Combining String Matching and Cost Minimization Algorithms for Automatically Geocoding Tabular Itineraries

RUI SANTOS, Instituto Superior Técnico, University of Lisbon
PATRICIA MURRIETA-FLORES, Digital Humanities Research Center, University of Chester
BRUNO MARTINS, Instituto Superior Técnico, University of Lisbon

Historical itineraries, often accessible as 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 cost optimization algorithms, specifically A* search for finding least-cost paths between pairs of locations over a raster encoding terrain slope, together with a dynamic programming method based on the Viterbi algorithm for finding sequences of locations that minimize the overall cost. 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. Moreover, the usage of least-cost paths for reconstruting the most likely routes between pairs of locations can enable new inquiries and inferences about historical routes. Our work shows that methods leveraging the intuition that travelers choose the least costly routes, in combination with approximate string matching for finding gazetteer entries matching the toponyms in the itineraries, are indeed effective for automatically geocoding these resources.

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

Additional Key Words and Phrases: automated geocoding; toponym matching; dynamic programming; digital humanities; geographic information retrieval

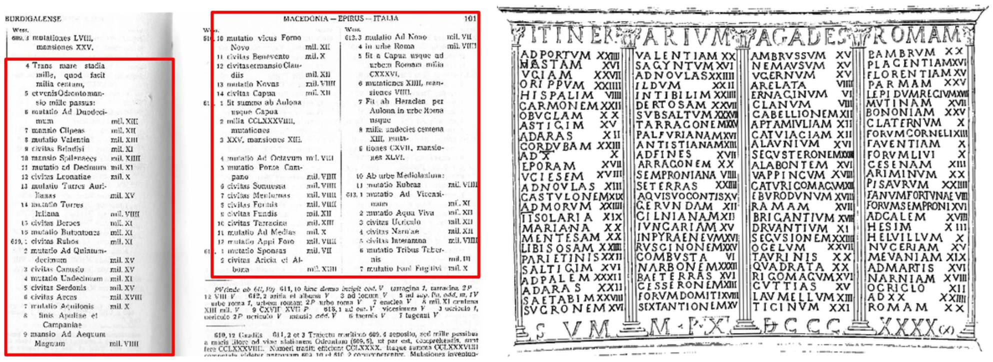
ACM Reference Format:
Rui Santos, Patricia Murrieta-Flores and Bruno Martins, 2017. An Automated Approach for Geocoding Tabular Itineraries. *ACM Trans. Spatial Algorithms Syst.* 9, 4, Article 39 (October 2017), 23 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Historical itineraries, often accessible as tables or as sequential lists of names for the places that were visited in the context of a particular journey, are abundant resources and also important objects of study for humanities scholars [Szabó 2009; Blank and Henrich 2016; Murrieta-Flores et al. 2016], for instance providing 'snapshots' of particular socio-cultural events, 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

Author's addresses: Rui Santos, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal; Patricia Murrieta-Flores, Digital Humanities Research Center, University of Chester, Chester, United Kingdom; Bruno Martins, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal.

DOI: 0000001.0000001



BURDIGALENSE

Wess.
609, 3 mutationes LVIII,
mansiones XXV.
4 Trans mare stadia mille, quod facit milia centum,
5 et venis Odronto mansio mille passus:
6 mutatio Ad Duodecimum mil. XIII
7 mansio Clipeas mil. XII
8 mutatio Valentia mil. XIII
9 civitas Brindisi mil. XI
10 mansio Spilenaees mil. XIIII
11 mutatio ad Decimum mil. XI
12 civitas Leonatiae mil. X
13 mutatio Turres Aurilianas mil. XV
14 mutatio Turres Iuliana mil. VIIII
15 civitas Beroes mil. XI
16 mutatio Butontones mil. XI
610, 1 civitas Rubos mil. XI
2 mutatio Ad Quintumdecimum mil. XV
3 civitas Canusio mil. XV
4 mutatio Undecimum mil. XI
5 civitas Serdonis mil. XV
6 civitas Aecas mil. XVIII
7 mutatio Aquilonis mil. X
8 finis Apuliae et Campaniae
9 mansio Ad Aequum Magnum mil. VIII

MACEDONIA — EPIRUS — ITALIA 101

Wess.
610, 10 mutatio vicus Forno Novo mil. XII
11 civitas Benevento mil. X
12 civitas et mansio Claudiis mil. XII
13 mutatio Novas mil. VIIII
14 civitas Capua mil. XII
611, 1 fit summa ab Aulona usque Capua
2 milia CCLXXXVIIII, mutationes
3 XXV, mansiones XIII.
4 mutatio Ad Octavum mil. VIII
5 mutatio Ponte Campano mil. VIIII
6 civitas Sonuessa mil. VIIII
7 civitas Menturnas mil. VIIII
8 civitas Formis mil. VIIII
9 civitas Fundis mil. XII
10 civitas Tarracina mil. XIII
11 mutatio Ad Medias mil. X
12 mutatio Appi Foro mil. VIIII
612, 1 mutatio Sponsas mil. VII
2 civitas Aricia et Albona mil. XIIII

Wess.
612, 3 mutatio Ad Nono mil. VII
4 in urbe Roma mil. VIIII
5 fit a Capua usque ad urbem Romam milia CXXXVI,
6 mutationes XIIII, mansiones VIIII.
7 Fit ab Heraclea per Aulona in urbe Roma usque
8 milia undecies centena XIII, muta-
9 tiones CXVII, mansiones XLVI.
10 Ab urbe Mediolanum:
11 mutatio Rubras mil. VIIII
613, 1 mutatio Ad Vicensimum mil. XI
2 mutatio Aqua Viva mil. XII
3 civitas Ucriculo mil. XII
4 civitas Narniae mil. XII
5 civitas Interamna mil. VIIII
6 mutatio Tribus Tabernis mil. III
7 mutatio Fani Fugitivi mil. X

PV (inde ab 611, 10) 611, 10 hinc demum incipit cod. V tarragina 1, tarracina 2P 12 VIII V 612, 2 aritia et albuna V 3 ad nonum V 5 ad sup. fin. add. m. 1V urbe roma 1, urbem romam 2P urbe roma V 7 eraclea V 8 mil. XI centena XIII mil. V 9 CXVII XVII P 613, 1 ad out. V vicesimum V 3 ucriculo 1, ucriculo 2P ucriculo V mansio add. V 6 ternis V 7 fugenui V

610, 12 Caudiis 611, 2 et 3 Traiectu maritimo 609, 4 seposito, sed mille passibus a maris litore ad viae stationem Odrontum (609, 5), ut par est, comprehensis, sunt fere CCLXXXVIIII. Numeri tradit efficiunt CCLXXXX. Itaque summa CCLXXXVIIII computata videtur [illegible]

