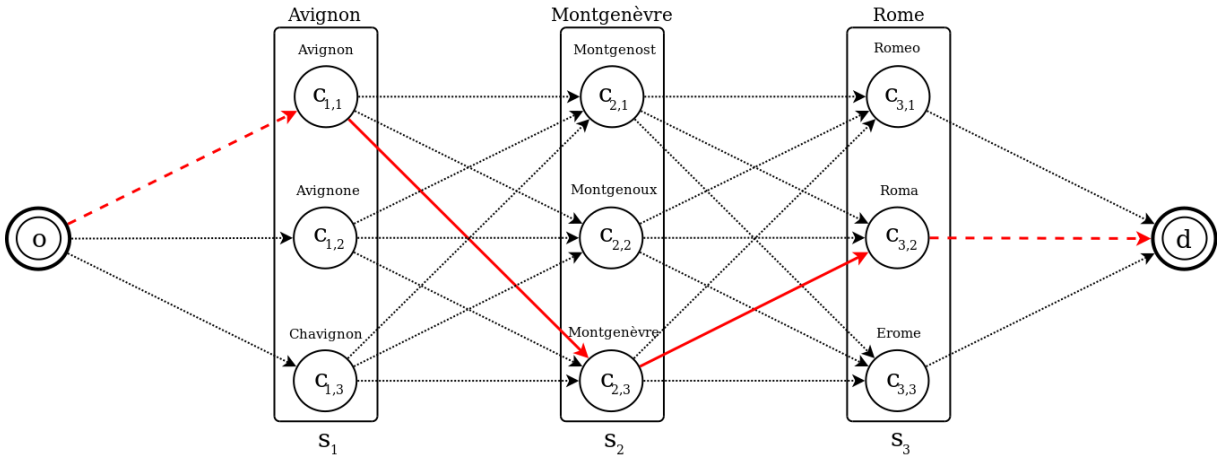

Figure 2: Example of a trellis encoding an itinerary that involves three different toponyms.

itinerary), by choosing the sequence of candidates that corresponds to the overall path of least distance. A well-known dynamic programming approach known as the Viterbi algorithm [10, 34], frequently used in the context of decoding sequences with hidden Markov models, can be adapted to compute the sequence of best candidate disambiguations.

Regarding Step 4 of the proposed geocoding method, we essentially rely on dynamic programming for efficiently picking the best disambiguation candidate for each place in the itinerary. The Viterbi algorithm is a dynamic programming approach to find the most likely path through a trellis, i.e. a graph where nodes are ordered into vertical slices representing sequential information, and where each node, at each position in the sequence (i.e., at each slice), is connected to at least one node at an earlier position, and at least one node at a later position. In our case, the trellis represents a graph of possible paths between candidate disambiguations for the places in the itinerary. Each node in this graph, apart for two special nodes that were included for illustration purposes and that encode the beginning and the termination of the sequence, represents a candidate disambiguation for a given toponym. Each edge represents the geodesic straight line distance between two candidates for consecutive places in the itinerary, as computed in Step 3 of our procedure. The edges from/to the special nodes at the extremes of the trellis can be considered to have a distance of zero towards the subsequent/preceding nodes. An example of a trellis, encoding an itinerary with 3 toponyms, is shown in Figure 2.

The motivation behind the use of dynamic programming arises from the fact that, given a maximum of N candidates per place and T places in the itinerary, naively selecting the most likely candidate for all places in the itinerary would involve N^T calculations. However, for any candidate at position t, there is only one most likely path to that candidate. Therefore, if several paths converge at a particular candidate for a toponym at position t, instead of recalculating them all when computing the most likely path from this candidate to candidates at position $t+1$, one can discard the less likely paths, and use only the most likely path in the calculations. When this

Algorithm 1: The dynamic programming method of Step 4.

