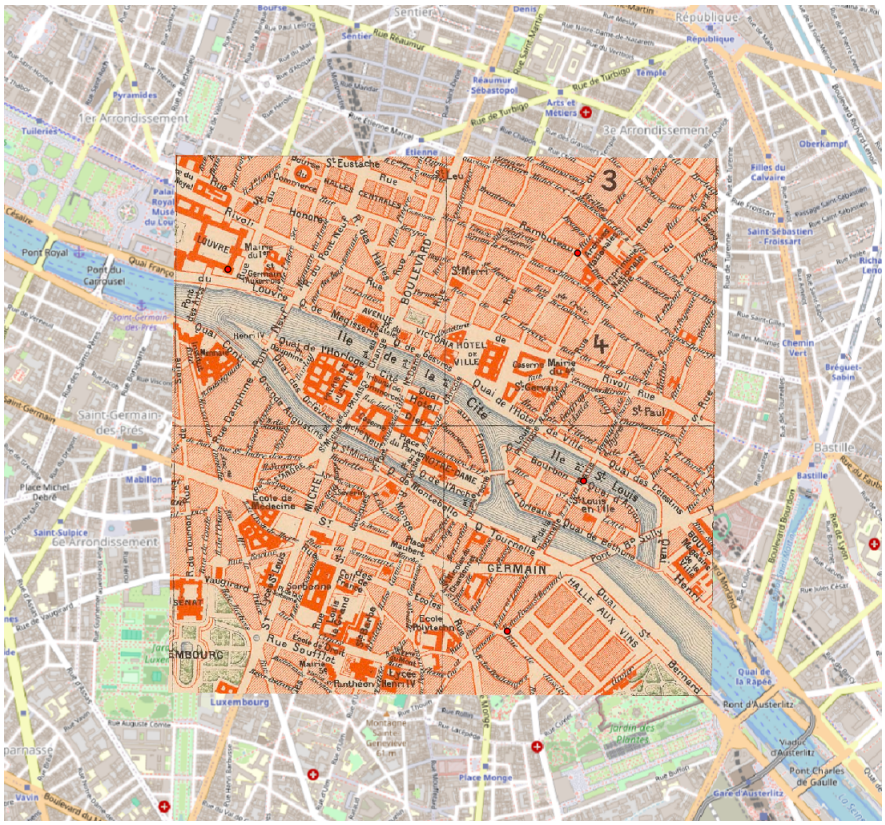
# Toponym-assisted Georeferencing for Historical Maps

*Semester Project 2024 Spring*




**Student:**

Tianhao Dai

**Supervisors:**

Beatrice Vaienti, Rémi Guillaume Petitpierre / Frédéric Kaplan


*07/06/2024*

# 1 Introduction

This project aims to build a pipeline for automated georeferencing of historical city maps using toponymic features. As a continuation of the text extraction project [12], this work first addresses the toponym assignment problem, which involves grouping single-word detections into phrase-level toponyms. We then discuss the method of automated georeferencing using the extracted toponyms to align historical maps with geographic coordinates.

## 1.1 Related Works

### 1.1.1 Toponym Assignment

Toponym assignment is an under-researched task. Current pipelines for extracting toponyms are limited to single word detection. The MapKurator project developed by the University of Minnesota team [6] applied the transformer-based model TESTR to perform scene text detection on historical maps, showing good results on the David Rumsey Map Collection. A course project from EPFL DH-405 [5] proposed a toponym pipeline that used Bézier curves as the main features to connect word bounding boxes. The test results on three map patches of Jerusalem achieved nearly or over half in both textual precision and recall.

### 1.1.2 Automatic Georeferencing

Automatic georeferencing is a crucial step in the digitization pipeline for historical maps and facilitates many downstream applications in digital humanities. Current research on automated map georeferencing can be classified into three main approaches:

- Using explicitly stated labels. This approach aims to detect explicit coordinate information on the map, such as grid lines and tick marks, and place control points at the grid intersections. Notable works in this line include [3].
- Visual feature-based approaches. These methods require an anchor map that is already georeferenced. They register other map images to the anchor map to obtain geographic coordinates. This involves detecting geometric features, such as the road network [9], and using algorithms like SIFT or deep learning-based methods (e.g., Superglue and Superpoint) to detect point correspondences on the anchor map and the map to be georeferenced. The maps are then aligned to the anchor map using a homography matrix [4, 8].
- Toponym-based approaches. These methods use toponyms as features to compute the transformation. Typically, users manually label toponyms on maps as control points for the transformation process. Existing works have employed various techniques to select control point pairs, including point pattern matching [2] and RANSAC [11]. However, these studies have mainly focused on large-scale maps where toponyms are cities and towns, rather than city maps.

## 1.2 Novelty and Contribution

This project focuses on detecting toponyms and georeferencing on historical maps of Paris and Jerusalem. To ensure the best performance of our pipeline, we first tried using the state-of-the-art model for scene text detection, DeepSolo, which we hoped would boost performance in word spotting. Since DeepSolo provides curve features of text bounding boxes, we attempted to utilize these features to group the words. Furthermore, considering how humans read toponyms, we observed that the order of words in a toponym can be easily predicted based on directional information. Therefore, we aimed to not only predict the binary group label but also the order of the words. After grouping all the toponyms on a map and obtaining their sorted text, we used a geocoder to automatically retrieve their geographic

coordinates, which are then used to compute the transformation from maps to a geographic coordinate system.

Our pipeline has been proven effective through extensive experiments. Specifically, our contributions are as follows:

- We designed a novel architecture for word grouping, based on an encoder-decoder and pointer network. This architecture not only effectively groups toponyms but also simultaneously determines the word order.
- We integrated the grouping results on a map scale using a consensus ranking algorithm and tested them on various sources of maps, including the ICDAR 2024 competition data [7] and our data for georeferencing.
- We developed an automatic pipeline for georeferencing historical maps of cities using detected toponyms, which showed more stable performance than the baseline [4].

We also discussed the effects and limitations of each component in detail throughout the report, hoping to offer insights to improve the pipeline in the future.