ITINERARIVM A GADES ROMAM

AD PORTVM	XXIIII	VALENTIAM	XX	AMBRVSSVM	XV	BAMBRVM	XX
HASTAM	XVI	SAGVNTVM	XVI	NEMAVSVM	XV	PLACENTIAM	XVI
VGIAM	XXVII	ADNOVLAS	XXIIII	VGERNVM	XV	FLORENTIAM	XV
ORIPPVM	XXIIII	ILDVM	XXII	ARELATA	VIIII	PARMAM	XXV
HISPALIM	VIIII	INTIBILIM	XXIIII	ERNAGINVM	VI	LEPIDVM REGIVM	XVIII
CARMONEM	XXII	DERTOSAM	XXVII	GLANVM	VIIII	MVTINAM	XVII
OBVCLAM	XX	SVBSALTVM	XXXVIII	CABELLIONEM	XII	BONONIAM	XXV
ASTIGIM	XV	TARRAGONEM	XXXV	APTAM IVLIAM	XII	CLATERNVM	X
ADARAS	XII	PALFVRIANAM	XVI	CATVIACIAM	XII	FORVM CORNELI	XIII
CORDVBAM	XXIII	ANTISTIANAM	XIII	ALAVNIVM	XVI	FAVENTIAM	X
AD X	X	ADFINES	XVII	SEGVSTERONEM	XXIIII	FORVM LIVI	X
EPORAM	XVII	ARRAGONEM	XX	ALABONTEM	XVI	CESENAM	XIII
VCIESEM	XVIII	SEMPRONIANA	VIIII	VAPPINCVM	XVIII	ARIMINVM	XX
ADNOVLAS	XIII	SETERRAS	XXIIII	CATVRIGOMAGVM	XII	PISAVRVM	XXIIII
CASTVLONEM	XIX	AQVIS VOCONTIS	XV	EBVRODVNVM	XVIII	FANVM FORTVNAE	VIII
AD MORVM	XXIIII	GERVNDAM	XII	RAMAM	XVII	FORVM SEMPRONI	XVI
ISOLARIA	XIX	CILNIANAM	XII	BRIGANTIVM	XVIII	ADCALEM	XVIII
MARIANA	XX	IVNCARIAM	XV	DRVANTIVM	XI	HESIM	XIII
MENTESAM	XX	INPYRAENEVM	XVI	SEGVSIONEM	XXIIII	HELVILLVM	X
LIBISOSAM	XXIII	RVSCINONEM	XXV	OGELVM	XXVII	NVCERIAM	XV
PARIETINIS	XXII	COMBVSTA	VI	TAVRINIS	XX	MEVANIAM	XIX
SALTIGIM	XVI	NARBONEM	XXXII	QVADRATA	XX	ADMARTIS	XVI
ADPALEM	XXXII	BAETERRAS	XVI	RIGOMAGVM	XVI	NARNIAM	XVIII
ADARAS	XXII	CESSERONEM	XIII	CVTTIAS	XV	OCRICLO	XII
SAETABIM	XXVIII	FORVM DOMITI	XVIII	LAVMELLVM	XIII	AD XX	XXIIII
SVCRONEM	XVI	SEXTANTIONEM	XV	TICINVM	XXI	ROMAM	XX

S VM M·P·X DCCC XXXX

Fig. 1. Two examples of historical itineraries.

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] and the Internet Archive[3], or in the context of Digital Humanities projects like Pelagios[4]. The left 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 right 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, inscribed on their outside with 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 [Blank and Henrich 2015; 2016; Adelfio and Samet 2014; Zhang et al. 2012; Moncla et al. 2016]).

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 [Navarro 2001; Recchia and Louwerse 2013] to look for candidate disambiguations in a large-coverage gazetteer. State-of-the-art string matching methods [Santos et al. 2017a; Santos et al. 2017b], leveraging supervised machine learning for combining multiple similarity metrics or, alternatively, a deep neural network, can then optionally be used to further filter/restrict the set of candidates associated to each place in the itinerary. A least-cost path

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

between pairs of candidates, visited in sequence over the itinerary, is afterwards estimated on the third stage. At this stage we can either consider a geodesic path over the Earth's surface, or a least-cost path calculation method that leverages terrain slope for estimating movement costs [Douglas 1994; Murrieta-Flores 2012]. 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 [Viterbi 1967; Forney 1973], 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 instances, corresponding to sequences of varied lengths, associated 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. Moreover, the usage of least-cost paths for reconstruting the most likely routes between pairs of locations can enable new inquiries and inferences about historical routes. Our work shows that 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 the two cost optimization procedures with the string matching method leveraging a deep neural network.

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 cost optimization procedures. 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 directions for future work.

2. RELATED WORK

Previous research with significant similarities towards the work that is presented in this article has been previously reported [Blank and Henrich 2015; 2016], 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).

In a first study, Blank and Henrich [2015] 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

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

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 [1990] 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 for representing a possible toponym disambiguation. The edges in the graph encode the possible sequences of place visits in 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 [2015] 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, Blank and Henrich [2016] experimented with several 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 the disambiguation 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 bigrams and trigrams, (iii) a method based on character skipgrams, (iv) the DAS distance metric, previously proposed by Kilinç specifically for toponym matching [Kilinç 2016], 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 again 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 expands the disambiguation candidates in decreasing order from the best to the worst candidates.

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%. 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 grants a better accuracy, while expanding first the best candidate grants 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 [2014] 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 geotagger 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 geotagging method based on spatial coherence, inspired on a previous publication [Adelfio and Samet 2013]. 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 often 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 $L = \{l_1 l_2 \ldots l_n >\}$ 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 as $\epsilon_2(L)$ in Equation 2.

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

$$\epsilon_2(L) = \frac{1}{n-22} \sum_{i=1}^{n-3} \sum_{j=i+2}^{n-1} \delta_{i,j}(L) \tag{2}$$

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. [2016] described a process for annotating spatial entities in text that, taking inspiration on previous research addressing toponym resolution [Moncla et al. 2014; Freire et al. 2011; Lieberman and Samet 2012; Speriosu and Baldridge 2013; Santos et al. 2015], 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 to-

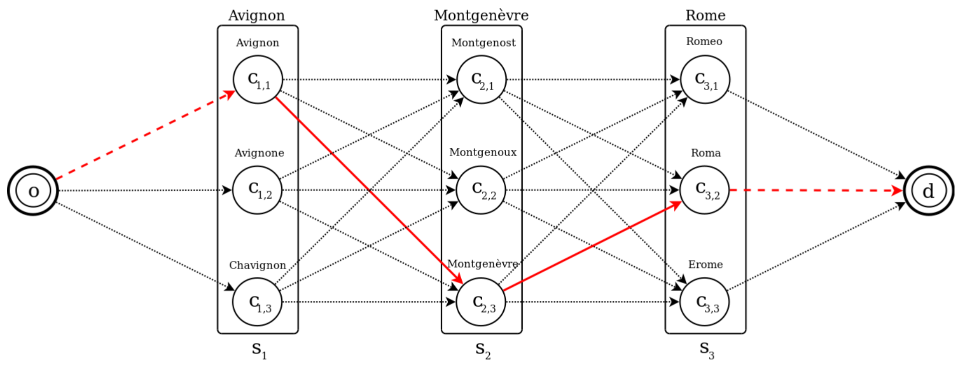


Fig. 2. Example of a trellis encoding an itinerary that involves three different toponyms.

ponyms 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 from 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 minimum 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 minimum 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 text descriptions for 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 conceptually simpler, sound, effective, 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 procedures. 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 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 20 most similar candidates.