input: a sequence $S = \{s_1, s_2, \ldots, s_T\}$, with each s_t equalling a set of disambiguation candidates for the toponym at step t
output: a sequence $I = \{c_1, c_2, \ldots, c_T\}$, with each c_t equalling the estimated disambiguation for the toponym at step t
function VITERBI(S)
  $R_1[1, \cdot\,] \leftarrow 0$
  for $t \leftarrow 2, 3, \ldots, T$ do
    for $n \leftarrow 1, 2, \ldots, N$ do
      $R_1[t, n] \leftarrow \min_k(R_1[t-1, k] + \Delta(s_{t-1,k}, s_{t,n}))$
      $R_2[t, n] \leftarrow \arg\min_k(R_1[t-1, k] + \Delta(s_{t-1,k}, s_{t,n}))$
    end for
  end for
  $z_T \leftarrow \arg\min_k(R_1[t, k])$
  $I_T \leftarrow s_{z_T}$
  for $t \leftarrow T, T-1, ..., 2$ do
    $z_{t-1} \leftarrow R_2[t, z_t]$
    $I_{t-1} \leftarrow s_{z_{t-1}}$
  end for
  return I
end function

is applied to each position in the itinerary, it greatly reduces the number of calculations to $T \times N^2$, which is much better than N^T.

Concretely, the algorithm looks at each candidate for a place at position t and, for all the paths that lead into that candidate, it decides which of them was the most likely, i.e. it chooses the path of least distance. In the unlikely case that two or more paths are found to be minimum (i.e. if their distances are exactly the same), then one of them is chosen randomly. The algorithm discards all other paths, and it appends the source candidate, from where the path originated, to a survivor path variable of the candidate for the toponym at position t. The corresponding path distance is also assigned to the toponym at position t of the itinerary.

Figure 3: Ground-truth trajectory associated to the pilgrimage of Jehan de Tournay from Valenciennes to Venice (on the left), compared to the estimated trajectory for the same itinerary with (on the right) or without (on the middle) cost optimization through dynamic programming for improving the results of approximate string matching in terms of the overall consistency.

The same operation is carried out on all the candidates for the place at position t, at which point the algorithm moves onto the candidates at position $t + 1$ and carries out the same operations. When we reach time $t = T$ (i.e., when we reach the final toponym in the itinerary), the algorithm determines the survivor path as before and it also has to make a decision on which sequence of these survivor paths is the most likely one. This is carried out by back-tracking the decisions regarding the choice of the survivor with the least distance. The sequence of candidate disambiguations associated to the path with the overall least distance is finally returned. The pseudo-code shown in Algorithm 1 formally describes the dynamic programming procedure used in Step 4 of the geocoding method, considering that $\Delta(s_{t-1,k}, s_{t,n})$ represents the distance between the candidates $s_{t-1,k}$ (i.e., toponym k that appears as a disambiguation candidate for the place immediately before position t in the itinerary) and $s_{t,n}$ (i.e., the candidate toponym n for the place at position t), as computed in Step 3.

4 EXPERIMENTAL EVALUATION

The experimental evaluation of the proposed method was based on tests with a set of 24 manually geocoded historical itineraries, originally made available by Peter Robins. On average, each itinerary contained 7 separate segments (i.e., in our tests we executed the algorithm separately for each segment) and involved a total of 167 different toponyms, with the extremes corresponding to sequences of 43 and 770 toponyms. Approximately 8.63% of the toponyms that were present in the itineraries had a unique interpretation in the considered gazetteer. Moreover, each toponym from the itineraries had, on average, 10.2 exactly matching candidates in the gazetteer, which confirms that these place names are highly ambiguous. The itineraries mostly involve places in South and Western Europe, and thus our tests involved a custom gazetteer built through a region-based filter on the contents of GeoNames (i.e., the Python library named Whoosh is used to index GeoNames contents from a region of the globe that discards most of Africa, Asia and America, associating each alternative name for the gazetteer entries to the corresponding geo-spatial coordinates and additional meta-data elements). Notice that GeoNames is mostly covering the modern administrative geography, but in many cases the entries are also described with historical toponyms or transliterations in multiple languages. The same index over the contents of GeoNames is used by all the methods compared in our tests. For future work, we can easily adapt our method in order to make use of other gazetteers focusing on historical placenames [4].

Table 2 lists the different itineraries that were considered in our tests, together with the corresponding number of places. In turn, Figure 3 presents a map illustrating the first itinerary from Table 2 (i.e., the map presents the multiple segments that constitute the complete itinerary) corresponding to the 1488's pilgrimage of Jehan de Tournay from Valenciennes, via Rome and Loreto, to Venice.

