Journeys of the Past: A Hidden Markov Approach to Georeferencing Historical Itineraries

Benedikt Budig
Chair for Computer Science I
University of Würzburg
benedikt.budig@uni-wuerzburg.de

Thomas C. van Dijk
Chair for Computer Science I
University of Würzburg
thomas.van.dijk@uni-wuerzburg.de

ABSTRACT

There are many historical itineraries that describe routes as a list of settlements and the travel distances along the way. They are an important source of information for various kinds of research in the humanities, providing insights into for example the development of human mobility and historical road networks.

In this paper, we develop an approach for aligning these itineraries with a modern gazetteer (database of places). We combine textual information (historical toponyms) and spatial information (travel distances) into a Hidden Markov model. Naively calculating a maximum likelihood explanation is slow, but careful algorithm engineering achieves high performance suitable for user interaction.

We demonstrate the practical potential of our approach by georeferencing 48 itineraries (containing 691 stops) from two important historical guidebooks published in 1563 and 1597: our approach is fast and accurate. Additionally, we show how to use sensitivity analysis to power an efficient user interface for quality assurance.

CCS CONCEPTS

• **Information systems → Geographic information systems**; **Information extraction**; **Users and interactive retrieval**;

KEYWORDS

Historical Itineraries, Deep Georeferencing, Itinerary Resolution, Toponyms, Optimization, Algorithms, Hidden Markov Models

ACM Reference format:
Benedikt Budig and Thomas C. van Dijk. 2017. Journeys of the Past: A Hidden Markov Approach to Georeferencing Historical Itineraries. In *Proceedings of 11th Workshop on Geographic Information Retrieval, Heidelberg, Germany, November 30-December 1, 2017 (GIR'17)*, 10 pages.
https://doi.org/10.1145/3155902.3155906

1 INTRODUCTION

Historical itineraries are fascinating documents that convey spatial information from the past. They describe routes by listing encountered settlements along the way, often including the travel distance between them. The information contained in these documents is of interest to several disciplines in the humanities and relevant for various research topics, including the investigation of early modern period road networks and the development of human mobility.

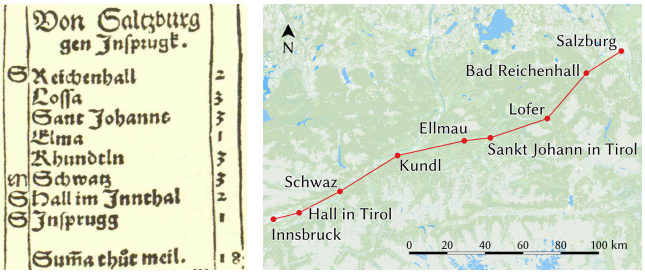

Figure 1: Left: Itinerary from the *Raißbüchlin*, describing the route from Salzburg to Innsbruck. Right: The same route on a modern map, showing corresponding modern places.

Georeferencing historical itineraries, that is, identifying the corresponding modern place for each historical stop, is a "tedious, but highly insightful" task that is not always undertaken [20, p. 9]. The main reason for not doing it is the amount of manual effort involved (using various sources of information, such as lexica and modern maps). Tools that support the process, while certainly useful, are insufficient to tip the balance for mass georeferencing: they merely provide visualization aids and usability improvements. The *Recogito* system [28], for example, provides a smooth user experience, but does not offer meaningful support for georeferencing itineraries.

In this paper, we present an approach that automates this information retrieval task to a large extent. This places our work in the scope of Chiang's research vision [9] of an intelligent pipeline for handling historical spatial data. In particular, our algorithm is able to assign historical stops in the itinerary to modern places with high accuracy. Our main contribution is a significant improvement over previous work on the same topic by Blank and Henrich [6]. Their algorithm is an ad-hoc heuristic without reasonable runtime guarantee. In contrast, our algorithm is based on proper mathematical modeling as an optimization problem. We give a polynomial-time algorithm to compute the optimal assignment and show that it is fast in practice. Additionally, we present an efficient user interaction for quality assurance, driven by sensitivity analysis.

Being able to automatically retrieve information from historical itineraries is beneficial for research practice, since a great number of historical itineraries remains in existence today. Among many others, the Austrian National Library alone has approximately 50 books of itineraries published between 1500 and 1800, with an additional

GIR'17, November 30-December 1, 2017, Heidelberg, Germany

ACM ISBN 978-1-4503-5338-0/17/11…$15.00
https://doi.org/10.1145/3155902.3155906

400 at SUB Göttingen, and 75 at HAB Wolfenbüttel. A particularly famous book is Jörg Gail's *Raißbüchlin*, a book of itineraries published in Augsburg in 1563, being the first independently printed guidebook[1] in the German language. [20, p. 1] It contains 161 individual routes throughout Europe, with a focus on Southern Germany. The itineraries are given as a sequence of stops, each of which has a place name and the travel distance from the previous stop. Some stops are also annotated with a class label that identifies them as a city, market town, village or monastery.

Figure 1 (left) shows Gail's itinerary from Salzburg to Innsbruck. Note that the destination Innsbruck is spelled in different ways even on this single page and neither time matches the modern spelling; in fact, none of the given toponyms equal their modern equivalents (right). Still, some phonetic similarity between corresponding place names is present, and the distances given on the right side of the page are plausible (at least in relation to each other).

The remainder of this paper is organized as follows. First we model the georeferencing task as an optimization problem (Section 3), for which we give an efficient algorithm (Section 4). We experimentally evaluate our approach (Section 5) and subsequently apply it to the Raißbüchlin in a case study (Section 6). For quality assurance of the georeferenced result, we use sensitivity analysis and introduce an efficient user interaction (Section 7).

2 RELATED WORK

The research presented in this paper is related to various fields. Most closely related is a series of papers by Blank and Henrich, which we will discuss first.

Historical Itineraries and Gazetteers. Blank and Henrich [5] recently introduced *itinerary resolution* as involving: (1) optical character recognition, (2) itinerary parsing, (3) toponym resolution, and (4) route finding. By route finding, the authors mean the problem of finding the actual paths underlying a particular itinerary.

In a second paper, Blank and Henrich [6] present a heuristic approach for step (3), matching an itinerary to a given gazetteer based on string distance and geometry. Their algorithm prunes the search space using textual and geospatial filters. This approach is reported to work well in practice, but lacks a clear optimization goal and requires runtimes of several minutes. Addressing an audience from the humanities, a further paper [7] presents another study of this approach, which however is not very accurate on the itinerary used for demonstration.

In 1974, Krüger [20] published a facsimile of the *Raißbüchlin*. This is our source for this document and the corresponding imagery in this paper. Together with the facsimile, Krüger provides an in-depth investigation of the *Raißbüchlin* in the context of contemporary itineraries from the perspective of historical road network research.

Southall et al. [29] postulate the need for historical gazetteers, by which they mean gazetteers that are enriched with historical places and name variations. They survey existing gazetteers and discuss requirements from a historian's standpoint, many of which would also be helpful for the task discussed in this paper. Note that our approach not only benefits from extensive historical gazetteers, but might also actively help adding additional name variations (by georeferencing itineraries that contain previously unknown toponyms). This concept is known as gazetteer enrichment and is highly relevant to the spatial humanities: for an overview, see [4].

Geocoding and Modern Itineraries. Geocoding – determining location based on textual description – is an active field of research in geographic information retrieval. For an extensive survey on automated geocoding of textual documents, see Melo and Martins [21].