(2) The list of candidate disambiguations can optionally be re-ranked through state-of-the-art string matching procedures, leveraging supervised machine learning for combining multiple similarity metrics [Santos et al. 2017a], or instead leveraging a deep neural network for comparing the strings [Santos et al. 2017b]. The string matching method combining multiple metrics essentially corresponds to an ensemble of decision trees (i.e., a random forest classifier) that verifies if a pair of toponyms matches 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 descriptive features for the pair. 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. The method based on a deep neural network was trained over the same dataset, and on comparative experiments achieved a accuracy of 88.71. Section 3.1 details the considered string matching procedures. The candidates are re-ranked according to the confidence of the classifiers on making assignments to the positive class (i.e., the confidence of matching decisions) 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 10). Note that result re-ranking is an optional step and, in some of our experiments, we directly used the top 20 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 either on an approach based on the shortest surface-path (i.e., geodesic) between the pairs of locations, using Vincenty's formulae for calculating the corresponding distance [Vincenty 1975], or on a method based on least-cost path computation that leverages a raster encoding terrain slope, using the A* search algorithm to find the most likely route between two locations in the raster grid [Yu et al. 2003; Douglas 1994; Murrieta-Flores 2012; Etherington 2016]. Section 3.2 details the least-cost path estimation method.

(4) Compute the most likely candidate for each place in the itinerary (i.e., globally disambiguate the toponyms in the 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 [Viterbi 1967; Forney 1973], frequently used in the context of decoding sequences with hidden Markov models, can be adapted to compute the sequence of best candidate disambiguations. Section 3.3 details the considered method.

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

3.1. A Deep Learning Method for Approximate String Matching

A typical approach for performing toponym matching involves computing a string similarity measure between the pair of names that are to be matched. For instance, the edit distance [Damerau 1964; Levenshtein 1966] or the Jaro-Winkler metric [Winkler 1990] are commonly used measures, but they need previous calibration upon its use. A threshold must be fine-tuned for effective classification. Past research focused both on toponym matching [Recchia and Louwerse 2013] and on person name matching [Cohen et al. 2003; Christen 2006; Moreau et al. 2008; Varol and Bayrak 2012] suggest that the performance of different string similarity algorithms is task-dependent, and that there is no single best technique.

An approach for performing toponym matching in this article uses a supervised machine learning algorithm denominated Random Forests. It operates by independently inferring a multitude of decision trees at training time, afterwards outputting the class that is the mode of the classes returned by the individual trees [Breiman 2001]. Each tree in the ensemble is trained over a random sample, taken with replacements, of instances from the training set. The process of inferring the trees also adds variability to each tree in the ensemble by selecting, at each candidate split in the learning process, a random subset of the features. The input instances are formed of different feature values. In this case, the features were formed of thirteen string similarity measures: Damerau-Levenshtein, Jaro, Jaro-Winkler, Reversed Jaro-Winkler, Sorted Jaro-Winkler, Permuted Jaro-Winkler, Cosine, Jaccard, Dice, Adapted Jaccard, Monge-Elkan, Soft-Jaccard, and a version of the procedure of Davis and De Salles [2007]. Another study addresses the details of this approach [Santos et al. 2017a].

The specific neural network architecture that is used in this article for addressing the toponym matching problem, where recurrent nodes are perhaps the most important components, is illustrated in Figure 3. This architecture takes its inspiration on models that have been previously proposed for natural language inference and for computing sentence similarities [Bowman et al. 2015; Rocktäschel et al. 2016; Yin et al. 2015; Wan et al. 2016; Liu et al. 2016; Mueller and Thyagarajan 2016]. The remaining paragraphs of this section briefly describe the neural network model and, for more information, the reader can refer to [Santos et al. 2017b].

The input to the network are two sequences of binary vectors that represent the strings to be compared. The strings are first converted to a unicode canonical normalized format (i.e., we use a fully decomposed UTF-8 representation, in which all combining character marks are placed in a pre-specified order) and they are also padded with a special symbol that denotes the beginning and termination of the toponym. The normalized strings are then represented as a sequence of one-hot binary vectors, in which a single bit is set to one for each byte in the sequence that corresponds to the unicode normalized string.

The binary vectors are provided as input to bi-directional Gated Recurrent Units (GRUs), which produce a vector of real values, also referred to as an *embedding*, for each of the toponyms being compared [Cho et al. 2014]. Bi-directional GRUs work by concatenating the outputs of two GRUs, one processing the sequence from left to right and the other from right to left [Schuster and Paliwal 1997]. Our neural network architecture actually uses two different layers of bi-directional recurrent units. The first bi-directional GRU layer generates a sequence of real-valued vectors, that is then passed as input to the second bi-directional GRU layer. The second bi-directional GRU layer outputs a single embedding for the input, resulting from the concatenation of the last outputs that are produced by the GRUs that process the input sequences in each direction.

