SPATIAL TRACING OF HISTORIC EXPEDITIONS: FROM TEXT TO TRAJECTORY

MAFKERESEB KASSAHUN BEKELE
March, 2014

SUPERVISORS:

Dr.Ir. R.A. de By
B.J. Köbben, M.Sc.

SPATIAL TRACING OF HISTORIC EXPEDITIONS: FROM TEXT TO TRAJECTORY

MAFKERESEB KASSAHUN BEKELE
Enschede, The Netherlands, March, 2014

Thesis submitted to the Faculty of Geo-information Science and Earth Observation of the University of Twente in partial fulfilment of the requirements for the degree of Master of Science in Geo-information Science and Earth Observation.
Specialization: GFM

SUPERVISORS:

Dr.Ir. R.A. de By
B.J. Köbben, M.Sc.

THESIS ASSESSMENT BOARD:

Dr. A.A. Voinov (chair)
Dr. Ir. M. van Keulen

This thesis is dedicated to my beloved father Kassahun Bekele Yimam.

ABSTRACT

Historic expeditions are events that are flavored by, often exploratory, scientific, military or geographic characteristics. Such events are documented in the literature, journey notes or personal diaries. A typical historic expedition involves multiple site visits and the description of these visits contains a spatial, temporal and attributive context. An expedition involves movements in a space and such movements are represented as a trajectory that consists a list of triplets (*location*, *time* and *expeditioner*). Tracing historic expeditions spatially requires recognizing and extracting the contextual information from the description of the site visits. This creates a possibility for their representation in a spatial data model and provides alternative ways of analysis to textual representations. To this end, we developed a Trajectory Extraction and Storage System (TESS) to: recognize and extract the contextual information, recognize the spatiotemporal relationships between features, extract and store the triplets of location visits, produce an expedition route (trajectory) from the stored triplets and conduct a reliability assessment on the extracted triplets. The system has six components, namely, Raw Data Extraction , Contextual Information Extraction, Trajectory Triplet Extraction, Temporal Inference, Expedition Route Production and Quantitative Spatiotemporal Reasoning. The Brazilian Ornithological Gazetteer was used as a source for the location descriptions that were used to experiment the built system. We produced expedition routes using this system for multiple expeditioners whose site visits were described in the Brazilian Ornithological Gazetteer. A reliability assessment was conducted on the produced expedition routes and the extracted triplets using the Quantitative Spatiotemporal Reasoning tool and visual inspections. Some triplets were identified as unreliable and outliers by the system. Multiple scenarios were suggested for the unreliably on the basis of manual investigations that were conducted on the outlier triplets.

Keywords

spatial tracing, contextual information extraction, trajectory triplet extraction, temporal inference, quantitative spatiotemporal reasoning

ACKNOWLEDGEMENTS

First and foremost, I must acknowledge and thank the almighty God for His provision, protection, guidance and comfort throughout my life. I could never have accomplished this without the mercy, grace and help of the heavenly father.

I would like to express my special appreciation and thanks to my first supervisor Dr.Ir. R.A. de By for his useful guidance, comments, motivation and encouragement throughout the thesis work. Furthermore I would like to thank my second supervisor B.J. Köbben for his support and guidance.

I would like to thank the Royal Netherlands Government for funding this study through the Netherlands fellowship program. I will never and ever forget the input and knowledge I have gotten from the staff members of GIP and EOS department at ITC university of Twente. You all have put a finger print and it is sincerely appreciated.

Ahun demo misiganawu be amarigna yiketilal. Kassiye (dad) min endemilih alakim, yihe zare yehonewu hulu bante newu eshi. Betamm yigerimihal sidist mi'eraf silante bitsif des yilegn neber (esunim kebeka newu), gin beka ayichalim.....

ahun demo le guadegnoche (Asme (tagay), Edu, Fiker, Fre, Lozi, Mahi, Maste, Micky, Mule (eke), Sari, Tg (kecho), Zola (yegna), (Jovani ena Zola (keshit neger nachu eshi))....hulachunim ewedachihualehu eshi. Last but not least, I thank all of my class mates.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

JDBC Java Database Connectivity

JAPE Java Annotation Pattern Engine

TESS Trajectory Extraction and Storage System

KML Keyhole Markup Language

XML Extensible Markup Language

RDE Raw Data Extraction

CIE Contextual Information Extraction

TTE Trajectory Triplet Extraction

ANNIE A Nearly-New Information Extraction System

LHS Left-Hand Side

RHS Right-Hnad Side

HTML HyperText Markup Language

GATE General Architecture for Text Engineering

POS Part of Speech

NER Named Entity Recognition

QSTR Quantitative Spatiotemporal Reasoning

SAX Simple API for XML

GIS Geographic Information Systems

IR Information Retrieval

API Application Programming Interface

SREP Spatiotemporal Relationship Extraction Pipeline

EEP Entity Extraction Pipeline

Chapter 1
Introduction

1.1 MOTIVATION AND PROBLEM STATEMENT

Spatial tracing can be defined as the discovery of events that have taken place in a specified time frame and space. An *expedition* is an excursion, journey, or voyage made for a specific purpose. Historic expeditions are flavored by, often exploratory, scientific, military or geographic characteristics. These expeditions and the space and time of their occurrence may have been documented in the literature, journey notes, or personal diaries. Therefore, the documents in which historic expeditions are recorded might have a contextual information that can describe and answer the spatial, temporal and thematic question about historic expeditions. The contexts that are comprised in such documents could be categorized as *spatial*, *temporal* and *attributive*.

Nowadays, though well-developed data models prevail for the abstraction and representation of reality, approximately 80% of all information is stored in textual documents [12]. On the other hand, as the amount of unstructured data increases, the demand of a contextual information extraction method also increases [22].

The expeditions that were undertaken in the past are likely documented in a form of unstructured data. Furthermore, past expeditions could have possibly left their marks in history. However, reading these textual documents is not adequate to study and fully understand these expeditions and the marks they left; it requires extracting the contextual information out of the textual documents. This creates a possibility for their representation in a spatial data model. Such representations of expeditions provide alternative ways of analysis to textual representations. The potential spatial computations will be easier if objects are represented in one of the recognized spatial data models.

An expedition is dynamic, because it involves a movement in space. The movement of a dynamic object (the expeditioner in our case) is represented by a trajectory that consists a list of triplets, each of which consists of the *expeditioner name*, the *location of the expeditioner* and the *time-stamp* [2]. The trajectory of any expedition may be the list *((P_{t1} E_{t1} t1) (P_{t2} E_{t2} t2).... (P_{ti} E_{ti} ti)....)*, where E_{ti} and P_{ti} denotes the expeditioner and position of the expeditioner, respectively, at the time *ti* [2]. Therefore, the spatial tracing of a historic expedition needs to extract the *expeditioner*, *location* and *time-stamp* from its textual representation and transform it into a trajectory.

For instance, the dataset that we use for this research project, namely the *Brazilian Ornithological Gazetteer* identifies approximately 6,000 sites at which birds have been located or observed [16, 17]. These ornithological site visit descriptions are recorded in textual form (see Appendix B.1). Reading such a textual representation of site visits is not sufficient to visualize them as a trajectory because the text contains entities that could be identified as *person name*, *place name*, *institute name*, *coordinates*, *spatial relationship* and *date* (see Figure 1.1). To aggregate and transform the textual representation of location descriptions into a trajectory or expedition route, those entities contained in the description need to be recognized and extracted.

The transformation of a narration of a visit description, which is in textual form, into trajectory is needed to study and understand the spatial and temporal dimension of those visits. How-

ever, identifying and disambiguating the entities in a text phrase is doubtful due to the fact that the identification process is easily affected by the caprices of language [1]. Furthermore, recognizing and extracting the entities from a textual representation is challenging because the textual documents could have *endonyms* or *exonyms*; the attributive entities to be extracted (i.e., *entity names*, *place names*) might have been changed through time; and the text may have a phrased spatial and temporal relationship, meaning one spatial or temporal entity is defined relative to another. Moreover, understanding the *spatiotemporal relationships* between features requires a spatiotemporal inference.

A location description may have a spatial and temporal vagueness. The spatial entity might be represented with a vague phrase such as *"few miles from place X"* or location indicators might be missing, such phrase leads to an ambiguous spatial entity extraction and computing a relative location will be in doubt. The prevalence of temporal vagueness causes ambiguity when extracting an implicit temporal entity such as duration, for instance, a time-stamp *"Feb 1978"* as Figure 1.1 shows is vague, because the *start* and *end* dates are not explicit, such vagueness lead to a doubtful conclusion about the duration of an event (may be the event took number of days, the whole month or only a single day).

The main computational challenge when extracting a spatial entity is to understand the spatiotemporal relationships between features. Recognizing and extracting a crisp temporal and attributive entity is relatively easy, nevertheless, a successful transformation of a textual representation into trajectory needs to consider the spatial and temporal relationship between features. Figure 1.1 shows a textual representation of a *location visit*[1] and the entities it contains. To transform this text into trajectory, extracting *expeditioner name*, *location* and *date* is enough, but the *date*, in this very Figure, is described with a vague time-stamp (the temporal boundary is not explicit), computing the missing temporal boundary requires a temporal inference and reasoning.

Geoparsing is a process of identifying geographic context from textual documents [11]. To compute a relative location of any spatial object, first the geographic/spatial coordinates of the reference object needs to be extracted; geoparsing can be used to extract the coordinate of a reference object. Figure 1.1 shows a *location* which is positioned relative to another spatial object (i.e., *"20 km off northern end of Ilha de Santa Catarina"*). The geographic/spatial coordinates of the reference object, in this case *"Ilha de Santa Catarina"*, can be extracted either from the text (provided coordinates or description) or other external sources such as *GeoNames*[2]. However, the coordinate of this location is *"20 km off northern end of"* the reference object. Computing such referred coordinate needs to understand the spatial relationship between spatial object, especially for those cases where coordinates are in doubt or even missing.

Hence, in this research project, we presented a *Trajectory Extraction and Storage System (TESS)* to: extract contextual information from a gazetteer, recognize spatial and temporal relationships between features, infer relative temporal boundaries (in case of vague time-stamps), infer spatial relations, extract triplets of location visits (expeditioner name, location and time), store the extracted information in a database, produce an expedition route from the stored triplets and conduct a reliability assessment on the produced expedition routes.

