Computing the Greeting Network in the Bullinger Collection

Yining Wang

1 Overview

The Bullinger Collection text analysis project analyzes historical letters stored in XML format, focusing on extracting, processing, and analyzing greeting information along with linguistic data across six folders. This project includes several stages of data manipulation, textual analysis, and predictive modeling to uncover insights into the greeting patterns within the collection.

Initially, the project targets the computational extraction of greeting information from the XML files. By parsing the XML structure, the script identifies and extracts sender and addressee information, translating the contents from Latin to English using OpenAI's GPT-4 model.

Subsequently, the processed data were further refined. Specific scripts were developed to clean and standardize the metadata associated with senders and addressees, ensuring data consistency and enhancing the utility of the dataset for analysis purposes. The cleaned data were then used to replace generic pronouns and references in the text with the actual names of the individuals involved, based on the refined sender and addressees information.

The project also introduces a machine learning component where predictive models are trained to identify the likely senders and receivers of each greeting based on the textual content. These models are developed using a deep learning approach, employing the T5 transformer architecture to learn from the annotated data.

In addition to textual analysis, the project assesses the linguistic composition of the collection. A section is implemented to count and analyze the frequency of various languages used in the letters, highlighting the predominance of Latin and German. This linguistic analysis provides significant insights into the cultural and historical context of the correspondence.

Finally, the project culminates in the quantification of identified names and keywords, focusing exclusively on those names associated with identifiers. This quantification helps to map the network of correspondences more clearly, identifying key figures and their interactions throughout the collection.

2 Work Steps

2.1 Data Processing and Translation of the Bullinger Collection

In this section, I outline the methodology used to extract and translate greeting information from the Bullinger Collection, which is stored in XML format. This process systematically

converts raw XML data into structured, translated text outputs for further analysis. Specifically, I focus on processing one selected folder from the collection (folders 10, 10A, 11, 12, 13, 14). This method generates a CSV file for each folder, consolidating the output of the entire correspondence for that folder. The objective is to extract the last ten sentences from each letter after data cleaning and then translate them using GPT-4. The process can be categorized into the following stages.

The process starts by importing basic Python libraries. These libraries were chosen for their specific functionality: ‘openai‘ provides an interface to machine translation, ’re‘ allows string manipulation via regular expressions, ’os‘ enables interaction with system file directories, ’xml.etree.ElementTree‘helps in parsing the XML file structure, while ’csv’ is essential for creating and manipulating CSV files that will form the final output of the process.

Next the script configures the OpenAI API key. This key is a critical part of authentication that grants the script access to OpenAI’s GPT-4 model, which was later used to translate the Latin text and German in the letters into English.

After authentication, the script specifies the directory path. It indicates the location of the XML files and creates a directory where the output CSV files will be stored. If the output directory does not exist, the script is programmed to create it, ensuring a location for the processed files. The core of the script is the ’process file’ function. This function parses each XML file and extracts the sender and addressee information. The focus is then on identifying and organizing the last ten sentences of the letter’s content. These sentences are cleaned up by removing unnecessary XML formatting, leaving only the basic text. If the script encounters certain predefined terms in the text, they are encoded to instruct the GPT-4 model to apply specific translations to those terms, ensuring historical and contextual accuracy.

Once the sentences were prepared, they were sent for translation only if the script confirmed the presence of at least five sentences to work with. If this condition was met, the GPT-4 model was called upon to translate the text from Latin to English, integrating the original context into the translated output. The final step in the function was to compile the file name, sender and addressee information, the cleaned original sentences, and their English translations into a returnable format.

The end result of this process was the writing of the processed information to a CSV file, with one CSV file designated for each folder of letters processed. As the script looped through the list of XML files, it processed each one in turn, outputting the information to the CSV file complete with headers that categorized the data into ’Filename,’ ’Sender Info,’ ’Addressee Info,’ ’Original Text,’ and ’GPT-4 Translation.’

Throughout the process, the script is able to handle errors, log any problems encountered and ensure that they do not interrupt the entire operation. In addition to error handling, the script also provides updates after each file has been successfully processed so that its progress can be tracked in real time.

2.2 Data Refinement of Translated Greetings in the Bullinger Collection

In this phase, the previously translated text stored in CSV format is further refined to extract and clean up specific greeting-related content. The first step is to find the keywords associated with the greeting in the translation column by using a regular expression pattern. Keywords such as ’regards,’ ’greet,’ and ’salute,’ along with their variants, are used to filter sentences containing greeting information. These extracted sentences are then appended to the existing CSV in a new column and saved as an intermediary file.