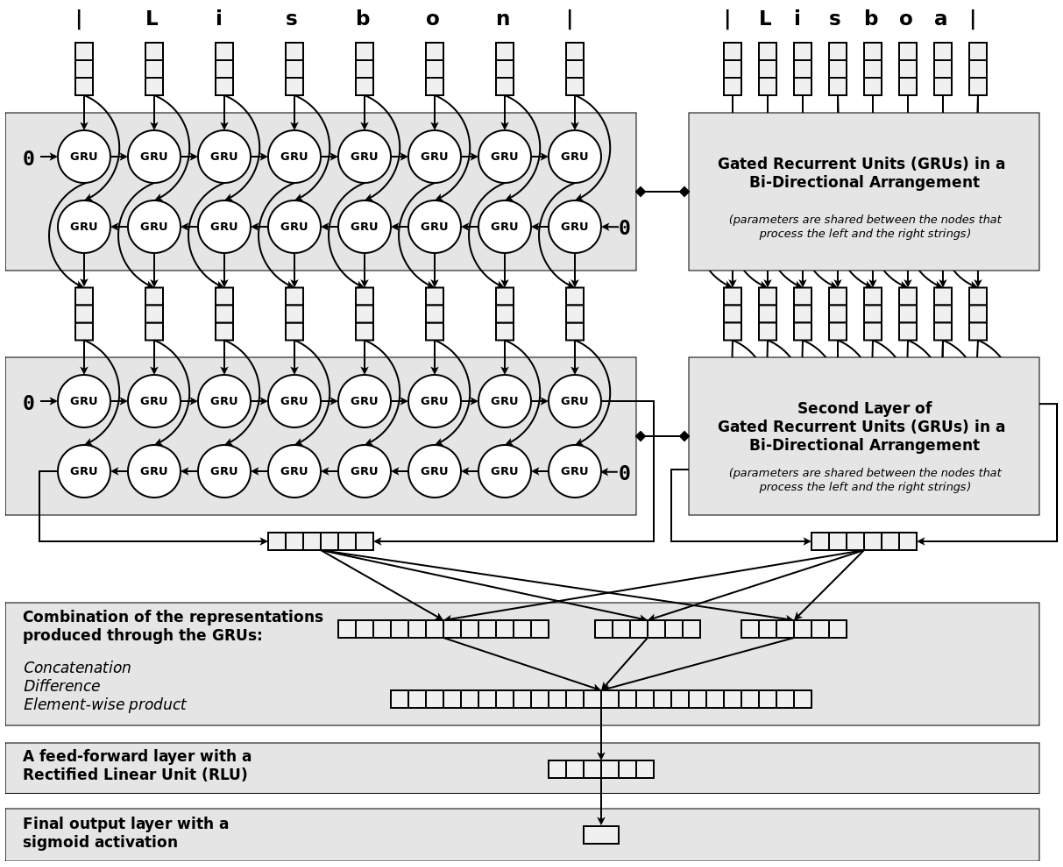

| L i s b o n |
| L i s b o a |
GRU
0
Gated Recurrent Units (GRUs) in a Bi-Directional Arrangement
(parameters are shared between the nodes that process the left and the right strings)
Second Layer of Gated Recurrent Units (GRUs) in a Bi-Directional Arrangement
(parameters are shared between the nodes that process the left and the right strings)
Combination of the representations produced through the GRUs:
Concatenation
Difference
Element-wise product
A feed-forward layer with a Rectified Linear Unit (RLU)
Final output layer with a sigmoid activation

Fig. 3. The neural network architecture proposed to address toponym matching.

Notice that each of the input toponyms is conceptually processed by individual recurrent layers, in order to produce a compact representation for its contents, although the parameters of these GRUs are shared across the parts of the network that process each input toponym. In other words, the two layers of bi-directional GRUs, which are used to process each of the input strings, have the same set of parameters. In the literature, these network architectures, where different parts have their parameters tied, are typically referred to as siamese networks [Mueller and Thyagarajan 2016].

The two embeddings produced by the bi-directional GRU layers are then combined and compared through a set of different operations. Taking inspiration on the network architecture from Mou et al. [2016], proposed to address the problem of natural language inference, we produce a combined representation for the pair of toponyms resulting from (i) the concatenation of the embedding vectors, (ii) the element-wise product of the embedding vectors, and (iii) the difference between the embedding vectors. This combined representation is then passed as input to feed-forward network layers: a first layer that uses a simple combination of the inputs together with a non-linear activation function (i.e., a rectified linear unit), followed by another simple layer that produces the final output and that uses a sigmoid activation function.

The entire network is trained end-to-end through the back-propagation [Rumelhart et al. 1988] in combination with the Adam optimization algorithm [Kingma and Ba 2015], using binary cross-entropy as the loss function to be optimized. In order to control overfitting and improve the generalization capabilities of the classification model, we use dropout regularization with a probability of 0.1 between each layer of the proposed neural network architecture. Dropout regularization is a simple procedure based on randomly dropping units, along with their connections, from the neural network during training. Each unit is dropped with a fixed probability p independent of other units, effectively preventing the network units from co-adapting too much [Srivastava et al. 2014].

Both the Random Forests and the neural network architecture have been evaluated on the same dataset. Using the Geonames gazetteer, a dataset with five million examples was constructed featuring half of the examples as positive matches and the other half as non-matching examples. It is remarkable the richness of different alphabets. While the majority are Latin based characters, more than 1.5 million examples feature at least one toponym written in other alphabet. The evaluation process was the same for both approaches using a 2-fold cross-validation experiment, i.e., the dataset was split in two parts. The algorithm is trained on one part and evaluated on the other, and then the opposite. The final accuracy result results from the average between the two results. The best performing method was the neural network architecture achieving an accuracy of 88.71, while the Random Forests settled on 78.67. The gain for the neural network can be largely attributed to its particularly effectiveness in capturing the complex character transliterations involved in some pairs of toponyms. For example when matching Asian or Arabic toponyms against their Western transliterations, which would be almost impossible for the similarity metrics as they do not detect this sort of patterns. In this article, we use the same models trained in past research [Santos et al. 2017a; Santos et al. 2017b], therefore, between the two confidence values obtained evaluating pair of toponyms, we always use the highest one.

3.2. Least-Cost Path Analysis for Inferring Likely Routes Between Pairs of Locations

Least-cost path analysis is fundamental spatial analysis method that aims at determining the most cost-effective route between a source and destination (e.g., the path between two adjacent locations in an itinerary, that costs the least to those traveling along it). The method is extensively used in engineering applications related to infrastructure planning (e.g., for pipeline routing or roadway planning), and also on computational archaeology and digital humanities studies [Gregory and Murrieta-Flores 2016; White and Surface-Evans 2012; Herzog 2014].

The procedure is based on a cost surface (i.e., a raster representation of the geographic space) where each of the cells encodes the cost, per unit distance, of passing through the corresponding terrain [Douglas 1994; Etherington 2016]. We specifically used two sets of input data, derived from satellite remote sensing procedures, to build an isotopic raster grid (i.e., a grid where costs are uniform for all directions [Xu and Jr. 1995]) with cells encoding traversal costs, namely water cover information (i.e., the cost associated to traversing water should be high) and a digital terrain elevation model (i.e., the traversal cost should be proportional to the topographic slope, as derived from a digital elevation model).

The water cover information was obtained from the MODIS land-water mask (MOD44W) product, made available in the Global Land Cover Facility (GLCF) website at a resolution of 250 meters per cell [Carroll et al. 2009; Salomon et al. 2004]. This raster provides a mask between land and water, including both the inland water bodies such as rivers and lakes, and the ocean. The data on terrain elevation was, in turn, obtained from the global digital surface model released by the Japan Aerospace

Exploration Agency [Tadono et al. 2014; Takaku et al. 2016]. This is currently the most precise global-scale terrain elevation dataset, being publicly available at a resolution of approximately 30-meters per cell. The GDAL[8] set of tools for analyzing and visualizing digital elevation models was used for computing the terrain slope (i.e., the percent slope, obtained through an algorithm originally outlined by Horn [1981]) from the elevation data at the resolution of 30-meters per cell. These results were then re-sampled to the resolution of 250 meters per cell (i.e., a resolution that is appropriate for our particular application, and that at the same time does not result in a significant computational effort when computing least-cost paths), and finally post-processed with basis on the raster encoding water coverage, so that cells corresponding to water were assigned to the maximum slope value.