## 1.3 Toponym Assignment

*This section "Toponym Assignment" is co-authored by Mengjie Zou and Tianhao Dai. Mengjie Zou is responsible for the general idea and motivation of auto-regressive transformer, providing training data and integrating the network into the pipeline, and Tianhao Dai is responsible for designing the architecture and implementation of auto-regressive transformer, training and evaluation of the network. Though not being an individual work, this section is kept for better consistency of the article.*

### 1.3.1 Motivation

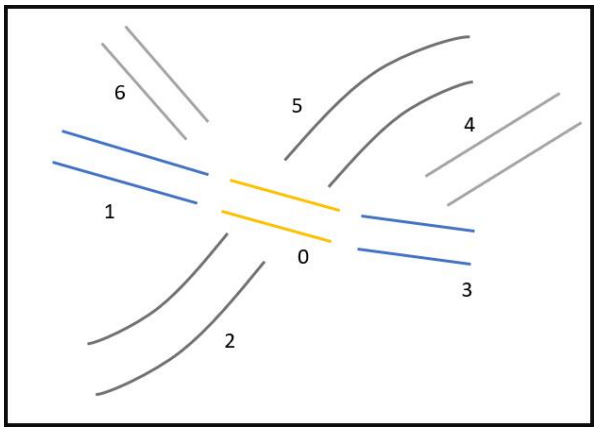


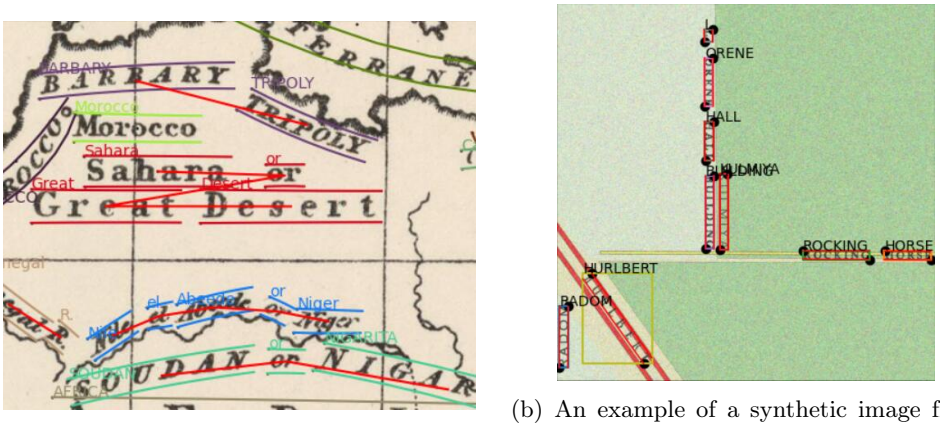

Figure 1: Illustration of what the model is exposed to in a local sample. Yellow indicates "query" word, blue signifies the words that belong to the same toponym as query (ground-truth), and grey signifies other sampled words. The auto-regressor doesn't know the words within the toponym in prior.

Suppose that we have a single detection for each word entity on the entire map, the next step is to assign each word entity to its corresponding toponym. Previously, this process involved minimizing the connecting cost between words [5], which, despite being mathematically sound, felt counter-intuitive, artificial, and cumbersome for integrating different features from a word detection. Instead, it seems more natural to employ a machine learning algorithm that can automatically learn to utilize available

detection features and identify the toponyms. In our experiment the features we selected for each word entity are the image coordinates of control points of the bounding Bézier curves (Fig. 1). Each vector

Considering how humans read words as toponyms, it's easily observed that human reading is an iterative process, locating the next word and determining its order simultaneously. Thus, an appropriate machine learning algorithm for this task should be auto-regressive, predicting both the words that belong to a specific toponym and their reading order concurrently. Drawing inspiration from auto-regressive transformers, we propose a model that uses the word of interest as a query and examines a local cluster of word detections around this word. The model then outputs a sequence of words that belong to the same toponym as the query word, in their reading order.

### 1.3.2 Datasets




(a) An example from MapText dataset.

(b) An example of a synthetic image from MapKurator's SynthMap dataset.

Figure 2: Examples from datasets used in Word Spotting stage of the pipeline.

**MapText**. The MapText dataset (Fig. 2a) is sourced from the ICDAR 2024 Competition on Historical Map Text Detection, Recognition, and Linking [7]. It comprises 200 training images and 40 testing images, each with a resolution of 2000x2000 pixels. The annotations include bounding polygons for word entities and groupings of word entities into toponyms for each image. These annotations are rich in detail, as the order of vertices in the bounding polygons maintains the reading direction of the words, and the word groups preserve the toponym information.

**MapKurator Dataset**. The MapKurator dataset, introduced within the MapKurator system (reference here), is specifically designed for training word detection models on historical maps. From the original dataset, we have selected two key components: SynthMap and Human Annotation. SynthMap comprises over 13,000 synthetic images that emulate the textual and background characteristics of genuine historical maps. This includes replicating the unique fonts, spacing, orientations, and distinctive background styles found in these maps (Fig. 2b).

In addition to SynthMap, the dataset includes a Human Annotation section with over 300 images. Although smaller in size, this section is highly valuable for fine-tuning detection models after comprehensive pretraining. Similar to the MapText dataset, the MapKurator dataset maintains the reading directions of word entities, though it does not include toponym notations.

### 1.3.3 Network Architecture

To achieve the goal of identifying words that belong to the same toponym as the selected word (the query) and outputting them in the correct sequence, we consider using an encoder-decoder architecture [1], which is a popular architecture for natural language generation tasks. In natural language generation, the output space is a large, fixed-length vocabulary, and at each step of the decoder, the

attention scores over the whole input sequence are calculated. The attention scores are then used to calculate a context vector, concatenated with the decoder hidden state, to predict an index of a token in the vocabulary. However, in our scenario, the output dictionary consists of all the surrounding word entities, and its size varies with the number of surrounding words. The decoder is expected to pick one word from these entities at each time step until the last word in the toponym is generated. To achieve this goal, we chose the pointer network [10] as our architecture. This is a modification of the traditional encoder-decoder architecture, which attends to the input sequence at each prediction step and take the position with the highest attention score as the output. Moreover, we chose the state-of-the-art transformer as the encoder and decoder.