1.2 RESEARCH IDENTIFICATION

The utilization of contextual information out of a gazetteer of historic ornithological site visits is mandatory to trace such events spatially. Therefore, our main aim throughout this research project is to develop an information extraction and storage system that can: recognize and extract

[1] Location visits are parts of an expedition. A single expedition typically has more than one location visit.
[2] http://www.geonames.org/

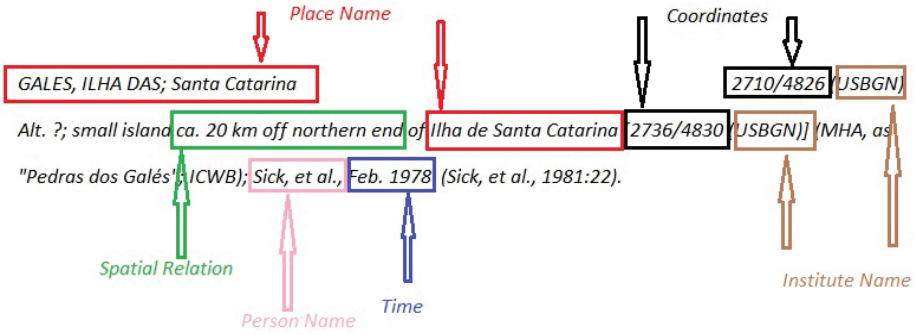


Figure 1.1: Entities to be extracted

contextual information; understand spatial relationships between features; understand temporal relationships between (possibly vague) time instants and intervals; infer relative temporal boundaries; and extract and store the extracted information. The dataset that is used throughout this research project is the Brazilian ornithological gazetteer [16, 17].

1.2.1 Research objectives

Objective: recognizing and extracting the *spatial*, *attributive* and *temporal* context of the location descriptions from a gazetteer of Ornithology.

- **Specific objectives**
  1. Recognizing and extracting the contextual information from a gazetteer of Ornithology.
  2. Developing pattern based entity recognition pipeline to recognize and annotate entities and the spatiotemporal relationships.
  3. Extracting triplets of an expedition trajectory (*expeditioner name*, *location*, *time*) and storing the extracted information in a database.
  4. Developing a temporal inference tool to infer relative temporal boundaries.
  5. Developing a data model to store the extracted contextual information.
  6. Producing an expedition route form the extracted triplets.
  7. Visualizing the produced expedition routes.
  8. Developing a quantitative spatiotemporal reasoning tool to assess the reliability of the produced expedition routes.
  9. Conducting a visual inspection-based reliability assessment on the produced expedition routes.

1.2.2 Research questions

1. What are the possible contexts in an expedition corpus?
2. What type of method can be used to recognize and extract contextual information?
3. What are the requirements to build an entity extraction pipeline?
4. What type of method can be used to recognize spatiotemporal relationships?

5. What should be the characteristics of a spatial and temporal inference techniques?
6. What should be the characteristics of a quantitative spatiotemporal reasoning techniques?
7. What are the computational requirements to produce an expedition route from extracted triplets?
8. What are the requirements to assess the reliability of a produced expedition route with a visual inspection-based technique?

1.2.3 Innovation aimed at

This research project comprises multiple procedural phases. The phases are: Raw Data Extraction (RDE), CIE, Trajectory Triplet Extraction, Trajectory Storage, Temporal Inference, Expedition Route Production and QSTR. These components are integrated each other. An output from one phase is used as an input for another phase, for instance, after extracting the raw data using the RDE, the CIE follows to annotate the entities of a location description. The Trajectory Triplet Extraction phase follows the CIE to extract and store the triplets of a given location visit. The Temporal Inference will be used infer relative boundaries when a vague temporal mark is encountered. The QSTR part of the system follows the Expedition Route Production phase and it is used to assess the reliability of a produced expedition route.

1.3 PROJECT SET-UP

1.3.1 Method adopted and plan of the project

To meet our objectives and address our research problem, the following computational phases are implemented. Each phase is designed and implemented to build the proposed system.

1. *CIE*: our research project is aimed at extracting the triplets (*location, person name* and *date*) of location visits from a gazetteer of Ornithology. A *reference dataset* for the spatial and attributive entities (*place name* and *person name*) and a gazetteer[3] are the main inputs to extract the *place name* and *person name*. The place names (such as *states* and *cities*) and person names can be extracted using a method that matches the input text against a list of place and person names. However, extracting the geographic coordinates and the temporal entities from the input text using the same method is not feasible. The geographic coordinates and the temporal entities must be extracted using a method that uses *Java Annotation Pattern Engine (JAPE)*[4] *Grammar Rules*. The JAPE Grammar Rules are pattern based expressions that comprise a Left-Hand Side (LHS) and Right-Hnad Side (RHS) rules. The input text is compared against the LHS pattern-based rule; and the RHS is executed whenever there is similarity between the LHS rule and the input text.

2. *Trajectory Triplets Extraction and Storage*: it is extracting, aggregating and storing the triplets (location, expeditioner name and date) of a location visits. A spatial database should be used to store the triplets with a relevant data type.

3. *Temporal Inference*: extracting the time-stamps of an expedition from a corpus is relatively easy, but capturing a vague temporal interval and duration of an expedition is not, because the availability of detailed time-stamps in a form of *"dd/mm/yyyy"* and temporal intervals

[3]The Brazilian ornithological gazetteer
[4]**J**ava **A**nnotation **P**atterns **E**ngine

in a form of *"dd/mm/yyyy-dd/mm/yyyy"* is not always true. The main concern of this phase is inferring the relative temporal boundaries of a location visit (i.e., if the time-stamp is non-crisp).

4. *Reliability assessment*: a trajectory narrates the story of an expedition in spatiotemporal dimension. The reliance on the story demands spatiotemporal reliability assessment. The reliability of a trajectory cannot be assessed on a single dimension (spatial reliability or temporal reliability) since the dimensions are not totally independent, for instance, the distance between two location visits may not exceed a threshold value over a distance while the duration is still a day, this shows the dependence between a spatial and temporal dimension. The following two alternatives are suggested to assess the reliability of a trajectory:

   (a) *Trajectory against constraints*: comparing the spatiotemporal property of a trajectory against predefined constraints of the spatiotemporal dimension can be a clue for a reliance. The constraints can be set based on the period of the expeditions, for instance, considering the accessibility of transportation in 1900s, constraints on a maximum distance and temporal interval between two location visits can be set; and a location visit that exceeds the constrained value might be regarded as an outlier.

   (b) *Trajectory against a reference trajectory*: the second option is to compare the extracted trajectory against an external source or manually computed trajectory, which serves as a reference. The functionality of the system depends on the reliance of the extracted trajectory. A geometrical and spatial situation comparison between the extracted trajectory and the reference (assuming that both expedition routes are , of the same period and expeditioner) serves as a clue to assess the reliability of both the system and an extracted trajectory.

1.3.2 Risks and contingencies

The problem we might face during this research project is when developing the spatial and temporal inference tool. Including all the possible spatial and temporal relation types we encounter in the gazetteer will enable our proposed system to solve ambiguities and extract as many triplets as possible, but getting through this will possibly face technical challenges and consume much time. Hence, we might be forced to include few of the spatial temporal relation cases only.

1.4 RESOURCES REQUIRED

1.4.1 Data

- The Brazilian Ornithological Gazetteer.
- Reference datasets for the state, city and place names in Brazil.
- Reference datasets for the person names in the Brazilian Ornithological Gazetteer.

1.4.2 Software and hardware

- Hardware
  Personal computer with all peripherals

- Software
  General Architecture for Text Engineering (GATE) Developer 7.1 [5]
  Eclipse
- Programming language
  Java
  JAPE
- Additional library
  Simple API for XML (SAX) [6]
  Java Database Connectivity (JDBC) [7]

[5] http://gate.ac.uk/

[6] SAX (Simple API for XML) is an event-based sequential access parser API developed by the XML-DEV mailing list for XML documents

[7] Java Database Connectivity is an API to create a connection between a Java application and database

Chapter 2
Literature Review

2.1 CONTEXTUAL INFORMATION EXTRACTION

Many research projects have been carried out to extract the geospatial entities from: a corpus, web queries, micro-text messages,[1] metadata and wikipedia[2] [6, 7, 9, 11, 22, 23]. The only concern of these projects was to recognize and extract geospatial entities. However, we proposed a trajectory extraction and storage system which has a contextual information extraction that is ideally required to recognize and extract the spatial, temporal and attributive contexts out of a textual document.

Geographic Information Retrieval is the study of methods that consist of the *Geographic Information Systems (GIS)* and *Information Retrieval (IR)* [4]. Such methods use an indexing structure to store and retrieve both a text document and the geographic context contained within the text. Two information retrieval approaches have been presented in [4]: a textual technique and a spatial technique; targeting on a language and a spatial aspects respectively. On the other hand, [3] have come up with a *temporal analysis framework* to utilize the temporal dimension of a corpus.

Travel guides and travel diaries have been used in [6], to correctly recognize geographic information to construct actual trajectory datasets that can be visualized on a map. In this research project the extraction of relative and absolute geographic information has been achieved. The main advantage of the method used in [6] is that only the linguistic, semantic and contextual information contained in the provided documents have been used.

Unlike the methods in the above projects, our proposed system focuses on extracting the spatial, temporal and attributive entities from an Ornithological Gazetteer. This proposed system has a bit of methodological similarity with [6]: the proposed system depends on the linguistic and contextual information contained in the provided Ornithological Gazetteer.

2.2 GEO-PARSING

Geoparsing is a method that recognizes and annotates geographic entities in documents [12]. A Geoparsing Web Service was developed by [1] to extract geographic coordinates from textual travel narrative documents. In this research project, Yahoo! Placemaker[3] was used as a geotagging tool. The geoparsing process in this web service has two steps: *entity extraction* and *disambiguation*. The weakness of the approach is that the issue of relative positioning of spatial objects was not addressed (i.e., the approach can extract only those geospatial entities with an absolute location). The web service can extract named entities and visualize them (Figure 2.1), but the spatial relationships between features and temporal relationships between entities cannot be captured. It is obvious that such a narrative textual document will possibly contain relative locations of spatial objects

[1] micro-text message is a short text possibly with many abbreviated words
[2] http://www.wikipedia.org/
[3] http://developer.yahoo.com/geo/placemaker/

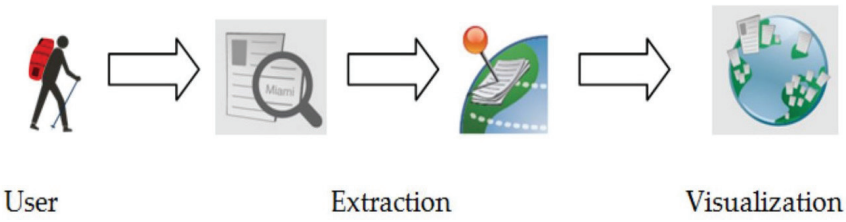


Figure 2.1: Geoparsing and visualization process

and temporal entities. However, the system we proposed uses a spatial and temporal inference tool to analyse and infer the spatial and temproal relatioships between features.

