Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

IKG
Institute of Cartography
and Geoinformation

Automatic Map Storytelling with Generative Pre-trained Transformer (GPT) Models

Geomatics Project MSc
Fall Semester 2023

Ziyi Liu
ziyliu@student.ethz.ch

Claudio Affolter
claudaff@student.ethz.ch

Advisors:
Sidi Wu, Dr. Yizi Chen

Supervisor:
Prof. Dr. Lorenz Hurni

January 12, 2024

Acknowledgements

We would like to thank our advisors, Sidi Wu and Dr. Yizi Chen, for their great support throughout the project and their encouragement to also go for a conference paper. In addition, we thank Prof. Dr. Lorenz Hurni for the opportunity to work on this project at the Chair of Cartography, Institute of Cartography and Geoinformation at ETH Zurich.

Abstract

Maps provide information and knowledge about the world. However, the diversity of map types as well as the wide range of map styles and thematic/temporal contexts make it challenging for non-experts in the field (i.e., non-cartographers) to easily identify and understand maps when lacking sufficient data sources. This is especially true for historical maps, as they often feature non-standard projections as well as hand-drawn styles and are usually more artistic than modern maps. Even though state-of-the-art image captioning methods such as CLIP and ClipCap are promising, for historical maps, they generate captions that are either too simple, too general, or even wrong. To address these challenges, this project focused on exploring Generative Pre-trained Transformer (GPT) models and fine-tuning existing image captioning methods for map storytelling. Given a historical input map (topographic or pictorial), a caption answering the questions *Where?*, *What?*, *When?* and *Why?* should be generated. A decision tree structure-based method combining fine-tuned CLIP models, and the generative capabilities of GPT was developed. The results reveal that this approach, despite the limited dataset, outperforms CLIP in terms of prediction accuracy overall by 34 percentage points, corresponding to an over 72% boost in performance. Provided that the content of the historical input map can be described by the present caption category classes, this method is capable of describing it with respectable accuracy in a storytelling fashion and could serve as a basis for future historical map captioning systems.

Contents

List of Figures

Chapter 1

Introduction

This chapter provides an introduction to the topic by first explaining the motivation and challenges behind it. Afterwards, the overall problem is presented and the objectives of this project are listed. The chapter concludes with a brief overview of the content covered in this report.

1.1 Motivation

There is a wide range of different map types, such as topographic and thematic maps, that allow people to learn more about the geography, history and culture of a particular location at a specific time. However, not only are the map types very diverse, but so are the map styles, layouts, and thematic/temporal contexts. For this reason, it can be difficult for people who are not experts in the field (i.e., non-cartographers) to correctly identify and understand maps, which is especially true for historical maps. Compared to modern maps, historical maps often contain less accurate geographic information, varying artistic or religious symbols and legends, non-standard projections, and hand-drawn styles, which makes it even more challenging to quickly capture key information in maps from different eras.

Image captioning provides descriptions for images in natural language, bridging the gap between visual and textual information. It serves as a powerful tool in various situations, including content understanding for individuals with visual impairments, image tagging for database management, efficient search and retrieval of images, etc. Recent advances in image captioning methods such as CLIP[1](Ilharco et al., 2021) and ClipCap[2] (Mokady et al., 2021) are promising to help overcome the challenges faced by non-domain experts in interpreting historical maps as they make it possible to automatically classify and describe images. These methods, combined with GPT models pre-trained on large datasets consisting of texts from books, articles, and websites are able to generate meaningful

[1]Contrastive Language-Image Pre-Training
[2]CLIP Prefix for Image Captioning

text that matches the input images. CLIP is designed to understand the connections between texts and images using contrastive learning, which can be used for zero-shot text prediction of images. Built on top of CLIP, ClipCap is designed especially for image captioning. It uses a fine-tuned language model and can automatically generate natural language descriptions for images without additional information.

1.2 Problem statement

Even though the state-of-the-art image captioning methods mentioned above are promising, their limitations become apparent when the input image is a historical map. The captions automatically generated by for example ClipCap are either too simple, too general, or even wrong. Table 1 below shows two input maps from the David Rumsey Historical Map Collection (Rumsey, David, and Cartography Associates, 2024) and corresponding output captions generated by ClipCap.

Input		
Output	An old map of the world is shown with a bridge in the background.	An old book with a map and a clock on it.

Table 1: Two maps captioned by ClipCap.

As one can see, the results are only partially correct. ClipCap has correctly recognized in both cases that the image content is a map, but the remaining information is not useful. Neither does it understand the subject of the map on the left, nor does it understand which area the map on the right depicts. Furthermore, the words *bridge* or *clock* would certainly confuse people when reading such a caption. Therefore, it can be said that the current captioning methods are not fully capable of producing accurate captions for historical maps.

1.3 Objectives

The main goal of this project is to explore and fine-tune image captioning and GPT models for map storytelling, where given a historical input map, the model should generate a caption answering the following questions:

1. **Where?** The area which is depicted on the input map.
2. **What?** The map type as well as the pictorial map topic and topographic map style.
3. **When?** The century in which the topographic map was created.
4. **Why?** The purpose of the map.

The focus will be set on topographic maps created between the 16th and 19th century, and pictorial maps featuring different topics.

1.4 Content overview

This project report first covers the relevant theory and related work. Afterwards, the methodology is presented, which includes the procedure for creating a map dataset with corresponding ground-truth captions and the process of developing the map storytelling method with in-depth explanations of code. Later on, all results are discussed, and possible consequences and limitations are pointed out.

Chapter 2

Theory and Related Work

This chapter explores relevant deep learning methods for historical map storytelling and explains two advancements and suitable models in image captioning in detail, i.e., CLIP and ClipCap. In the end, this chapter covers Generative Pre-trained Transformer (GPT) models, particularly Text-davinci-003, GPT-3.5-turbo, and GPT-4, detailing their capabilities and applications in natural language processing.

The diverse styles of maps, ranging from topographic maps to pictorial maps, can significantly aid people in understanding map content, but their variety also presents challenges for automatic map interpretation and classification. The distinction of map styles is helpful for further effective map analysis. The study by Zhou et al. (2018) uses state-of-the-art deep convolutional neural networks (CNNs) for automatic map-type classification. A dataset of seven categories of map types was employed, including topographic maps, terrain maps, physical maps, urban scene maps, national maps, 3D maps, etc. Another research by Raimund Schnürer and Hurni (2021) distinguishes maps from images and then separates pictorial maps from others to conduct object identification and classification within pictorial maps using CNNs.

To achieve optimal results for deep learning methods, a substantial amount of training data is essential for training. There are several data augmentation methods. Yingjie Hu and Li (2022) developed a GIS-based data augmentation method that can generate labeled training map images from shapefiles, which can be used to enrich metadata concerning spatial extents and place names.

Regarding the content of maps, deep learning models also have shown their capabilities in understanding detailed aspects of maps. Touya et al. (2020) used convolutional neural networks trained with maps, showcasing promising results in inferring labels of the map including boundaries, cities, hydrography, etc., and the extent of the map including World, Europe, France, and Paris. Recent advancements in image captioning technologies, which utilize one pipeline for understanding and integrating map contents, show considerable promise in generating comprehensive and coherent outputs from map data.

2.1 Image captioning

Image captioning is the task of generating a natural language description of a given image. It lies in the fields of both computer vision and natural language processing, which involves understanding the content of the image and turning the visual information into textual representations. This process is crucial to map storytelling, as it relies on accurate extraction and interpretation of key features of a map and constructs a compelling story.

2.1.1 CLIP

CLIP (Radford et al., 2021) is a neural network that can predict the most relevant text snippet as the label for an image. As shown in Figure 1, it applies a text transformer model as the text encoder and a ResNet or Vision Transformer as the image encoder, and jointly trains these two encoders at the same time to learn a multi-modal embedding space. CLIP uses the idea of natural language supervision and it is trained with text as a whole instead of an exact word as a label, which helps to establish connections between image representations and natural language and makes it more efficient at zero-short transfer. This model is trained over a dataset of over 400 million image and text pairs that are retrieved from public sources on the Internet, and maximizes the cosine similarity of the correct pairs while minimizing that of the incorrect ones during training. For data augmentation, it only applies a random crop to resized images. Given an image and a list of potential label texts, CLIP takes the image and returns the possibilities for each item in the list, and thus gets the most suitable one as the final output.