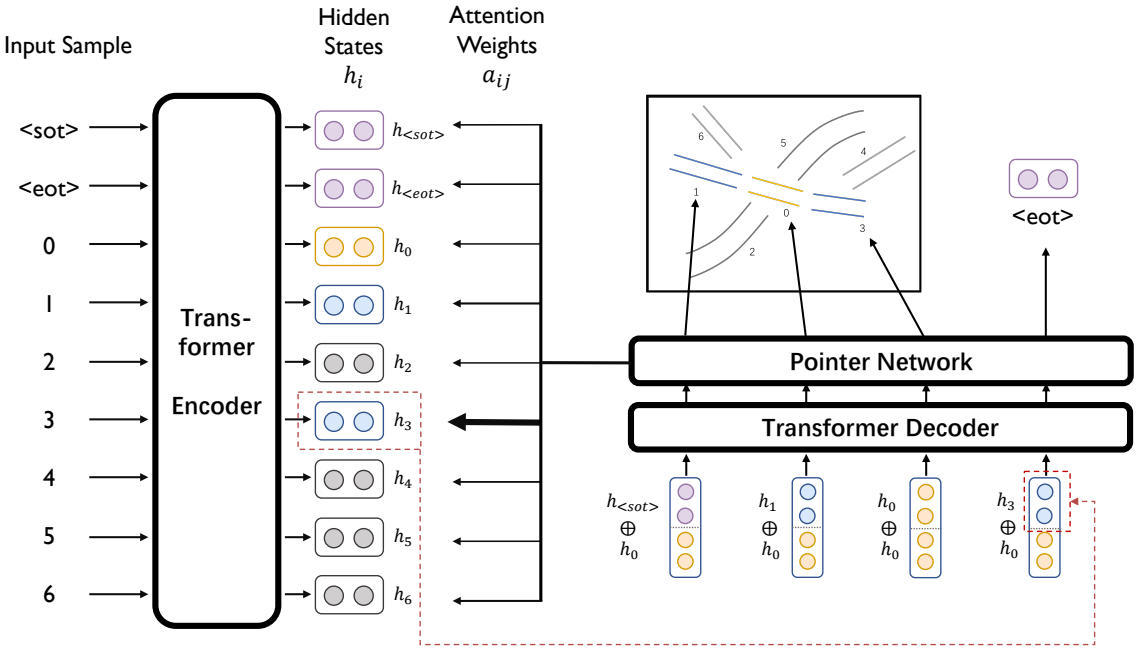



Figure 3: The encoder-decoder network for toponym grouping. In this example, sample 0 is selected as the query, and samples 1 and 3 are the words that belong to the same toponym as the query (the ground-truth words), while non-relevant samples are shown in grey. The pointer network calculates attention scores (indicated by the thickness of the arrows) over all encoder hidden states. The one with the highest score is chosen as the output and used as the next input for the decoder. The ground-truth output is the indices of the same-group samples in the correct order, i.e., 1-0-3. After the last word is found, the decoder should generate an <eot> token to indicate the end of the toponym.

The detailed illustration of the network architecture is shown in Fig. 3. It is composed of one layer of a transformer encoder and one layer of a transformer decoder. A pointer network, which calculates the attention weights over all encoded hidden states at each decoding step, is appended on top of the decoder. The input features are encoded as hidden states $h_i$, which are then taken as the inputs of the decoder. To condition on the query sample, the hidden state of the query is concatenated with the decoder input. The output of the decoder is then passed to the pointer network to calculate the additive attention weights over all the encoder hidden states [1]. The sample with the highest attention score is selected as the next input for the decoder, forming an auto-regressive generation process.

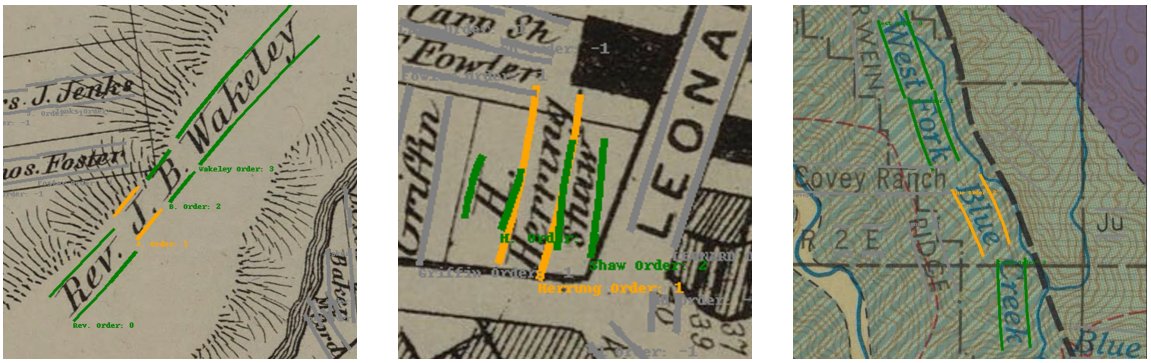
### 1.3.4 Model Training

The auto-regressive grouping model was trained on maps from the MapText dataset, as described in 1.3.2. Out of the 200 maps in the training set, only 101 contained toponymic labels. From these,

we held out 10 maps for testing and used the remaining 91 for training. Due to the scarcity of data, we used all the detected words and sampled their nearest 15 words, resulting in 15,505 samples. These samples were further augmented by shuffling the word order, bringing the total to 222,907 training samples. The test set included 1,909 samples without augmentation.

The input samples were represented as 16-dimensional vectors, consisting of 8 pairs of image coordinates of the control points on the bounding Bézier curves. These features were converted to the offsets to the center of the query sample before being fed to the model. To represent the special tokens <sot> (start of toponym) and <eot> (end of toponym), a two-dimensional binary vector was concatenated, making the size of the feature vector 18. We set the maximum length of the input to 16 (excluding the two special tokens), padding when necessary.

We used teacher forcing during training, i.e., using the ground-truth output sequence as input for the decoder, and greedy decoding during inference, i.e., taking the argmax at each decoding step. We applied negative log-likelihood loss on the log softmax of the predicted logits over non-padding outputs and used Adam as the optimizer, with an initial learning rate of 0.001. Due to the relatively small size and low diversity of the data, the model converged after one epoch, which took approximately 5 minutes on a laptop with one GPU. Evaluated on a binary labeling task (whether the word is from the same group or not), the best model achieved over 95% accuracy and and approximately a 75% F1 score for the same-group labels.




(a) Toponym in one line (b) Toponym in a twisted line (c) Toponym in multiple lines

Figure 4: Successful grouping examples. The text bounded by yellow curves represents the query, and texts bounded by green curves are the correctly predicted words. The associated number is the predicted order of each word.