The second step is to take the intermediary CSV file as its input and split the content of the 'Extracted Sentences' column into individual sentences, removing XML tags that annotate sentence numbers. The cleaned sentences are written to a new CSV file, excluding the original text and GPT-4 translation columns from the previous dataset, thus creating a more focused dataset on the extracted greetings.

The final stage utilizes the pandas library to read the cleaned CSV file and apply a transformation to the 'Extracted Sentences' column. This transformation involves a regex function that restructures references to individuals by appending an identifier number to their names, formatted as Name_ID. The modified DataFrame is then exported to a new CSV file.

The outcome of this part is a clean dataset that include sentences containing greetings from the Bullinger Collection, removing extraneous XML tags, and structured for clarity with consistent naming conventions for individuals referenced within the text.

2.3 Model Development for Sender and Receiver Prediction

In the next stage of analysis, the focus is to develop predictive models for identifying the senders and receivers of greetings within the Bullinger Collection. This process involves combining manually annotated greetings from two separate folders to form a robust training dataset. The folders chosen for this purpose are '10' and '13', yielding a set of 162 annotated sentences.

The first step involves using the refined data from the previous steps and merging the output CSV files from folders '10' and '13'. This merged dataset serves as the training set for the predictive models. Next, the same refinement process applied to folders '10' and '13' is extended to the remaining folders in the collection, preparing them for subsequent analysis. The resulting CSV files from each folder are then consolidated into a single dataset.

With the training dataset prepared, a predictive model for senders is developed using a deep learning approach with the T5 transformer model. The model is trained on the annotated sentences, learning to predict the sender of each greeting. The receiver model follows a similar logic and architecture, being trained separately on the same dataset to predict the receivers.

The sender model's development begins with loading the annotated data and initializing the tokenizer associated with the T5 model. The data is then partitioned using K-Fold cross-validation to enhance the robustness of the model through multiple training iterations. For each fold, a DataLoader is created to batch the data, and the model is trained over several epochs using the AdamW optimizer and a learning rate scheduler.

After training, the best-performing model is saved and then loaded into an evaluation state to ensure its readiness for making predictions. The model's performance is then assessed on a separate test dataset that has not been seen during the training phase. The predictions made by the model are appended to the test dataset, providing an initial prediction of the senders for each greeting.

A similar process is then undertaken with the receiver model, using the same deep learning framework to train a predictive model for identifying the receivers in the collection's greetings.

The final phase involves a manual review of the model-generated predictions to ensure accuracy. Any discrepancies found between the predictions and the actual senders or receivers are corrected. This step is crucial to maintaining the integrity of the dataset and improving the model's performance.

Once the manual verification is complete, all the data, including the original greetings, sender and receiver annotations, and model predictions, are compiled into a comprehensive CSV file. This file serves as the final output for current stage.

2.4 Cleaning of Sender and Addressee Information

In the current phase, the script targets the refinement of 'Sender Info' and 'Addressee Info' columns within the CSV file, aiming to normalize and structure the metadata associated with each greeting. The process employs a Python script that uses the pandas library for data manipulation, along with regular expressions for string pattern matching.

The script loads the dataset from a CSV file and defines a function 'process_addressee_info' which is intended to parse and reformat the addressee information. This function is applied to the 'Addressee Info' column. The function checks if the data is a string and then uses regular expressions to identify the addressee's ID, first name, and last name within the provided XML tags.

If all components of the addressee information are successfully found, the script constructs a new string that concatenates the first name and last name, followed by an underscore and the ID number. If the function fails to locate any of these parts, it returns the original content without alteration.

After processing, the refined sender and addressee information is saved back into the same CSV file, replacing the original columns with the cleaned and reformatted data.

2.5 Integration of Sender and Receiver Metadata in Predicted Text

Then, it focuses on refining the text within the 'predicted_sender' and 'predicted_receiver' columns of a CSV file, by substitively integrating the detailed sender and receiver information from their respective columns. This task is executed through a combination of Python's pandas library for data manipulation and regular expressions for pattern matching.

The process starts by loading the CSV file into a pandas DataFrame. This structure facilitates row-wise operations and data transformations. The script defines a set of keywords associated with sender and receiver references, such as 'I', 'me', 'my' for sender keywords, and 'you', 'your' for receiver keywords.