With basis on the ground-truth annotations for the itineraries in the considered dataset, we measured the quality of the obtained results in terms of the geo-spatial distance between the true coordinates for each toponym in each itinerary, and the coordinates of the estimated disambiguations. We specifically measured the mean and the median geo-spatial distances, similarly to previous studies

Table 1: Experimental results for the proposed method, in comparison against simpler baselines.

Method	Average Distance	Median Distance	Accuracy					
			<500m	<1km	<5km	<50km	<100km	<500km
Whoosh	795.20 ± 1710.83	2.11	0.37	0.49	0.59	0.65	0.67	0.78
Whoosh + Population	440.52 ± 1273.57	0.82	0.43	0.55	0.71	0.82	0.84	0.89
Whoosh + Random Forests + Population	358.56 ± 1076.01	0.75	0.44	0.56	0.71	0.83	0.85	0.91
Whoosh + Dynamic Programming	142.29 ± 297.80	2.04	0.33	0.44	0.61	0.75	0.81	0.93
Whoosh + Dynamic Programming (Spatial+String)	146.96 ± 304.59	2.09	0.33	0.44	0.61	0.74	0.79	0.93
Whoosh + Random Forests + Dynamic Programming	126.71 ± 339.39	1.32	0.37	0.50	0.69	0.82	0.86	0.95

on toponym disambiguation and text geocoding [20, 30, 35]. Using both metrics can better inform on cases where the disambiguations are very close to the ground truth coordinates, and on cases where the incorrect disambiguations are nonetheless very close to the true locations. Distances were measured with Vincenty's geodetic formulae [33]. We also measured results in terms of disambiguation accuracy, thresholding the aforementioned geo-spatial distance with values ranging from 500 Meters to 500 Kilometers. Notice that a distance of zero is highly unlikely, given that the coordinates for places in GeoNames will be different from the ones considered in the ground-truth annotations for the itineraries. The final threshold value of 500 Kilometers, although impractical for real-world applications, can provide an estimate for the number of cases in which the disambiguation procedure provided results that are significantly distant from the ground-truth and from the remaining places in the corresponding itinerary.

Table 1 presents the results obtained with the proposed method, over the full set of annotated itineraries and contrasting the results against simpler baselines. The first 3 rows in Table 1 correspond to baselines that do not use dynamic programming to optimize the total distance involved in each itinerary. These are as follows:

(1) A baseline method using Whoosh to retrieve, for each toponym in each itinerary, the single most similar entry from the GeoNames gazetteer. Recall that Whoosh considers a similarity metric based on the degree of overlap between sets of character n-grams in both strings, specifically considering character bi-grams and tri-grams.
(2) Similar to the first baseline method, but using the population counts associated to the gazetteer entries as a second disambiguation criteria. For each toponym in the itinerary, if there are multiple candidate disambiguations in the top positions of the list retrieved by Whoosh, all with the same similarity score, we consider the top candidate with the highest value in terms of the population count. Notice that GeoNames only contains modern values for the population counts, although higher values in terms of modern population are also frequently associated to important historical places.
(3) Similar to the first baseline method, but using a more sophisticated procedure to rank the gazetteer entries according to their estimated similarity. We specifically use our previously proposed state-of-the-art string matching method [29], which leverages supervised machine learning (i.e., a random forest classifier trained with a large set of pairs of toponyms extracted from GeoNames) for combining multiple similarity metrics. The classifier attempts to decide if a given pair of toponyms (i.e., the toponym in the original data and a candidate retrieved by Whoosh) indeed correspond to the same real-world location. For each toponym in the itinerary, we return the GeoNames entry, from the list of 30 candidates retrieved by Whoosh, for which the classifier had the highest confidence in saying that it matches the toponym in the input data. In case of ties (e.g., for disambiguating candidates sharing the exact same name, but with different geo-spatial coordinates), the population count is used as a second-level ranking/selection criteria.

The last three rows in Table 1 correspond to alternative methods using dynamic programming. The middle row in this part of the table corresponds to an approach that combines geo-spatial and string distances (i.e., the Whoosh similarity score, transformed into a distance function through a logarithmic transformation that attempts to place the string similarity scores within the same range of values as the geo-spatial distances) within the dynamic programming algorithm, instead of just using string similarity for ranking the top candidates. In this case, the pseudo-code given in Algorithm 1 is adapted so that the update for the intermediate costs $R_1[t, n]$ becomes as follows:

$$R_1[t, n] \leftarrow \min_k (R_1[t-1, k] + \Delta(s_{t-1,k}, s_{t,n})) + d(s_{t,n}, o_t) \quad (3)$$

In the previous equation, $d(s_{t,n}, o_t)$ corresponds to the Whoosh distance between the candidate $s_{t,n}$ for the toponym at position t in the itinerary, and the actual/original toponym o_t at position t. In our adapted version of the Viterbi procedure, these string distances can be seen as equivalent to the emission costs that are considered in the context of hidden Markov models.