Not all historical guidebooks have the tabular format of the *Raißbüchlin*, but many exhibit structure that may enable the automatic extraction of the toponyms (for example: Khan et al. [18]). In this paper we assume that any such processing is already done and require the itinerary to be given as a list of toponyms.

In terms of modern itineraries, Adelfio and Samet [1] present an approach to identify and extract modern itineraries from spreadsheets and websites. Moncla et al. [22] reconstruct itineraries from annotated text, for example on a corpus of hiking descriptions [23].

Historical Spellings. Handling historical spelling variants of toponyms is one of the main challenges in identifying corresponding modern places for itinerary entries. Ernst-Gerlach and Fuhr [11] present a system for text retrieval supporting non-standard historical spellings. Rather than applying a manually-created set of rules, we *learn* transformations from a training corpus. For this we use a lexicon of historical spelling variants of places in Franconia compiled by Von Reitzenstein [14][2]

Butler et al. [8] recently gave an overview of historical onomastic variations of place names in the context of geotagging. In particular, the authors discuss common issues with place names from sources from the 17th to the 19th century and how non-standardized spelling makes automated place recognition challenging.

String Distances and Toponyms. There is some research on matching modern toponyms (to each other) based on string distance. Recchia and Louwerse [26] compare various string similarity measures in this context. Kılınç [19] presents an approximate string matching approach for toponym matching and shows that it outperforms some of the traditional string distance measures.

There is less research about matching historical toponyms to modern ones. Blank and Henrich [6] evaluate 13 string distance measures on historical toponyms from itineraries. We use the probabilistic string edit distance introduced by Ristad and Yianilos [27], which has the advantage that it integrates well with our probabilistic modeling and can be trained for specific applications using semantically meaningful training data.

Connection to Map Matching. There are some conceptual similarities between our itinerary resolution problem and the more well-known problem of map matching. In their landmark paper on the topic, Newson and Krumm present "a novel, principled map matching algorithm" [24]. Our algorithm is similarly based on a probabilistic model and finding a most-likely path in a hidden Markov model (HMM), but the ingredients to the model are fairly different.

[1] Appropriately, *Raißbüchlin* literally translates to "travel book."

[2] There are other similar databases, e.g. THELO by the Academy of Sciences and Literature Mainz, http://www.personalschriften.de/datenbanken/thelo.html

3 PROBLEM STATEMENT AND MODELING

We now formalize the input to our georeferencing algorithm: an itinerary and a gazetteer. The itinerary $\mathcal{I}$ is a sequence of stops. Let $k = |\mathcal{I}|$. Each stop in the itinerary consists of a historical toponym (as a string), possibly the type of settlement (as a string), and a distance from the previous stop (a number).

The gazetteer $\mathcal{G}$ is a set of places. Let $n = |\mathcal{G}|$. Each place consists of a modern toponym (as a string) and a latitude/longitude pair. In case alternative or historical place names are available, we model these as additional gazetteer entries with the same geolocation.

This input contains several kinds of noise. Firstly, toponyms have changed over the centuries. Sometimes, they are seemingly unrelated at a string level (now it's Istanbul, not Constantinople): in this case the historical toponym must be in the gazetteer or we have no chance to georeference with high confidence. Luckily, in many cases the historical toponym is phonetically similar to the modern one. Transcription (or OCR) errors introduce further noise, but string similarity measures can be applied to some success.

The distances given in the itinerary are imprecise for a variety of reasons. We do not generally know what (if any) road network underlies the reported travel distances. The conversion factor between the distances in the itinerary and real-world geodistances is also not necessarily clear: see Section 6.4. Furthermore, in our itineraries, many distances are given as integers between 1 and 5, which is rather coarse-grained. Still, these numbers clearly contain information about where to find the places on a modern map.

We now combine the textual and spatial information into a Bayesian model for georeferencing sets of itineraries. When restricted to a single itinerary, this model reduces to a hidden Markov model. Future work could consider the potential benefit of combining information from multiple itineraries, but will likely face a harder inference problem.

Consider a single itinerary. We introduce three variables for each stop in the itinerary: the historical toponym $\mathbf{T}_i$ (domain: strings), the reported distance from the previous stop $\mathbf{D}_i$ (domain: reals), and the modern place $\mathbf{P}_i$ (domain: gazetteer $\mathcal{G}$). We have observed the first two variables at each stop and the third variable gives the solution to our problem, for which we will infer a most likely assignment. It may be noted that $\mathbf{T}_i$ and $\mathbf{D}_i$ have infinite domain, but this is not a problem since we have observed these variables and therefore they have singleton support.

As a modeling decision, we postulate the following independences. Conditioned on $\mathbf{P}_i$, that is, given a decision to georeference step i as a certain place: (a) other places are conditionally independent of $\mathbf{T}_i$ and $\mathbf{D}_i$: if we have decided on a place, it no longer matters what the itinerary says; (b) later places are conditionally independent of all places before $\mathbf{P}_i$, that is, $\mathbf{P}$ is Markov: if we have decided a place, it does not matter what came before; and, (c) the previous place $\mathbf{P}_{i-1}$ combined with the distance $\mathbf{D}_i$ is conditionally independent of $\mathbf{T}_i$: our a priori assessment of how historical and modern toponyms relate is independent of our a priori assessment of how reported distances in the itinerary relate to geodistances.

We start from a uniform prior on each of the place variables $\mathbf{P}_i$ and then *fuse* our evidence [25] using the conditional probability distributions given in the next three subsections. Each one can be fused separately because of the assumed independences.

3.1 Toponym Evidence

The influence of the historical toponym on the selected place is achieved through an evidential term $\mathbb{P}_{\mathrm{t}}(\,\mathbf{P}_i \,|\, \mathbf{T}_i\,)$, that is, the a priori conditional probability distribution over places in the gazetteer given a historical toponym. We set this distribution based on the modern toponyms in the gazetteer and the statistical string similarity measure of Ristad and Yianilos [27]. Their similarity measure has the advantage of being rigorously grounded in probability theory and being trainable by expectation maximization on an appropriate corpus. With $t_{\mathrm{modern}}(\mathbf{P}_i)$ the modern toponym of place $\mathbf{P}_i$, and $\mathrm{sim}_{RY}(\,\cdot\,,\,\cdot\,)$ the Ristad-Yianilos similarity, we let

$$\mathbb{P}_{\mathrm{t}}(\,\mathbf{P}_i \,|\, \mathbf{T}_i\,) \stackrel{\Delta}{=} \mathrm{sim}_{RY}(\,t_{\mathrm{modern}}(\mathbf{P}_i),\, \mathbf{T}_i\,). \tag{1}$$

Note that $\mathbf{T}_i$ is in fact observed; we set it equal to the historical toponym at stop i.

The basic version of Ristad and Yianilos's measure favors short strings over long ones and performs poorly on our data. Instead, we use their alternate model conditioned on length [27, Appendix B]. With regard to length, we state that every modern character independently corresponds to either zero, one or two characters of length in the historical toponym, with probabilities p_0, p_1 and $p_2 = 1 - p_0 - p_1$. Training the similarity measure and choosing appropriate values for p_0 and p_1 is discussed in Section 6.2.

3.2 Distance Evidence

The influence of distance information on the selected place is achieved through an evidential term $\mathbb{P}_{\mathrm{d}}(\,\mathbf{P}_i \,|\, \mathbf{P}_{i-1}, \mathbf{D}_i\,)$. Here we consider two consecutive places and the reported distance between them. Through the gazetteer's latitude/longitude pairs, we have the actual distance between these two places. (In the absence of information about a road network, we take the great-circle distance.)