2.3 TRAJECTORY REPRESENTATION

A historic expedition is an aggregate of the location visits during an undertaken journey. The aggregation of spatial, descriptive and temporal components of such visits is characterized as a trajectory, meaning that an expeditioner can be considered as a moving object. When there are no observations to determine the path or the location of a moving object, a proper method is required to fill the observation gap [14]. Since the main concern of our research is *transforming historic location descriptions to trajectory*, a positional interpolation method presented in [14] might be used to compute the missing spatial components of a given trajectory. However, in [14], a positional interpolation is introduced that is entirely dependent on the characteristics of a moving object. Therefore, depending on the characteristics of an expeditioner, the proposed system might be equipped with a tool that interpolates the position of an expeditioner at a given time (i.e., when the location is not defined explicitly).

2.4 SPATIAL REASONING

The ability to represent and reason about space is the fundamental feature of general intelligence [13]. In the Brazilian Ornithological Gazetteer (i.e., our dataset), a location description has a spatial coordinate that informs where an expeditioner is. These coordinates along with a phrase that describes the spatial relation are used to recognize the spatial relationships between spatial entities. Reasoning the spatial relationships quantitatively is required mainly, to compute the relative location of a location visit, assuming that the location of a reference spatial object and a phrase that describes a spatial relation are provided in the location description. The proposed system has a spatial inference tool to quantitatively reason the spatial relations that are detected during a contextual information extraction process.

2.5 TEMPORAL REASONING

A reasoning activity in a dynamic domain needs to include a temporal characteristic [10]. *Time semi-interval* is a temporal primitive that is the *start* or *end* point of an event (i.e., an event is a *location visit* in our case). In [8], *time semi-intervals* and their relationships are used as the basic units of temporal knowledge. Temporal reasoning between *time semi-intervals* requires a reasoning capability to compute the missing temporal member of the primitive, either the start or end of an event. In the Brazilian Ornithological Gazetteer (i.e., our dataset), there are location descriptions with non-crisp temporal marks; in such chases a temporal inference method is required to relatively

infer the temporal boundaries of a given location visit. To do so, the proposed system has a tool that infers a relative temporal mark for the vaguely defined location visits relative other crisply defined location visits.

2.6 TEMPORAL KNOWLEDGE REPRESENTATION

Historical descriptions are dependent on a temporal dimension, hence, time is a fundamental concern when modeling historical information [15]. If the temporal description of a historical event is vague, then the temporal information to be extracted will be subjected to uncertainty [15].

Historical events are not always represented with crisp temporal information, but with imprecise and subjective ones [15]. In [15], it is claimed that the existing approaches for a temporal modeling are based on the assumption that the representation of time is crisp . These approaches cannot be applied to any temporal modeling tasks. To overcome the difficulties of vague temporal information representation, [15] presented *a fuzzy interval-based temporal model* that is capable of capturing vague temporal information.

2.6.1 Fuzzy temporal model

Time instants and *time intervals* are mentioned as a basic time primitive in [15]. However, *time instant* is a *time interval* if the granularity is increased and the interval is one of the usual understandable time intervals (such as day, week and month) [15]. For instance, a month is considered as time instance when it is counted in a given year, but a month itself is a time interval when the days of a given month are considered as time instants. Though it is common to see temporal statements in historical documents, the characterization is not always crisp as the documentation might be subjective and vague. Even if it is common to see non-crisp temporal statements in ordinary text, based on the temporal statements, the temporal model can be either continuous or discrete.

Time intervals as fuzzy sets

Given that the temporal boundaries of an event (i.e. start and end) are crisp, an assumption of the *time interval* as crisp also holds true. Let i represent crisp interval with i^- and i^+ denoting crisp start and end time points respectively [15]. An interval is called *imprecise* if the start or end points are unknown, this can be represented as a *fuzzy set* $\tilde{I}$; which is a membership function $\tilde{I} : T \longrightarrow [0, 1]$; $\tilde{I}(t)$ represents the confidence level that time t is in an interval i; If $\tilde{I}(t) = 0$, t is not in i; if $\tilde{I}(t) = 1$, t is in i [15].

2.6.2 Fuzzy temporal relations

If time intervals are not crisp, the temporal relations between temporal entities will also be non-crisp. It can't be determined that two time intervals are consecutive if time intervals are vague [15]. Given two crisp time intervals i and j and a crisp temporal relationship θ; the *fuzzy temporal relation* $\tilde{\theta}$ considers *fuzzy sets* $\tilde{I}$ and $\tilde{J}$ and compute a confidence level $c \in [0, 1]$ that represents the confidence that a fuzzy temporal relation θ is between i and j [15].

Chapter 3
Data Sources, Tools and Methods

3.1 DATA SOURCES

We discussed in the first chapter that the target of this research is to extract triplets of a location visit from a gazetteer and aggregate them into a trajectory. It is obvious that a gazetteer of any specific theme is subjected to that theme only; it is like an English dictionary is subjected to providing definition of English vocabulary. There is, of course, a number of gazetteers to work with, but the Brazilian Ornithological Gazetteer was chosen for this project.

3.1.1 Ornithological Gazetteer of Brazilian

The Ornithological Gazetteer of Brazilian was written by Raymond A. Paynter and Melvin A. Traylor in 1991 [16, 17]. This Ornithological Gazetteer has more than 6,000 description of sites where birds have been observed and located throughout Brazil [16, 17]. There are many expeditioners mentioned in this Ornithological Gazetteer. One of them is Tadeusz Chrostowski (1878-1923), a Polish ornithologist and explorer whose name is mentioned in 58 descriptions in the Brazilian Ornithological Gazetteer. These 58 descriptions were extracted from the spatial database in which the Gazetteer was stored and restored in a GATE Developer Datastore as a text entry with UTF-8 encoding. Section 3.2.2 discusses GATE Developer.

3.2 TOOLS AND METHODS

3.2.1 List of Named Entities

The contextual information extraction phase (see Section 4.3) has two ways to identify and annotate Named Entities from an input text: (1) by matching the entities with a reference dataset that contains a list of Named Entities; and (2) by matching the patterns of the entities with JAPE Grammar Rule LHS pattern expressions. Appendix B.1 contains Named Entities such as *Santa Catarina* and *Chrostowski* that can be annotated as *State* and *Person* respectively. These are not the only entities to be identified. There are entities such as 'S' and 'SE' that cannot be considered as an explicit Named Entity but they can be treated as one. Therefore, it is essential to compile lists of different Named Entities. The followings are types of frequently occuring Named Entities.

Direction indicators

We discussed that there are entities such as 'S' and 'SE' in an input text. These entities are direction indicators. The direction indicators are not always written in uppercase letters. As a result the direction indicators written with lowercase letters must be considered. All the abbreviated direction indicators were compiled to facilitate the contextual information extraction phase. The indicators are not extracted or recognized as an independent entity, but they are used to detect spatial relationships. The spatial relationships are often described with a phrase that combines *distance, direction indicator* and *place name*, such as in *"12 km SE of Curitiba"*.

Figure 3.1: Tadeusz Chrostowski: 1878-1923 (Source: Wikipedia)

Month

Though the month names are fixed, the way they are written can be quite different. All the possible month names have been compiled as a list of months. The compiled list is used as an input for the contextual information extraction machine to identify and annotate temporal entities, for instance, *"April"* can be written as one of the following ways.

```
APR
APRIL
Apr
April
apr
april
```

Organization

The names of the organizations in the Gazetteer are all written as uppercase abbreviations. A simple JAPE Grammar Rule can be defined to annotate words that are written with all uppercase letters as an Organization. However, such a rule may confuse the names of organizations with direction indicators. Therefore, it is mandatory to compile the lists of organizations that are mentioned in the Gazetteer. An alternative to avoid the annotation confusion could be to annotate the direction indicators first and consider the remaining uppercase abbreviations as organization names.

Person

The name of a *person* and the name of a *place* are both written as a first letter capitalized word. If the information extraction machine is using JAPE Grammar Rule to annotate these two types of

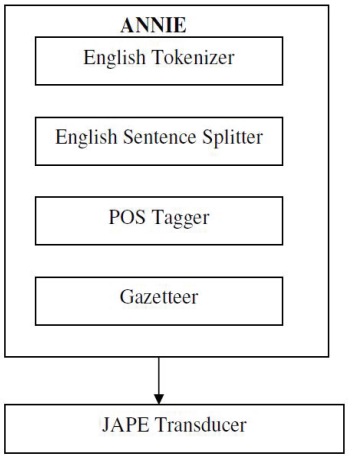


Figure 3.2: Typical GATE system components

named entities, it is quite common for the machine to confuse them. To overcome the annotation confusion, a list of names must be compiled either for the *person* or the *place* entity type or for both. The choice of the entity type to compile the list for is subjective. Lists of person names and place names were compiled based on the fact that the database in which the Gazetteer is stored has the list for both person and place names.

State, City and any Place Name

It can be argued that if any first letter capitalized word is not a person name, then it is a place name. However, identifying a word as a place name is not adequate to capture the notion behind the name. The spatial extent that is noted along with a city name is not the same as that of a country or a state name. Therefore, we need to annotate place names according to their administrative levels (i.e., as State, City and Place). Compiling the list of names for the states, cities and places is a good solution to make the information extraction machine capable of annotating the entities in an input text without any confusion. The list of the state and city names was compiled surfing the internet whereas the place names list was compiled from the database in which both the Ornithological Gazetteer and list of name of places in Brazil are stored.

3.2.2 GATE Developer

GATE is a platform to develop an application that processes natural language [5]. This platform consists of processing components that can be used for an Information Extraction System. GATE has various types of component, known as resources, that are reusable specialized types of *JavaBean*[1] [5]. These resources come in three different flavors; *LanguageResources (LRs)*, *ProcessingResources (PRs)* and *VisualResources (VRs)*.

3.2.3 ANNIE

A Nearly-New Information Extraction System (ANNIE) is an information extraction tool distributed with GATE that relies on algorithms and the JAPE language [5].