The last row in this second part of Table 1 corresponds to the complete method described in Section 3, whereas the first row corresponds to a baseline method in which the top 15 candidates retrieved by Whoosh are considered at each slice of the trellis (i.e., ignoring Step 2 of the method described in Section 3, and retrieving only 15 candidates at Step 1 of the algorithm). In all cases, the ties for the ranking scores based on string matching are again resolved through the maximum population heuristic.

The results in Table 1 show that while approximate string matching (i.e., the first baselines) can already achieve very low median errors, the combination with cost optimization can significantly improve results in terms of the average distance towards the correct disambiguations. A manual inspection of the results showed that

Table 2: Results for each of the 24 itineraries, with the method that corresponds to the complete procedure given in Section 3.

Itinerary	# Places (Segments)	Average Dist. (km)	Median Dist. (km)	Accuracy					
				<500m	<1km	<5km	<50km	<100km	<500km
Jehan de Tournay, Valenciennes-Venice 1488	104 (7)	14.34 ± 25.88	0.96	0.26	0.50	0.74	0.95	0.96	1.00
Anonymous, Bordeaux-Milan, 333	130 (7)	172.28 ± 496.95	0.76	0.39	0.49	0.66	0.79	0.81	0.93
Bertrandon de la Broquière, Ghent-Dijon, 1432-1433	51 (2)	63.91 ± 82.99	3.16	0.23	0.34	0.48	0.69	0.82	1.00
Bruges, road inventory, late 15th-century	770 (39)	97.50 ± 194.97	0.87	0.40	0.52	0.69	0.83	0.86	0.94
Nompar de Caumont, Caumont-Fisterra, 1417	61 (3)	152.3 ± 279.08	0.77	0.39	0.48	0.56	0.63	0.85	0.92
Nompar de Caumont, Caumont, 1418-1419	89 (2)	260.68 ± 1267.11	0.51	0.48	0.66	0.80	0.93	0.94	0.95
Codex Calixtinus, Aquitaine-Santiago, ca. 1140	88 (3)	137.93 ± 143.05	1.12	0.43	0.53	0.73	0.76	0.79	0.88
Emo, Frisia-Rome-Cologne, 1211-1212	43 (2)	55.79 ± 169.77	4.43	0.23	0.33	0.52	0.83	0.90	0.99
Charles Estienne, France, 1552-1553	623 (17)	212.83 ± 391.51	0.76	0.42	0.51	0.72	0.81	0.85	0.95
Arnold von Harff, Cologne-Cologne, 1496-1499	419 (6)	76.19 ± 313.06	0.50	0.48	0.62	0.79	0.89	0.91	0.96
Anonymous, Avignon-Santiago, 14th century	59 (6)	51.46 ± 112.82	0.27	0.64	0.70	0.85	0.89	0.92	0.95
Künig von Vach, Einsiedeln-Aachen, 1495	130 (2)	62.41 ± 165.44	0.48	0.50	0.61	0.80	0.91	0.92	0.94
Nikulas of Munkathvera, Iceland-Apulia, 1151	85 (5)	92.82 ± 148.20	1.15	0.28	0.41	0.58	0.65	0.71	0.97
Matthew Paris, London-Apulia, 1250	103 (8)	99.46 ± 116.17	2.62	0.27	0.37	0.58	0.76	0.81	0.93
Pvrchas his Pilgrimage, Plymouth-Calais, ca.1425	142 (6)	151.37 ± 409.42	1.96	0.32	0.43	0.68	0.87	0.90	0.95
Eudes Rigaud, Rouen-Rouen, 1253-1254	110 (2)	25.36 ± 91.76	0.58	0.45	0.58	0.74	0.92	0.95	0.99
Romweg-Karte, Germany-Rome, 1500	226 (16)	55.13 ± 53.02	1.85	0.19	0.36	0.61	0.77	0.81	0.99
Sigeric, Rome-England, 990	79 (2)	688.63 ± 1924.02	1.54	0.34	0.47	0.59	0.67	0.70	0.83
Annales Stadenses, Stade-Rome, ca.1250	217 (6)	64.73 ± 252.93	1.09	0.40	0.52	0.72	0.87	0.91	0.98
Barthélemy Bonis, Avignon-Rome, 1350	45 (1)	57.62 ± 192.18	0.39	0.53	0.62	0.73	0.89	0.89	0.96
Adam of Usk, Bergen op Zoom-Bruges, 1402-1406	49 (2)	39.90 ± 86.73	2.84	0.23	0.38	0.62	0.83	0.89	1.00
Pedro Juan Villuga, Spain, 1546	225 (8)	275.88 ± 743.25	0.95	0.38	0.48	0.74	0.77	0.81	0.91
William Wey 1st journey, Calais-Calais, 1458	104 (4)	101.05 ± 401.86	1.10	0.39	0.57	0.71	0.89	0.94	0.98
William Wey 2nd journey, Eton-Venice, 1462	47 (2)	31.52 ± 83.28	1.13	0.24	0.44	0.81	0.89	0.89	1.00
Average	167 (7)	126.71 ± 339.39	1.32	0.37	0.50	0.69	0.82	0.86	0.95