We say the historical and modern distance should be approximately equal: we set the relative probabilities of places based on a normal distribution around an expected difference of zero. Here it is relevant that $\mathbf{D}_i$ is given in historical units: we multiply by some constant conversion factor λ to get modern units. (We will see in Section 6.4 that $\lambda = 7.5$ works well for the "historical German miles" in the test itineraries.)

With $\mathrm{dist}(\,\cdot\,,\,\cdot\,)$ the great-circle distance, and $\mathcal{N}(\,\cdot\,;\sigma)$ the normal distribution around zero with standard deviation σ, we let

$$\mathbb{P}_{\mathrm{d}}(\,\mathbf{P}_i \,|\, \mathbf{P}_{i-1}, \mathbf{D}_i\,) \stackrel{\Delta}{=} \mathcal{N}(\,\mathrm{dist}(\mathbf{P}_{i-1}, \mathbf{P}) - \lambda \cdot \mathbf{D}_i\,;\sigma_d). \tag{2}$$

The value of σ_d is discussed in Section 4.1. Note that $\mathbf{D}_i$ and $\mathbf{P}_{i-1}$ are known constants when evaluating this expression.

It is, in principle, possible to include the conversion factor λ as a variable to be inferred. However, this makes exact inference infeasible since all $\mathbf{P}_i$ become dependent. A MLESAC-based approach like that of Weinman [32] could work, but we have not found picking λ to be a problem in practice.

3.3 Bearing Evidence

An additional spatial term can be based on the bearing from $\mathbf{P}_{i-1}$ to $\mathbf{P}_i$. We take it to be independent of the other evidence terms.

Many historical itineraries describe routes with a fairly consistent bearing [7]. If we are given some overall bearing B for the route, then any two consecutive places should have approximately

this bearing. We model this using an evidential term $\mathbb{P}_b(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,)$. With $\mathrm{bear}(\cdot,\cdot)$ the initial geodetic bearing from the first to the second place, $\mathrm{diff}(\cdot,\cdot)$ the difference between two bearings, and $\mathcal{N}(\cdot;\sigma)$ as before, we let

$$\mathbb{P}_b(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,) \triangleq \mathcal{N}(\,\mathrm{diff}(\mathrm{bear}(\mathbf{P}_{i-1},\mathbf{P}),B)\,;\sigma_b). \tag{3}$$

The value of σ_b is discussed in Section 4.1.

The input as specified before does not contain B, but we can get reasonable values in various ways. We may require the user to pick a bearing, for example using a rough dragging gesture in a graphical user interface. We could hope to georeference the first and the last stop purely based on string similarity, since they are nearly always major cities. In our experiments in Section 5, we have used the initial bearing from first to last place, which we have manually georeferenced for this purpose. (We present experiments with and without bearing information.)

3.4 Representation as Hidden Markov Model

The Bayesian model above has an intuitive interpretation as a hidden Markov model: there is a historical journey (a sequence of actual places) that we have not observed, and these places have "emitted" observations in the form of historical toponyms written down in the itinerary. The reported distances and the bearing information get folded into the transition probabilities between hidden states. Putting some evidence in the emission and some in the transitions is justified since the resulting inference procedure for $\mathbf{P}_i$ is equivalent to the fusion rules of Bayesian inference [25].

As hidden states, we take the places $\mathbf{P}_i$. Each has as domain all places in the gazetteer. It remains to specify the transition distributions $\mathbb{P}(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,)$ and the emission distributions $\mathbb{P}(\,\mathbf{T}_i \mid \mathbf{P}_i\,)$. The emission distribution is given through the toponym evidence term:

$$\mathbb{P}(\,\mathbf{T}_i \mid \mathbf{P}_i\,) \triangleq \mathrm{sim}_{RY}(\,\mathbf{T}_i,\, t_{\mathrm{modern}}(\mathbf{P}_i)\,) \tag{4}$$

Note that the arguments to the string similarity function are flipped compared to Eq. (1).

The transition distributions are given by multiplying the distance and bearing evidence, which is valid by their assumed independence. Let $e_d = \mathrm{dist}(\mathbf{P}_{i-1},\mathbf{P}) - \lambda\cdot\mathbf{D}_i$ and $e_b = \mathrm{diff}(\mathrm{bear}(\mathbf{P}_{i-1},\mathbf{P}),B)$. Then:

$$\mathbb{P}(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,) \triangleq \mathcal{N}(e_d;\sigma_d)\cdot\mathcal{N}(e_b;\sigma_b) \tag{5}$$

We compute a maximum-likelihood sequence of states for this HMM and this is our georeferenced output: a globally most-likely set of values $\mathbf{P}_i$ incorporating all evidence.

4 ALGORITHM ENGINEERING

Inference in HMMs can famously be done using the Viterbi algorithm [30]. Directly applying it to our HMM yields the following.

Theorem 4.1. *A most-likely sequence of places can be computed in $O(kn^2)$ time and $O(kn)$ space.*

Our HMM has the curious property that the Markov chain is short (length k), but the state space is large (n gazetteer entries). This is reversed from the common situation and is unfortunate for the runtime, since the dependence on n is quadratic.

All algorithms were implemented in C++ and the experiments were run on an Intel® Core™ i5-4670 CPU at 3.4 GHz with 8 GB of RAM running Ubuntu 14.04.

4.1 Textbook Viterbi

As baseline we have implemented the "textbook" version of the Viterbi algorithm: eagerly filling a table of dynamic-programming values and back pointers. There are $O(kn)$ states and each takes $O(n)$ time to evaluate since it considers all previous states.

We are purely interested in the most-likely path and not its actual probability. Hence we need not normalize the transition distributions: every sequence of states involves exactly one term from each $\mathbb{P}(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,)$, so all paths are off by the same factor (namely the product of all missing normalization constants).

Next we apply the well-known transformation of taking logarithms and using addition, rather than working with raw probabilities and multiplication. Recall that $\mathbb{P}(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,)$ is the product of two normal distributions. Taking Eq. (5) and expanding the normal distributions, we get:

$$\mathbb{P}(\,\mathbf{P}_i \mid \mathbf{P}_{i-1}\,) \triangleq \frac{1}{\sqrt{2\pi\sigma_d^2}}\cdot\exp\left(\frac{-e_d^2}{2\sigma_d^2}\right)\cdot\frac{1}{\sqrt{2\pi\sigma_b^2}}\cdot\exp\left(\frac{-e_b^2}{2\sigma_b^2}\right) \tag{6}$$

Similarly to the normalization factors, we can drop the first and third factor since they are the same everywhere. Then, taking logarithms cancels the exponentiation and turns multiplication into addition. This leaves $-e_d^2/2\sigma_d^2 - e_b^2/2\sigma_b^2$. Here we see that σ_d and σ_b act as inverse weights on the corresponding error term. In the remainder of the paper, let $\delta = 1/2\sigma_d^2$ and $\beta = 1/2\sigma_b^2$: it is more convenient to discuss linear weight factors. When we later experimentally pick good values for δ and β, the above relation gives us the implied standard deviations.

After these transformations, the values computed by the Viterbi algorithm are not easily and directly interpretable as probabilities. However, as argued, these values suffice for our purposes. From now on we refer to them as *Viterbi values*.

The Viterbi algorithm looks at the distance and bearing between every pair of places: we can precompute a complete lookup table. This does nothing to the asymptotic runtime (clearly these computations take constant time each), but building the lookup table can be trivially parallelized. Using a single OpenMP [10] directive, we achieve near-100% utilization of our quadcore machine while filling the table, giving a 3.96-fold speedup of this step. As a downside, this table increases the memory usage from $O(kn)$ to $\Omega(n^2)$. This can be an obstacle in practice: for example at $n = 24{,}000$ the tables exceed 4 GB each when stored at double precision.