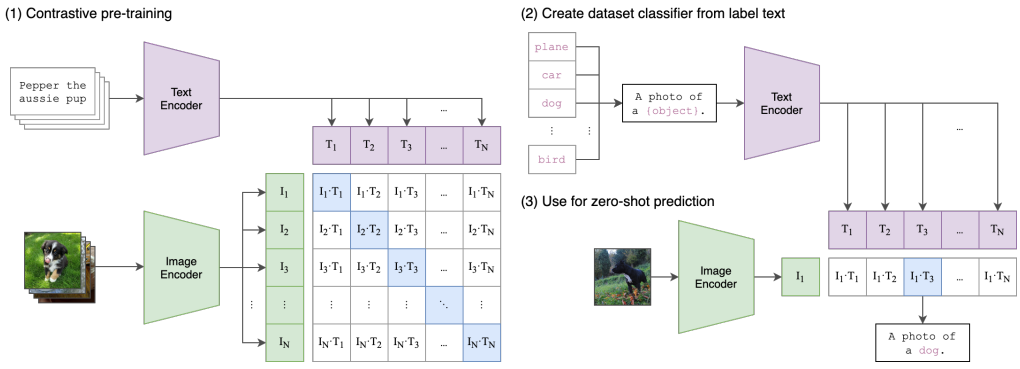

Figure 1: CLIP model architecture

2.1.2 ClipCap

ClipCap (Mokady et al., 2021) is designed for image captioning. Since CLIP is trained over a large amount of data and can generate representative semantic encodings for an image without extra supervision, ClipCap applies CLIP as its image encoder. As shown in Figure 2, it trains a lightweight transformer-based mapping network to generate a fixed length prefix from CLIP encoding and a learned constant to GPT-2, and then fine-tune the language model, i.e. GPT-2 taking the generated prefix together with captions of the image as input and captions themselves as target. When inference, GPT-2 is utilized to generate meaningful captions directly from prefix embeddings of the image. There are two modes of training, including training only transformer mapping network or also with fine-tuning of GPT-2. Pre-trained ClipCap models are provided as well, which were trained over data from COCO dataset (Lin et al., 2014) and Conceptual Captions dataset (Sharma et al., 2018). These two datasets have different styles, with Conceptual Captions, sourced from the web, offering a wider variety of styles. Meanwhile, ClipCap's training is quite fast. Different from CLIP, ClipCap can generate a sentence as a caption for a given image, without the need for additional information or sources.

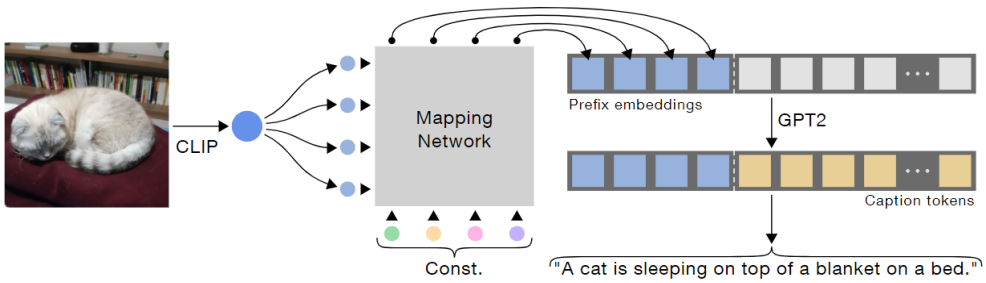

Figure 2: ClipCap model architecture

2.2 Generative Pre-trained Transformer Models

Generative pre-trained transformer (GPT) models (Yenduri et al., 2023) are large-scale, transformer-based deep-learning neural network architectures developed by OpenAI in the field of Natural Language Processing. Large language models in the GPT family are designed to complete tasks including answering questions, translating languages, summarizing texts, generating creative writing, etc., in human-like text, basically through predicting the next word or the likelihood of a sequence of words. With millions to billions or even trillions of parameters, they are pre-trained over extremely vast amounts of data and have

shown extraordinary abilities to understand and generate complex text in natural language. Most GPT models are accessible through APIs. While GPT-4 is now the most advanced system, here are the models chosen for our task.

2.2.1 Text-davinci-003

Text-davinci-003 was developed by OpenAI and published in 2022. It is noted for its advanced ability of natural language understanding and generation, especially for contextually relevant text completion. It is proficient in tasks like content creation, summarization, and complex problem-solving. However, to be an older completion and embedding model, it was shut down in on January 4th, 2024.

2.2.2 GPT-3.5-turbo

GPT-3.5-turbo (OpenAI, 2022), a model in the GPT-3.5 series, is optimized for quick interactions. It is suitable for applications requiring rapid response times while with language processing in good quality. Its performance is notable in the context of chat applications and real-time language processing tasks.

2.2.3 GPT-4

GPT-4 (OpenAI, 2023) is the latest and most advanced version. It is more capable of understanding and generating text and can handle complex tasks better than GPT-3.5. It is also known for its multi-modal capabilities of understanding and generating content in other modalities, such as images.

Chapter 3

Methodology

In this chapter, it is explained how exactly and with what means a method for automatic map storytelling was developed.

3.1 Procedure

3.1.1 Creation of map dataset

As with any project that involves fine-tuning models, the first step was to assemble a map dataset with corresponding ground-truth captions. For this task, the David Rumsey Historical Map Collection (Rumsey, David, and Cartography Associates, 2024) was the ideal data source. This map collection contains more than 200'000 historical maps from all over the world and each map is complemented by detailed metadata. The metadata contains, for example, the map title, date and location and, in some cases, detailed descriptions of the map content and background information. An example of how the metadata is structured can be seen in Appendix A.

Since, as mentioned in 1.3, the focus would be topographic maps created between the 16th and 19th century, and pictorial maps, only the maps in the categories *Classical*[1] and *Pictorial map*[2] were considered. In order to then automatically download all maps of interest, a Python script (AutomaticMapDownload.py) was written that allows to do so with the help of Selenium WebDriver (SeleniumHQ, 2023). This script will be explained in more detail in subsection 3.2.1.

This script was then used to download all necessary maps and metadata from the David Rumsey Historical Map Collection. In total, 1'524 maps belonging to the category *Classical* and 5'234 pictorial maps were downloaded. To keep the memory size low, all maps were exported as *Small* (i.e., up to 768 px). After carefully looking through the downloaded maps, it became clear that not all of them were useful since some of the "maps" featured images of book covers, letters

[1] https://www.davidrumsey.com/luna/servlet/view/all/what/Classical
[2] https://www.davidrumsey.com/luna/servlet/view/all/what/Pictorial+map

or buildings, which had to be removed. This required some manual work, but fortunately the file names made it obvious which files had to be deleted from the dataset. As a result of this filtering process, the dataset was reduced to 1'334 *Classical* and 3'183 pictorial maps, which together take up less than 2 GB of space.

3.1.2 Creation of ground-truth captions

To create ground-truth captions answering the questions introduced in 1.3, the necessary information had to be extracted from the metadata belonging to each map. For this task, two Python scripts (CaptionGenerationClassical.py and CaptionGenerationPictorial.py) were created that first load in all maps and metadata. Then, for each map, its path is saved and the corresponding metadata is scanned line for line for keywords, e.g., *Date* and *Country* etc., as the metadata is structured similar to an attribute table. The needed information could then be saved in separate lists. These two scripts will be explained in more detail in subsection 3.2.2.

After applying the above-mentioned approach to both the *Classical* and pictorial maps, the problems with this approach became clear and will be addressed in the following, as well as how they were (partially) solved:

Classical maps:

1. **Where?** Extracting the ground-truth answering this question from the metadata created major challenges. Firstly, some maps contain several location attributes that are not ordered by importance, i.e., the first location showing up in the metadata does not always correspond to the area taking up the most space in the map. Secondly, in a couple of cases, the location attribute is incorrect. Finally, there are maps with a very general location attribute, e.g., *Europe*, while the depicted country is Italy. To overcome these problems, the scrips had to be extended to be able to also derive locations from the title and in many cases, the location attribute had to be manually corrected to ensure accurate ground-truth captions.

2. **What?** Since separate scripts for *Classical* and pictorial maps were made, the map type (topographic map) had not to be derived from the metadata itself. For the map style part, information from the *Note* attribute had to be used. Because the content of this attribute usually varied a lot across all maps (and would lead to too many style categories) and sometimes was in other languages instead of English, ways to create fewer distinct style groups had to be explored. The most successful approach was using the Python Counter class to automatically generate a dictionary where the note information is stored as key and the count as value. This way, identical note information and the number of occurrences could be made visible which then helped to reduce the number of style categories to just six. Note that

this only worked for maps with a *Note* attribute. For the remaining maps (about 15%), no style ground-truth could be created.

3. **When?** This information was extracted by scanning the *Date* attribute in the metadata and since this field was never incomplete, no problems arose.
4. **Why?** As the metadata did not contain information about the purpose of a given map, it was decided to let GPT-3.5 (OpenAI, 2022) answer this question with its generative capabilities instead. Therefore, no ground-truth captions were created for this part.

Pictorial maps:

1. **Where?** Using the Python Counter class to get an idea about the class imbalance regarding the locations, the following was revealed: Overall, the pictorial maps are depicting 1'349 different locations (with 3'183 maps in total) and over 600 maps are either depicting the world or the United States. The majority of the remaining locations are only featured on a single map each. Due to this huge class imbalance, it was decided to only focus on the two largest classes, namely pictorial maps depicting either the whole world or the United States.
2. **What?** Analogous to the *Classical* maps, the map type (pictorial map) had not to be derived from the metadata itself. For the map topic part, at first the information from the title was extracted, which led to way too many topic classes. Once again, solutions had to be found to reduce the number of topics or generalize them. Extracting the information from the *Subject* attribute led to fewer classes but it was mostly too general and not all maps even contained this attribute field. Finally, it was decided to download pictorial maps depicting either the world or the United States featuring common topics separately using the *Subject* attribute (e.g., pictorial maps with topic *Military*) and then manually try to group them into certain topic classes.
3. **Why?** Just like for *Classical* maps, it was decided to let GPT-3.5 (OpenAI, 2022) answer this question with its generative capabilities.

The results of this ground-truth creation process are separate lists for each of the caption categories containing the captions as well as corresponding lists containing the paths to the maps. Here, the ground-truth captions are typically only composed of keywords, e.g., *Italy* or *16th century*, and not full sentences.