many of the toponyms in the historical itineraries match exactly with names for entries in the GeoNames gazetteer, and thus the median distance towards the correct disambiguations is quite low. However, it is also often the case that Whoosh retrieves many different candidates, corresponding to places located far apart, whose names match exactly with the input toponyms (e.g., GeoNames contains many different entries for places named *Rome*, which are all retrieved by Whoosh for a query using that toponym). In these cases, the first baseline in Table 1 (i.e., the method that only leverages the Whoosh retrieval scores) often failed to identify the correct candidate, although the combination with the maximum population heuristic was quite effective (i.e., the methods corresponding to the second and third rows in Table 1 achieved the overall best median distances, outperforming all methods that used dynamic programming for optimizing costs). In cases involving toponyms not matching exactly with GeoNames entries, and/or ambiguous cases in which the correct disambiguation was nonetheless retrieved in the top k candidates, the procedure based on dynamic programming for minimizing costs lead to improved results, effectively capturing the intuition that travelers tend to choose the least costly routes. The best trade-off between the median and the mean errors was achieved by the method that combined dynamic programming for minimizing geo-spatial distances with the string matching method that leverages the random forest classifier (i.e., the more advanced string matching procedure helped to further filter the candidates retrieved by Whoosh).

Table 2 summarizes the results with the complete method outlined in Section 3, for each of the 24 individual itineraries in the considered dataset. In Figure 3, we also present maps illustrating the ground-truth and the estimated trajectory for the first itinerary in Table 2, leveraging either the best method from the first part of Table 1, which does not use cost minimization (i.e., the map on the middle), or the complete method described on Section 3 (i.e., the map on the right). These results again confirm that methods leveraging dynamic programming for minimizing the overall geo-spatial distance can indeed be quite accurate, with the average distance ranging between 14 and 364 Kilometers and leading to overall trajectories that appear more consistent. In most of our itineraries, more than half of the individual toponyms are disambiguated to places located between 1 and 5 Kilometers of the correct coordinates, when considering the complete method.

Figure 4 presents a violin plot [15] with the distribution of the error values measured for the alternative algorithms in Table 1, in terms of the geo-spatial distance between the ground-truth coordinates of each toponym in the different itineraries, and the coordinates of the resulting disambiguations. Figure 5 presents a similar plot, in this case illustrating the errors for the 5 individual itineraries at Table 2 that involve a higher number of toponyms, and comparing the complete method described in Section 3 against the simplest baseline (i.e., the first row from Table 1). From the plots shown in these figures, one can see that a large majority of the toponyms are indeed disambiguated with a high accuracy. The distribution for the errors is skewed, with most of the toponyms disambiguated with

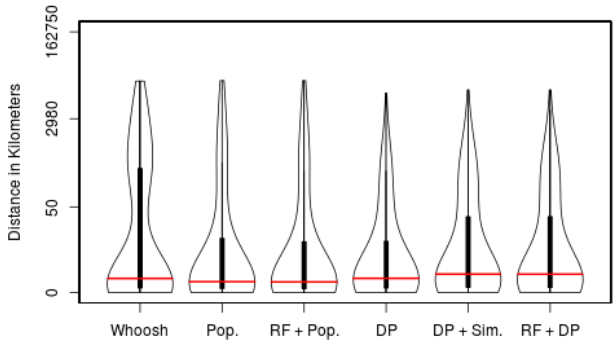

Figure 4: Distribution for the errors obtained over each toponym, for each of the methods shown in Table 1.