On one of our larger (but otherwise typical) instances ($k = 18$, $n \approx 10{,}000$), our basic implementation takes 698.8 s. A more careful implementation of the evaluation of dynamic-programming states leads to 68.5 s runtime.[3] Eliding exponentiation gives 30.0 s. We use this version as our baseline implementation in Section 5.

Adding the lookup table with multithreaded precomputation gives runtime 12.9 s. The next algorithm will improve this further to 2.6 s, but for ease of implementation one might prefer this one.

[3] When evaluating a dynamic programming state we have to loop over the possible preceding states and keep a running maximum of where the best value comes from. If the DP value of that state is already lower than our running maximum, we can conclude that this state will not improve our value, since all transition costs are negative. Then we do not need to calculate the distance and bearing terms for this transition. We similarly elide calculation of the bearing term if the distance term shows that this preceding state is useless.

4.2 Lazy Evaluation

Calculating a most-likely path in a hidden Markov model can be reduced to finding a longest path in a directed acyclic graph called a *trellis* (see e.g. [3]). This suggests a Dijkstra-like search based on a priority queue. In the coding-theory community, such algorithms are called "lazy Viterbi" [13, 15], since they do not necessarily evaluate all states. Therefore our implementation also does not use a lookup table of distances and bearings: we hope to ignore most.

The worst-case runtime in general suffers by a factor $O(\log n)$ because of priority-queue operations, but since fewer nodes are inspected in practice, the actual runtime improves. For a significant improvement, the edge weights should be transformed as follows [13]: between any two layers in the trellis, add a constant to each weight such that the maximum becomes 0. (As argued before, this is safe since it does not influence which path is optimal.) We additionally implemented bidirectional search on this trellis.

On the same large instance as before, the basic lazy algorithm runs in 13.3 s: slightly slower than our best implementation of "eager" Viterbi. After adjusting the edge weights as described, this improves to 4.6 s. Bidirectional search improves this further to 2.6 s. Unfortunately, bidirectional search is not consistently faster on all of our itineraries. Future work could investigate improved search strategies.

4.3 A Heuristic

As a heuristic, we can filter the gazetteer based on string similarity, rejecting most states as implausible. In particular, we restrict the domain of each $\mathbf{P}_i$ to the τ most string-similar entries at that stop. The algorithm first finds appropriate gazetteer entries by brute force, and then runs the eager Viterbi algorithm. This gives the following.

Theorem 4.2. *The most-likely sequence of places, restricted to the top-τ places at each stop, can be computed in $O(k\tau^2 + kn)$ time and $O(k\tau)$ memory.*

This is a significant improvement since the reduction is in the quadratic term. The disadvantage is that we are no longer guaranteed to find an optimal solution (to the original problem: clearly we find an optimal solution to this restricted problem). In Section 5.3 we evaluate suitable values of τ if speed is more important than quality. This can be the case in a system with real-time user interaction: present the result with a low-τ solution as soon as possible, then run the full inference in the background.

Taking the top-τ states is rank-based filter; we could also filter with a similarity threshold. Potentially this is a more sensible threshold, but does not give a predictably-low runtime runtime like the rank filter.

4.4 Sensitivity Analysis

Consider a stop i and let its most-likely assignment be $\mathbf{P}_i = g^*$, for some place $g^* \in \mathcal{G}$. We can find a "second-best" solution by computing the most-likely assignment for $\mathbf{P}_i$ conditioned on $\mathbf{P}_i \neq g^*$. (In general, this may also change the most-likely assignment of other places.) If this alternative solution has almost the same Viterbi value, the output of the algorithm could have easily been different if the input had been slightly different: this does not inspire confidence in the solution. If this alternative solution is much worse, on the other hand, the solution is robust in the sense that there is no close competition. These sensitivity values can be used to power an interactive user interface for quality assurance: see Section 7. Here we discuss their efficient computation.

We can calculate the above second-best sensitivity by running any of the previous algorithms and making it skip the possibility that $\mathbf{P}_i = g^*$. Doing this for each stop in the itinerary takes $O(k^2n^2)$ time. We can do better with a variant of the eager Viterbi algorithm – this will have worse constant factors but the asymptotic improvement makes up for this.

Theorem 4.3. *The Viterbi value of the most-likely sequence of places, conditioned on $\mathbf{P}_i = g$, can be calculated for all $1 \leq i \leq k$ and all $g \in \mathcal{G}$ in a total of $O(kn^2)$ time and $O(kn)$ space.*

Proof (sketch). Run the eager Viterbi algorithm twice, once as normal and once with the itinerary reversed, and keep the dynamic-programming tables. This is within the time and space bounds. Then for any i and g, the conditional Viterbi value for $\mathbf{P}_i = g$ can be read from the dynamic-programming tables by adding the forward value of "stop i equals g" to the value in the other direction (taking care not to count stop i twice). This takes $O(1)$ time per combination of a stop and a place, which again falls within the time bound. □

This result also gives us the second-best solution for each stop simply by inspecting all places for a fixed stop and taking the second best.

5 EXPERIMENTS

We evaluate our algorithms on real-world data from two historical guidebooks: Gail's *RaiSSbüchlin* from 1563 as taken from Krüger's facsimile [20] and the *Kronn und Auszbunde aller Wegweiser* [2], which was published anonymously in 1597. We work with three sets of itineraries taken from these publications:

- RB1: 21 routes from first 35 pages of the *RaiSSbüchlin*,
- RB2: a selection of 15 edited routes from the *RaiSSbüchlin* used by Blank and Henrich [6], and
- KR: the 12 routes originating in Würzburg from the *Kronn*.

For RB1 and KR, we manually created ground truth, identifying a gazetteer entry for each stop; we did not edit or cut the routes. For comparison we include the data set RB2, which was used by Blank and Henrich. (Some of their itineraries do not match the source material exactly: some are reversed and some cover only part of a *RaiSSbüchlin* itinerary.) In all three data sets, there is a small number of stops for which we could not identify a modern place, most likely due to deserted villages. For these, we accepted any solution. See Table 1 for an overview of the data sets. These data sets are culturally and temporally specific, but do note that our system can (and should) be trained on appropriate data: to georeference Latin itineraries (for example), one should train the string similarity on different data and use a different distance conversion factor λ.

We have used two different publicly available sources to generate our input gazetteers: the GeoNames geographical database[4] and the Getty Thesaurus of Geographic Names[5] (Getty TGN). For each

[4] http://www.geonames.org/

[5] http://www.getty.edu/research/tools/vocabularies/tgn/index.html

Table 1: Overview of the three data sets. Lengths are calculated according to the ground-truth modern places.

data set	RB1	RB2	KR
source	*RaiSSbüchlin*	*RaiSSbüchlin*	*Kronn*
year of publication	1563	1563	1597
# of itineraries	21	15	12
total # of stops	354	218	119
median # of stops	14	14	10
unidentified stops	5	3	8
total length [km]	5551.9 km	3051.8 km	1480.4 km
median length	209.8 km	178.9 km	103.9 km

Table 2: Statistics on the two gazetteers used for experimentation, giving the average number of entries per itineraries for the three data sets as well as the total number of entries.

gazetteer	∅ RB1	∅ RB2	∅ KR	total
GeoNames	6342.1	3798.5	1881.3	212 737
Getty TGN	4736.4	3781.4	2028.3	180 524

itinerary individually, we extracted the set of places from either source contained in a bounding box around the itinerary, padded by 0.1° latitude and longitude. This filtering step requires a rough knowledge of where the described route is located. This is a reasonable assumption, since the place of departure and the destination are given in the title of each itinerary, and are usually well-known cities. Blank and Henrich [6] describe a very similar filtering step based on the same assumption. See Table 2 for an overview.