Fig. 4 shows successful grouping examples in three typical cases: a toponym in one line, in a twisted line, and in multiple lines. The numbers associated with each word indicate that the order of the words (starting from zero) is also correctly predicted.


(a) False negative example 1 (b) False negative example 2 (c) False positive example

Figure 5: Failed grouping examples. The text bounded by yellow curves represents the query. Texts bounded by green curves are correctly predicted words, while texts bounded by blue curves are ground-truth words that were not grouped (false negatives). Texts bounded by red curves are words that were mistakenly predicted as belonging to the same group (false positives). The associated number is the predicted order of each word.

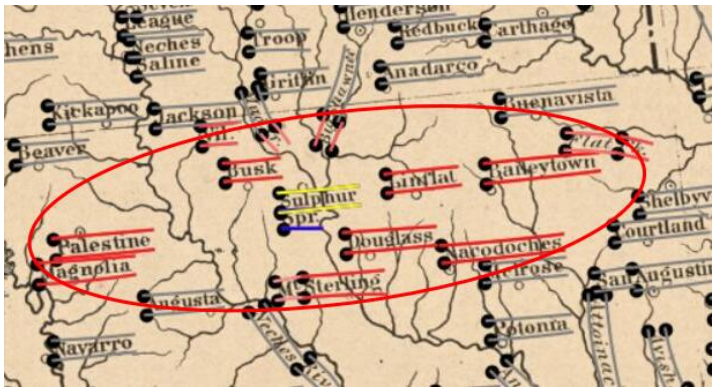



Figure 6: Illustration of Sampling algorithm. Yellow word detection is the 'query' and red ones are the sampled words.

There are also failed cases, as shown in Fig. 5, each representing a typical failure mode that could potentially be addressed by different heuristics. For instance, Fig. 5a failed to group the name of a canal, potentially due to the irregular layout of the words. In this case, visual background information about the canal would help tackle this issue (imagine how a human recognizes a river name). The failure in Fig. 5b highlights the necessity of textual heuristics when grouping highly coherent words; in this example, "Avenue" would never appear alone and should be grouped with surrounding words. Lastly, font style may help differentiate different toponyms to reduce the false positive rate, as shown in Fig. 5c, where the wrongly predicted word has a distinct style from the query. Besides these failure, the grouping model sporadically produces output sequence without the <eot> token (4 out of 1909 testing samples). Hence, a larger and more diverse training dataset would help enhance the accuracy and reliability of the grouping model. However, we must acknowledge that, due to the unconstrained nature of cartography, these heuristics might not be exhaustive, and there are always exceptions that may lead to wrong predictions. For example, some toponyms appear in different font styles corresponding to the original text and the translated text.

### 1.3.5 Map-scale Integration

Our auto-regressive transformer predicts one toponym at a time, in a local scope. To get all the toponyms at map scale, we designed our entire Toponym Assignment algorithm which includes the following four sub-steps.

**Sampling**: For each detected word, use it as a query and sample a fixed number **n_samples** of the nearest word detections as input. As illustrated in Fig. 6, the sampling region is elliptical, with a higher likelihood of selecting words along the direction of the query word.

**Inferencing**: Run the neural network to generate a predicted toponym with word order for each sample. This ensures that every word detection is used as a query once.

**Toponym Criterion**: Each prediction forms several directed edges from the query word to the predicted words in the toponym. Consequently, a directed graph can be constructed at map scale. We say a set of words belong to the same toponym if and only if any two words are mutually reachable via directed edges in aforementioned graph. We identify Strongly Connected Components (SCC) in the directed graph as toponyms.

**Algorithm 1** Consensus Ranking with Weight-based Cycle Removal

**Input:** List of partial observations
**Output:** Consensus ranking
Initialize an empty directed graph $G$
Initialize an empty dictionary $weights$
**for** each partial observation $obs$ **do**
  **for** each pair of consecutive nodes $(u, v)$ in $obs$ **do**
    **if** $(u, v) \in G$ **then**
      $weights[(u, v)] \leftarrow weights[(u, v)] + 1$
    **else**
      Add edge $(u, v)$ to $G$
      $weights[(u, v)] \leftarrow 1$
    **end if**
  **end for**
**end for**
**while** $G$ contains a cycle **do**
  Find the edge $(u, v)$ with the lowest weight in the cycle
  Remove edge $(u, v)$ from $G$
**end while**
Apply topological sorting to $G$ to get the consensus ranking
**return** Consensus ranking

**Order Estimation**: The words in the toponym grouped by SCC is order-less. Recovering the word order within a SCC can be framed as a Consensus Ranking problem. For each prediction, the order of the elements in the SCC is partially observed and potentially noisy. The Topological Sorting with Weight-based Heuristic Cycle Removal algorithm (Algorithm 1) is applied to address this issue.

### 1.3.6 Results

The first experiment focuses on the selection of the hyper-parameter **n_samples**. As shown in Fig. 7, the probability of obtaining a complete sample—one that includes the entire toponym of interest—is determined given a random word as the sampling center (query word). This probability is derived by repeatedly executing the sampling algorithm on the MapText dataset and calculating the proportion of complete samples among all generated samples.

Selecting the hyper-parameter **n_samples** involves balancing efficiency and performance. As illustrated in Fig. 7, achieving a $99\overline{\%}$ probability of generating a complete sample requires setting

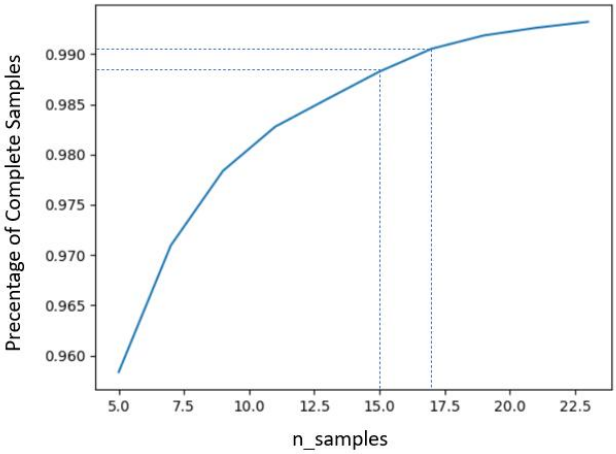



Figure 7: Percentage of samples containing the entire toponym of the query words at different **n_samples** values. Evaluated on MapText dataset.