[1]JavaBeans are reusable software components for Java that can be manipulated visually in a builder tool.

ANNIE Tokeniser

The ANNIE Tokeniser is a tool that chunks a text into number of token types such as words, punctuations and numbers [5]. The tokeniser uses a rule which has a LHS and a RHS; the LHS and RHS are separated by '>'. The LHS is always a regular expression which has to be compared against an input text; the RHS contains the action to be carried out when the LHS expression is matched with the input text [5]. The RHS uses ';' as separator.

1. Tokeniser Rules

   {LHS}>{AnnotationType};{attribute 1}={value 1};...;{attribute n}={value n}

2. Token Types

   Word is any set of UPPERCASE or lowercase letters, including '-'.

   Number is any combination of string of digits.

   Punctuation is one of the English language punctuations.

   SpaceToken is the white space in a sentence.

3. LHS operators

   (or) |

   (0 or more occurrences) *

   (0 or one occurrence) ?

   (1 or more occurrences) +

ANNIE Sentence Splitter

The ANNIE Sentence Splitter is a transducer that chunks an input text into a number of sentences. In most cases a sentence splitter is preceded by a Tokeniser because the punctuations in a text are used to split the text into sentences. The sentence splitter uses a gazetteer list of abbreviations to help it identify a sentence-marking full stop [5], for instance *"Mr. Bean was born in Feb 1989."*; the full stops after *'Mr'* is not a sentence-marking. The gazetteer list of abbreviations is application-dependent and subjective to the characteristics of the text processing machine. After splitting, each sentence is annotated as *"Sentence"* and each sentence break is annotated as *"Split"* [5].

ANNIE Part of Speech (POS) Tagger

The ANNIE POS Tagger follows the Tokeniser and the Splitter. The Tagger produces a POS tag as an annotation class on each word or number token types. The annotation class produced by the Tagger is used by a pipeline[2] to extract Named Entities. Each POS tag is considered as a token category by other applications, assuming the applications need tagged POS, that go after the POS Tagger in the information extraction pipeline. For instance, given *"Snethlage, 1925"*, the POS Tagger annotates the parts of this sentence:

'Snethlage' is annotated as *Token* with a POS category of NNP (proper noun) and a token kind of *word*.

',' is annotated as *Token* with a token kind of *punctuation*, a POS category is not provided because comma is not recognized as a POS.

[2] A Pipeline is a module that consists of number of sequential applications

'1925' is annotated as *Token* with a POS category of CD (cardinal number) and a token kind of *number*.

After a sentence is tagged by the POS Tagger, the output annotation classes along with the POS categories are used in the JAPE Grammar Rule to define the LHS rules of the entity pattern expressions (see Section 3.2.4).

ANNIE Gazetteer

The ANNIE Gazetteer tags entities using a method that consumes an input text and list of reference items. It identifies entity names in the input text by checking their existence in the item list. The lists are plain text files with one entry per line. Each list file represents a set of entity names such as cities, organizations, days of the week and months. Entities of similar category must be stored with their kind only. This tagging application can be tuned to be case-sensitive or insensitive. If the application is tuned to case-insensitive the tagging works without having a distinction between upper and lower cases.

The lists of entity names are stored as "*.list" file. An index file, conventionally, "lists.def" is used to access the "*.list" files. The "lists.def" file provides the definition of each list files. The definition includes the *file name*, *major type*, *minor type*, *language* and *annotation type* as *column one*, *column two*, *column three*, *column four* and *column five* respectively. The columns are defined here:

Column one defines the *file name* of a list file. The "lists.def" file and all the "*.list" files must be stored in a similar directory.

Column two defines the major or primary identifier to be used when an entity from an input text is found in the items list file. For instance, the *major type* of 'Jhon' might be 'Person'. If the *major type* is is not defined, the default type is 'lookup'.

Column three defines the *minor type* or the secondary identifier. The *minor type* always comes with the *major type* and it gives a detail property of the identified entity. For instance, the *major type* of 'Jhon' and 'Maria' might be 'Person' and the *minor type* might be 'Male' and 'Female', respectively. A default value is not assigned for the *minor type*, meaning that it needs to be defined explicitly.

Column four defines the language of the input text being processed.

Column five defines the *annotation type* to be used when an entity is found in the list file. For instance, the *major type* of 'Jhon' might be 'Person'; the *minor type* might be 'Male' and the *annotation type* might be 'PersonMale'. Therefore, if the string 'Jhon' is found when a text is processed, it is tagged as 'PersonMale'.

```
1 Direction.list:Direction
2 Person.list:person:::Person
3 Organization.list::::Organization
4 Months.list:Month
5 Time.list:time
6 SpatialRelation.list:SRelation
7 State.list::::State
```

The *column one* is required to be defined; *column three* and *column four* are optional but either *column two* or *column five* must be provided with an explicit value. The columns in the "lists.def" file are separated by ":". In the example above (i.e., used in this research project) the *major type* of the *"Direction.list"*, *"Time.list"*, *"SpatialRelation.list"* and *"Months.list"* is specified; the *major type* and *annotation type* of *"Person.list"* are explicitly defined; the rest are provided with explicit annotation type.

3.2.4 JAPE: Regular Expressions over Annotations

JAPE[3] allows to recognize predefined regular expressions in annotations over textual documents [5]: a regular expression is set of strings, it does not include graphs. The JAPE Transducer always follows the Tokeniser, Splitter, POS Tagger and/or Gazetteer Processing. The tagged POS of an input text, lookup annotation, annotation types created by the Gazetteer Processing Machine and the JAPE Grammar Rules are used by the JAPE Transducer to annotate an input text.

JAPE Grammar Rule

A JAPE grammar is a pattern-based rule that consists of a set of phases [21]. These rules are stored as "*.jape" file. An index file conventionally 'main.jape' is used to access the JAPE Grammar phases (if multiple phases are defined). Each of these phases consists of a set of pattern/action rules. The rule has a LHS and RHS. The rule on the LHS contains the annotation pattern which, often, contains operators such as '*', '?', '|' and '+'. These operators are listed in Section 3.2.3. The RHS rule contains the action to be taken whenever the annotation pattern on the LHS is matched in an input text. A JAPE rule has the shape: *LHS->RHS*. The following is an example of a JAPE rule.

```
Phase:distancefinder
Input:
Options: control = appelt
Rule: distance
Priority:50
((
 ({Token.kind==word,Token.category==MD,Token.string=="Ca"}|
 {Token.kind==word,Token.category==MD,Token.string=="ca"})
 ({Token.kind==punctuation,Token.string=="."})?
 ({SpaceToken}))?
 (
 {Token.kind == number, Token.length == "1"}|
 {Token.kind == number, Token.length == "2"}|
 {Token.kind == number, Token.length == "3"}|
 {Token.kind == number, Token.length == "4"}|
 {Token.kind == number, Token.length == "5"}
 )
 (({Token.kind==punctuation,Token.string=="."})?
 ({Token.kind == number, Token.length == "1"})?
 ({SpaceToken})
 ({Token.kind==word, Token.category==NN, Token.orth==lowercase, Token.string=="km"}))
)
:distance
-->
 :distance.Distance= {rule = "distance"}
```

[3] Java Annotation Patterns Engine

Line 1 defines the phase name. Each of the phases in the JAPE grammar must have unique name, for instance, here the phase is named *distancefinder*.

Line 2 defines the input annotations, which the LHS rule uses for pattern matching, must be defined at the start of each grammar. In the absence of explicit definition of the input annotations; the defaults are *Token*, *SpaceToken* and *Lookup*.

Line 3 defines the option. There are two types of options that can be set at the beginning of each grammar rule:

1. *control* is a rule matching method. The control options are *Appelt, Brill, All or Once*. For instance, the *Appelt* forces the JAPE Grammar Rule to trigger a rule with higher priority first.
2. *debug* can be set to either *true* or *false*. It notifies a conflict between more than one possible match if it is set true.

Line 4 defines the name of the rule, in this example the name is *distance*

Line 5 defines the priority of the rule. If there are multiple rules in a single phase, the rules with higher priority is triggered and matched prior to the rest.

Line 6 – 23 it is the LHS of the rule. Here, the rule searches for a part of an input text that is a combination of word and number. This LHS pattern rule has three sub-patterns:

sub-pattern one matches a combination of word, punctuation and white space that equals "Ca. " or "ca. ", note the white space before the closing quotations (Line 6 – 10).

sub-pattern two matches a string of digits in one of the following formats: '9', '99', '99', '999', '9999', '99999' (Line 11 – 17).

sub-pattern three matches a combination of punctuation, number, white space and word that resembles ".1 km" (Line 18 – 21).

The sub-patterns in combination create a pattern rule that matches a distance in a text (e.g, Ca. 45 km). When a part of a text is matched with this pattern, the LHS rule tags the matched part with a temporary label, in this example the temporary label is *"distance"*.

Line 23 defines the temporary annotation class.

Line 24 separates the LHS and RHS.

Line 25 here, the RHS of the rule renames the temporary label (Line 23) into permanent annotation class. In this example the temporary label *"distance"* is renamed into a permanent label (*"Distance"*). The new label is recognized as an annotation class by other JAPE phases.