For the experiments presented in this section, we assume all input data has been correctly parsed, organized as an itinerary $\mathcal{I}$, and that the gazetteer toponyms are formatted similarly to the itinerary toponyms. Section 6 describes how this was achieved.

5.1 Parameter Choice

We begin by discussing the choice of the parameters, in particular δ and β, and their impact on the quality of the solution. We measure the accuracy (that is, the fraction of places that are assigned correctly) and a distance error: the average distance between the assigned place and the correct place according to ground truth (which is zero for correctly-assigned places).

For RB1 in combination with GeoNames gazetteers, we can pick parameters ($\delta = 0.005$, $\beta = 0.001$) that achieve 85.9% accuracy and a low average distance error (2.4 km). In fact, the algorithm is quite robust in terms of parameter choice: there is an interval of about an order of magnitude for both δ and β in which any combination of the two parameters yields accuracy values greater than 80% (see Figure 2). For these intervals of parameter values, the average distance error stays below 3.5 km.

Figure 3 (left) shows a solution computed by our algorithm. It contains a single error: "Jesta" in the itinerary in fact corresponds to the modern place Oestheim, while the algorithm picked Esbach.

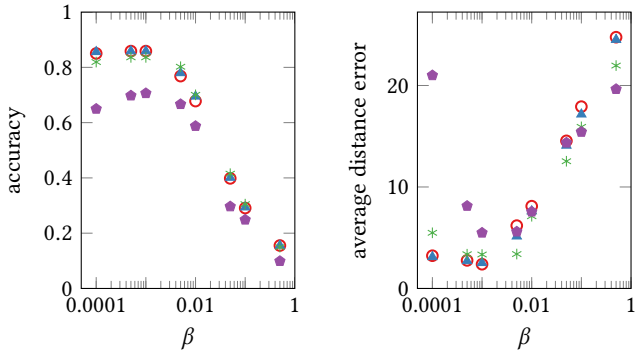

Figure 2: Different parameter values on RB1 using GeoNames. Parameter δ is included as 0.005 (○), 0.01 (▲), 0.05 (✳), and 0.5 (⬟). Both the highest accuracy (85.9%) and lowest distance error (2.4 km) are achieved with $\delta = 0.005$ and $\beta = 0.001$.

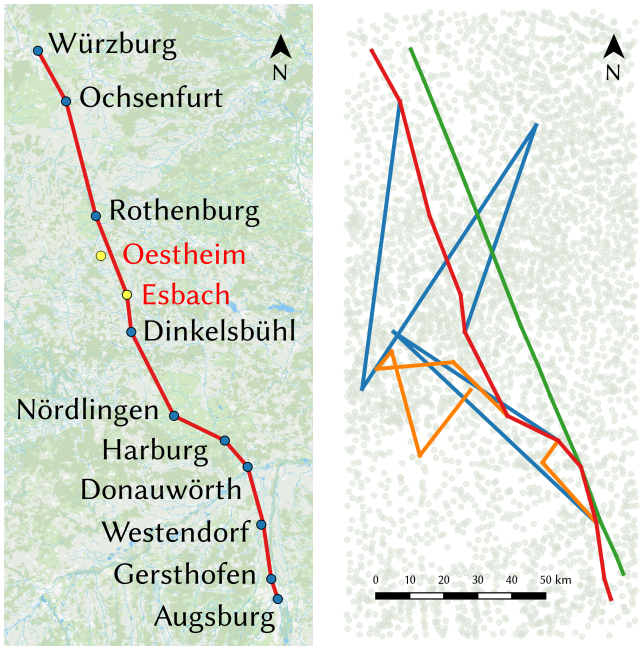

Figure 3: Route from Augsburg to Würzburg. Left: Solution with recommended parameters (red) and one error (the yellow places). Right: Solution using no spatial evidence (blue), too much weight on distance ($\delta = 1$, orange), and too much weight on bearing ($\beta = 1$, green). Points in the background indicate gazetteer places.

Note the significant change in the place name; still, the distances and bearing allowed the algorithm to pick a place that is geographically plausible. Elsewhere on the route, the algorithm was able to correctly assign for example Donauwörth ("Thonawerdt") and Dinkelsbühl ("Dinckelspihel").

Using GeoNames gazetteers, the algorithm achieves high accuracies on the remaining data sets as well; see Table 3 for details. This shows that our approach is able to generate accurate solutions in a variety of settings and somewhat robust in terms of parameter choice: based on our experiments, we recommend setting $\delta = 0.05$ and $\beta = 0.001$ for previously unseen data sets. This is a clear methodological improvement over Blank and Henrich, who achieve similar accuracies, but only if researchers "adequately choose the parameters" for each itinerary individually [6].

In fact, Blank and Henrich's paper [6] leads us to believe there does not exist a consistently-good set of parameters for their approach. In their experiments, they report an accuracy of 83.7% on data set RB2, but use an individually tuned set of parameters for each of the 15 itineraries: the authors do not report results for a consistent set of parameters. In addition, they run their algorithm in three different modes for each itinerary and only count the best result. Note in particular that the parameters and modes can only be evaluated using the ground truth, which is of course not available in practical applications. Reporting on a consistent choice of mode, the best choice leads to an accuracy of 71.2% (still allowing individual parameters for each itinerary). Our approach clearly outperforms this result with a global set of parameters.

Next, we briefly consider extreme choices for the parameters. Disabling the distance and bearing terms leads the algorithm to greedily assign each stop independently to the most string-similar place. This can be considered a baseline algorithm and it is much less accurate (for example 64.97% for RB1 using GeoNames), demonstrating that the spatial evidence is useful. In the other extreme, putting too much weight on a particular factor is also detrimental. For RB1 and GeoNames, Figure 2 shows the results of $\delta = 1$ and $\beta = 1$, which pushes the accuracy considerably below 40%. We see in Figure 3 (right) that extreme parameter values indeed lead to nonsensical solutions.

Finally we analyze the impact of a lack of bearing information on the accuracy. (Recall that in the present experiments, we assume the initial bearing from first to last place to be known, but the global bearing of the itineraries might in practice as well be unknown). For RB1 using GeoNames itineraries, the accuracy slightly decreases, from 85.9% to 82.2%. This shows that bearing information does help increase accuracy, but is not absolutely required to achieve good results. Our approach is thus not limited to itineraries with a straight direction, but could handle round trips as well.

5.2 GeoNames vs. Getty TGN

For all three sets of itineraries, our algorithm is less accurate when run with the Getty-based gazetteers than GeoNames: see Table 3. Except on KR, the distance errors are also higher. There are several factors leading to this behavior. First, the coordinates given by Getty are quite imprecise; indeed, the Getty trust states that it "is not a GIS" and the coordinates are meant for personal reference only.[6] Second, it is particularly sparse outside of Germany. This affects RB1, which has several itineraries to Vienna and Prague. Despite these shortcomings of the Getty gazetteers, our algorithm still achieves accuracy upwards of 70%.

[6] http://www.getty.edu/research/tools/vocabularies/obtain/download.html

Table 3: Top: Parameters that achieve the best accuracy on each data set, with the corresponding distance error. Bottom: Same measures, but with our recommended parameter set.