**n_samples** to approximately 16. Which means if this hyperparameter is 16, there is only 1% chance that no matter how good our auto-regressive transformer is, at least one word in the toponym of interest cannot be predicted. And thus, we think 16 is a good trade-off between efficiency and performance.

# 2 Automated Georeferencing

In this section, we present the automated georeferencing pipeline that builds on the results from Section 1.3. Georeferencing involves converting a set of data points from an image coordinate space to a geographic coordinate space. Our toponym-assisted georeferencing pipeline includes three main steps: toponym normalization, automatic geocoding, and transformation estimation. To evaluate the pipeline's performance, control points are selected for each map, and the average distance error is calculated.

## 2.1 Dataset

Our pipeline is tested and built on a collection of historical maps of Paris and Jerusalem. Specifically, we used the same 50 maps of Paris as Jan et al. [4], published between the 1800s and 1950s. Additionally, we randomly sampled 10 maps of Jerusalem, ranging from the 1820s to the 1910s. On each map of Paris, five control points corresponding to landmarks were selected, while on the maps of Jerusalem, over a hundred control points were chosen, with the exact number varying across different maps.

## 2.2 Toponym Normalization

As the results from the toponym extraction pipeline do not guarantee exact matches to the ground-truth text, and misspellings and word omissions frequently occur, a toponym correction step is necessary. However, even when the texts are correctly recognized, variations in names referring to the same geographic entity, such as abbreviations (e.g., "Notre Dame" -> "Ne. De."), multilingual names (e.g., "Damaskustor" -> "Damascus Gate"), and different transliterations (e.g., "Al-Aqsa Mosque" or "El

Aksa" -> المسجد الأقصى), can also make it challenging for the geocoder to accurately identify and provide the precise location.

To address this, we chose to use the GPT-3.5 API to unify toponym correction and normalization into a single step. By using the prompt below:

> *This is a toponym extracted from an old map of [THE CITY]. Please normalize it into its most likely full name that is recognizable by the GoogleV3 Geocoder: [TOPONYM]. Please only respond with the normalized name; if it's unclear, simply return 'Unclear'.*

we can obtain normalized toponyms that will be fed to the geocoder.

## 2.3 Automatic Geocoding

As part of our automated pipeline, we retrieve the geographic coordinates (longitude and latitude) for each toponym by calling a geocoding API. We chose GoogleV3 over OpenStreetMap's Nominatim due to its empirically higher accuracy and support for fuzzy search, making it more robust for handling various toponyms in our scenario. We disambiguated toponyms by specifying the city when calling the API, and got their geographic coordinates in the WGS84 system.

The geocoding step relies on two main assumptions:

1. **The invariance of toponyms over time:** Since we are using a modern geocoding service, we assume that most toponyms have not changed their names since the maps were published. This project aims to achieve the best georeferencing accuracy, and we maintain this assumption as long as it proves effective. However, in a broader context, a more specialized geocoder, such as one using a gazetteer dataset, will eventually be developed to support the digital humanities research.

2. **The placement of the toponym at the center of its polygon:** Various methods have been used to determine the placement of a toponym by finding its nearest marker symbol (such as a dot or square [2]). These methods are mostly effective for regional or global-scale maps, where toponyms typically refer to cities and towns. However, this approach is less feasible for city maps, where toponyms are more densely packed and marker symbols are usually not identifiable. Using the centroid of the polygon is the simplest way to represent the location of a toponym. As the number of toponyms increases, the artifact introduced by this assumption can be minimized.

## 2.4 Transformation Estimation

It is inevitable that the geocoded results contain some noise, as there are always less-documented places and toponyms that have changed names. Therefore, we adopted the idea from [11] to use RANSAC to filter out erroneous point correspondences and estimate the transformation model on inliers. RANSAC is a robust method for estimating model parameters from a set of data points that contain outliers. By setting an inlier threshold—a distance error threshold below which a predicted value is considered an inlier—and specifying the maximum number of iterations, we can estimate the model repeatedly. This process continues until we identify the maximum number of inliers. The final transformation estimate is then fitted to these inlier data points. To our most recent knowledge, we are the first to use RANSAC for estimating transformations on city-scale maps.

RANSAC naturally suits the task of georeferencing city-scale maps because the ambiguity of locating toponyms within a city is much less than at a larger scale, minimizing the number of outliers in the geocoded results and increasing the success rate of fitting a model. Additionally, RANSAC unifies toponym disambiguation and transformation estimation into one step, allowing full use of all available toponyms without manual selection beforehand.

The commonly-used models for georeferencing are polynomial transformations, which approximate the process of map projection. We chose to use its simplest form, the first-order transformation, i.e.,

the affine transformation, due to its simplicity, widely accepted effectiveness, and lower likelihood of overfitting the data.

However, due to the randomness in the RANSAC method, the inliers used to estimate the model parameters vary each time, causing slight variations in the estimation results. During inference, this could lead to inaccurate predictions if we do not have control points to assess the estimation quality. Therefore, we recompute the RANSAC model several times and use the mean distance error to evaluate the transformation.