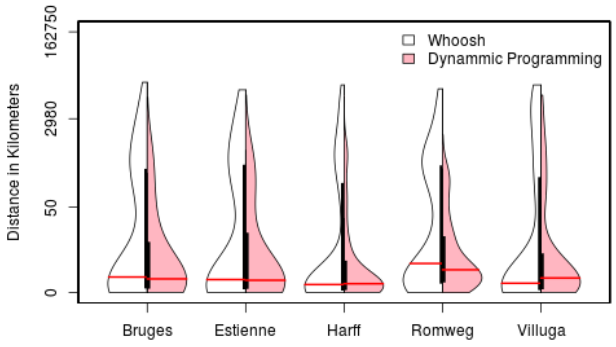

Figure 5: Distribution for the errors obtained over each of the 5 longest itineraries shown in Table 2.

very low errors. The distributions, particularly on Figure 5, also illustrate the fact that cost optimization with dynamic programming contributed to lowering the mean errors.

5 CONCLUSIONS AND FUTURE WORK

Historical itineraries, i.e. sequences of toponyms corresponding to particular journeys, are abundant resources and also important objects of study for humanities scholars. Geographical text analysis (i.e., methods for addressing tasks such as document geocoding [20, 35], toponym resolution [13, 22, 30, 35], or others [7, 11]) is increasingly being used in the Digital Humanities and related fields [14, 35], although very few studies have specifically addressed the problem of geocoding itineraries.

In this article, we described a novel approach for addressing the specific problem of automatically geocoding historical itinerary presented in tabular format, effectively combining string similarity and cost optimization techniques. The proposed method was evaluated with a set of manually annotated historical itineraries, and the obtained results attest to its effectiveness. The use of dynamic programming to minimize the total distance between the places involved in the journey obtained a significant improvement in terms of the average distance towards the correct disambiguations, although our experiments also showed that simpler baselines (e.g., combining approximate string matching with a maximum population heuristic) is already sufficient for disambiguating many of the toponyms and achieving a low median distance.

Despite the interesting results, there are also many ideas for future developments. Currently ongoing efforts focus on extending the method advanced in this paper in two different directions, namely (i) improving Step 2 with a new string matching method, leveraging a deep neural network architecture for modeling the strings being compared [28], and (ii) improving Step 3 with a novel approach for computing least-cost paths between consecutive locations in the itineraries, leveraging raster datasets encoding terrain slope and/or historical land-coverage [3, 8, 23].

The idea of leveraging least-cost path analysis for reconstructing the trajectories between pairs of locations is particularly interesting in the context of geocoding historical itineraries, supporting a better estimation of the cost for moving between locations, and also a detailed analysis and possible reconstruction of the actual routes. State-of-the-art least-cost path analysis methods are based on applying search procedures such as the A* algorithm, over anisotropic cost surfaces built through heuristics for combining multiple sources of information. Given that we can have access to a significant amount of itineraries already manually assigned to the corresponding geo-spatial trajectories, a particular idea that we would like to pursue relates to using reinforcement learning [32] for inferring the parameters of a model capable of reconstructing movement decisions, with basis on raster data sources (e.g., historical land-coverage information for the European territory from the Historic Land Dynamics Assessment project [12]).

The dynamic programming algorithm used in Step 4 of the proposed method takes its inspiration on Viterbi decoding for Hidden Markov Models (HMMs). For future work, it would also be interesting to test alternative methods for finding the best overall sequence of disambiguations. In the context of HMMs, a popular alternative is posterior decoding, based on taking the decisions that are individually most likely for each position. These and other alternative methods can perhaps also be adapted to our task [17].