LHS Macros

LHS Macros are methods that allow to create a definition of a regular expression that can be used multiple times in the JAPE rules [5]. The following is one of the JAPE Grammar Rules used in our project to find a spatial relationship from an input next. This rule has three LHS Macros.

```
Phase:SpatialRelation
Input: Token Lookup SpaceToken PlaceName RiverName CityName StateName
Options: control = appelt

Macro:DI
(
(
 ({Token.kind==word,Token.category==MD,Token.string=="Ca"}|
  {Token.kind==word,Token.category==MD,Token.string=="ca"})
 ({Token.kind==punctuation,Token.string=="."})?
 ({SpaceToken}))?
 (
 {Token.kind == number, Token.length == "1"}|
 {Token.kind == number, Token.length == "2"}|
 {Token.kind == number, Token.length == "3"}|
 {Token.kind == number, Token.length == "4"}|
 {Token.kind == number, Token.length == "5"}
 )(
 ({Token.kind==punctuation,Token.string=="."})
 ({Token.kind == number, Token.length == "1"})
 )?
 (
 ({SpaceToken})
 {Token.kind==word, Token.category==NN, Token.orth==lowercase, Token.string=="km"}
 )
)
Macro:RI
(
 ({Token.string == "Rio"}|{Token.string == "rio"})
 ({SpaceToken})?
 ({Token.kind==word, Token.category==NNP, Token.orth==upperInitial}|
 {Token.kind==word})?
 ({SpaceToken})?
 ({Token.kind==word, Token.category==NNP, Token.orth==upperInitial}|
 {Token.kind==word})?
)
Macro:CO
(
 {Token.kind == number, Token.length == "4"}
 {Token.string=="/"}
 {Token.kind == number, Token.length == "4"}
)
Rule: spatialrelation
(
(DI)
({SpaceToken})
({Lookup.majorType=="Direction"})
({SpaceToken})?
({Token.string=="of"})?
({SpaceToken})?
({RiverName}|{CityName}|{StateName}|{PlaceName})
({SpaceToken})?
({Token.kind==punctuation,Token.string=="["})
(CO)
({Token.kind==punctuation,Token.string=="]"})?
):spatialrelation
-->
:spatialrelation.SpatialRelation={rule="spatialrelation"}
```

A phrased spatial relationship such as "12 km SE of Curitiba" is a combination of distance, direction indicator and place name. Defining single JAPE Grammar Rule that matches a spatial relationship might be complicated. However, using LHS Macro, the repetitive parts of a spatial relationship phrase such as distance and place name can be defined once and used many times. For instance the above JAPE Grammar rule has three LHS macros:

Macro:DI (Line 5 – 25) is a pattern rule that matches a distance.

Macro:RI (Line 26 – 35) is a pattern rule that matches a river name.

Macro:CO (Line 36 – 41) is a pattern rule that matches a coordinate.

These LHS macros are not independent rules that annotate an entity, but they are used as sub-patterns of the JAPE Grammar rule that matches a spatial relationship. These macros are called inside the rule defined to match a spatial relationship (i.e., Line 43 – 58 in the above illustration).

JAPE Transducer

In the context this research project, a transducer is a method with an input and output phase. A transducer translates the contents of its input (LHS rule) to new content of output (RHS rule). In our context it takes an input text (location visit descriptions) and returns a text with annotation classes (annotated descriptions).

3.2.5 JDBC

JDBC is an Application Programming Interface (API) that serves as a standard tool for a database-independent connectivity. It allows connectivity between a Java program and a database. This API allows to use the Java programming language to query and update relations in a relational database. This API is used in our research project as a communication tool between our Java application and the database that is both a source for the input location visit descriptions and a store for the extracted trajectory triplets.

Chapter 4

Trajectory Extraction and Storage System (TESS): Design and Implementation

4.1 INTRODUCTION

This chapter discusses the concepts, procedural requirements, design and implementation of the Trajectory Extraction and Storage System. Figure 4.1 shows the conceptual model of the system. The system has six components, namely, *Raw Data Extraction*, *Contextual Information Extraction*, *Trajectory Triplet Extraction*, *Temporal Inference*, *Expedition Route Production* and *Quantitative Spatiotemporal Reasoning*. In Section 4.2 we discuss the Raw Data Extraction phase. It is the first phase of the system that prepares the location descriptions as an Extensible Markup Language (XML) Document for the Contextual Information Extraction phase (see Section 4.3). All the phases of the built system except the Contextual Information Extraction are developed as Java applications. The Contextual Information Extraction Phase is developed in GATE application development environment with two text processing pipelines: Entity Extraction Pipeline and Spatiotemporal Relationship Extraction Pipeline. These pipelines are used to recognize and annotate Named Entities and spatiotemporal relationships, respectively. The Trajectory Triplet Extraction phase is the main component of the system, which is used to extract and store the triplets of a location visit (expeditioner name, location and time) from the location descriptions (see Section 4.4). Since the temporal information of those location visits is not often provided with a crisp fashion, the Temporal Inference phase which infers a relative temporal boundary follows the Trajectory Triplet Extraction process (see 4.5). The Expedition Trajectory Storage phase follows the Trajectory Triplet Extraction and the Temporal Inference phases to categorize and store the extracted and inferred triplets into a number of expedition trajectories (see Section 4.6). After the Expedition Trajectory Storage process is completed, the Expedition Route Production phase creates a number of expedition routes for a given expeditioner on the basis of the user provided value of a $temporal\ gap$ that is a temporal benchmark between two consecutive expeditions (see Section 4.7). This phase creates KML line and point files to visualize the produced route in the Google Earth environment. The system has a reliability assessment phase which is conducted in two ways: on the basis of visual inspection and the Quantitative Spatiotemporal Reasoning phase. The Quantitative Spatiotemporal Reasoning phase has a tool that assesses the reliability of the extracted triplets and the produced expedition routes on the basis of constraints that are provided by the user (see Section 4.8).

4.2 RAW DATA EXTRACTION (RDE)

Though each entry in the Ornithological Gazetteer is a description of a spatially defined location. The content of these entries regarding a number of expeditioners is different. Some entries might consist of a single expeditioner (see Appendix B.1) while some consists of multiple expeditioners (see Appendix B.2).

The Raw Data Extraction is the first block of the system. It is defined as a process of preparing the descriptions for the Contextual Information Extraction phase. It extracts the location de-

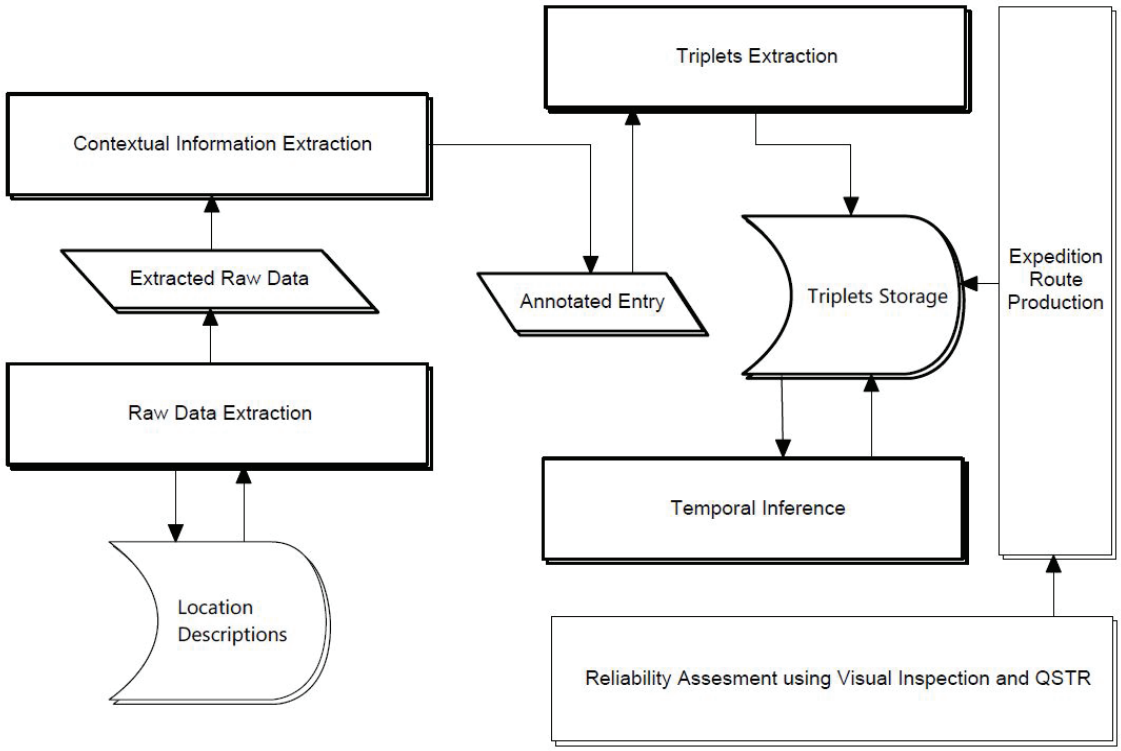


Figure 4.1: Conceptual Model of Trajectory Extraction and Storage System (TESS)

scriptions of a given expeditioner and store them as XML document in which one XML element contains a sentence that has the temporal, spatial and attributive phrases for one expeditioner. The preparation of the descriptions to the Contextual Information Extraction phase is handled by three methods: *parseParagraph()*, *storeSubParagraph()* and *fromDatabaseToXML()* (see Algorithm 1). The algorithm is discussed here:

Data: the database that contains the location visit descriptions

Process:

Line 2 – 4: the data extraction process requires two attributes to be provided by the user: the expeditioner name and the sentence splitting punctuation. Since the location descriptions in the Ornithological Gazetteer contain the spatial and historic descriptions in the same paragraph, the sentence splitting punctuation is used to: separate the spatial and historic descriptions and separate between text describing different expeditioners.

Line 5: the location descriptions of the mentioned expeditioner are brought in from the database and separated into sub-paragraphs.

Line 6 – 12: the separated sub-paragraphs are concatenated with other attributes that were brought in and the concatenated sub-paragraphs are stored in the database.

Line 13: the stored sub-paragraphs are brought in from the database to create the XML Document that transports the location descriptions to the Contextual Information Extraction phase.

Result: the result is an XML Document in which each element is a location description that contains the attributive, temporal and spatial phrases.

Algorithm 1: Raw data extraction

Data: $G =$ Ornithological Gazetteer database
Result: $R =$ XML document of a set of location descriptions such that $R \in G$

```
1  begin
2      expeditioner ⟵ enter expeditioner name;
3      P = set of location descriptions of a given expeditioner; SS = sentence separator;
4      SS ⟵ enter sentence separator;
5      P ⟵ parseParagraph(G, expeditioner, SS);
6      while P_x ∈ P do
7          description ⟵ P_x(sub_paragraph);
8          location ⟵ P_x(location);
9          name ⟵ P_x(name);
10         id ⟵ P_x(id);
11         storeSubParagraph(token, location, name, id);
12     end
13     R ⟵ fromDatabaseToXML(expeditioner);
14 end
```

The *parseParagraph()* method parses a location description that is in a form of paragraph into number of sub-paragraphs using a semicolon as a benchmark between two sub-paragraphs. The gazetteer that we use as a data source treats location descriptions as a single paragraph each of which uses a semicolon to separate spatial and historic descriptions. Within historic descriptions, the semicolon is often also used to separate between text describing different expeditioners (see Appendix B.2). The *parseParagraph()* method brings a location description and other additional attributes from the database in which the descriptions are stored and chunks each of them into sub-paragraphs and stores them using the *storeSubParagraph()* method.