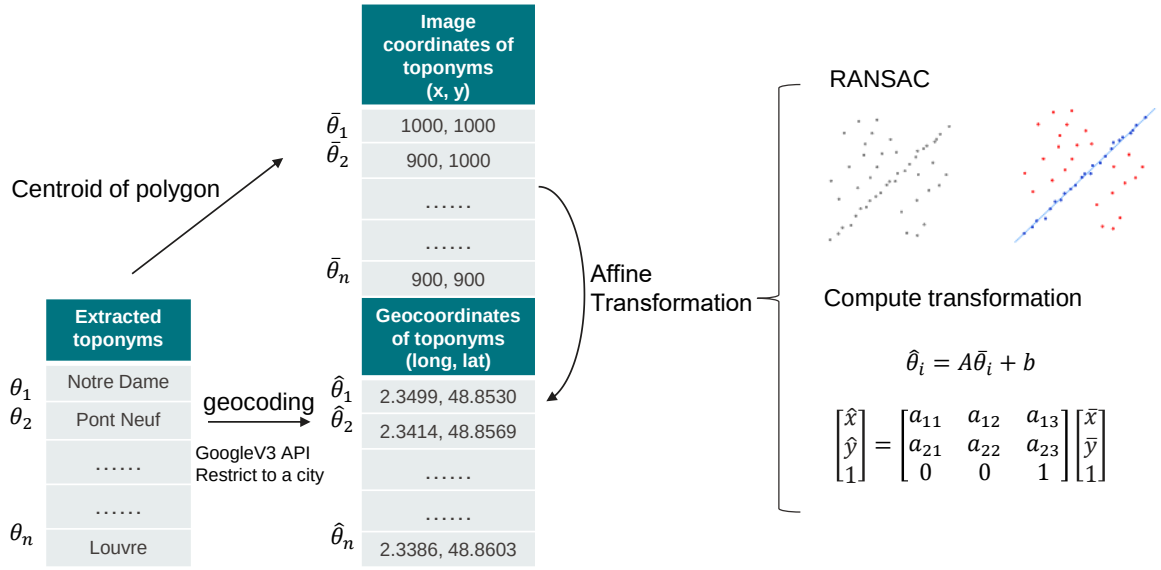



Figure 8: An illustration of georeferencing pipeline.

The illustration of the transformation is shown in Fig. 8. Formally, we have a set of already normalized toponyms $\{\theta_i\}$, with their image coordinates represented by the centroids of the polygons $\{\bar{\theta}_i\} = \{\bar{x}_i, \bar{y}_i\}$. After geocoding, we obtain their geographic coordinates $\{\hat{\theta}_i\} = \{\hat{x}_i, \hat{y}_i\}$, where $\hat{x}_i$ and $\hat{y}_i$ are longitude and latitude in the WGS84 system. We then applied RANSAC to iteratively compute the affine transformation $A$ with six parameters (degrees of freedom). Note that the longitude and latitude are computed separately, and the inlier threshold is set as the L2 distance between the ground-truth points and the predicted points in the geographic coordinates.

## 2.5 Results and Discussion

### 2.5.1 Experiment Details

The only manual process involved in this pipeline was wiping out all the marginal texts containing the index of toponyms for all the maps of Paris and Jerusalem. We believed that these texts would introduce significant noise, and would negatively impact the georeferencing results.

To further denoise the data, before running RANSAC, we also performed an automatic filtering process and only kept toponyms that met these criteria:

1. A clear normalized name was returned by the ChatGPT API, rather than 'Unclear', 'Unrecognized', or similar responses.

2. The toponym contains at least two tokens (i.e., it includes at least one space). This is because we observed that most single-word toponyms were either false predictions or carried little information, making them unsuitable for the geocoder.

3. Its geocoded coordinates fall within the boundary of the city. Even though we restricted the geocoding to a specific city, exceptions still occurred. For Paris, we set the boundary as [1.443, 3.563] for longitude and [48.140, 49.050] for latitude. For Jerusalem, the boundary was [35.00, 35.40] for longitude and [31.60, 32.00] for latitude (in the WGS84 system).

The distance was measured using a local coordinate system to enhance precision. We used EPSG:27561 for maps of Paris and EPSG:28193 for maps of Jerusalem. The pyproj library was used to convert the WGS84 coordinates to these local systems. We then calculated the Euclidean distance (in meters) between each pair of control point and the predicted point. We repeated RANSAC five times for each inlier threshold to get more robust results. The mean distance error over all control points and its confidence interval on maps of Paris and Jerusalem are reported in Table 1 and Table 2, respectively.

We also compared the results of our pipeline with one of the baseline methods [4] that performed homography transformation from source maps to an anchor map, using SIFT-detected road network point correspondences. The distance measure for this baseline was on based on pixel distance. Following the calculation of [4], one pixel distance corresponds to 1.19 meters.

### 2.5.2 Georeferencing Results

As shown in Table 1, for 41 maps of Paris, our pipeline demonstrates a lower variance in mean distance error, indicating more stable predictions compared to the baseline. According to the metric used in [4], a realignment is considered successful if the average distance error is below 75 meters. The success rates of our pipeline is 29 out of 41, one more map than the baseline.

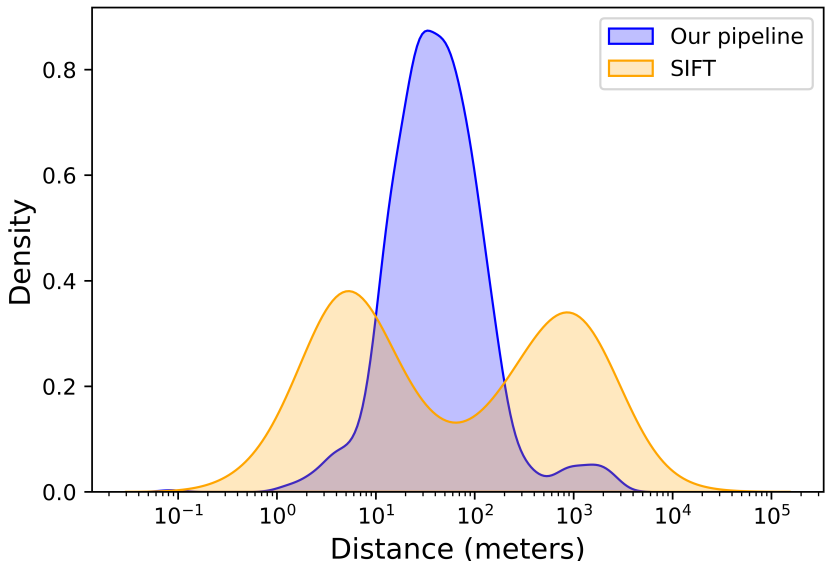



Figure 9: Kernel density estimations of the distributions of distance error of each pair of control points.

Figure 9 illustrates that most of the distance errors from our pipeline range between 10 and 100 meters, whereas the baseline method either achieves very high precision (below 10 meters) or very poor precision (over 1000 meters). This comparison highlights a trade-off between bias and variance in georeferencing. The bias in our pipeline likely stems from the inconsistency between the toponyms' image coordinates and their geographic coordinates. By taking the center of the polygon as the toponym's image coordinate, we inevitably introduced and accumulated bias, reducing prediction precision.