A function, 'replace_keywords', is crafted to replace occurrences of these keywords within the text with the actual names from the 'Sender Info' and 'Addressee Info' columns. This function constructs a regex pattern from the list of keywords and uses this pattern to identify and replace the keywords with the respective sender or receiver name. The replacement is done in a case-insensitive manner to ensure all variations of the keywords are captured and replaced appropriately.

The function is then applied to the 'predicted_sender' and 'predicted_receiver' columns. For the 'predicted_sender' column, sender keywords are replaced with the sender's name, and receiver keywords are replaced with the receiver's name. Similarly, the 'predicted_receiver' column undergoes a transformation where receiver keywords are replaced by the receiver's name, and sender keywords by the sender's name. This dual application ensures that each predicted text field accurately reflects the personal references as per the data in the sender and receiver information columns. Finally, the script saves the transformed data back into a new CSV file.

2.6 Quantifying Named Entities and Keywords in the Predicted Text

The final step in analyzing the Bullinger Collection involves quantifying the occurrences of specific named entities with identifiers and frequently used keywords in the predicted greeting sender and receiver texts. This analysis aims to provide quantitative insights into the commonality of certain names and terms within the dataset. Not all names in the data provided have identifier numbers, but for convenience, only names with identifiers are counted in this step.

First, the script loads the previously modified data from a CSV file into a pandas DataFrame. It then defines a function to extract and count occurrences of names paired with identifiers from both 'predicted_sender' and 'predicted_receiver' columns. This function uses a regular expression pattern to detect sequences where a name and an identifier are concatenated (e.g., 'Name Surname_ID'). Each match is captured and categorized by identifiers, allowing for the aggregation of unique name occurrences associated with each identifier. The results are grouped by identifier, and each unique name under the same identifier is compiled into a newline-separated list.

Simultaneously, another part of the script focuses on counting specific relational and familial keywords such as 'brother,' 'mother,' and 'family.' This is achieved by another function that assesses the frequency of each keyword within the text, considering variations in capitalization and partial matches.

Both sets of results—the identifier-linked names and the keyword frequencies—are then saved into separate CSV files. These files contain comprehensive counts and details of the named entities with identifiers and the frequency of specific keywords across the dataset.

2.7 Language Frequency Analysis Across the Bullinger Collection Folders

This part is an analysis to quantify the frequency of languages used in XML files across all 13 folders of the Bullinger Collection. The analysis identifies the count and percentage of each language's appearance within the text elements labeled with a language attribute.

The count_languages_in_all_folders function iterates through each directory and subdirectory in the specified root path, checking only XML files. Each file is parsed to extract and check for the presence of the 'lang' attribute in any sentence ('s') elements. Language types are recorded and occurrences counted to build a comprehensive dictionary reflecting the frequency of each language across the entire dataset.

During the XML file parsing, if a file is found to be corrupted or improperly formatted, the script will note an error, ensuring robustness in data processing. After collecting the language data, the total occurrences of all languages are aggregated to calculate the proportional representation of each language in the dataset.

The results show that Latin ('la') is the predominant language, comprising 77.4664% of the language attributes found, followed by German ('de') with 22.4004%, Greek ('el') at 0.1315%, and Hebrew ('he') with a minimal presence of 0.0017%. These percentages provide a clear quantitative breakdown of the linguistic distribution in the Bullinger letters, indicating a strong prevalence of Latin and German in the communications.

3 Proposed Further Steps

Building upon the initial phases of the Bullinger Collection Text Analysis Project, several further enhancements can significantly deepen the analysis and expand the project's scope.

3.1 Inclusion of Additional Data and Annotation

To enhance the robustness of the predictive models and broaden the analysis, it is proposed to include additional XML files from other collections. Expanding the dataset will provide a richer basis for training and improve the model's ability to generalize across different contexts. Additionally, increasing the quantity and quality of manual annotations will cover more greetings within the letters, enriching the dataset with detailed tags on sender and receiver information.

3.2 Advanced Post-Processing Techniques

The implementation of named entity recognition technology will improve the extraction of names and potential relationships from text. For example, identifying a reference like 'Heinrich Bullinger's wife' requires the model to identify 'Heinrich Bullinger' as a person and 'wife' as a relational entity related to him. Extending this entity extraction to identify and classify the types of relationships (family, professional, social) mentioned in the text allows for a deeper analysis of the social networks in the collection.

3.3 Visualization of Greeting Relations

Developing visualizations to map greeting relationships between individuals in a collection is critical. Implementing interactive visualization tools that allow users to explore various aspects of the network (e.g., filtering by collection, greeting type, or relationship) provides dynamic insight into the data using platforms such as Gephi or the web-based D3.js visualization.