It is important to notice that the aforementioned procedure is relatively naive in terms of the underlying assumptions that govern how travelers prefer particular routes, given that the final raster encoding traversal costs is exclusively derived from modern data on terrain coverage and elevation. We also restricted our procedure to using an isotropic cost surface, although this is a crude simplification (e.g., when terrain surfaces are complex, slopes in different directions are not constant, and traversal costs are usually more accurately modeled by taking anisotropic cost surfaces into consideration). For future work, we plan to experiment with more sophisticated cost surfaces, for instance by taking into account other sources of ancillary data (e.g., historical land coverage information [Horn 1981] or proximity towards historical highly populated places). However, the development of a cost surface that takes into account anisotropy and multiple contributing factors, better capturing the constraints involved in historical routes, would likely involve a significant effort from domain experts, e.g. for tuning the contribution of different parameters.

After building the isotropic raster with the traversal costs associated to each cell, we use a standard search algorithm based on dynamic programming for finding the least-cost path. The A* search algorithm, which is an extension of Edsger Dijkstra's 1959 algorithm that achieves better performance by using heuristics guide the search, is one of the most widely used algorithms for finding least-cost paths. The algorithm starts by creating two lists for storing references to raster cells, namely an open cell list and a closed cell list. Initially, the open list contains only the source cell. We take from the open list the element n with the least estimated cost towards the destination cell, minimizing the following function where $gScore(n)$ is the cost of the path from the start node to n (i.e., zero in case n is the origin, and otherwise corresponding to the number of cells than need to be traversed from the origin until the n, multiplied by the diagonal length of each cell), and $geo_distance(n)$ is a heuristic that estimates the cost of the cheapest path from n to the goal (i.e., the geospatial distance towards the destination cell, computed through Vincenty's formulae).

$$fScore(n) = gScore(n) + geo\_distance(n) \tag{3}$$

From the open list element n with the least value for $fScore(n)$, we generate all possible successors (i.e., the 8 immediate neighbors in the raster grid). If one of the successors is the destination, we can stop the search, and use backtracking to recover the least-cost path. Otherwise, for each successor, we retrieve its traversal cost and add the node to the open list. At the end of this iteration, we put the cell we took off the open list on the closed list. The pseudocode in Algorithm 1 details the aforementioned procedure. Notice that we may search the same point a few times, but only if the new path is more promising than the last time we searched it.

[8]http://www.gdal.org/gdaldem.html

ALGORITHM 1: The A* implementation used for Least-Cost Path Analysis.

```
input: a $start$ and a $goal$ nodes
output: a path $total\_path$ of nodes in case of success, failure otherwise
function A*(start, goal)
    $closedList \leftarrow \{\}$
    $openList \leftarrow \{start\}$
    $cameFrom \leftarrow theemptymap$
    $gScore[\,\cdot\,] \leftarrow +\infty$
    $gScore[start] \leftarrow 0$
    $fScore[\,\cdot\,] \leftarrow +\infty$
    $fScore[start] \leftarrow geo\_distance(start)$
    while $openList$ is not $empty$
        $current \leftarrow \min_{fScore[k]} openList[k]$
        if $current = goal$
            $total\_path \leftarrow [current]$
            while $current$ in $cameFrom.Keys$:
                $current \leftarrow cameFrom[current]$
                $total\_path.append(current)$
            return $total\_path$
        $openList.Remove(current)$
        $closedList.Add(current)$
        foreach $neighbor$ of $current$
            if $neighbor$ in $closedList$
                continue
            if $neighbor$ not in $openList$
                $openList.Add(neighbor)$
            $tentative\_gScore \leftarrow gScore[current] + (1 * diag\_length)$
            if $tentative\_gScore \geq gScore[neighbor]$
                continue
            $cameFrom[neighbor] \leftarrow current$
            $gScore[neighbor] \leftarrow tentative\_gScore$
            $fScore[neighbor] \leftarrow gScore[neighbor] + geo\_distance(neighbor)$
        end for
    return failure
end function
```

3.3. Using Dynamic Programming for Globally Disambiguating Toponyms in an Itinerary

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.

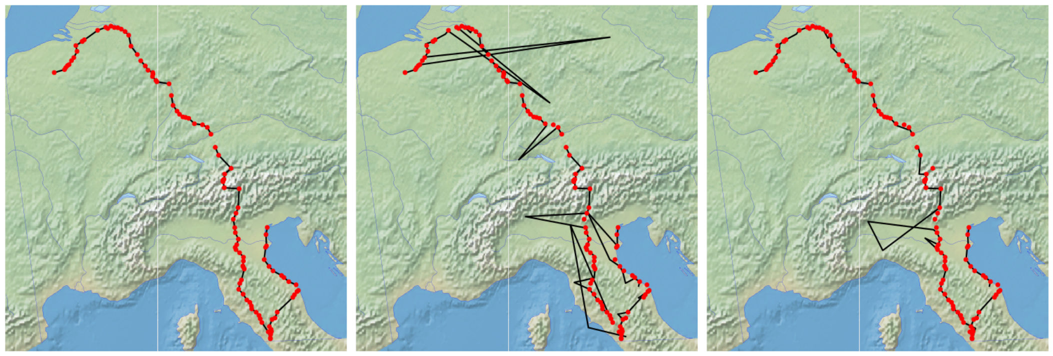


Fig. 4. Ground-truth trajectory associated to the pilgrimage of Jehan de Tournay from Valenciennes to Venice (on the left), compared to the estimated trajectory for the same itinerary, using straight-line routes of the baseline using only Whoosh (on the middle) or using Whoosh with random forests classifier and dynamic programming (on the right).

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 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.

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 pseudocode shown in Algorithm 1 formally describes the dynamic programming procedure.