	GeoNames			Getty		
data set	**RB1**	**RB2**	**KR**	**RB1**	**RB2**	**KR**
distance factor δ	0.005	0.05	0.05	0.005	0.05	0.05
bearing factor β	0.001	0.005	0.001	0.001	0.005	0.001
accuracy	85.9%	82.1%	86.6%	70.6%	73.4%	77.3%
⌞ dist. error [km]	2.4	1.3	2.6	4.1	2.0	2.2
With recommended values $\delta = 0.05$ and $\beta = 0.001$:						
accuracy	83.6%	80.7%	86.6%	68.6%	70.2%	77.3%
⌞ dist. error [km]	3.3	1.3	2.6	4.1	2.2	2.2

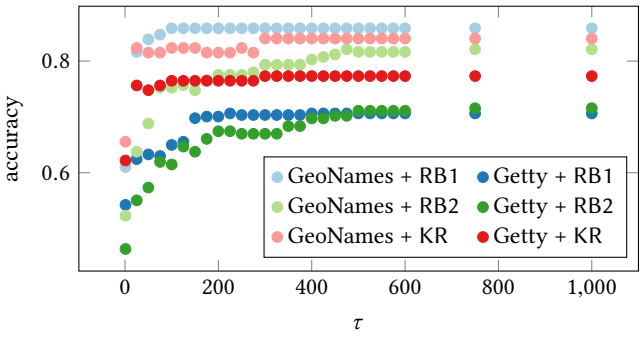

Figure 4: Accuracy of the heuristic as a function of τ.

5.3 Quality of the Heuristic Algorithm

In Section 4.3 we presented a faster, heuristic algorithm. Using the parameter sets δ and β that performed best in Section 5.1, Figure 4 shows that all sets of itineraries and both gazetteers can be run with $\tau \approx 500$ fairly safely: this achieves accuracy within 0.5 percentage points of the exact algorithm. A more restrictive choice of $\tau = 200$ still yields accuracy within 5 percentage points. The improvement in runtime is discussed in the next section.

5.4 Runtime

We have run each algorithm on all six combinations of a set of itineraries and a gazetteer, using the parameter set that yields the best accuracy for this pair. Table 4 gives the average runtime measured in these experiments.

As a baseline we have run all data sets with the "textbook implementation" of the Viterbi algorithm. The baseline is outperformed by all variants of our algorithm, except for "sensitivity", which calculates additional information. The lazy variant is faster than the baseline by approximately a factor 4 and is, in turn, slightly outperformed by the bidirectional lazy algorithm. The two versions of the heuristic are faster than the exact approaches by a full order of magnitude. Runtime in the low tenths-of-a-second range mean the heuristic can be used in real-time interactive applications.

Processing itineraries from RB1 takes the longest on average. This is because RB1 contains several itineraries that span a large geographical area and consequently require a large gazetteer. For

Table 4: Average runtime in seconds for processing one itinerary with the different variants of our algorithm.

	GeoNames			Getty		
data set	**RB1**	**RB2**	**KR**	**RB1**	**RB2**	**KR**
textbook Viterbi	24.88	8.83	2.77	13.46	8.72	2.80
lazy	6.44	2.19	0.45	3.92	2.26	0.58
bidirectional lazy	6.08	2.05	0.38	3.83	2.25	0.49
heuristic ($\tau = 500$)	0.21	0.15	0.06	0.18	0.15	0.06
heuristic ($\tau = 200$)	0.17	0.09	0.04	0.13	0.09	0.04
sensitivity	28.55	7.26	2.39	14.89	7.23	2.38

example, the bidirectional lazy algorithm takes 33 seconds to process an itinerary from Innsbruck to Vienna containing 24 stops (on a GeoNames gazetteer of size $n = 16{,}129$). On the other hand, the same algorithm solves a further 11 itineraries from this data set in less than 0.2 seconds each.

Note that all variants of our algorithm clearly outperform the algorithm by Blank and Henrich, who report that the exact version of their algorithm has runtime over five minutes for all but the shortest itineraries.

6 CASE STUDY: DATA PREPARATION

In this section, we discuss important data preparation steps necessary to run our algorithm on real-world data. These steps are: Optical character recognition for extracting the input for our algorithm from scans, training string similarity weights for use in $\mathbb{P}(\mathbf{T}_i \mid \mathbf{P}_i)$, and obtaining suitable gazetteer data. In addition, we discuss picking a conversion factor from historical German miles to kilometers. We describe our approach to the data preparation as a case study on the *RaiSSbüchlin*, for which we thereby obtain a nearly complete digitization pipeline.

6.1 Manual Transcription vs. OCR

For the experiments described in Section 5, we have manually transcribed the input itineraries. In order to efficiently deal with large sets of itineraries, one might consider applying optical character recognition (OCR). Höhn has applied his work-in-progress OCR system [16] to our scans of the *RaiSSbüchlin*, in particular the itineraries in RB1. From the resulting text strings, we (manually) selected the 354 lines that correspond to itinerary stops.

The OCR system was trained on 9 of the 21 itineraries and achieves a character error rate of approximately 5% on previously unseen text from the *RaiSSbüchlin*. Evaluating the remaining 12 itineraries (151 stops), our algorithm achieves 76.8% accuracy, compared to 86.0% on a manual transcription of these 12 itineraries.[7] This is far from a comprehensive study of the performance OCR approaches could have in this domain. Instead, it is a promising proof of concept that shows the general applicability of OCR to the problem.

[7]Using slightly different parameters than those used with the manual transcription, our algorithm achieves 78.2% accuracy with the OCR results.

6.2 Training for String Distance Weights

The statistical string similarity measure of Ristad and Yianilos [27] mentioned in Section 3.1 relies on a transition model learned from training data. The similarity measure can thus be tailored to a specific domain (here: matching historical and modern German toponyms) when provided with a sufficient number of appropriate training examples. We used von Reitzenstein's lexicon of Franconian place names [14] for this purpose. The lexicon contains approximately 800 entries of modern places and lists historical name variations for each of them. In total, we extracted 6432 pairs of corresponding modern and historical spellings from this lexicon. We picked p_0, p_1 and p_2 to approximate the joint distribution of lengths of historical toponyms and the corresponding modern toponym.

6.3 Gazetteer and Itinerary Preparation

As mentioned before, we have based our gazetteers on two sources: GeoNames and Getty TGN. GeoNames offers SQL dumps[8] and Getty provides an open SPARQL endpoint[9].

As a first step, the databases have been filtered for entries that correspond to populated places. Both databases contain alternative names for places; in the case of Getty, we have included all of these. With GeoNames, alternative names are often tagged with a language, so we only selected German ones. There is a trade-off between having many toponym variants in the gazetteer (more, but possibly confusing information) and having a small gazetteer (better runtime, but possibly missing relevant information).

We apply a set of simple transformations to all toponyms, both from the gazetteers and the itineraries, in order to somewhat normalize them. This is unfortunately rather ad-hoc and future work could attempt to put this step on proper foundation. At present it is necessary for two reasons. First, both gazetteer sources contain quite a number of malformed or incorrectly tagged place names. Second, the character set in the gazetteers must match the character set in the training data for our string distance measure.

Our transformations include dropping strings that are excessively long or contain characters that are implausible for our domain (e.g. non-western characters). Then we tokenize each place name (splitting on whitespace) and keep only the first token, with two exceptions. If the token is part of a stop list containing prepositions like "gen" ("toward"), we drop it and continue to the next token. If the token is part of a second list containing common prefixes like "Markt" or "St.", we merge it with the token after it. So for example "gen markt bibart" becomes "marktbibart." Finally, we remove diacritics and replace special characters with similar standard characters.