Note that instead of creating the ground-truth captions with the above-mentioned semi-automatic method, it was also explored to create them fully automatic from the metadata using large language models, such as GPT-4 (OpenAI, 2023) and HuggingChat Python API (Soulter, 2023). The idea was to utilize the advanced generative capabilities of these models by calling their APIs, enabling the automatic expansion of the dataset while solving the problems mentioned above.

These include the following: 1.) multi-location attributes; 2.) categorizing attributes with high variety into fewer distinct groups; 3.) no direct information for the *Why?* part; 4.) metadata in other languages instead of English and 5.) reducing the amount of tedious manual work. This method not only augments the dataset size but also inserts the model's linguistic expressions and styles into the ground-truth captions which are easy to understand for large language models in the later steps, potentially enhancing the robustness of the trained map captioning models. Text-davinci-003, GPT-3.5-turbo, GPT-4, and HuggingChat were used for trials. To construct an optimal prompt for GPT-4, specific attributes of metadata were selected: *Short Title*, *World Area*, *Subject* and *Full Title*.

The structured prompt consists of two main parts, including the attributes to be used and the requirements for the generated captions, which can also present the challenges of prompt engineering for ground-truth caption creation, as shown below:

```
f"Summarize this info as a reasonable sentence in English for image
    captioning of historical maps, based on: {attributes}.
    Requirements: 1. translate all other languages into English 2. as
     short as possible 3. no abbreviations 4. without any punctuation
     mark 5. no longer than 12 words 6. only 'what' and 'where',
    without time 7. only mention location info once 8. without people
    's name 9. all in lowercases with the general format: map
    depicting [short title] in [world area]."
```

However, even after extensive prompt engineering experiments, this approach had to be discarded. Details are further explained in Section 5. Table 2 below gives an overview of the final number of classes and maps used for each of the caption categories. In addition, Appendix B gives more insight into the classes.

Caption category	Number of classes	Number of maps
Map Type (What?)	2	4'517
Area (Where?)^T	27	723
Style (What?)^T	6	1'132
Century (When?)^T	4	1'334
Area (Where?)^P	2	290
Topic (What?)^P	13	284

Table 2: Overview of number of classes and maps for each of the caption categories. The superscript letters T and P stand for *Topographic* and *Pictorial* maps, i.e., 723 topographic maps are depicting 27 different areas.

3.1.3 Fine-tuning

ClipCap:

Initially, the idea was to fine-tune the ClipCap model since it is capable of generating full-sentence captions. For this approach, the ground-truth captions generated with the methods explained in 3.1.2 had to be extended to full sentences as well. This was done by simply embedding the keyword captions into predefined sentences, as shown in the following code snippet:

```
captionsArea.append(f"Topographic map depicting {area}.")
captionsDate.append(f"Created in the {date}.")
captionsStyle.append(f"{note}.")
```

It was then planned to fine-tune models for each of the caption categories, thus the separate caption lists. Later, the generated sentences should be concatenated into a full caption. The reason for this is that otherwise the output string, which had to be constructed by a general ClipCap model at once, would have been too long. Furthermore, it was believed that having multiple specialized models would create a more flexible method for caption generation, especially since the number of maps in the dataset is not large enough to expose a general model to all possible caption variations.

Unfortunately, even after considerable efforts to fine-tune ClipCap, this approach did not lead to satisfying results, which ultimately led to this model being discarded. The reason for the failure of this approach is assumed to be the following: ClipCap's underlying image encoder CLIP model comes in two variants. The first one was pre-trained on the COCO image dataset (Lin et al., 2014) and the second one on the Conceptual Captions dataset (Sharma et al., 2018). These datasets mostly consist of images featuring real-life situations that are too distinct from historical maps. Furthermore, since during the fine-tuning of ClipCap, only the transformer mapping network and GPT-2 are actually fine-tuned, the underlying image encoder CLIP model stays frozen and is not exposed to these new historical map images.

CLIP:

After realizing that ClipCap was not suitable for this dataset, it was tested to see how the base CLIP model performs when it receives historical maps as input. Surprisingly, this model is already able to recognize certain countries and topics. This is why it was decided to continue fine-tuning CLIP instead.

The fine-tuning process was adopted from Jhon Parra (2022) and adapted for this specific case. Overall, the idea was once again to fine-tune specialized models for each of the six caption categories. Here, the keyword ground-truth captions, created with the methods described in subsection 3.1.2 were used without having to extend them to full sentences. For training each model, the following hyperparameters were selected:

- Maximum number of epochs: 32
- Batch size: 10
- Loss: Cross-entropy loss
- Optimizer: Adam
- Initial learning rate: 1e-5
- Scheduler: Cosine Annealing Learning Rate scheduler

The models were all trained on a single NVIDIA RTX A4000 GPU (16 GB) with a maximum memory consumption of 4500 MB and a maximum training time of 20 minutes. This whole fine-tuning process will be explained in more detail in subsection 3.2.3. Furthermore, subsection 3.2.4 shows how the accuracy of fine-tuned models was assessed.

3.1.4 Combining fine-tuned models

Overall, there are two approaches with different logic to integrating GPT models into storytelling. The first involves leveraging models like ClipCap that incorporate GPT to directly generate stories. The second approach enhances the natural language processing part of the pipeline by applying GPT capabilities individually, thus improving the overall quality of textual output in storytelling. Since the first approach was already discarded, a pipeline with an underlying decision tree structure was chosen to be the implementation of the second approach. The idea was to use fine-tuned CLIP models to predict keyword captions and then use a GPT model to tell stories based on the generated keywords.

The fine-tuned CLIP models for the six categories, including map type, area of topographic map, century of topographic map, style of topographic map, area of pictorial map, and topic of pictorial map, are combined by the above-mentioned decision tree structure, as shown in Figure 3. An input map is first categorized by map type, distinguishing between topographic and pictorial maps. Each map type is further analyzed and its area, century, style or topic is predicted by a corresponding CLIP model. After completing this process, the method has generated a set of four keyword labels for a topographic map or three for a pictorial map.

Large language models (LLMs) are then utilized to generate the final output. Taking into account factors like speed of response, price of the API, quality of answers, etc., GPT-3.5 has been chosen. Unlike the prompt to generate a ground truth caption from metadata in subsection 3.1.2, except for generating a basic caption as a description for the map, the prompt is structured to be able to emphasize and respond to different aspects in the generated story as well, by embedding key inquiries into the construction. This serves as the extension to

the story content, and also as the response to the *why?* question. As the utilization situation of the map and the interests of its users can significantly vary, the question of 'why' remains open-ended. In this context, we offer a plausible explanation for the potential usage of the map. The following are tailored prompts for each question:

- What?: "What is this map about?"
- Where?: "Where is this map about?"
- When?: "When is this map about?"
- Why?: "What can this map be used for?"

The prompt of GPT-3.5 is as shown below, and the implementation of the above-mentioned is explained in more detail in subsection 3.2.5.

```
f"Please create a concise sentence that encapsulates these keywords:
    {keywords}. Additionally, provide a brief explanation, in under
   30 words, about: {questions}
```

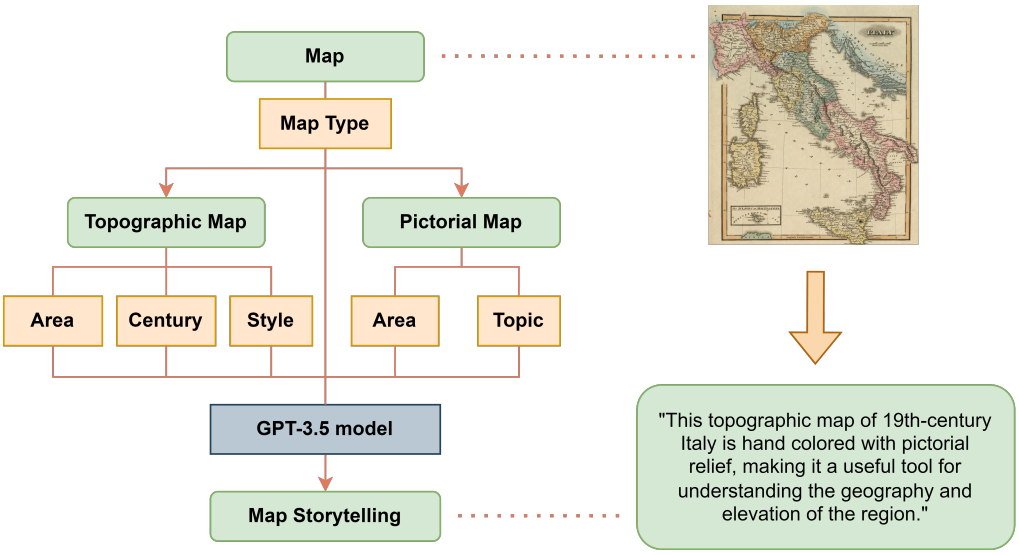

Figure 3: Decision tree structure

3.2 Details concerning the methodology

This section explains the underlying code of the most important Python scripts, which were briefly discussed in the previous section 3.1, in more detail. Imports and variable initializations are omitted. Note that the line numbers displayed here do not correspond to the ones in the actual code and slight simplifications were applied.