ALGORITHM 2: 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 )
    $R_1[0, \cdot\,] \leftarrow 0$
    for $t \leftarrow 1, 2, ..., T$ do
        for $n \leftarrow 1, 2, ..., N$ 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
```

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. An itinerary is usually sub-divided into segments of different lengths. On average, each itinerary is divided in 7 segments and involves 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 region-based filter on the contents of the GeoNames gazetteer (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.

Table II lists the different itineraries that were considered in our tests, together with the corresponding number of places. In turn, Figure 4 presents a map illustrating the first itinerary from Table II, 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. Distances were measured with Vincenty's geodetic formulae [Vincenty 1975]. We also measured results in terms of disambiguation accuracy, thresholding the aforementioned geo-spatial distance with values rang-

Table I. Experimental results for the proposed method, in comparison against simpler baselines (average and median distance in kilometers).

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	450.92 ± 1294.95	0.83	0.43	0.55	0.71	0.82	0.84	0.89
Whoosh + RF + Population	372.42 ± 1125.31	0.74	0.44	0.57	0.71	0.83	0.84	0.90
Whoosh + DNN + Population	394.76 ± 1173.65	0.69	0.44	0.58	0.74	0.82	0.84	0.90
Whoosh + DP	135.26 ± 309.80	1.35	0.36	0.48	0.65	0.77	0.81	0.93
Whoosh + DP (String Sim.)	133.66 ± 306.54	1.35	0.36	0.47	0.65	0.76	0.81	0.93
Whoosh + RF + DP	118.42 ± 323.77	1.04	0.40	0.53	0.71	0.82	0.86	0.95
Whoosh + DNN + DP	136.66 ± 360.96	1.18	0.39	0.51	0.70	0.82	0.86	0.95
Whoosh + LCP + DP	152.53 ± 332.30	1.69	0.35	0.45	0.62	0.73	0.78	0.92

ing 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.

Table I presents the results obtained with the proposed method, over the full set of annotated itineraries and contrasting the results against simpler baselines. The first four rows in Table I 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 bigrams and trigrams.
(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.
(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 [Santos et al. 2017a], 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 20 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 disambiguation 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 same process was done for the neural network approach (i.e., it was also trained on the same set of toponyms whose classification confidence values are used for ordering).

The following four rows in Table I (i.e., from the fifth to the eighth line) are alternative methods using dynamic programming (DP), where the first line, of the four, corresponds to a baseline method in which the top 10 candidates retrieved by Whoosh are considered at each slice of the trellis (i.e., not using Step 2 and 4 of the method described in Section 3, thus considering fewer candidates). Also, in the following line, we have a variation of the same method, but reusing the string similarity score combined with the distance to compose a cost value. The string similarity score is transformed to a distance function using a logarithmic transformation. Adapting the Algorithm 2, this line's approach replaces the intermediate cost of $R_1[t,n]$ for:

$$R_1[t,n] \leftarrow \min_k (R_1[t-1,k] + \Delta(s_{t-1,k}, s_{t,n})) + d(s_{t,n}, o_t) \qquad (4)$$

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. Then, the last two lines of the same group of four include the Step 2 in its pipeline, using, respectively, the Random Forest and the Deep Neural Network approaches for the candidate disambiguation.The last line in Table I reports to the complete system, using all the steps described in Section 3 including the least-cost path analysis.

The results in Table I 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 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. It is 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 for that toponym). In these cases, the first baseline in Table I (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 method corresponding to the second row in Table I achieved the overall best median distance). 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 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 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 II summarizes the results with the complete method outlined in Section 3, for each of the 24 individual itineraries in the considered dataset. In Figure 4, using the same method, we also present a map illustrating the ground-truth and the estimated trajectory for the first itinerary in Table II. These results again confirm that methods leveraging dynamic programming can indeed be quite accurate, with the average distance ranging between 14 and 364 Kilometers. 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.

Figure 5 presents a violin plot [Hintze and Nelson 1998] with the distribution of the error values measured for the alternative algorithms in Table I, in terms of the geospatial distance between the ground-truth coordinates of each toponym in the different itineraries, and the coordinates of the resulting disambiguations. Figure 6 presents a similar plot, in this case illustrating the errors for the 5 individual itineraries at Table II 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 I). 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 very low errors. The distributions, particularly on Figure 6, also illustrate the fact that cost optimization with dynamic programming contributed to lowering the mean errors.

In the context of the least-cost path analysis application, only one approach was tested and it is reported in Table III. While it was part of our plan to do more extensive tests, the conditions needed for computing least-cost paths in an efficient way were not achieved. The lack of efficiency determined to be impossible gathering results with the rest of the approaches. Table III shows that using the dynamic programming baseline with least-cost path analysis achieves results almost on pair with the application of the machine learning approaches. This is a great promise for future work on combining both methods.

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 [Melo and Martins 2016; Wing 2016], toponym resolution [Speriosu and Baldridge 2013; Moncla et al. 2014; Santos et al. 2015; Gregory et al. 2015; Rupp et al. 2013], or others [Freire et al. 2011; Derungs and Purves 2014]) is increasingly being used in the Digital Humanities and related fields [Gregory and Murrieta-Flores 2016; Wing 2016], 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. For instance, besides the historical itinerary data made available by Peter Robins, there are many other examples of datasets that can could have been used, 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). A particular example would be the dataset from the al-Ṯurayyā Gazetteer[9], 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*.