Ideally, a person with (historical) domain knowledge defines these transformation rules, and we trust someone with digital-humanities training will be able to write a corresponding script. (The rules for our data are mostly simple search-and-replace.)

6.4 Conversion Factor for German Miles

In both the *RaiSSbüchlin* and *Kronn*, distances between stops are given in historical German miles. According to the literature, one

[8]http://download.geonames.org/export/dump/
[9]http://vocab.getty.edu/sparql

historical German mile corresponds to 10,000 steps, which meant a travel time of approximately 2 hours by foot, and a conversion factor to kilometers of (very) roughly $\lambda = 7.5$. [31, p. 245f.]

We have verified this conversion factor using a least-squares fit between the reported distances and the ground truth (assuming great-circle distances). This results in $\lambda = 6.767$ (RB1), 7.259 (RB2), and 7.373 (KR). Our algorithm is robust against such inaccuracies: for example, the results on RB1 are the same for $\lambda = 6.767$ and $\lambda = 7.5$. We have used $\lambda = 7.5$ for all other experiments.

7 SMART USER INTERACTION

In Section 5, we have shown that our algorithm works accurately in general, but does not always get all places right. Indeed, for the given problem an accuracy of 100% can hardly be expected from a computer system, since there are difficult semantics involved. (Onomastics, the study of the history of proper names such as toponyms is an entire discipline in the humanities.)

For quality assurance we need to involve a human to verify the output of our algorithm. Since this process costs valuable experts' time, it should be efficient in terms of user interaction – in particular, we do not want to require the user to carefully inspect all results. Rather, the system should present parts of the solution that most require manual inspection. For these, it should in addition present alternatives that (hopefully) include the correct solution. We achieve both goals through sensitivity analysis, which can be efficiently computed (see Section 4.4).

7.1 Classification

Based on the sensitivity values for each assigned place, we consider the following classifier to decide which assignments warrant user inspection – that is, whether there is a good chance that the assignment might be wrong. Sort the places in the solution according to their sensitivity in increasing order, then classify all places with sensitivity values larger than θ as Inspect, and all others as Assume Correct, for some parameter θ. (Alternatively, one can choose the threshold to be a rank rather than a value.)

Our goal is for the user to be presented with most errors, while having to look at only relatively few places. We evaluate this classifier using receiver operating characteristic (ROC) curves, which plot the false positive rate versus the false negative rate: see Figure 5. Following standard methods in ROC analysis [12], we calculated the *area under the curve* (AUC). The AUC values in our experiments are 0.94 for RB1, 0.84 for RB2, and 0.95 for KR, all using GeoNames gazetteers. In general, AUC values between 0.8 and 0.9 can be considered excellent, while values over 0.9 are outstanding [17, p. 177]. In our specific case, this means that the classifier reliably discriminates between correct and incorrect assignments.

Having a means to put the user's attention on assignments that might be wrong, the next question is how to support the user in finding the correct assignment. The sensitivity analysis yields not only scores for the assigned places, but also for all other places. For a stop with an doubtful place assignment (as identified by the classifier above), we use this information to select likely alternatives (in descending order of Viterbi value).

Using our ground truth, we have evaluated at which rank in this sequence of alternatives the actually correct place is located. For all

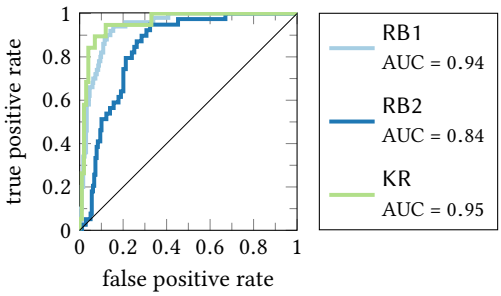

Figure 5: ROC curves, using best parameter values and GeoNames gazetteers.

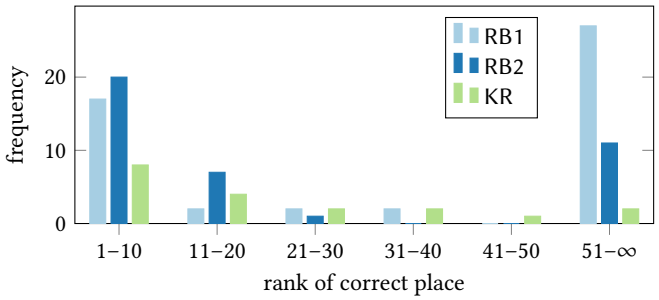

Figure 6: Rank of the correct place when alternatives are presented in order of Viterbi value, using GeoNames gazetteers and the parameters from Table 3.

three itinerary data sets, using GeoNames gazetteers, we find that a considerable fraction of the correct places lies within the first 20 alternatives (see Figure 6). This means that we can present the user with a relatively small set of alternatives and expect the correct solution to be among them.

7.2 User Interface

The classifier and the selection of alternatives described above are suitable for powering a user interface. For instance, we can (visually) draw the user's attention to places with high sensitivity using color coding. Figure 7 shows an itinerary from RB1: the color of the stops corresponds to sensitivity values from highly sensitive (red) to fairly robust (green). Indeed, four of the six stops colored in red are incorrect: Hlina[10], Schafhof, Schönberg, and Frankfurt.[11] There are no other errors in this solution, so having inspected only six out of 18 stops the user would have found all errors.

The line segments connecting the stops are also color-coded: here the color refers to the distance term $\mathbb{P}_{\mathrm{d}}(\mathbf{P}_i \mid \mathbf{P}_{i-1}, \mathbf{D}_i)$ given by Eq. (2). This presentation helps the user to assess whether the distances between the assigned places are plausible: if they are displayed in red, either the assignment is wrong or the distance given in the itinerary is particularly imprecise – either of which is potentially interesting.

[10] The toponym in the itinerary corresponds to Schlan, which is the German name of the Czech town Slaný. However, this alternative spelling is not present in $\mathcal{G}$.
[11] The gazetteer contains two entries called Frankfurt, very near to each other, and the algorithm was unlucky.

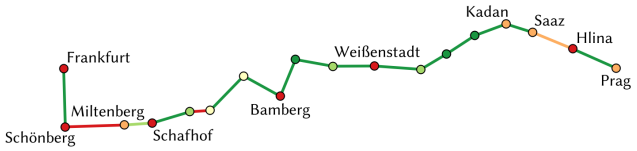

Figure 7: A solution, color-coded based on sensitivity values.

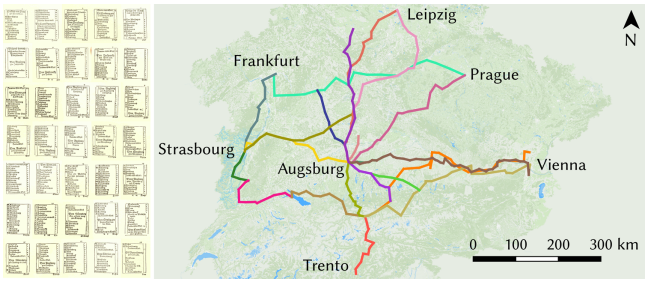

Figure 8: Left: 35 pages of the *RaiSSbüchlin*. Right: Our solutions for those 21 itineraries.

In addition we can present a set of alternatives for any stop, for example on click or mouse-over. As was demonstrated in Sections 4 and 5, our algorithms are fast enough to support real-time interactive systems like this.