The *storeSubParagraph()* method concatenates the parsed sub-paragraphs with a coordinate mark before they are stored into the database. The *storeSubParagraph()* method, after concatenating the *location* column value with the parsed sub-paragraphs, stores the *id*, *name* and *sub-paragraph* into a separate relation. The coordinate marks are brought in from the same relation that stores the location descriptions. This relation has many columns but only the *id*, *name*, *location* and *description* columns are used in the Raw Data Extraction phase:

- *id*: a location description identifier which is used in the contextual information extraction and visualization phases to track upon which entry a triplet is parsed from.
- *name*: a place name that a location description belongs to.
- *location*: a coordinate in a form of *9999/9999*, *ca. 9999/9999*, *9999N/9999* or *ca. 9999/9999 ?* that specifies the spatial context of a location visit or a coordinate denoting a region in a form of *9999/9999-9999/9999*.
- *description*: a paragraph that describes a location with descriptive sentences about the expeditioner, location and time of a visit. This column is used as a main source to extract the triplets and generate a trajectory of an expedition.

The *fromDatabaseToXML()* creates an XML file (see XML. 4.1) from the parsed sub-paragraphs. This XML file is used in the Contextual Information Extraction block to annotate different entities contained in each of the parsed location descriptions. As XML. 4.1 shows the element *Location-Visit* contains two attributes *id* , *name* and the parsed *sub-paragraph*. A parsed sub-paragraph is

semantically better than the description it is parsed from, because it always puts the spatial , attributive and temporal phrases in the same order, which improves the speed and quality of the trajectory triplet extraction task.

XML Document 4.1: Extracted raw data

```
<?xml version="1.0" encoding="utf-8" ?>
<Expedition>
       <LocationVisit id="3059" name="FERNANDES PINHEIRO">
       2525/5033 : Chrostowski, 21 Dec. 1910, 4, 11 Jan. 1911 (Chrostowski,
           1912:498, 499).
       </LocationVisit>

       <LocationVisit id="279" name="APUCARANA">
       2447/5110 : Chrostowski, Aug. 1922 (Sztolcman, 1926:124)"
       </LocationVisit>
</Expedition>
```

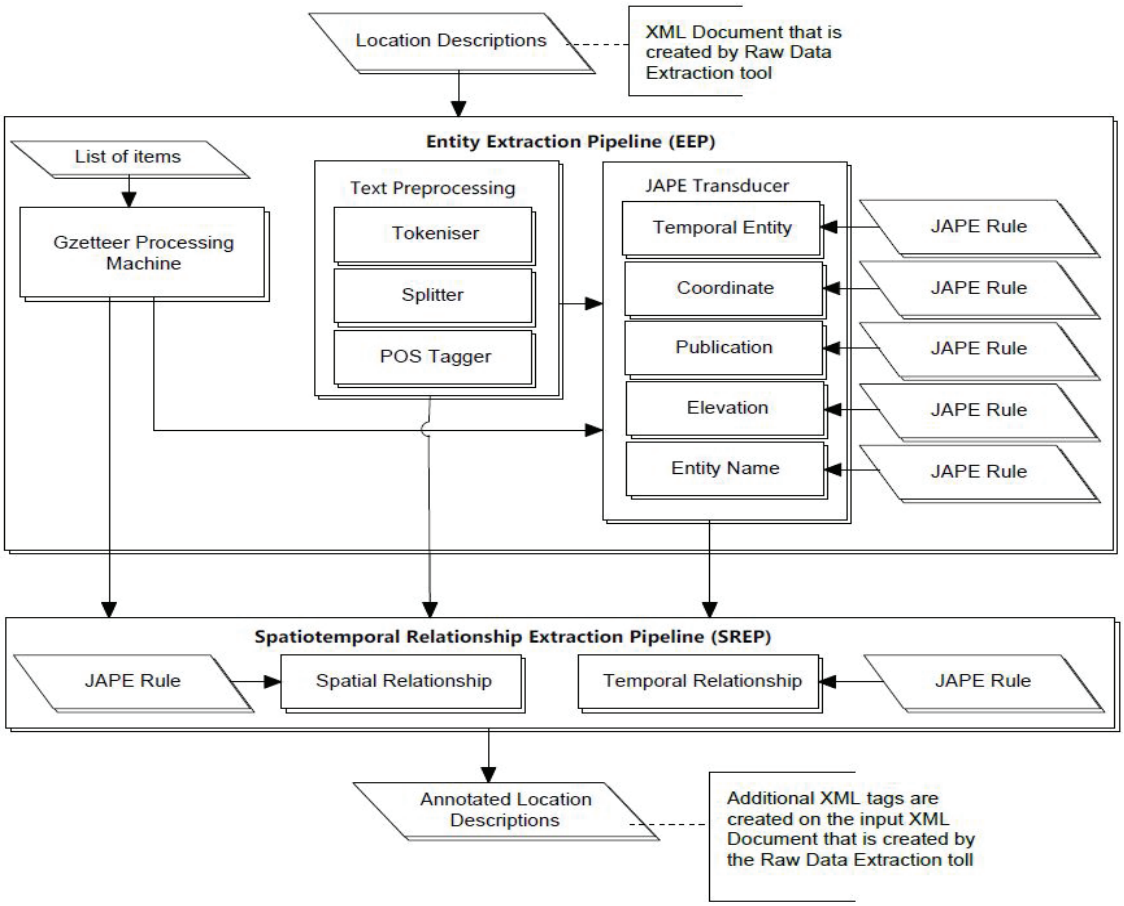


Figure 4.2: Conceptual Model of Contextual Information Extraction Process

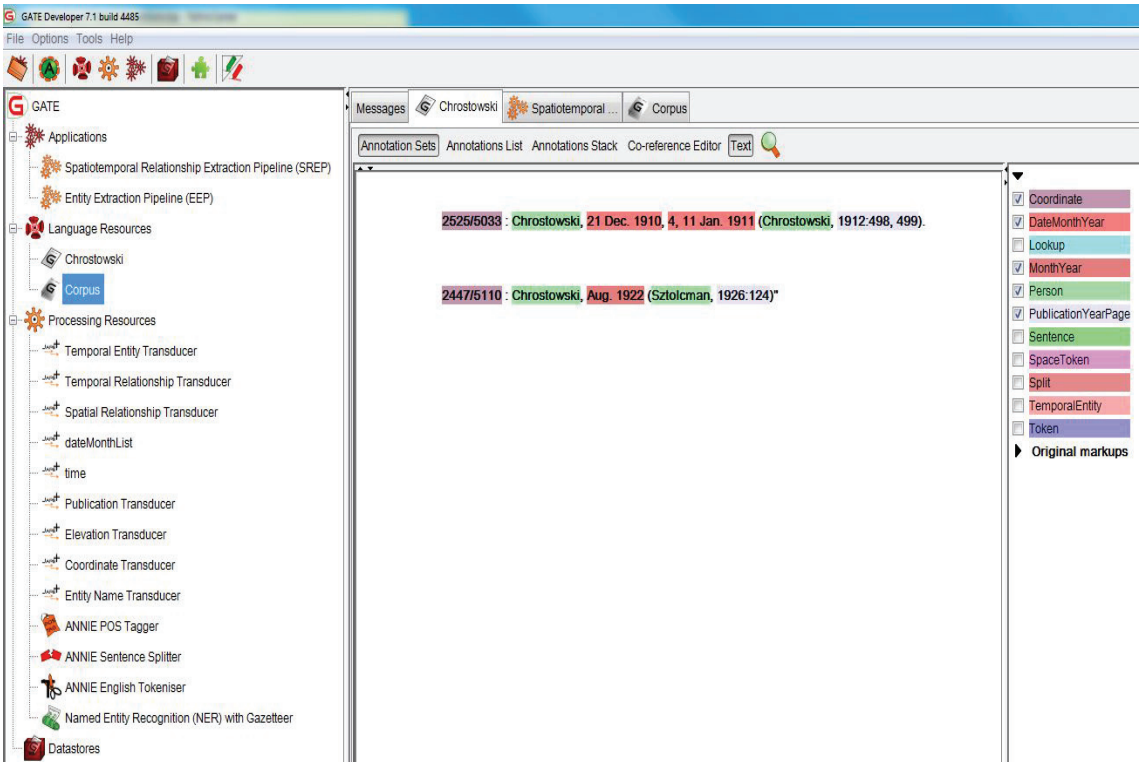


Figure 4.3: Entity and Spatiotemporal relationship extraction Pipelines

4.3 CONTEXTUAL INFORMATION EXTRACTION (CIE)

The Contextual Information Extraction is the second block of the *Trajectory Extraction and Storage system (TESS)*. This phase is a an independent application that has two text processing pipelines; the *Entity Extraction Pipeline* and the *Spatiotemporal Relationship Extraction Pipeline*. These text processing pipelines are built using GATE Developer as an application development platform. Each component of the pipelines are built using the tools of ANNIE and JAPE Grammar Rules. The components of ANNIE that are used to built these text processing pipelines and the principles of JAPE Grammar Rules are discussed in Section 3.2.3 and 3.2.4, respectively. These two text processing pipelines are executed in a row; always the *Entity Extraction Pipeline* is executed first and followed by the *Spatiotemporal Relationship Extraction Pipeline*. The reason behind this execution order is that the second pipeline uses the outputs(Annotation classes) from the first. Figure 4.2 shows the conceptual model of the Contextual Information Extraction block.

The Contextual Information Extraction block consumes an XML file (the extracted raw data of Section 4.2) and returns the same file with annotation classes. Each of the annotation classes is created by the components in the text processing pipelines. The first pipeline (Entity Extraction Pipeline) takes care of recognizing and tagging different entities out of the input XML file. The second pipeline uses (Spatiotemporal Relationship Extraction Pipeline) the same file and the tagged entities to recognize and annotate relationships between entities. The principle to annotate the relationships between the entities is to recognize the entities first. Every component in each of these pipelines has its own task; the task being recognizing and annotating a specific and single entity type, meaning that there is a reserved tagging component in each of the pipelines for every entity kind. The following two sections (4.3.1 and 4.3.2) discuss these two text processing pipelines in detail. Figure 4.3 shows the Entity Extraction and Spatiotemporal Relationship Extraction

pipelines in the GATE Developer environment and annotated location descriptions (the extracted raw data of Section 4.2 was used for this illustration).