3.2.1 Automatic map download

In the following, the most important parts of the Python script *AutomaticMap-Download.py* are explained with the help of code snippets:

The three lines in the code snippet below set up a connection to the Chrome browser, specify a target website (in this case leading to the *Classical* maps on David Rumsey Historical Map Collection), and open that website in the browser using the Selenium WebDriver.

```
1 driver = webdriver.Chrome()
2 website = "davidrumsey.com/luna/servlet/view/all/what/Classical"
3 driver.get(website)
```

Then, at line 4 in the next snippet, the number of available pages showing the maps is entered manually. After that, the actual automated interaction with the browser happens. The first for-loop (see line 6) iterates over all pages, and the variable *map_elements* (see line 7) always contains the number of maps located per page. At line 9, the second for-loop then iterates over all maps per page and always the same procedure is executed: At line 11, the current map is clicked on, then starting from line 12, it is tried to locate an export button. If it is clickable, that button is pressed (see lines 21 and 22), or else the back button is pressed and the next map is inspected. After successfully clicking the export button, the desired resolution is selected (see line 25 and 27). Finally, at lines 28 and 29, the current map is left and the a new one will be selected in the next loop iteration.

```
4 amount_pages = 31
5
6 for page in range(amount_pages):
7     map_elements = driver.find_elements(By.CSS_SELECTOR,"img")
8
9     for m in range(len(map_elements) - 1):
10        map_element=driver.find_elements(By.CSS_SELECTOR,"img")[m]
11        driver.execute_script("arguments[0].click();", map_element)
12        try:
13            export_button = WebDriverWait(driver, 2).until(
14                EC.element_to_be_clickable((By.ID, "ExportButton"))
15            )
16        except TimeoutException:
17            print("No Export Button found")
```

```
18              driver.back()
19              continue
20
21          export_button.click()
22          export_button.click()
23
24          resolution = WebDriverWait(driver, 20).until(
25              EC.presence_of_element_located((By.ID, "ExportSize3"))
26          )
27          resolution.click()
28          driver.back()
29          driver.back()
```

3.2.2 Ground-truth caption generation

In the following, the most important parts of the Python scripts *CaptionGenerationClassical.py* and *CaptionGenerationPictorial.py* are explained. Note that because the structure of both scripts is almost identical, the former will be presented in more detail.

The code snippet below shows how each map and its metadata have to be loaded from the directory *ClassicalMaps* containing all *Classical* maps and corresponding metadata. After line 2, where the correct directory is chosen, each sub-folder in this main folder is iterated over (see line 4). In each of these sub-folders, there are two files: A .jpg image which is showing a map and a .txt file where its metadata is stored. At lines 10 and 14 it is then checked what the type of the current file is. The image paths (map paths) are simply stored in *image_paths* (see line 12), while the text content is prepared to be scanned at line 18.

```
1  maps_directories = ["ClassicalMaps"]
2  for maps_directory in maps_directories:
3
4      for root, dirs, files in os.walk(maps_directory):
5
6          for file in files:
7
8              file_path = os.path.join(root, file)
9
10             if file.lower().endswith(".jpg"):
11
12                 image_paths.append(file_path)
13
14             elif file.lower().endswith(".txt"):
15
16                 with open(file_path,"r","utf-8") as txt_file:
17
18                     txt_content = txt_file.read()
```

Starting from line 19, the metadata of the current map is scanned line for line for certain keywords, e.g., "Note" as seen in line 21, which correspond to the

attributes. The information is then stored in separate variables (see lines 22, 25 and 29).

```
19 for line in txt_content.split("\n"):
20
21     if line.startswith("Note"):
22         note = line.split("\t", 1)[1]
23
24     elif line.startswith("World Area"):
25         area = line.split("\t", 1)[1]
26     ...
27
28     elif line.startswith("Date"):
29         date = line.split("\t", 1)[1]
```

Later, some of these saved attribute values had to be further processed, especially the location attribute, which is denoted as *area* in the code snippet below. In many cases it was necessary to bring the area variable to a common value due to typos or different languages as seen at lines 30 and 33. On the other hand, it was needed that the area variable had to be derived from the title of the map itself because the metadata was not accurate enough. This can be seen at lines 37 and 40.

```
30 elif "romain, empire" in area.lower():
31     area = "Roman Empire"
32
33 elif "balkan" in area.lower() or "balkans" in area.lower():
34     area = "Balkans"
35 ...
36
37 elif "belgie" in short_title.lower():
38     area = "Belgium"
39
40 elif "graecia" in short_title.lower():
41     area = "Greece"
```

Finally, the processed variables belonging to each caption category for the current map are stored in separate lists. When all maps and metadata have been processed, the lists are saved as NumPy arrays for later use.

```
42 captions_area.append(area.lower())
43 captions_note.append(note.lower())
44 ...
45 captions_npy = np.array(captions_area)
46 np.save("classicalMapsCaptionsArea.npy", captions_npy)
47 captions_npy = np.array(captions_note)
48 np.save("classicalMapsCaptionsNote.npy", captions_npy)
49 ...
```

The following code snippet shows how the ground-truth caption generation looked like for pictorial maps (*CaptionGenerationPictorial.py*). Here, the difference was that, as already mentioned in 3.1.2, separate folders containing a single general

topic each were processed. In the case below (see line 1), the contents of the folder *MilitaryWorld* containing military pictorial world maps is checked. Then, based on keywords found in the map's title, the map is assigned to a bit more specific topic class.

```
1  elif maps_directory == "MilitaryWorld":
2      if re.search(r"World News", short_title) or re.search(
3          r"Newsmap", short_title
4      ):
5          captions.append(f"news during world war 2")
6
7      elif re.search(r"Dated", short_title):
8          captions.append(f"world war 2")
9
10     else:
11         captions.append(f"world war 2")
```

3.2.3 Fine-tuning CLIP

In this subsection, the fine-tuning process of CLIP is presented in more detail. Since in total six models were fine-tuned, there also exist six separate Python scripts for training. Their structure is almost identical, this is why only the one used for the caption category *Map Type* (fineTuneCLIPMapType.py) will be explained in more detail:

The first step was to load the base CLIP model and set the device to cuda (see lines 1 to 4). After that, at lines 6 and 7, the ground-truth captions and corresponding map paths are loaded.

```
1  model = CLIPModel.from_pretrained("clip-vit-base-patch32")
2  processor=CLIPProcessor.from_pretrained("clip-vit-base-patch32")
3  device = "cuda" if torch.cuda.is_available() else "cpu"
4  model, preprocess = clip.load("ViT-B/32", device=device, jit=False)
5
6  classpic_captions = np.load("classPictorialCaptions.npy")
7  classpic_paths = np.load("classPictorialPaths.npy")
```

In the second step, due to the fact that the map dataset is unbalanced regarding map types (1'334 *Classical* maps vs. 3'183 pictorial maps), sub-sampling was necessary. Therefore, at lines 8 to 14, the two map types are separated and stored in separate lists. Then, with help of the *sample* function provided by Python's *random module* (see lines 17 and 19), exactly 1'334 pictorial maps are sampled to keep the two classes balanced.

```
8  captions_nopic=classpic_captions[classpic_captions!="pictorial map"]
9  indices_nopic = np.argwhere(classpic_captions != "pictorial map")
10 paths_nopic = classpic_paths[indices_no_pic]
11
12 captions_pic = classpic_captions[classpic_captions=="pictorial map"]
13 indices_pic = np.argwhere(classpic_captions == "pictorial map")
14 paths_pic = class_pic_paths[indices_pic]
```

```
15
16 random.seed(24)
17 subsampled_captions_pic = sample(list(captions_pic),k= 1334)
18 random.seed(24)
19 subsampled_paths_pic = sample(list(paths_pic), k=1334)
```

After balancing the classes, it was time to define the train, validation and test sets. In order to get optimal splits, scikit-learn's *StratifiedShuffleSplit* (see line 20) was used which preserves the percentage of samples for each class. The code snippet below shows how the test split was created and similarly, the validation split was defined. It was decided to use 90% of maps for training, 10% for validation and the rest for testing.

```
20 sss=StratifiedShuffleSplit(n_splits=1,test_size=0.1,random_state=42)
21
22 for train_index, test_index in sss.split(image_paths,captions):
23
24     train_captions, train_image_paths = [captions[i] for i in
25     train_index], [image_paths[i] for i in train_index]
26
27     test_captions, test_image_paths = [captions[i] for i in
28     test_index], [image_paths[i] for i in test_index]
```

Finally, after initializing the dataloader, fine-tuning of the base CLIP model could begin. For this, the training and validation loop was adopted by Jhon Parra (2022), as already mentioned in 3.1.3.

The result of the whole fine-tuning process were six separate models, one for each caption category:

- **Map Type:** best_model_MapType.pt
- **Area (Topographic):** best_model_27Countries.pt
- **Century:** best_model_Date.pt
- **Style:** best_model_Note.pt
- **Area (Pictorial):** best_model_Pictorial_Area.pt
- **Topic:** best_model_Pictorial_Topic_V2.pt

3.2.4 Accuracy assessment

This subsection explains how the accuracy of the fine-tuned CLIP models was evaluated. Here, the procedure is shown for the model *best_model_Date.pt* which was fine-tuned for the caption category *Century (Topographic)*.