Another contributing factor could be the residual noise in RANSAC. To investigate this, we manually annotated 74 highly recognizable toponyms on map btv1b53027745s, which we considered "clean" data. We then computed the best transformation and found the mean distance error to be $11.30 \pm 3.02$ meters, slightly better than our pipeline's result of $18.67 \pm 2.87$ meters. This suggests that additional noise in the transformation estimation can lead to larger errors. However, since toponyms are nearly ubiquitous on the map and only six toponym coordinate pairs are needed to compute the affine transformation at a minimum, the number of failing cases is lower compared to using the road network feature.

As shown in Table 2, the success rate on the maps of Jerusalem was 3 out of 10, with a larger mean distance error compared to the maps of Paris. One potential explanation for this is the prevalence of Biblical names, which makes it difficult for the geocoder to provide accurate coordinates. This inaccuracy can adversely affect the affine transformation estimation when too few toponyms are correctly geocoded.

### 2.5.3 Toponym Normalization Results

We observed that the success of our pipeline largely depends on the toponym normalization process. Without this step, the transformation estimation rarely succeeds. To study the effect normalization, we sampled 20 normalized toponyms on map btv1b53087758d, as shown in Table 3. In most cases, normalization successfully corrected spelling errors and completed abbreviations (e.g., "R. Montgalet" to "Rue Montgallet" and "NT PARNASSE" to "Montparnasse"). However, normalization can sometimes overpredict when the original name lacks sufficient information; for example, "Quai des" was normalized to "Quai des Orfèvres". Although these false predictions are mostly filtered out by RANSAC, improving the normalization process is a promising approach to enhancing the georeferencing pipeline.

# 3 General Discussion and Conclusion

To summarize the achievements of this project, we first designed a novel architecture for word grouping, enabling the model to effectively group toponyms in various scenarios, including single lines, multiple lines, and twisted lines. We then applied a consensus ranking algorithm to obtain grouping results on a map scale, which were directly fed into the georeferencing pipeline. Through toponym normalization, geocoding, and transformation estimation, our pipeline successfully aligned 29 out of 41 Paris maps and 3 out of 10 Jerusalem maps. Additionally, the mean distance error and variance were significantly lower than those produced by the SIFT-based approach.

The technical limitations of this project are as follows:

- **Scarcity of data.** The model effectively learns the geometric patterns of toponyms, but it is still under-fitted for broader scenarios. With additional training data, the grouper could adapt to more situations.
- **Insufficient features.** Our grouper sometimes falls short in grouping highly coherent words as we only considered curve line features and did not include textual features. Additionally, incorporating visual information, such as rivers, could further enhance the performance of the grouping method.
- **Meta optimization problem.** Minimizing manual labor requires further research on meta optimization. One significant limitation of our georeferencing pipeline is the need to set different inlier thresholds for different maps. Additionally, removing marginal text would become costly as the amount of data increases. This report mainly serves to prove the concept, so we did not investigate how this hyper-parameter relates to factors such as the density of toponyms or the scale of the map, as well as how to deal with marginal text in an automated way. These are what we plan to study in future.

Currently, the pipeline is one-directional, progressing from word detection to toponym grouping to georeferencing the entire map. Within the scope of this project, the results are satisfactory. However, in a broader context, the georeferenced maps and their toponyms will be used to enhance historical gazetteers, so the precisions of all components are important. We therefore seek to minimize the artifacts and noises introduced by the pipeline, so that we could provide a more robust method of digitizing historical maps with minimum information loss.

# References


[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2016. arXiv:1409.0473.

[2] Runfola D Bahgat K. Toponym-assisted map georeferencing: Evaluating the use of toponyms for the digitization of map collections. 2021. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8601528/, doi:10.1371/journal.pone.0260039.

[3] Gregory Allord Kenneth M. Then James E. Burt, Jeremy White and A-Xing Zhu. Automated and semi-automated map georeferencing. *Cartography and Geographic Information Science*, 47(1):46–66, 2020. arXiv:https://doi.org/10.1080/15230406.2019.1604161, doi:10.1080/15230406.2019.1604161.

[4] Maxime Jan. A generic method for cartographic realignment using local feature matching. 2022. URL: https://github.com/JanMaxime/PDM/blob/master/report/PDM%20report%20Maxime%20Jan.pdf.

[5] Mengjie Zou Kaede Johnson, Tianhao Dai. Extracting toponyms from maps of jerusalem. 2023. URL: https://fdh.epfl.ch/index.php/Extracting_Toponyms_from_Maps_of_Jerusalem.

[6] Jina Kim, Zekun Li, Yijun Lin, Min Namgung, Leeje Jang, and Yao-Yi Chiang. The mapkurator system: A complete pipeline for extracting and linking text from historical maps. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*, pages 1–4, 2023.

[7] Zekun Li, Yijun Lin, Yao-Yi Chiang, Jerod Weinman, Joseph Chazalon, Solenn Tual, Julien Perret, Bertrand Duménieu, and Nathalie Abadie. ICDAR 2024 competition on historical map text detection, recognition, and linking. https://rrc.cvc.uab.es/?ch=28, 2024. Dataset organizers.

[8] Rémi Petitpierre. Bnf-jadis/supermaprealigner: v0.1-initial, May 2023. doi:10.5281/zenodo.638211649.

[9] Rémi Petitpierre. *Neural networks for semantic segmentation of historical city maps: Cross-cultural performance and the impact of figurative diversity*. PhD thesis, 07 2020. doi:10.13140/RG.2.2.10973.64484.

[10] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks, 2017. arXiv:1506.03134.

[11] Jerod Weinman. Geographic and style models for historical map alignment and toponym recognition. In *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*, volume 01, pages 957–964, 2017. doi:10.1109/ICDAR.2017.160.