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. In our study we used a least-cost path estimation method based on applying the A* algorithm over an anisotropic cost surface built through heuristics for combining two sources of information (i.e., terrain slope and water bodies). For future work, first, we plan to conclude the current experiments, which include using the string similarity approaches with the least-cost path estimation. Then experiment with the incorpora-

[9]https://althurayya.github.io/

Table II. Results for each of the 24 itineraries with the complete procedure given in Section 3 (average and median distance in kilometers).

Itinerary	# Places (Segments)	Average Distance	Median Distance	Accuracy (<)					
				500m	1km	5km	50km	100km	500km
Jehan de Tournay, Valenciennes-Venice, 1488	104 (7)	6.98 ± 14.77	0.85	0.29	0.56	0.82	0.98	0.99	1.00
Anonymous, Bordeaux-Milan, 333	130 (7)	189.49 ± 548.23	0.64	0.41	0.52	0.67	0.78	0.81	0.92
Bertrandon de la Broquière, Ghent-Dijon, 1432-1433	51 (2)	53.42 ± 88.51	1.21	0.28	0.43	0.63	0.76	0.79	1.00
Bruges, road inventory, 15th cent.	770 (39)	104.90 ± 215.78	0.78	0.42	0.53	0.69	0.80	0.83	0.93
Nompar de Caumont, Caumont-Fisterra, 1417	61 (3)	208.82 ± 423.6	0.87	0.39	0.48	0.53	0.58	0.79	0.90
Nompar de Caumont, Caumont, 1418-1419	89 (2)	258.78 ± 1386.91	0.48	0.51	0.69	0.83	0.92	0.94	0.94
Codex Calixtinus, Aquitaine-Santiago, ca. 1140	88 (3)	164.38 ± 246.48	1.10	0.49	0.57	0.73	0.75	0.77	0.86
Emo, Frisia-Rome-Cologne, 1211-1212	43 (2)	58.29 ± 174.78	2.33	0.25	0.36	0.49	0.84	0.90	0.99
Charles Estienne, France, 1552-1553	623 (17)	245.61 ± 492.05	0.69	0.43	0.55	0.73	0.81	0.85	0.94
Arnold von Harff, Cologne-Cologne, 1496-1499	419 (6)	63.09 ± 228.67	0.40	0.52	0.66	0.82	0.88	0.91	0.96
Anonymous, Avignon-Santiago, 14th cent.	59 (6)	243.12 ± 453.05	0.38	0.62	0.68	0.77	0.77	0.82	0.85
Künig von Vach, Einsiedeln-Aachen, 1495	130 (2)	65.70 ± 176.54	0.43	0.54	0.64	0.85	0.92	0.92	0.94
Nikulas of Munkathvera, Iceland-Apulia, 1151	85 (5)	139.61 ± 323.78	1.68	0.28	0.41	0.55	0.73	0.78	0.96
Matthew Paris, London-Apulia, 1250	103 (8)	142.3 ± 243.02	2.75	0.24	0.34	0.56	0.71	0.76	0.91
Pvrchas his Pilgrimage, Plymouth-Calais, ca.1425	142 (6)	37.44 ± 102.25	1.14	0.32	0.45	0.74	0.90	0.92	0.97
Eudes Rigaud, Rouen-Rouen, 1253-1254	110 (2)	24.58 ± 91.22	0.52	0.48	0.62	0.77	0.92	0.95	0.99
Romweg-Karte, Germany-Rome, 1500	226 (16)	47.32 ± 58.91	1.42	0.25	0.43	0.69	0.82	0.86	0.98
Sigeric, Rome-England, 990	79 (2)	234.70 ± 751.65	1.14	0.39	0.51	0.63	0.73	0.78	0.93
Annales Stadenses, Stade-Rome, ca.1250	217 (6)	68.94 ± 254.08	0.94	0.41	0.54	0.71	0.86	0.90	0.98
Barthélemy Bonis, Avignon-Rome, 1350	45 (1)	57.5 ± 192.21	0.39	0.53	0.62	0.76	0.89	0.89	0.96
Adam of Usk, Bergen op Zoom-Bruges, 1402-1406	49 (2)	28.42 ± 67.95	2.10	0.29	0.43	0.7	0.88	0.93	1.00
Pedro Juan Villuga, Spain, 1546	225 (8)	253.18 ± 707.54	0.92	0.41	0.51	0.74	0.77	0.81	0.92
William Wey 1st journey, Calais-Calais, 1458	104 (4)	105.65 ± 440.63	0.78	0.45	0.64	0.76	0.90	0.94	0.97
William Wey 2nd journey, Eton-Venice, 1462	47 (2)	39.74 ± 87.78	0.91	0.29	0.49	0.80	0.85	0.85	1.00
Average	167 (7)	118.42 ± 323.77	1.04	0.40	0.53	0.71	0.82	0.86	0.95

tion of additional sources of information, including material land-coverage. Moreover, 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 [Tamar et al. 2016] for inferring the parameters of a model capable of reconstructing movement decisions, with basis on raster data sources (e.g., historical land-coverage information from the Historic Land Dynamics Assessment project [Fuchs et al. 2012]).

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 over-

Table III. Results for the 24 itineraries with the least-cost path procedure presented (average and median distance in kilometers).

Itinerary	# Places (Segments)	Average Distance	Median Distance	Accuracy (<) 500m	1km	5km	50km	100km	500km
Jehan de Tournay, Valenciennes-Venice, 1488	104 (7)	87.27 ± 178.24	1.12	0.29	0.46	0.67	0.81	0.81	0.95
Anonymous, Bordeaux-Milan, 333	130 (7)	515.88 ± 1110.11	1.69	0.28	0.37	0.47	0.61	0.63	0.81
Bertrandon de la Broquière, Ghent-Dijon, 1432-1433	51 (2)	85.82 ± 144.32	2.37	0.22	0.33	0.57	0.74	0.78	0.95
Bruges, road inventory, 15th cent.	770 (39)	141.64 ± 287.78	1.13	0.34	0.43	0.59	0.67	0.73	0.93
Nompar de Caumont, Caumont-Fisterra, 1417	61 (3)	121.58 ± 220.78	0.53	0.35	0.37	0.49	0.6	0.75	0.91
Nompar de Caumont, Caumont, 1418-1419	89 (2)	69.13 ± 214.76	0.65	0.46	0.62	0.74	0.86	0.89	0.94
Codex Calixtinus, Aquitaine-Santiago, ca. 1140	88 (3)	163.85 ± 277.81	1.51	0.41	0.49	0.65	0.68	0.73	0.88
Emo, Frisia-Rome-Cologne, 1211-1212	43 (2)	113.87 ± 215.06	4.47	0.22	0.29	0.45	0.67	0.77	0.96
Charles Estienne, France, 1552-1553	623 (17)	137.24 ± 390.11	0.84	0.43	0.51	0.68	0.77	0.80	0.94
Arnold von Harff, Cologne-Cologne, 1496-1499	419 (6)	100.59 ± 294.85	0.53	0.50	0.58	0.69	0.78	0.82	0.95
Anonymous, Avignon-Santiago, 14th cent.	59 (6)	97.85 ± 182.85	0.53	0.47	0.54	0.65	0.72	0.78	0.97
Künig von Vach, Einsiedeln-Aachen, 1495	130 (2)	53.64 ± 147.66	0.42	0.57	0.66	0.82	0.87	0.88	0.97
Nikulas of Munkathvera, Iceland-Apulia, 1151	85 (5)	114.67 ± 360.79	1.36	0.23	0.37	0.61	0.84	0.9	0.94
Matthew Paris, London-Apulia, 1250	103 (8)	139.5 ± 219.16	1.10	0.31	0.41	0.63	0.72	0.75	0.88
Pvrchas his Pilgrimage, Plymouth-Calais, ca.1425	142 (6)	179.94 ± 453.97	1.57	0.31	0.40	0.71	0.82	0.86	0.92
Eudes Rigaud, Rouen-Rouen, 1253-1254	110 (2)	297.19 ± 1007.04	0.67	0.44	0.54	0.64	0.73	0.75	0.89
Romweg-Karte, Germany-Rome, 1500	226 (16)	154.40 ± 264.89	1.68	0.26	0.39	0.59	0.70	0.75	0.91
Sigeric, Rome-England, 990	79 (2)	164.94 ± 279.53	4.82	0.34	0.45	0.54	0.62	0.74	0.87
Annales Stadenses, Stade-Rome, ca.1250	217 (6)	129.28 ± 221.62	2.05	0.31	0.45	0.62	0.73	0.76	0.89
Barthélemy Bonis, Avignon-Rome, 1350	45 (1)	288.59 ± 517.10	4.62	0.38	0.40	0.51	0.56	0.58	0.84
Adam of Usk, Bergen op Zoom-Bruges, 1402-1406	49 (2)	171.19 ± 244.96	3.63	0.24	0.28	0.55	0.67	0.71	0.83
Pedro Juan Villuga, Spain, 1546	225 (8)	202.41 ± 495.47	1.09	0.46	0.57	0.75	0.79	0.83	0.91
William Wey 1st journey, Calais-Calais, 1458	104 (4)	48.21 ± 107.97	1.14	0.39	0.53	0.68	0.84	0.87	0.99
William Wey 2nd journey, Eton-Venice, 1462	47 (2)	82.0 ± 138.42	1.14	0.20	0.40	0.68	0.76	0.76	0.97
Average	167 (7)	152.53 ± 332.3	1.69	0.35	0.45	0.62	0.73	0.78	0.92

all 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 [Lember and Koloydenko 2014].

Another objective for future work involves expanding the set of experiments with the proposed method, for instance combining string similarity and geo-spatial distance when choosing the most likely paths (instead of just using string similarity for ranking the top candidates to be included), or considering different thresholds for the number of disambiguation candidates considered for each toponym. For now, we restricted our tests to 10 candidates per toponym, although increasing this number can perhaps