The following code snippet shows how after first loading the base CLIP model (see line 1), the saved weights are loaded at line 7. To be able to conveniently

switch between both base and fine-tuned model, the varibale *useFineTuned* was introduced.

```
1  model, preprocess = clip.load("ViT-B/32", device=device)
2  useFineTuned = True
3
4  if useFineTuned:
5
6      model_path = "checkpoints/best_model_Date.pt"
7      model.load_state_dict(torch.load(model_path, map_location=device
       ))
8      model = model.eval()
9      model = model.to(device)
10     print("Using Fine-tuned model")
```

Line 11 and 12 define a list consisting of all classes belonging to the *Century* caption category (see Table 2). Then, starting from the for-loop at line 17, all test maps are iterated and corresponding ground-truth captions are through. First, each map is prepossessed and the classes tokenized (see lines 19 and 20). Later at line 27, the variable *probs* contains the softmax probabilities for each class based on the test input map. These probabilities represent the model's confidence in assigning the image to each class. Because we are interested in the class assigned to the largest probability, at line 29, this class is extracted from the *classes* list. If the obtained prediction corresponds to the ground-truth, then the *count* variable is incremented (see line 32 and 33). The overall accuracy is then calculated as seen at line 35.

```
11 classes = [
12 "19th century","18th century","17th century","16th century"]
13
14 count = 0
15 total = 0
16
17 for testmap, groundtruth in zip(test_image_paths, test_captions):
18
19     image = preprocess(Image.open(testmap)).unsqueeze(0).to(device)
20     text = clip.tokenize(classes).to(device)
21
22     with torch.no_grad():
23         image_features = model.encode_image(image)
24         text_features = model.encode_text(text)
25
26         logits_per_image, logits_per_text = model(image, text)
27         probs = logits_per_image.softmax(dim=-1).cpu().numpy()
28
29     prediction = classes[np.argmax(probs)]
30     total += 1
31
32     if prediction == groundtruth:
33         count += 1
34
35 accuracy = count / total
```

3.2.5 Caption inference and GUI

In this subsection the Python script *CaptionInferenceGUI.py* is explained in detail. This script carries out two tasks: construction of the GUI, and combination of the six fine-tuned CLIP models with a GPT-3.5 API prompt. The GUI's webpage is built by Gradio[3], which is an open-source Python package that helps to demo deep learning models. The following code snippet shows the block construction of the interface.

This Gradio interface has two tabs: "Demo" and "README". The "Demo" tab contains an image upload section where the user can upload or drag an image for analysis, four checkboxes for selecting map details, a submit button, and a text box for displaying the generated storytelling caption. Those four checkboxes correspond to the four questions of *What?*, *Where?*, *When?*, and *Why?*. When selected, the generated story will emphasize the chosen aspects, providing a detailed narrative. If none are selected, a basic version of the story will be generated, depicting essential content without extension. The click action of the submit button will then transfer the input of components to the interaction function *map_interface*. The other "README" tab includes a Markdown section that gives an overview of this demo, describing its features and capabilities, and providing instructions on usage, technical background and notes.

```
with gr.Blocks() as demo:
    with gr.Tab("Demo"):
        with gr.Row("Map Details"):
            with gr.Column("Map"):
                image_input = gr.Image(label="Upload or Drag Map
    Here", type='numpy')
                with gr.Row("Map Details"):
                    what = gr.Checkbox(label="What")
                    where = gr.Checkbox(label="Where")
                    when = gr.Checkbox(label="When")
                    why = gr.Checkbox(label="Why")

                submit_button = gr.Button("Submit")

            output_text = gr.Textbox(label="Caption")

        submit_button.click(
            fn=map_interface,
            inputs=[image_input, what, where, when, why],
            outputs=output_text
        )
    with gr.Tab("README"):
        gr.Markdown("""
            # Map Storytelling Tool
            ...
            """)
```

[3] https://www.gradio.app/

The code snippet below shows how one of the six fine-tuned models is loaded. First, the device is set to cuda and then the base CLIP model is initialized. Afterwards, the fine-tuned weights are loaded and all parameters in the model are frozen to keep the model unchanged during inference.

```
26 device = "cuda" if torch.cuda.is_available() else "cpu"
27 model, preprocess = clip.load("ViT-B/32", device=device)
28
29 model_maptype = copy.deepcopy(model)
30 def freeze_network(model):
31     for p in model.parameters():
32         p.requires_grad = False
33     return model
34
35 model_path_maptype = "Models_CLIP/best_model_MapType.pt"
36 model_maptype.load_state_dict(torch.load(model_path_maptype,
       map_location=device))
37 freeze_network(model_maptype)
```

In the following, the code snippet of the combined model class is explained in detail. The class takes six models as input. In the *forward* method, it processes the input through the models to predict keywords for each category. Given a map image, it first predicts the type of the map, i.e., topographic or pictorial, as the root node in the decision tree. Then, based on the type, it further predicts area, century, and style category keywords for topographic maps, and area and topic category keywords for pictorial maps. During prediction, the items of each category are tokenized by CLIP's tokenizer, and the model returns a list of possibilities for each item, as shown in lines 46 to 49.

```
38 class Combined_model(nn.Module):
39     def __init__(self, model_maptype, model_location, model_century,
        model_note, model_area, model_topic):
40         super(Combined_model, self).__init__()
41         self.model_maptype = model_maptype
42         self.model_location = model_location
43         ...
44     def forward(self, x):
45         maptypes = ["topographic map", "pictorial map"]
46         text = clip.tokenize(maptypes).to(device)
47         logits_per_image, logits_per_text = self.model_maptype(x,
       text)
48         probs = logits_per_image.softmax(dim=-1).cpu().numpy()
49         maptype = maptypes[np.argmax(probs)]
50
51         if maptype == "topographic map":
52             ...
53         elif maptype == "pictorial map":
54             areas = ["united states", "world"]
55             ...
56             topics = ['flight network', ...]
57             ...
58             return maptype, area, topic
```

The next code snippet presents the interaction function for the demo. This function begins with pre-processing map input as a NumPy array, and then the combined model is applied to predict keyword captions. Subsequently, based on the user's selection of the four aspects (*What?*, *Where?*, *When?* and *Why?*), they are embedded into the GPT-3.5-turbo API prompt and an output is generated within 60 tokens. Note that the API usage requires an OpenAI account key. After generation, the output will be displayed in the text box on the right side of the interface. Gradio also provides the *share* mode when launching the demo, which can generate a temporary link to the webpage GUI for sharing with the public.

```
59 def map_interface(map, what, where, when, why):
60     if type(map) == "str":
61         image = preprocess(Image.open(map)).unsqueeze(0).to(device)
62     else:
63         map = Image.fromarray(map)
64         image = preprocess(map).unsqueeze(0).to(device)
65 
66     results = []
67     combined_model = Combined_model(model_maptype, model_location,
       model_century, model_note, model_area, model_topic)
68     combined_model.eval()
69     with torch.no_grad():
70         results = combined_model(image)
71 
72     prompt = ""
73     if what:
74         prompt += f"What is this map about? "
75     if where:
76         prompt += f"Where is this map about? "
77     if when:
78         prompt += f"When is this map about? "
79     if why:
80         prompt += f"What can this map be used for? "
81 
82     response = client.chat.completions.create(
83             model="gpt-3.5-turbo",
84             messages=[
85                 {"role": "system",
86                  "content": "You are a helpful assistant"},
87                 {"role": "user",
88                  "content": f"Please create a concise sentence that
       encapsulates these keywords: {results}. Additionally, provide a
       brief explanation, in under 30 words, about: {prompt}."
89                  }
90             ],
91             max_tokens=60
92         )
93     results = response.choices[0].message.content
94     results = results.strip('"')
95 
96     return results
```

Chapter 4

Results

This chapter deals with the results of this project. Section 4.1 uses plots to visualize what the fine-tuning process looked like. Afterwards, in Section 4.2, the quality of the fine-tuned CLIP models is shown. The chapter concludes with Section 4.3, where a developed Graphical User Interface (GUI) for map storytelling is presented.

4.1 Fine-tuning Process

The following Figures 4 to 9 depict training and validation loss curves for each of the six fine-tuned CLIP models.

Figure 4: Map Type

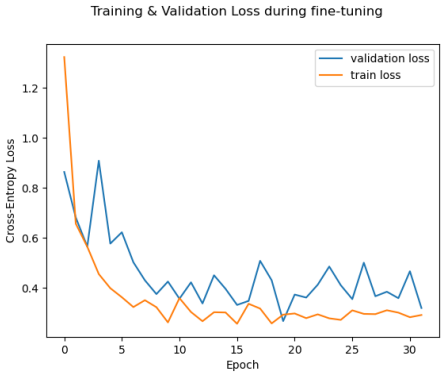

Figure 5: Area (Topographic map)

Figure 6: Style

Figure 7: Century

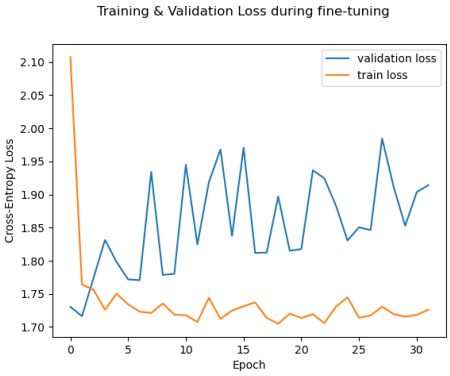

Figure 8: Area (Pictorial map)

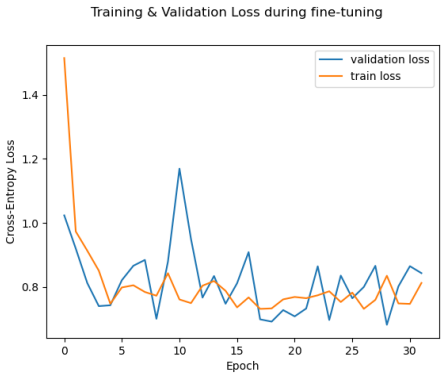

Figure 9: Topic

Upon observation, one can see that the training loss decreased quite rapidly for all models, although it is still worth noting the sudden spike in Figure 4. Interestingly, the training loss never reached values very close to zero. The lowest value of roughly 0.2 was reached in Figure 5. When taking a look at the validation loss curves, a variety of patterns can be observed. In Figures 4, 5 and 9 the curves can be described as decreasing or rather decreasing. On the other hand, the validation loss in Figures 7 and 8 is increasing, the latter quite strongly. A rather constant validation loss curve can be seen in Figure 6. Finally, it has to be mentioned that the validation loss was subject to greater fluctuations overall than the training loss.