8 CONCLUSION

We have taken a problem from the digital (geo-)humanities, formulated it properly as an optimization problem, and developed an efficient way of solving it – both asymptotically and in practice. We show experimentally that our algorithm outperforms the state of the art on this problem, both in accuracy and runtime. Our proper modeling enables automatic sensitivity analysis and we show that this forms a good basis for an algorithmically-supported user interface.

Some directions for future work have been addressed throughout the text. We mention one more: Blank and Henrich [6] filter routes based on angles (rather than bearings). Our model could handle this by using to a higher-order HMM, but then a straightforward application of the Viterbi algorithm takes $\Theta(kn^3)$ time which would require improvement.

We conclude with Figure 8, which shows 35 pages of the *RaiSSbüchlin* and our computed solutions on the 21 contained itineraries (RB1). This network, with 354 stops spanning a considerable part of Europe, was computed in about two minutes total runtime by the bidirectional lazy algorithm and has an accuracy of 85.9%.

ACKNOWLEDGMENTS

We thank Daniel Blank for providing RB2 and Winfried Höhn for applying his work-in-progress OCR system to the *Raißbüchlin*. Thomas C. van Dijk is supported by the German Research Foundation (DFG), grant number Di 2161/2-1.

REFERENCES

[1] Marco D. Adelfio and Hanan Samet. 2014. Itinerary Retrieval: Travelers, Like Traveling Salesmen, Prefer Efficient Routes. In *Proceedings of the 8th Workshop on Geographic Information Retrieval (GIR '14)*. 1:1–1:8.
[2] Anonymous. 1597. *Kronn und Auszbunde aller Wegweiser*. Lambertus Andree, Köln. http://data.onb.ac.at/rec/AC10037959 Austrian National Library.
[3] Lalit Bahl, John Cocke, Frederick Jelinek, and Josef Raviv. 1974. Optimal decoding of linear codes for minimizing symbol error rate. *IEEE Transactions on Information Theory* 20, 2 (1974), 284–287.
[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 *Proc. 9th Workshop on Geo. Inf. Retrieval (GIR '15)*. 9:1–9:2.
[6] Daniel Blank and Andreas Henrich. 2016. A Depth-first Branch-and-bound Algorithm for Geocoding Historic Itinerary Tables. In *Proceedings of the 10th Workshop on Geographic Information Retrieval (GIR '16)*. 3:1–3:10.
[7] Daniel Blank and Andreas Henrich. 2016. Die computergestützte ErschlieSSung und Visualisierung historischer Itinerare. In *Conference Abstracts of the 3. Tagung des Verbands Digital Humanities im deutschsprachigen Raum (DHd '16)*. 277–281.
[8] James O. Butler, Christopher E. Donaldson, Joanna E. Taylor, and Ian N. Gregory. 2017. Alts, Abbreviations, and AKAs: Historical Onomastic Variation and Automated Named Entity Recognition. *J. Map & Geo. Lib.* 13, 1 (2017), 58–81.
[9] Yao-Yi Chiang. 2015. Querying Historical Maps as a Unified, Structured, and Linked Spatiotemporal Source. In *Proc. of the 23rd ACM SIGSPATIAL Int. Conference on Advances in Geographic Information Systems (SIGSPATIAL '15)*. 270–273.
[10] Leonardo Dagum and Ramesh Menon. 1998. OpenMP: An Industry-Standard API for Shared-Memory Programming. *IEEE Comp. Sc. & Eng.* 5, 1 (1998), 46–55.
[11] Andrea Ernst-Gerlach and Norbert Fuhr. 2007. Retrieval in Text Collections with Historic Spelling Using Linguistic and Spelling Variants. In *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL '07)*. 333–341.
[12] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.
[13] J. Feldman, I. Abou-Faycal, and M. Frigo. 2002. A fast maximum-likelihood decoder for convolutional codes. In *Proceedings of the 56th IEEE Vehicular Technology Conference*, Vol. 1. 371–375.
[14] Wolf-Armin Freiherr von Reitzenstein. 2009. *Lexikon fränkischer Ortsnamen*. C.H.Beck, München.
[15] Y. S. Han, C. R. P. Hartmann, and Chih-Chieh Chen. 1993. Efficient priority-first search maximum-likelihood soft-decision decoding of linear block codes. *IEEE Transactions on Information Theory* 39, 5 (1993), 1514–1523.
[16] Winfried Höhn. 2017. Deep Learning for Place Name OCR in Early Maps. In *Proceedings of the 2nd Workshop on Exploring Old Maps (EOM '17)*. 17–18.
[17] David W. Hosmer Jr. and Stanley Lemeshow. 2004. *Applied Logistic Regression*. John Wiley & Sons.
[18] Arbaz Khan, Maria Vasardani, and Stephan Winter. 2013. Extracting Spatial Information From Place Descriptions. In *Proceedings of the 1st ACM SIGSPATIAL Int. Workshop on Computational Models of Place (COMP '13)*. 62:62–62:69.
[19] Deniz Klnç. 2016. An accurate toponym-matching measure based on approximate string matching. *Journal of Information Science* 42, 2 (2016), 138–149.
[20] Herbert Krüger. 1974. *Das älteste deutsche Routenhandbuch. Jörg Gails RaiSSbüchlin*. Akademische Druck- und Verlagsanstalt, Graz.
[21] Fernando Melo and Bruno Martins. 2017. Automated Geocoding of Textual Documents: A Survey of Current Approaches. *Trans. in GIS* 21, 1 (2017), 3–38.
[22] Ludovic Moncla, Mauro Gaio, Javier Nogueras-Iso, and Sébastien Mustière. 2016. Reconstruction of itineraries from annotated text with an informed spanning tree algorithm. *Int. J. of Geographical Information Science* 30, 6 (2016), 1137–1160.
[23] 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 *Proc. of the 22nd ACM SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems (SIGSPATIAL '14)*. 183–192.
[24] Paul Newson and John Krumm. 2009. Hidden Markov Map Matching Through Noise and Sparseness. In *Proceedings of the 17th ACM SIGSPATIAL Int. Conference on Advances in Geographic Information Systems (SIGSPATIAL '09)*. 336–343.
[25] Judea Pearl. 1986. Fusion, propagation, and structuring in belief networks. *Artificial intelligence* 29, 3 (1986), 241–288.
[26] Gabriel Recchia and Max Louwerse. 2013. A Comparison of String Similarity Measures for Toponym Matching. In *Proceedings of the 1st ACM SIGSPATIAL Int. Workshop on Computational Models of Place (COMP '13)*. 54:54–54:61.
[27] Eric Sven Ristad and Peter N. Yianilos. 1998. Learning String-Edit Distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, 5 (1998), 522–532.
[28] Rainer Simon, Elton Barker, Leif Isaksen, and Pau de Soto Cañamares. 2015. Linking Early Geospatial Documents, One Place at a Time: Annotation of Geographic Documents with Recogito. *e-Perimetron* 10, 2 (2015), 49–59.
[29] Humphrey Southall, Ruth Mostern, and Merrick Lex Berman. 2011. On historical gazetteers. *Int. Journal of Humanities and Arts Computing* 5, 2 (2011), 127–145.
[30] A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Inf. Theory* 13, 2 (1967), 260–269.
[31] Hans-Joachim von Alberti. 1957. *MaSS und Gewicht*. Akademie-Verlag, Berlin.
[32] Jerod Weinman. 2013. Toponym Recognition in Historical Maps by Gazetteer Alignment. In *Proc. 12th Int. Conf. on Doc. Ana. & Recog. (ICDAR '13)*. 1044–1048.