4.3.1 Entity Extraction Pipeline (EEP)

The *Entity Extraction Pipeline* has three text processing blocks: *Gazetteer Processing Machine*, *Text Preprocessing* and *JAPE Transducer*. The inputs for this pipeline are: *XML* document (location description), *list of items* (list of Direction indicators, Organization, Month, State, Person and City) and *JAPE rules*. The pipeline has two flavors (Document and Corpus). The document pipeline is executed on a single GATE Document and the Corpus pipeline is executed on a GATE Corpus; GATE Document is a container of a single TEXT, HyperText Markup Language (HTML) or XML file, whereas GATE Corpus is a container of one or more GATE Documents. The *Gazetteer Processing Machine* is the first annotator in the *entity extraction pipeline*. It tags *State*, *Person*, *Organization* and *City names*. The *Text Preprocessing Machine* is a temporary annotator in this pipeline; the annotation classes are used only as an input for the *Spatiotemporal Relationship Extraction Pipeline* and JAPE Transducer of the *entity extraction pipeline*. The *Text Preprocessing Machine* contains the *ANNIE Tokeniser*, *ANNIE Splitter* and *POS Tagger*. Conventionally, the Text Preprocessing block precedes the JAPE Transducer. This block is responsible to chunk a paragraph into sentences, chunk a sentence into tokens and tag each token with its POS category. The *JAPE Transducer* is the second annotator in the Entity Extraction Pipeline. The inputs of this annotator are the tagged POS from the Text Preprocessing Machine, annotation classes from the Gazetteer Processing Machine, an input XML file and JAPE rules. This annotator and tags *Coordinate*, *Elevation*, *City Name* (i.e., those missed by the Gazetteer Processing Machine), *Place Name* (neither State nor City), *Publication*, *Journal*, *River Name* and *Temporal Entity*. Tables 4.2 and 4.1 show the entity types to be annotated by the Gazetteer Processing Machine and by the Transducers, respectively.

Named Entity Recognition (NER) with Gazetteer

We discussed in Section 4.3.1 that an entity is recognized and annotated by either the Gazetteer Processing Machine (the first annotator) or the JAPE Transducer (the second annotator). Entity recognition and annotation by the Gazetteer Processing Machine is feasible and restrictive to Named Entities with an accessible reference dataset, for instance, State names and City names. This annotator extends the characteristics of the ANNIE Gazetteer (see Section 3.2.3); it consumes a location description and *lists.def* file. This file is a definition of the reference item lists for each Named Entity type. The definition includes the *Major type*, *Minor type* and *Annotation class* of an entity type. This annotator inserts each of the words from an input location description into the processing machine, and checks if there is a match for the inserted word in the reference items list. If the outcome is positive, the annotator tags the matched word according to its annotation class as defined in the *lists.def* file. During the matching process, the processing machine can be set to be either case-sensitive or insensitive.

Coordinate Transducer

The Coordinate Transducer is a component of the JAPE Transducer in the Entity Extraction Pipeline. This transducer uses an explicitly defined JAPE Rule that is capable of matching a coordinate(latitude/longitude) patterns. An input location description may have different patterns for a coordinate. The typical patterns of a coordinate in the *Brazilian Ornithological Gazetteer* are: *9999/9999*, *Place Name 9999/9999*, *ca. 9999/9999*, *9999N/9999*, *ca. 9999/9999 ?*, *location?* or *not located*. There are three possible annotation classes for any matched coordinate patterns:

Table 4.1 Entities annotated by JAPE Transducer

No.	Entity	Pattern	Annotaiton Class
1	Coordinate	9999/9999, ca. 9999/9999, 9999N/9999, ca. 9999/9999 ? or Place Name 9999/9999	Coordinate
2	Unknown Coordinate	not located or location?	CoordinateUnknown
3	Date	12-28 Mar.	DateMonth
4	Date	13 Jan. - 28 Feb. 1900	DateMonthDuration
5	Date	14 July-Dec. 1817	DateMonthMonthDuration
6	Date	1-2 May 1930	DateMonthYear
7	Date	26 Dec. 1902-5 Jan. 1903	DateMonthYearDuration
8	Date	Feb. 1922	MonthYear
9	Date	July-Dec. 1817	MonthDuration
10	Date	3, 8, 18, 26-27 Apr., 6, 10-11, 17, 26 May, 17, 24 Nov. 1914	DateMonthListYear
11	Date	Feb.[?] or 1817[?]	DateVague
12	Elevation	999 m, 999.9 m, 999 ft,...	Elevation
13	Optional Place Name	as "Barro do Rio do Peixe"	OptionalPlaceName
14	Place Name	Fazenda Ferreira	PlaceName
15	Publication	1925:331-333	PublicationYearPage
16	River Name	Rio Ivaí	RiverName
17	Journal	Ann. Zool. Mus. Polonici Hist. Nat	JournalName
18	City Name	city of Rio Negro	CityName

Table 4.2 Entities annotated by Gazetteer Processing Machine

No.	Entity	Sample	Annotaiton Class
1	City	city of Rio Negro	CityName
2	Organization	USBGN	Organization
3	Person	Chrostowski	Person
4	State	Parana	State

Table 4.3 Relationships annotated by JAPE Transducer

No.	Relationship	Pattern	Annotaiton Class
1	Spatial	ca. 12 km SE of Curitiba	SpatialRelation
2	Temporal	sometimes between May. and June.	TemporalRelation

1. *Coordinate* is assigned for the *9999/9999*, *ca. 9999/9999*, *9999N/9999* and *ca. 9999/9999 ?* coordinate patterns.

2. *PlaceCoordinate* is assigned for coordinate which is situated next to a place name; *PlaceName 9999/9999*.

3. *CoordinateUnknown* is assigned for *location?* and *not located* coordinate patterns.

Elevation Transducer

Like the other transducers the Elevation Transducer also uses JAPE rules to match an elevation pattern in the provided location description. The JAPE Rule that is used for this transducer is capable of detecting an elevation entity with one of the following patterns: *ca. 999 m*, *ca. 999m*, *ca. 9,999 m*, *ca. 999.99 m*, *ca. 999 ft* and *Alt.?*. This transducer has two annotation classes:

1. *Elevation* is assigned for on of the following matched patterns; *ca. 999 m*, *ca. 999m*,*ca. 9,999 m*, *ca. 999.99 m* and *ca. 999 ft*.

2. *ElevationUnknown* is assigned for an elevation entity matched with *Alt.?* pattern.

Entity Name Transducer

We discussed in Section 3.2.4 that a JAPE grammar rule contains a set of phases. Each of these phases consists of a set of pattern/action rules. The Entity Name Transducer uses a JAPE grammar which contains a single phase with nine different rules for nine different patterns of entities that frequently occur in the text corpus. The nine rules of the Entity Name Transducer are categorized into five main annotation classes, namely *RiverName*, *PlaceName*, *OptionalPlaceName*, *JournalName* and *CityName*. These rules are discussed here:

1. *RiverName*: almost all river names in the *Brazilian Ornithological Gazetteer* start with the word *Rio*. The word *Rio* has two notions behind; (1) it represents the short name for *Rio de Janeiro* or (2) it is a Portuguese word for *river*. Therefore any consecutive string of words that starts with *Rio/rio* and/or followed by *da/de/do/des* (the Portuguese language connectors) or any first letter capitalized word must be a river name, unless it is listed exclusively in one of the item lists of the Gazetteer Processing Machine. The output (annotation classes) from the Gazetteer Processing Machine is used in the JAPE Transducer to avoid an annotation confusions. The JAPE rule defined to annotate a river name is capable of excluding City, State or any place names that starts with *Rio/rio*. The list of items to be excluded from the river annotation class are the output of the Gazetteer Processing Machine. Any word or string of words with positive response to this specific rule is annotated as *RiverName*. There is an exception to this rule: if a city or a place name starts with *Rio/rio*, it is not annotated as *RiverName*. This rule can be extended to include other Portuguese words that denote a river, such as *Igarapé*, *Iguapo*, *Ribeirão*, *Barra*, *Canal*, *Lagoa* and *Braço*.

2. *PlaceName*: a place name, in the *Brazilian Ornithological Gazetteer*, can be a single first letter capitalized word or string of first letter capitalized words optionally with Portuguese connectors such as *da/de/do/des* and adverbs such as *de lá/de las*. If the rule depends only on the pattern of the words, any place name can be easily annotated as a state, city, river or person name. For instance, *Rio Ivaí* is a river name, but can be annotated as a *PlaceName*, provided the rule that any consecutive of first letter capitalized words is a place name. The inclusion of items not to be annotated as place name helps to avoid the annotation error. A

part of an input location description is positivity responsive to *PlaceName* rule if and only if the pattern rule is matched and the matched item is not one of those already annotated entities. The annotation class for a part of a description with a positive response to this rule is *PlaceName*.

3. *OptionalPlaceName*: this annotation class extends all the characteristics of the *PlaceName* annotation class rule. There are two additional characteristics to this rule: (1) every *OptionalPlaceName* is preceded by *'as'* and (2) every *OptionalPlaceName* is situated between quotation marks, for instance, *as "Barro do Rio do Peixe"*. The transducer matches a part of an input location description against this rule, and any word or string of words with positive response for the *OptionalPlaceName* rule is annotated as *OptionalPlaceName*.

4. *JournalName*: as there are few journals mentioned in the Brazilian Ornithological Gazetteer, the easiest way of annotating a journal is the Gazetteer Processing Machine. However, annotating the journals name with JAPE Transducer is a way ahead in the annotation process, because it gives the possibility of annotating journals that are not mentioned in the item lists. The JAPE rule of this annotation class makes use of the punctuation tokens a lot. Many of the journal names in the Gazetteer are strings of first letter capitalized words followed by '.' (Here it is worth noting, that the full stop is not sentence marking) or situated inside brackets, for instance, *"Ann. Zool. Mus. Polonici Hist. Nat"*. Any part of an input text with positive response to this rule is be tagged as *JournalName*.

5. *CityName*: we discussed in Section 4.3.1 that *CityName* is one of those entities to be annotated by the Gazetteer Processing Machine. However, if there are any city names missed by the Gazetteer Processing machine, they are annotated by the JAPE Transducer. If any first letter capitalized word or string of its kind is preceded by a phrase *"city of"*, then it is a city name, for instance, *city of Rio Negro*. To avoid double annotation, the *CityName* JAPE rule excludes the city names that are already identified by the Gazetteer Processing Machine. Any part of a location description that matches this rule is annotated as *CityName*.