4.2 Fine-tuned CLIP Models

Table 3 below compares the prediction accuracy achieved for each of the six caption categories with the base CLIP model and with the fine-tuned CLIP models. These numbers are based on 113 test maps (68 topographic maps and 45 pictorial maps).

Caption Category	Base CLIP	Fine-tuned CLIP
Map Type (What?)	0.43	**0.96**
Area (Where?)^T	0.28	**0.78**
Style (What?)^T	0.29	**0.75**
Century (When?)^T	0.40	**0.76**
Area (Where?)^P	**0.96**	0.93
Topic (What?)^P	0.47	**0.67**
Average Accuracy	0.47	**0.81**

Table 3: Comparison of prediction accuracy achieved (per caption category) with the base CLIP model and fine-tuned CLIP models. The superscript letters T and P stand for *Topographic* and *Pictorial* maps.

When studying the table, it becomes evident that the fine-tuned CLIP models significantly outperform the base CLIP model in five out of six caption categories. Only in the pictorial area category does the base model perform slightly better. The prediction accuracy was determined by giving each test map as an input image to the base CLIP model and to all supporting fine-tuned CLIP models. Then it was checked whether the generated keyword caption for each caption category matches the ground-truth caption. Note that because some of the test maps only contain incomplete or no metadata information at all (to derive ground-truth captions), in these cases, the accuracy assessment had to be performed by manually inspecting both the generated caption and the input map.

The two Figures 10[1] and 11[2] on the next page depict two of these test maps. Tables 4 and 5 show the corresponding captions generated by base CLIP and the fine-tuned CLIP models.

[1] https://www.davidrumsey.com/luna/servlet/detail/RUMSEY~8~1~260648~5522977:Air-France--Reseau-Aerien-Postal-?

[2] https://www.davidrumsey.com/luna/servlet/detail/RUMSEY~8~1~263393~5524234:142--Carte-Physique-et-Politique-de?

Figure 10: Orginal title: Air France. Reseau Aerien Postal.

Caption Category	Base CLIP	Fine-tuned CLIP
Map Type (What?)	**pictorial map**	**pictorial map**
Area (Where?)^P	**world**	**world**
Topic (What?)^P	world war 2	**flight network**

Table 4: Captions generated by base CLIP and fine-tuned CLIP models for each category. Bold captions match the ground-truth.

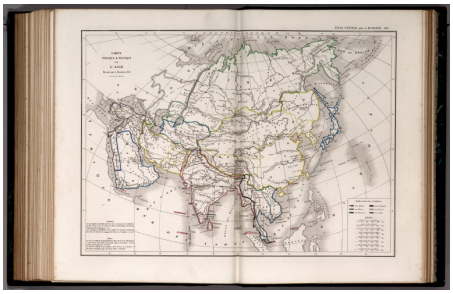

Figure 11: Orginal title: Carte Physique et Politique de L'Asie.

Caption Category	Base CLIP	Fine-tuned CLIP
Map Type (What?)	pictorial map	**topographic map**
Area (Where?)^T	eastern hemisphere	**asia**
Style (What?)^T	hand colored with decorative elemnts and pictorial relief	**hand colored**
Century (When?)^T	18th century	**19th century**

Table 5: Captions generated by base CLIP and fine-tuned CLIP models for each category. Bold captions match the ground-truth.

When taking a look at Table 4, one can observe that both models correctly predicted the map type and area. But base CLIP was unable to generate the correct topic caption, as the test map is showing an Air France flight network map.

Upon observing Table 5, it becomes clear that the fine-tuned models generated the correct captions for each caption category. Even though the base CLIP model was not too far away from the ground truth when inspecting the *Area* and *Century* categories, it was not able to distinguish the map types correctly, which is crucial.

For the sake of completeness, following Table 6 shows the resulting captions generated by the discarded model ClipCap.

Test Map	Base ClipCap
Figure 10	A map of the world is on display in a room.
Figure 11	A map of the world is shown on a table.

Table 6: Full captions generated by the discarded base ClipCap model for both of the example test maps.

4.3 Map Storytelling and GUI

Figure 12 shows the interface of the map storytelling tool demoed by Gradio[3], taking a map[4] of the United States as an example. This tool applies the decision tree structure introduced in subsection 3.1.4, utilizing the fine-tuned CLIP models to analyze uploaded map images and generate relevant descriptions as storytelling using OpenAI's GPT-3.5 model.

To use this GUI, one uploads a map image by clicking or dragging. This application will identify the map type first (i.e., topographic or pictorial). Then, based on that information, the app will recognize specific details (e.g., area, century, topic, and style). After extracting keyword captions, the GPT-3.5 model is employed through its API to generate a detailed textual description of the map based on the identified information. There are four check-boxes provided for users to select different details they wish to analyze, i.e., the four questions: *What?*, *Where?*, *When?*, and *Why?*. Upon clicking the "Submit" button and waiting for the system to process, a description of the map with emphasis on the chosen aspects is generated. Since the fine-tuned models are pre-loaded when launching the demo, the generation process is efficient and usually takes less than twenty seconds, often just several seconds. Table 7 shows the final captions generated by our approach.

[3] https://www.gradio.app/

[4] https://www.davidrumsey.com/luna/servlet/detail/RUMSEY~8~1~349878~90117266:America-the-Wonderland---A-Pictoria?

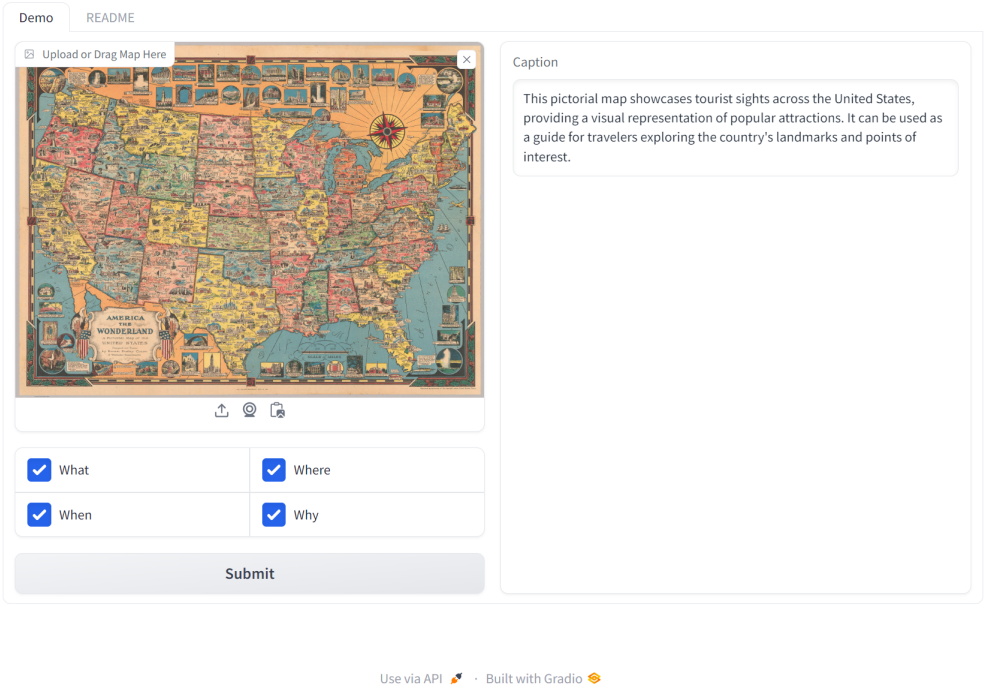

Figure 12: Demo layout

Test Map	Decision Tree Method
Figure 10	This pictorial map illustrates the global flight network, showcasing worldwide destinations and travel routes. It is a visual representation of the world, providing information about flight connections and can be used for planning and visualizing travel itineraries.
Figure 11	A hand-colored topographic map of 19th-century Asia, illustrating landforms and elevations, providing geographic information for various purposes.

Table 7: Full storytelling captions generated by the decision tree structured method for both of the example test maps.

Chapter 5

Discussion

In this chapter, the results of this project are interpreted and discussed under the given related work. Furthermore, possible consequences and limitations are pointed out.

5.1 Interpretation and discussion of results

5.1.1 Fine-tuning Process

The training and validation loss curves (see Figures 4 to 9) in Section 4.1 show that the behavior can look vastly different depending on the previous knowledge of the base CLIP model. It turns out that more epochs had a positive impact on fine-tuning the caption categories *Map Type* (Figure 4), *Topic* (Figure 9), and especially *Area (Topographic)* (Figure 5), as both the validation loss and training loss kept gradually decreasing. This means that the model is learning and improving its performance on the given task. In the other plots, the opposite is the case: after very few epochs (< 5), the validation loss almost plateaus or increases again, meaning that the model is already over-fitting on the training maps. An exact reason for the sudden spike in Figure 4 could not be found but it is assumed that the learning rate was set too high for these epochs and caused the model to overshoot the optimal parameters.