Another objective for future work involves expanding the set of experiments with the proposed method, for instance through further tests involving the combination of string similarity and geo-spatial distance when choosing the most likely paths (i.e., the experiments in which we tried to combine geo-spatial and string distances, within the Viterbi procedure, were limited in the sense that we did not systematically attempt to tune the contribution of each component), or considering different thresholds for the number of disambiguation candidates considered for each toponym in an itinerary. For now, we restricted our tests to 15 candidates per toponym, although increasing this number can perhaps lead to improvements on the overall results, by passing more disambiguation options to the dynamic programming algorithm. Significantly

increasing the number of candidates will involve additional strains in terms of computational performance, although we can consider highly optimized implementations for the dynamic programming method, e.g. leveraging Graphical Processing Units (GPUs) [19].

Also in terms of further extending the experimental evaluation, it would be interesting to evaluate the proposed method with other datasets besides the historical itineraries made available by Peter Robins, e.g. referring to different regions of the globe, and/or using other types of toponyms (e.g., involving different alphabets, and different challenges in terms of performing matches against gazetteer entries [28]). A particular example would be the dataset from the al-Thurayyā Gazetteer[8], which includes almost 2,000 route sections geo-referenced from Georgette Cornu's *atlas du monde arabo-islamique à l'époque classique: IXe-Xe siècles*.

ACKNOWLEDGMENTS

We would like to thank Peter Robins for openly publishing the historical itinerary data that supported our validation experiments.

This work was partially supported by the Trans-Atlantic Platform for the Social Science and Humanities, through the Digging into Data project with reference HJ-253525. The researchers from INESC-ID also had financial dupport from Fundação para a Ciência e Tecnologia (FCT), through the project grant with reference PTDC/EEI-SCR/1743/2014 (Saturn), as well as through the INESC-ID multi-annual funding from the PIDDAC programme, which corresponds to the project with reference UID/CEC/50021/2013.

REFERENCES

[1] Marco D Adelfio and Hanan Samet. 2013. Structured Toponym Resolution Using Combined Hierarchical Place Categories. In *Proceedings of the ACM Workshop on Geographic Information Retrieval*. ACM.

[2] Marco D Adelfio and Hanan Samet. 2014. Itinerary Retrieval: Travelers, Like Traveling Salesmen, Prefer Efficient Routes. In *Proceedings of the ACM Workshop on Geographic Information Retrieval*. ACM.

[3] Jieun Baek and Yosoon Choi. 2017. A new algorithm to find raster-based least-cost paths using cut and fill operations. *International Journal of Geographical Information Science* 31 (2017). Issue 11.

[4] Merrick Lex Berman, Ruth Mostern, and Humphrey Southall (Eds.). 2016. *Placing Names : Enriching and Integrating Gazetteers*. Indiana University Press.

[5] Daniel Blank and Andreas Henrich. 2015. Geocoding Place Names from Historic Route Descriptions. In *Proceedings of the ACM Workshop on Geographic Information Retrieval*. ACM.

[6] Daniel Blank 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.

[7] Curdin Derungs and Ross S Purves. 2014. From text to landscape: locating, identifying and mapping the use of landscape features in a Swiss Alpine corpus. *International Journal of Geographical Information Science* 28 (2014). Issue 6.

[8] David H. Douglas. 1994. Least-cost Path in GIS Using an Accumulated Cost Surface and Slopelines. *Cartographica: The International Journal for Geographic Information and Geovisualization* 31 (1994). Issue 3.

[9] Thomas R Etherington. 2016. Least-cost modelling and landscape ecology: concepts, applications, and opportunities. *Current Landscape Ecology Reports* 1, 1 (2016).

[10] G. D. Forney. 1973. the Viterbi Algorithm. 61 (1973). Issue 3.

[11] Nuno Freire, 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.

[12] Richard Fuchs, M. Herold, P. H. Verburg, and J. G. P. W. Clevers. 2012. A high-resolution and harmonized model approach for reconstructing and analyzing historic land changes in Europe. *Biogeosciences* 10 (2012).

[13] Ian Gregory, 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 (2015). Issue 1.

[14] Ian Gregory and Patricia Murrieta-Flores. 2016. *Doing Digital Humanities: Practice, Training, Research*. Routledge, Chapter Geographical Information Systems as a Tool for Exploring the Spatial Humanities.

[15] Jerry L Hintze and Ray D Nelson. 1998. Violin Plots: A Box Plot-Density Trace Synergism. *The American Statistician* 52 (1998). Issue 2.