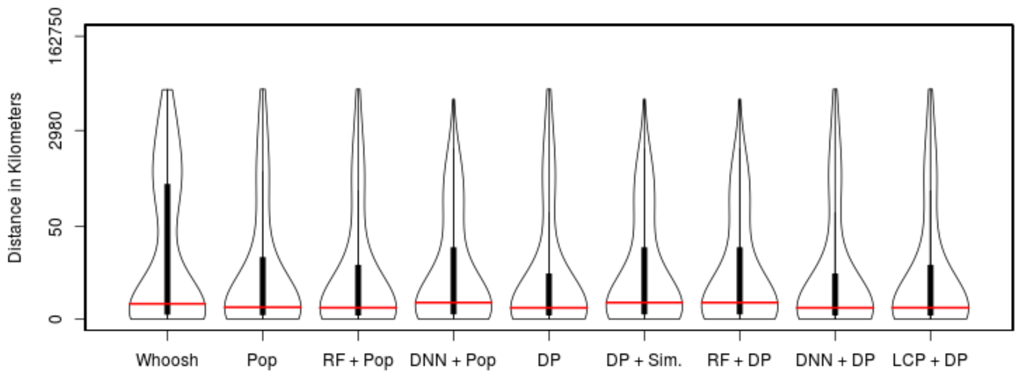


Fig. 5. Distribution for the errors obtained over each toponym, for each of the methods shown in Table I.

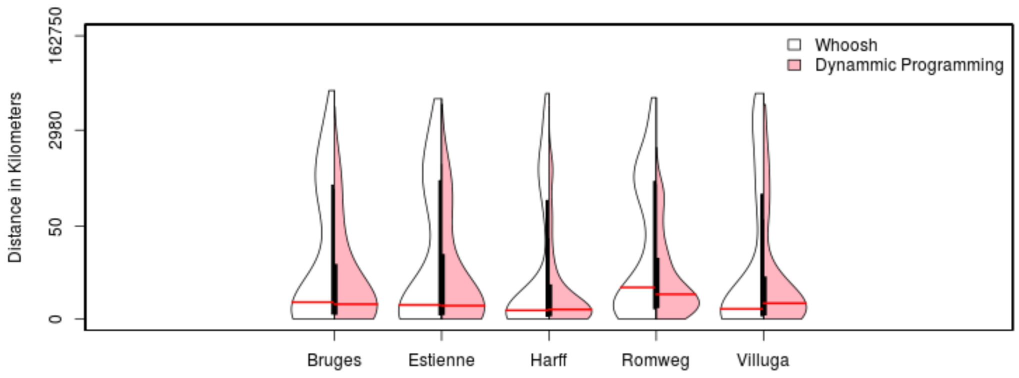


Fig. 6. Distribution for the errors obtained over each of the 5 longest itineraries shown in Table II.

lead to improvements on the overall results. 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) [Maleki et al. 2016].

ACKNOWLEDGMENTS

We would like to thank Pável Calado for his comments on preliminary versions of this work. We would also like to thank Peter Robins for openly publishing the historical itinerary data that supported our validation experiments.

This work was partially supported through Fundação para a Ciência e Tecnologia (FCT), through project grants with references PTDC/EEI-SCR/1743/2014 (Saturn) and CMUPERI/TIC/0046/2014 (GoLocal), as well as through the INESC-ID multi-annual funding from the PIDDAC programme (UID/CEC/50021/2013).

REFERENCES

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.

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.

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

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.

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

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

M L Carroll, J R Townshend, C M DiMiceli, P Noojipady, and R A Sohlberg. 2009. A new global raster water mask at 250 m resolution. *International Journal of Digital Earth* 2, 4 (2009), 291–308.

Kyunghyun Cho, B van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. *On the properties of neural machine translation: Encoder-decoder approaches*.

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

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

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

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

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, 6 (2014).

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, 3 (1994).

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

G D Forney. 1973. the Viterbi Algorithm. *Proc. IEEE* 61, 3 (1973).

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.

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).

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, 1 (2015).

Ian Gregory and Patricia Murrieta-Flores. 2016. *Doing Digital Humanities: Practice, Training, Research*. Routledge, Chapter Geographic.

Irmela Herzog. 2014. A review of case studies in archaeological least-cost analysis. *Archeologia e Calcolatori* 25 (2014).

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

B K P Horn. 1981. Hill shading and the reflectance map. *Proc. IEEE* 69, 1 (1981), 14–47.

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

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference for Learning Representations*.

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, 1 (2014).

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

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.

Yang Liu, Chengjie Sun, Lei Lin, and Xiaolong Wang. 2016. Learning Natural Language Inference using Bidirectional LSTM model and Inner-Attention. *arXiv preprint arXiv:1605.09090* (2016).

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

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

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, 6 (2016).

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.

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

Lili Mou, Rui Men, Ge Li, Yan Xu, Lu Zhang, Rui Yan, and Zhi Jin. 2016. Natural language inference by tree-based convolution and heuristic matching. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*. ACL.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI.

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.

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, 4 (2016).

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

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.

Tim Rocktäschel, Edward Grefenstette, Karl Moritz Hermann, Tomáš Kočisk\'y, and Phil Blunsom. 2016. Reasoning about entailment with neural attention. In *Proceedings of the International Conference on Learning Representations*.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling* 5, 3 (1988), 1.

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

J Salomon, J C F Hodges, M Friedl, C Schaaf, A Strahler, F Gao, A Schneider, X Zhang, N El Saleous, and R E Wolfe. 2004. Global land-water mask derived from MODIS Nadir BRDF-adjusted reflectances (NBAR) and the MODIS land cover algorithm. In *IGARSS 2004. 2004 IEEE International Geoscience and Remote Sensing Symposium*, Vol. 1. 241.

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

Rui Santos, Patricia Murrieta-Flores, Pável Calado, and Bruno Martins. 2017b. Toponym Matching Through Deep Neural Networks. *(currently under review)* (2017).

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

M Schuster and K K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45, 11 (1997). DOI:http://dx.doi.org/10.1109/78.650093

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*. ACL.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 15, 1 (2014).

Thomas Szabó. 2009. Die Itinerarforschung als Methode zur Erschlie{ß}ung des mittelalterlichen Stra{ß}ennetzes. *Die Welt der europ{ä}ischen Stra{ß}en - von der Antike bis zur fr{ü}hen Neuzeit* (2009).

T Tadono, H Ishida, F Oda, S Naito, K Minakawa, and H Iwamoto. 2014. Precise Global DEM Generation by ALOS PRISM. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* II-4 (2014), 71–76.

J Takaku, T Tadono, K Tsutsui, and M Ichikawa. 2016. VALIDATION OF "AW3D" GLOBAL DSM GENERATED FROM ALOS PRISM. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences* III-4 (2016), 25–31.

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.

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

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

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

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

D.A. White and S.L. Surface-Evans. 2012. *Least Cost Analysis of Social Landscapes: Archaeological Case Studies*. University of Utah Press.

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

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).

Jianping Xu and Richard G Lathrop Jr. 1995. Improving simulation accuracy of spread phenomena in a raster-based Geographic Information System. *International Journal of Geographical Information Systems* 9, 2 (1995), 153–168.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. ABCNN: Attention-based convolutional neural network for modeling sentence pairs. *arXiv preprint arXiv:1512.05193* (2015).

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, 4 (2003).

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.