5.1.2 Fine-tuned CLIP Models

As already mentioned in Section 4.2, the fine-tuned CLIP models outperform the base CLIP model in five out of six categories. On average, fine-tuning increased CLIP's performance by 34 percentage points (i.e., roughly 72%) in these tasks. The reason why the base model performed slightly better in the *Area (Pictorial)* caption category is that, as seen in Table 2, only 290 maps were used for fine-tuning of that category. On the other hand, base CLIP has very likely seen illustrations showing the United States or the world several thousand times with way more variations and is therefore able to distinguish these two landmasses a tiny bit more accurately. This might also explain why in Figure 5 the model

starts to overfit after just one epoch.

Furthermore, it has to be said that while Table 3 shows that the fine-tuned models overall outperform base CLIP, the degree of it is actually higher. This has to do with following two points:

1. The base CLIP model generated for 109 out of all 113 test maps *pictorial map* as map type, even though only 45 are truly pictorial. Calculating precision and recall shows: $\text{Precision}^P = 41\%$ and $\text{Recall}^P = 100\%$, while $\text{Precision}^T = 100\%$ and $\text{Recall}^T = 6\%$. This confirms that the base CLIP model would perform much worse if the test set contained even more topographic maps, as it would have assigned most of them the label *pictorial map*. For comparison, the corresponding fine-tuned CLIP model achieved: $\text{Precision}^P = 92\%$, $\text{Recall}^P = 100\%$, $\text{Precision}^T = 100\%$ and $\text{Recall}^T = 94\%$.

2. Evaluating the prediction accuracy per caption category and not by combined caption (see decision tree in Figure 3) favors the base CLIP model. The reason is that, since as seen before, the CLIP model almost always says that a map is pictorial, the probability that the wrong sub-tree is chosen in the decision tree is rather high for topographic input maps. Since determining the map type is the root node of the tree and therefore the most crucial decision, incorrect predictions will lead to combined captions that make no sense. For the example test map seen in Figure 10, the actual generated caption will not be made out of the keywords seen in the left column in Table 5 but would instead be the ones shown in Table 8. Nevertheless, the approach to evaluate the prediction accuracy per caption category was chosen in order to not fully discredit the generative abilities of the base CLIP model.

Caption Category	Base CLIP
Map Type	pictorial map
Area (Pictorial map)	united states
Topic	transport routes

Table 8: Captions generated by the base CLIP model for each category following the logic of the decision tree. Note how these predicted keywords make no sense when compared with the input test map depicted in Figure 11.

Overall, apart from the *Map Type* category, the base CLIP model still performed better than random guessing. Especially in the categories *Area (Topographic)*, with 27 classes and *Topic (Pictorial)*, with 13 classes, it was already able to predict correct captions surprisingly well. This is why the jump from 47% to 67% in the latter might not look that impressive but considering the low amount

of maps (i.e., 284 maps) used for fine-tuning that category, it is still a significant improvement.

5.1.3 Decision Tree Structure

Through a decision tree structure, fine-tuned models are combined and the generative capabilities of GPT models are directly utilized. This integration provides a way for systematic processing. Presently, there are six CLIP models, each fine-tuned for a specific category. If additional aspects of the map are required to be explored, a new CLIP model can be fine-tuned and then integrated into the existing decision tree framework. This approach is not only efficient but also straightforward to implement. It offers several advantages: 1.) It allows for the manual selection of aspects crucial for comprehending a map; 2.) It utilizes models in a way where each CLIP model specializes in its respective category, thereby enhancing the overall accuracy and relevance of the captions generated; 3.) It facilitates the repeated utilization of the same data to learn distinct information separately, thereby providing a solution to the challenge of limited data availability in complicated deep learning tasks.

5.1.4 GPT API Usage

The quality of the story generated by LLMs depends to a large extent on how complete the information in the prompt is perceived and how well the generation is carried out. The former is reflected in the length, content, structure, language, etc. of the generated story, which are already listed in the prompt as generation criteria. The latter is reflected in the level of detail, topic relevance, fidelity to the facts, and perplexity, which will be reflected in the balance of the four questions and the syntax of the sentences, especially in the *Why?* part of the story. Unlike *What?*, *Where?*, and *When?*, there is no clearly identifiable keyword for *Why?* that represents the extension to the content. Therefore, this part can mainly be used to measure the performance of the model based on its fact fidelity and topic relevance. In practice, HuggingChat often generated irrelevant content about the map, providing expansive explanations without constraints, while GPT performed better. Therefore HuggingChat was discarded.

Even though ChatGPT's underlying model is GPT-3.5 or GPT-4, the responses from ChatGPT conversational interactions often differ from the same model's API responses. The reason is that ChatGPT is fine-tuned for conversational use and takes contextual information when processing. Typically, ChatGPT tends to yield more satisfactory answers to the given questions, even though it may additionally require several emphasized or repeated explanations sometimes. However, responses from the API frequently lack this level of satisfaction, particularly in maintaining the perplexity of the generated sentences at a low level. It is observed that no matter in which kind of way to use the GPT model, it does not always fully retrieve information in the prompt or in the right way.

Therefore, the generated answers are likely to disqualify all requirements at the same time, and it's often more effective to be clear and concise instead of overly long or complex. For queries with rigorous and multiple requirements, it listing answer criteria separately in the prompt rather than condensing them into several sentences performs more effectively. Besides, individually utilizing multiple API calls to answer different parts of the requirements can also be helpful in executing the prompt. Furthermore, using GPT itself to craft the API prompt can be highly beneficial as it aligns with the GPT model's inherent understanding capabilities.

Overall, to generate a compelling story with GPT models, the construction of an effective prompt is critical and should have these characteristics:

- Logically well-structured
- Concise and brief
- Clear requirements
- Appropriate length

Experiments reveal that both GPT-3.5-turbo and GPT-4 outperform another candidate model i.e., Text-davinci-003, on similar prompts. Compared to GPT-3.5-turbo, GPT-4 can understand the prompt better, but still has its limitations and does not achieve absolute perfection. Concerning the pricing of GPT models, it is based on the number of tokens for both input and output, and GPT-4 is approximately 30 times more expensive than GPT-3.5-turbo. Consequently, GPT-3.5 was selected for its good performance and cost-effectiveness, with a cost of less than 0.0002 USD for each request for storytelling in the demo.

5.2 Discussion of ClipCap

As the initial approach chosen to achieve the goals, ClipCap utilizes GPT-2 as part of its model to have a strong understanding of textual data. However, it failed to achieve expectations and was finally discarded even after a lot of effort. The reason why ClipCap failed has been briefly discussed in subsection 3.1.3. The most important reason is that the underlying image encoder, i.e., CLIP model, which was pre-trained on datasets of real-life situations and thus is not well capable of retrieving information from historical maps, stays frozen during training or fine-tuning. This could also explain why in the past, unlike CLIP, ClipCap has only been fine-tuned on subsets of the COCO dataset (Lin et al., 2014), e.g., Cafagna et al. (2023). Apart from this factor, a large, balanced, expansive, and well-structured dataset, especially with clearly structured and categorized ground truth captions, plays an important role in training the ClipCap. This is due to the high variability in captions, particularly in terms of topics and content descriptions of the map. When there is a vast diversity in

captions but insufficient data within corresponding categories, the model will not be able to establish a good connection between visual and textual information and therefore not perform well in caption generations.

In preliminary experiments, ClipCap managed to generate correct captions with an average character similarity of 80%. This might seem high, but in reality, 100% character similarity was rarely achieved (meaning that a generated caption and corresponding ground-truth are equal). One has to also keep in mind that similarity on the character level and semantic level are completely different. This is why for CLIP, the accuracy was determined by pure string equality, which is much more strict but also more clear and unambiguous. In addition, unlike the test map sets used for CLIP, the ones for ClipCap were not guaranteed to be balanced. This means that the average character similarity of 80% cannot truly be seen as a "fair" or deterministic value but rather inflated.

GPT models were employed to automatically create ground-truth captions for experiments. Even though GPT-4 is a powerful tool for understanding and generating sentences in natural language, it did not always follow all requirements in the prompt, for example, punctuation marks, abbreviations, and repeated location descriptions still occasionally appeared in the generated captions, which required additional post-processing work before being put into use. Moreover, it is also challenging to extract just keywords from map content without including any additional contextual descriptions meanwhile efficiently categorizing them into an appropriate number of categories. The balance between the diversity of the ground truth data and the model's generalization ability is difficult to maintain as well. Therefore, the CLIP model was subsequently adopted in the later stages of the project.

5.3 Consequences

Based on the results presented in Chapter 4, the following can be said:

- The developed decision tree method, which combines keyword captions generated by fine-tuned CLIP models, is capable of describing historical maps with respectable accuracy in a storytelling fashion.
- The fine-tuning process of CLIP is quite efficient and does not necessarily require a huge dataset consisting of tens of thousands of maps to improve the base CLIP model.
- The base CLIP model has much higher generalizability than ClipCap and already performs better than random guessing.