[16] Deniz Kilinç. 2016. An accurate toponym-matching measure based on approximate string matching. *Journal of Information Science* 42 (2016). Issue 2.

[17] Jüri Lember and Alexey A Koloydenko. 2014. Bridging Viterbi and posterior decoding: a generalized risk approach to hidden path inference based on hidden Markov models. *Journal of Machine Learning Research* 15 (2014). Issue 1.

[18] Michael D Lieberman and Hanan Samet. 2012. Adaptive context features for toponym resolution in streaming news. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[19] Saeed Maleki, Madanlal Musuvathi, and Todd Mytkowicz. 2016. Efficient parallelization using rank convergence in dynamic programming algorithms. *Commun. ACM* 59 (2016). Issue 10.

[20] Fernando Melo and Bruno Martins. 2016. Automated geocoding of textual documents: A survey of current approaches. *Transactions in GIS* 21 (2016). Issue 1.

[21] Ludovic Moncla, Mauro Gaio, Javier Nogueras-Iso, and Sébastien Mustière. 2016. Reconstruction of itineraries from annotated text with an informed spanning tree algorithm. *International Journal of Geographical Information Science* 30 (2016). Issue 6.

[22] Ludovic Moncla, Walter Renteria-Agualimpia, Javier Nogueras-Iso, and Mauro Gaio. 2014. Geocoding for texts with fine-grain toponyms: an experiment on a geoparsed hiking descriptions corpus. In *Proceedings of the ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM.

[23] Patricia Murrieta-Flores. 2012. *Travelling through past landscapes - Analysing the dynamics of movement during Late Prehistory in Southern Iberia with spatial technologies*. Ph.D. Dissertation. University of Southampton.

[24] Patricia Murrieta-Flores, Christopher Elliott Donaldson, and Ian Norman Gregory. 2016. GIS and literary history: advancing digital humanities research through the spatial analysis of historical travel writing and topographical literature. *Digital Humanities Quarterly* 10 (2016). Issue 4.

[25] Gonzalo Navarro. 2001. A Guided Tour to Approximate String Matching. *Comput. Surveys* 33 (2001). Issue 1.

[26] Gabriel Recchia 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.

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

[28] Rui Santos, Patricia Murrieta-Flores, Pável Calado, and Bruno Martins. 2017. Toponym Matching Through Deep Neural Networks. *International Journal on Geographical Information Science* (2017).

[29] Rui Santos, Patricia Murrieta-Flores, and Bruno Martins. 2017. Learning to Combine Multiple String Similarity Metrics for Effective Toponym Matching. *International Journal of Digital Earth* (2017).

[30] Michael Speriosu and Jason Baldridge. 2013. Text-Driven Toponym Resolution using Indirect Supervision.. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

[31] Thomas Szabó. 2009. Die Itinerarforschung als Methode zur Erschließung des mittelalterlichen Straßennetzes. *Die Welt der europäischen Straßen - von der Antike bis zur frühen Neuzeit* (2009).

[32] Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel. 2016. Value iteration networks. In *Proceedings of the Conference on Neural Information Processing Systems*. Curran Associates, Inc.

[33] Thaddeus Vincenty. 1975. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey review* 23 (1975). Issue 176.

[34] Andrew Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13 (1967). Issue 2.

[35] Benjamin Patai Wing. 2016. *Text-based document geolocation and its application to the digital humanities*. Ph.D. Dissertation. University of Texas at Austin.

[36] William E Winkler. 1990. String Comparator Metrics and Enhanced Decision Rules in the Fellegi-Sunter Model of Record Linkage. *Proceedings of the Section on Survey Research Methods of the American Statistical Association* (1990).

[37] Chaoqing Yu, Jay Lee, and Mandy J. Munro-Stasiuk. 2003. Extensions to least-cost path algorithms for roadway planning. *International Journal of Geographical Information Science* 17 (2003). Issue 4.

[38] Xiao Zhang, Baojun Qiu, Prasenjit Mitra, Sen Xu, Alexander Klippel, and Alan M MacEachren. 2012. Disambiguating Road Names in Text Route Descriptions Using Exact-All-Hop Shortest Path Algorithm. In *Proceedings of the European Conference on Artificial Intelligence*. IOS Press.

[8] http://althurayya.github.io/