[12] Mengjie Zou. Automated toponym extraction pipeline for historical maps. 2024. URL: https://github.com/SesamePaste233/ToponymExtractor.

| Map | Mean Dist. (m) | Best Threshold | # toponyms | SIFT (m) |
|---|---|---|---|---|
| btv1b53010772z | **69.39 ± 18.38** | 0.005 | 134 | 179.25 |
| btv1b53020877w | 26.08 ± 4.47 | 0.01 | 279 | **4.50** |
| btv1b530277395 | 46.06 ± 9.32 | 0.004 | 172 | **6.54** |
| btv1b53027745s | **18.67 ± 2.87** | 0.006 | 680 | 870.11 |
| btv1b53027781n | **41.34 ± 8.94** | 0.007 | 337 | 1398.89 |
| btv1b53028002g | 21.81 ± 3.73 | 0.003 | 270 | **4.12** |
| btv1b53035270h | 53.53 ± 13.14 | 0.004 | 136 | - |
| btv1b53057644w | 22.53 ± 4.68 | 0.009 | 199 | **13.71** |
| btv1b530589233 | 21.45 ± 3.03 | 0.003 | 364 | **8.76** |
| btv1b530620624 | 62.02 ± 11.74 | 0.003 | 592 | **7.32** |
| btv1b53062149t | **107.29 ± 20.46** | 0.004 | 208 | 685.21 |
| btv1b53062261f | **90.78 ± 18.23** | 0.006 | 267 | 1452.68 |
| btv1b530633990 | 8.78 ± 1.77 | 0.001 | 1196 | **4.82** |
| btv1b53069957s | 68.33 ± 10.10 | 0.011 | 96 | **4.00** |
| btv1b530706426 | **64.19 ± 20.13** | 0.002 | 123 | 377.38 |
| btv1b53085158f | **44.63 ± 11.85** | 0.006 | 251 | 334.40 |
| btv1b530851699 | 39.93 ± 8.59 | 0.004 | 342 | **9.78** |
| btv1b530851803 | **36.61 ± 10.53** | 0.002 | 216 | 369.74 |
| btv1b530852257 | 16.74 ± 2.47 | 0.003 | 335 | **5.83** |
| btv1b53085242q | **49.10 ± 16.52** | 0.003 | 263 | 4308.63 |
| btv1b530852452 | 103.23 ± 14.32 | 0.005 | 524 | **52.64** |
| btv1b530852507 | 19.21 ± 4.16 | 0.002 | 359 | **3.52** |
| btv1b53085266b | 41.36 ± 9.88 | 0.003 | 423 | **14.64** |
| btv1b53085541d | 16.32 ± 4.30 | 0.003 | 166 | **8.74** |
| btv1b53085542v | 29.53 ± 6.88 | 0.002 | 408 | **8.38** |
| btv1b530874370 | 619.17 ± 232.11 | 0.004 | 32 | - |
| btv1b530875593 | **104.82 ± 30.22** | 0.004 | 263 | 627.15 |
| btv1b53087560g | **103.16 ± 20.00** | 0.005 | 265 | 454.08 |
| btv1b53087758d | 18.11 ± 3.46 | 0.002 | 271 | **6.97** |
| btv1b53121232b | **17.56 ± 4.26** | 0.012 | 1084 | 830.97 |
| btv1b531896457 | 80.53 ± 27.04 | 0.011 | 241 | **3.96** |
| btv1b53192547n | **19.47 ± 4.33** | 0.009 | 857 | 2166.34 |
| btv1b550044782 | **46.83 ± 7.56** | 0.003 | 254 | 954.36 |
| btv1b55013275x | 1463.73 ± 314.21 | 0.015 | 59 | - |
| btv1b8439886z | 28.60 ± 7.13 | 0.008 | 349 | **8.29** |
| btv1b8440652z | **220.00 ± 57.03** | 0.011 | 105 | 1319.67 |
| btv1b8440780m | **105.78 ± 25.22** | 0.018 | 34 | 1209.19 |
| btv1b8444443j | 73.77 ± 11.15 | 0.004 | 61 | **5.03** |
| btv1b8445269d | 75.33 ± 18.89 | 0.006 | 102 | - |
| btv1b84599611 | **37.28 ± 7.94** | 0.004 | 284 | 926.36 |
| btv1b8492690n | **82.31 ± 17.70** | 0.005 | 87 | 835.81 |
| **# Success** | **29/41** | - | - | 28/41 |
| **Mean** | **102.81** | - | - | 526.53 |
| **Std.** | **238.30** | - | - | 844.61 |

Table 1: Georeferencing results on maps of Paris. The mean distance error of our pipeline (Mean Dist.) and the baseline (SIFT) are compared. The inlier threshold used in RANSAC and the number of remaining toponyms after filtering are also reported.

| Map | Mean Dist. (m) | Best Threshold | # toponyms |
|---|---|---|---|
| bartlett (1842) | $72.65 \pm 4.81$ | 0.003 | 13 |
| dufour (1837) | $438.80 \pm 18.05$ | 0.019 | 10 |
| gelis (1866) | - | - | 0 |
| kiepert (1845) | $121.23 \pm 7.84$ | 0.001 | 13 |
| pierotti (1858) | $322.02 \pm 5.31$ | 0.007 | 8 |
| sieber (1818) | - | - | 3 |
| unknown (1821) | $313.80 \pm 25.64$ | 0.003 | 10 |
| unknown (1868) | $46.37 \pm 2.37$ | 0.001 | 10 |
| vandevelde (1846) | $63.82 \pm 2.49$ | 0.008 | 14 |
| weller (1860) | $96.33 \pm 3.94$ | 0.005 | 14 |
| **# Success** | 3/10 | - | - |
| **Mean** | 184.38 | - | - |
| **Std.** | 150.37 | - | - |

Table 2: Georeferencing results on maps of Jerusalem. The inlier threshold used in RANSAC and the number of remaining toponyms after filtering are also reported.

| Detected Name | Normalized Name |
|---|---|
| Rae St. Honore | Rue Saint-Honoré |
| R. de Menilmontant | Rue de Menilmontant |
| Montimartre B. | Montmartre, Paris |
| Bre. Rochcchouart | Boulevard de Rochechouart. |
| Rue Malesherbes | Rue Malesherbes |
| Pte Rue de Reuilly | Petite Rue de Reuilly |
| Poissonniere Rue | Rue du Faubourg Poissonnière |
| Charbonmiere Rue | Rue de la Charbonnière |
| D. des ictoires martre | Rue des Victoires, Paris |
| R. Montgalet | Rue Montgallet |
| Quai des | Quai des Orfèvres |
| R. de Bourgogne | Rue de Bourgogne |
| Fosses St. | Fosses Saint-Jacques |
| NT PARNASSE | Montparnasse |
| Champs Bourbe | Champs-Élysées |
| Rue Mechin | Rue Méchin |
| Rempart Capucines | Rue des Capucines |
| des Mathurins | Rue des Mathurins |
| R. des Fourneanx Rue | Rue des Fourneaux |
| R. de Beaune Royal | Rue de Beaune Royal |

Table 3: Twenty toponyms normalized by GPT-3.5 on map btv1b53087758d.

# Appendix