5.4 Limitations

The developed method, in its current state, can only be used for map storytelling if the input map's content can be described by the present caption category classes. Meaning that the input map should for example not depict an area that the fine-tuned models have never seen. If this is still the case, then the model will check which of the already-known areas looks closest to the one in the input map and output the area corresponding to the highest probability. Therefore, the main limitation here is the limited number of fine-tuned classes per caption category. This can be attributed to the available maps on David Rumsey Historical Map Collection (Rumsey, David, and Cartography Associates, 2024) and the laborious ground-truth caption process that made it challenging to create a balanced map dataset. Especially for the *Topic (Pictorial)* category it would have been desirable to have access to more easily classifiable pictorial maps. In addition, increasing the number of test maps would have made the accuracy assessment more robust, but it was already difficult to find more pictorial maps online corresponding to any of the topic classes. This is why it was decided not to just add even more topographic maps instead, to increase the test set, but to keep it at its current size at which the map types are more or less balanced.

Moreover, six distinct CLIP models are now used to capture the crucial aspects of map content. This method provides an efficient approach to extracting and processing diverse map features. However, a potential drawback is the risk of overlooking finer details and thus losing information in the map content, because the focus is primarily on key aspects identified in advance. Prerequisite knowledge about the map content is also needed for prediction as mentioned before.

Chapter 6

Conclusion and Outlook

In this project, the main objective was to explore and fine-tune GPT models for map storytelling, where given a historical input map, the model should generate a caption answering the four questions *Where?*, *What?*, *When?* and *Why?* (see section 1.3). The results presented and discussed in chapters 4 and 5 show that a method for map storytelling answering the above questions was successfully developed. Provided that the content of the historical input map can be described by the present caption category classes, the method is capable of describing it with respectable accuracy while significantly outperforming the related state-of-the-art captioning methods CLIP and ClipCap.

Potential future work could include finding ways to create a larger and more diverse (by also taking advantage of other map collections) map dataset with corresponding ground-truth captions more efficiently. Such a dataset, in combination with the decision tree approach used in this project, would allow the development of a way more powerful (historical) map captioning method, overcoming the current limitations. In addition, this dataset can also provide an opportunity to investigate the integration of various distinct models or to develop just one more comprehensive CLIP model. The enhanced model can then be employed to encode visual information from maps in ClipCap, facilitating the effective generation of more complex stories with richer content and more details in the future.

Bibliography

Cafagna, M., van Deemter, K., and Gatt, A. (2023). Hl dataset: Grounding high-level linguistic concepts in vision. Retrieved January 09, 2024, from https://github.com/michelecafagna26/CLIPCap?tab=readme-ov-file.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. (2021). Openclip. Retrieved January 02, 2024, from https://github.com/mlfoundations/open_clip.

Jhon Parra (2022). Fine-tuning open ai clip for image encoding. Retrieved January 02, 2024, from https://github.com/statscol/clip-fine-tuning.

Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Doll'a r, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. Retrieved January 02, 2024, from https://cocodataset.org/#home.

Mokady, R., Hertz, A., and Bermano, A. H. (2021). Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734*. Retrieved January 02, 2024, from https://github.com/rmokady/CLIP_prefix_caption.

OpenAI (2022). GPT-3.5. Retrieved January 02, 2024, from https://platform.openai.com/docs/models/gpt-3-5.

OpenAI (2023). GPT-4. Retrieved January 02, 2024, from https://platform.openai.com/docs/models/gpt-4-and-gpt-4-turbo.

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., Krueger, G., and Sutskever, I. (2021). Learning transferable visual models from natural language supervision.

Raimund Schnürer, René Sieber, J. S.-L. A. C. and Hurni, L. (2021). Detection of pictorial map objects with convolutional neural networks. *The Cartographic Journal*, 58(1):50–68.

Rumsey, David, and Cartography Associates (2024). David Rumsey Map Collection. Retrieved January 02, 2024, from https://www.davidrumsey.com.

SeleniumHQ (2023). Selenium webdriver. Retrieved January 02, 2024, from https://github.com/SeleniumHQ/selenium/tree/trunk/py.

Sharma, P., Ding, N., Goodman, S., and Soricut, R. (2018). Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of ACL*. Retrieved January 02, 2024, from https://github.com/google-research-datasets/conceptual-captions.

Soulter (2023). HuggingChat Python API. Retrieved January 02, 2024, from https://github.com/Soulter/hugging-chat-api.

Touya, G., Brisebard, F., Quinton, F., and Courtial, A. (2020). Inferring the scale and content of a map using deep learning. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLIII-B4-2020:17–24.

Yenduri, G., M, R., G, C. S., Y, S., Srivastava, G., Maddikunta, P. K. R., G, D. R., Jhaveri, R. H., B, P., Wang, W., Vasilakos, A. V., and Gadekallu, T. R. (2023). Generative pre-trained transformer: A comprehensive review on enabling technologies, potential applications, emerging challenges, and future directions.

Yingjie Hu, Zhipeng Gui, J. W. and Li, M. (2022). Enriching the metadata of map images: a deep learning approach with gis-based data augmentation. *International Journal of Geographical Information Science*, 36(4):799–821.

Zhou, X., Li, W., Arundel, S. T., and Liu, J. (2018). Deep convolutional neural networks for map-type classification.

Appendix A

Example Metadata

Author W. & A.K. Johnston Limited
Date 1839
Short Title Orbis veteribus notus.
Publisher W. & A.K. Johnston
Publisher Location Edinburgh
Type Atlas Map
Obj Height cm 21
Obj Width cm 29
Scale 1 75,000,000
Note Map of the eastern hemisphere, according to ancient geographical knowledge (from a western perspective). Title, in Latin; translates to: [The world known to the ancients]. Shows continental boundaries, cities, routes, topography, deserts, drainage, coastlines and routes of exploration, as well as "Terra incognita, according to Ptolemey". Relief shown pictorially. Includes latitudinal and longitudinal lines, as well as three bar scales and an elaborate compass rose. "Longit ab Ins: Fortun. versus Orientum." Map is 21 x 29 cm, on sheet 27 x 35 cm. Hand-colored engraving. In first section of atlas, Ancient maps.
World Area World
World Area Eastern Hemisphere
Subject Historical
Subject Classical
Full Title Orbis veteribus notus. Edinburgh W. & A.K. Johnston ; Glasgow; Robert Weir ; Lumsden & Son.
List No 14296.010
Page No 1
Series No 10
Engraver or Printer W. & A.K. Johnston
Publication Author W. & A.K. Johnston Limited
Pub Date 1839
Pub TitleDedicated to Her Most Gracious Majesty The Queen. The Edinburgh cabinet atlas comprising maps illustrating the modern geography of every country of the world and the most interesting portions of ancient geography; constructed from the latest & most authentic sources. Edinburgh, W. & A.K. Johnston Engravers & Printers to The Queen. Glasgow Robert Weir; Lumsden & Son: London, Whittaker & Co. Dublin, John Cumming: Paris J.P. Aillaud. 1839.
Pub Note Edinburgh cabinet atlas by W. & A.K. Johnston Limited, 1839. World atlas containing forty-five maps, as well as two comparative charts - one comparing mountain heights around the globe, and the other river lengths. Bound in black board with brown leather spine and corners. Front cover and spine both have gilded title. Collation: [1] leaf, [1-3], 4-9, [10-11], 12-13 pages, 1-45, [46-47] leaves. Includes an engraved title page, table of contents, a historical text and an index listing place names with coordinate points. Maps appear in two sections: Ancient maps and Modern maps, respectively. Geographic coverage spans Europe, the Middle East, Asia, Oceania, Africa, the Americas and Arctic regions, featuring the exploration routes of Arctic explorers. Maps show political boundaries, cities, routes, topography, deserts, drainage, coastlines and islands.
Pub List No 14296.000
Pub Type World Atlas
Pub Maps 47
Pub Height cm 36
Pub Width cm 28
Image No 14296010.jp2
Download 1 Full Image Download in JP2 Format
Download 2 GeoViewer for JP2 and SID files
Authors W. & A.K. Johnston Limited

Figure 13: Example metadata from David Rumsey Map Collection

Appendix B

Caption Classes

Caption Category	Classes (Number of maps per class)
Map Type	pictorial map (3'183), topographic map (1'334)
Area (Topographic)	europe (62), greece (56), eastern hemisphere (54), part of italy (54), italy (51), part of germany (41), france (40), middle east (36), iberian peninsula (34), part of france (34), asia minor (29), germany (27), part of greece (27), british isles (25), world (21), egypt (19), india (18), asia (17), holy land (16), caucasus (12), africa (11), netherlands (9), switzerland (7), americas (6), south america (6), scandinavia (6), sri lanka (5)
Style	hand colored (469), hand colored with decorative elements and pictorial relief (224), pictorial relief (189), hand colored with pictorial relief (125), engraved (86), decorative elements and pictorial relief (39)
Century	19th century (711), 16th century (221), 17th century (216), 18th century (186)
Area (Pictorial)	world (210), united states (80)
Topic	flight network (81), news during world war 2 (76), world war 2 (28), transport routes (18), tourist sights (14), playing card (12), animals (11), satirical representation (11), people (9), stamps (7), educational drawings (6), food and agriculture (6), military (5)

Table 9: Overview of all classes per caption category. In brackets, the number of maps per class.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Automatic Map Storytelling with Generative Pre-trained Transformer (GPT) Models

Authored by (in block letters):

For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Liu	Ziyi
Affolter	Claudio

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
Zurich, 12.01.2024	刘子怡 Affolter

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.