Publication Transducer

The Publication Transducer annotates those entities with one of the following patterns: *1925:331-333*, *1925* and *5:331-333*; these patterns are annotated as *PublicationYearPage*, *PublicationYear* and *PublicationVolumePage* respectively. This transducer has three different rules and three annotation classes in a single phase. Here, it is worth noting that if four digit numbers were considered as publication year, coordinates with 9999/9999 pattern would be annotated as *PublicationYear/PublicationYear*; but the rule used to annotate a publication year is restrictive to those four digit numbers that are followed by a punctuation token ':'.

Temporal Entity Transducer

A location description might consist of multiple patterns for a temporal entity. The Temporal Entity Transducer uses a single phase JAPE rule with nine different rules for different patterns. This Transducer uses annotation classes of the Gazetteer Processing Machine and the Text Preprocessing block. According to the location descriptions, the possible patterns for any temporal entity are those listed in Table 4.1. The JAPE Rule that is used for a temporal entity annotation in this research project is shown in Appendix A.1. The annotation classes of this Transducer and their respective patterns are discussed here:

1. *DateMonth*: if one or two digit numbers or string of this kind is followed by a *Month*, then it is annotated as *DateMonth*, for instance, *12-28 Mar*.

2. *DateMonthYear*: one or two digit numbers or string of this kind is followed by a *Month* and a four digit number is annotated as *DateMonthYear*, for instance, *28 Feb 1900*.

3. *DateMonthListYear*: list of *DateMonth* ended by *DateMonthYear* is annotated as *DateMonthListYear*, for instance, *3, 8, 18, 26-27 Apr., 6, 10-11, 17, 26 May*, *17, 24 Nov. 1914*.

4. *DateMonthDuration*: one or two digit numbers followed by a *Month*, '-', one or two digit number, *Month* and four digit number is annotated as *DateMonthDuration*, for instance, *31 June.-28 Feb. 1900*.

5. *DateMonthMonthDuration*: one or two digit numbers followed by a *Month*, '-', *Month* and four digit number is annotated as *DateMonthMonthDuration*, for instance, *14 July-Dec. 1817*.

6. *MonthYear*: a *Month* followed by four digit number is annotated as *MonthYear*, for instance, *Dec. 1817*.

7. *DateMonthYearDuration*: one or two digit number followed by a *Month*, four digit number, '-', one or two digit number, *Month* and four digit number is annotated as *DateMonthYearDuration*, for instance, *26 Dec. 1902-5 Jan. 1903*.

8. *MonthDuration*: a *Month* followed by '-', *Month* and four digit number is annotated as *MonthDuration*, for instance, *July-Dec. 1817*.

9. *DateVague*: a *Month* or four digit number followed by '[?]' is annotated as *DateVague*, for instance, *Feb*[?] or *1817*[?].

4.3.2 Spatiotemporal Relationship Extraction Pipeline (SREP)

The Spatiotemporal Relationship Extraction Pipeline is the second pipeline in the contextual information extraction process, and it is dependent on the Entity Extraction Pipeline. The annotation process over entities relationships should be preceded by the entity extraction process. This pipeline has two basic JAPE transducers: (1) Spatial Relationship Transducer and (2) Temporal Relationship Transducer.

Spatial Relationship Transducer

The Spatial Relationship Transducer can be considered as an annotator with a semantic flavor. This transducer consumes a location description and its annotated entities (the outputs from the Entity Extraction Pipeline) and annotates the relationship amongst the spatial entities. The principle behind annotating a spatial relationship is annotating the spatial entities first. The JAPE rule used for a spatial relation annotation has 1 annotation class *SpatialRelation* and four rules:

Rule 1: 'ca. ' followed by a distance, direction indicator and place name (State, CityName, PlaceName); for instance, *ca. 12 km SE of Curitiba*.

Rule 2: a direction indicator followed by any of Place, City or State names; for instance, *SE of Curitiba*.

Rule 3: any of Place, City or State names followed by spatial relationship indicators (A word that indicates spatial relationship, such as *between*, *border* and *near*); for instance, *Paraná border*.

Rule 4: a spatial relationship indicator followed by a place name, 'and' and a place name; for instance, *between Vitória da Conquista and Poções*

Temporal Relationship Transducer

The Temporal Relationship Transducer works with a similar principle as of the Spatial Relationship Transducer. The JAPE rule used for this transducer has one rule and one annotation class *TemporalRelation*. The temporal relationship pattern defined by this rule is: a temporal relation indicator followed by a *Month*, temporal relation indicator and a *Month* again. A typical example for this pattern is *sometimes between May and June*.

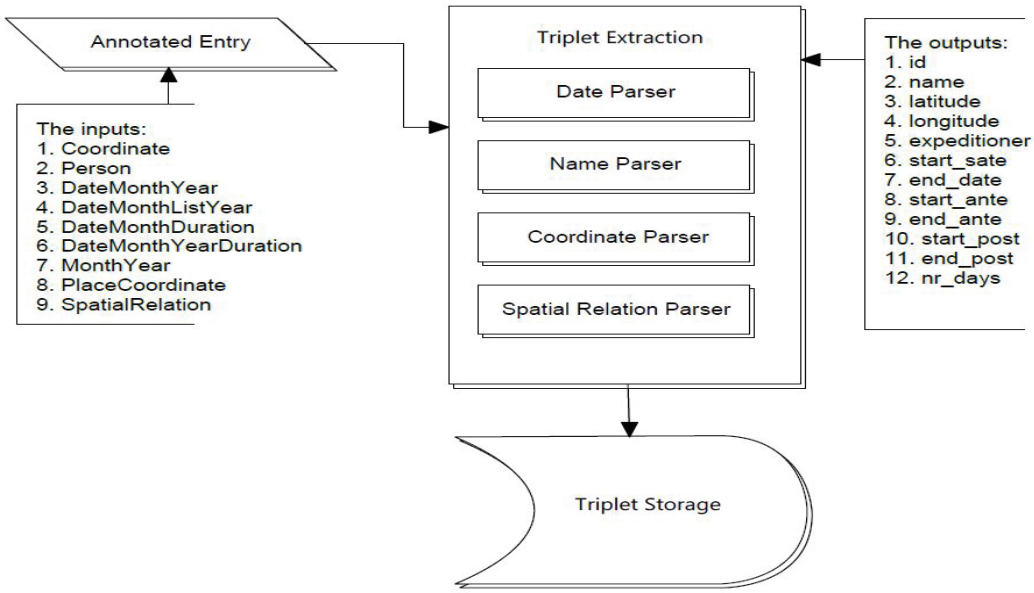


Figure 4.4: Conceptual Model of Triplet Extraction Process

4.4 TRAJECTORY TRIPLET EXTRACTION (TTE)

The Trajectory Triplet Extraction block is a component of TESS that parses and stores the triplet of a given location visit. The output from the Raw Data Extraction block is the input for the Contextual Information Extraction block. In Section 4.3 we discussed that the Contextual Information Extraction process consumes an XML file and recognizes and annotates entities in the consumed document. This block is a stepping stone between a location description and its trajectory, because this is the phase in which the aggregation and storage of the triplets takes place. The parsed triplets are automatically stored in a database.

The basic difference between the Contextual Information Extraction and the Trajectory Triplet Extraction blocks is the analytical capability attached to the last. The Contextual Information Extraction process is dependent on a JAPE Grammar rule and a Gazetteer list matching. Moreover, its task is only to annotate entities with positive response to pattern-based rules and a Gazetteer list matching. On the other hand, the Trajectory Triplet Extraction block parses the annotated entities, applies further analysis to extract targeted attributes and stores the triplets into a specified database. The XML document (XML. 4.2) is a typical example of how an annotated location description looks; this document is the annotated version of the location descriptions in XML. 4.1.

XML Document 4.2: Annotated location description

```
<?xml version="1.0" encoding="windows-1252" ?>
<Expedition>
	<LocationVisit id="3059" name="FERNANDES PINHEIRO">
		<Coordinate>2525/5033</Coordinate>
		<Person>Chrostowski</Person>
		<DateMonthYear>21 Dec. 1910</DateMonthYear>,
		<DateMonthYear>4, 11 Jan. 1911</DateMonthYear>
		<Person>Chrostowski</Person>
		<PublicationYearPage>1912:498, 499</PublicationYearPage>
	</LocationVisit>

	<LocationVisit id="279" name="APUCARANA">
		<Coordinate>2447/5110</Coordinate>
		<Person>Chrostowski</Person>
		<MonthYear>Aug. 1922</MonthYear>
		<Person>Sztolcman</Person>
		<PublicationYearPage>1926:124</PublicationYearPage>
	</LocationVisit>
</Expedition>
```

In the Trajectory Triplet Extraction process, parsing and storing *Person* and *Coordinate* are relatively easy, but extracting the Temporal marks and coordinates from spatial relationships requires extensive text tokenization and processing. The annotated temporal entities are not always bounded by crisp start or end points. The temporal entity annotation classes, such as *DateMonthYear*, *DateMonthListYear*, *DateMonthDuration* and *DateMonthYearDuration* have crisp temporal boundaries, whereas the temporal entities annotated with the *MonthYear* annotation class have a vague temporal boundary. The Trajectory Triplet Extraction process is capable to parse and store both crisp and vague temporal entities. The process has two phases: triplet extraction and temporal inference phase. The temporal inference is intended for vague temporal entities. The attributes that are collected from each location description are discussed here:

1. *id*: an identifier of the description.
2. *name*: name of the place that the visit description belongs to.
3. *latitude*: latitude of the visit.
4. *longitude*: longitude of the visit.
5. *expeditioner*: the person who visited the mentioned location.
6. *start_date*: the start date of the visit.
7. *end_date*: the end date of the visit.
8. *start_ante*: the number of days that the expeditioner could have started the visit prior to the *start_date*.
9. *end_ante*: the number of days that the expeditioner could have ended the visit prior to the *end_date*.
10. *start_post*: the number of days that the expeditioner could have started the visit after the *start_date*.

11. *end_post*: the number of days that the expeditioner could have ended the visit after the *end_date*.

12. *nr_days*: the number of days that the visit took.

There are two types of triplets: those with crisp dates and those with vague dates, the first one being easy to aggregate, while the second one requires temporal inference after the extraction process (see Section 4.5). This block has four main methods: *parseTrajectoryTripletCoordinate()*, *parseTrajectoryTripletPlaceCoordinate()*, *parseTrajectoryTripletSpatialRelation()* and *aggregateTrajectoryTriplet()* (see Algorithm 2). The algorithm is discussed here:

Data: the XML Document that is created by the Raw Data Extraction tool. This document contains the location descriptions of a given expeditioner.

Process: the spatial, attributive and temporal entities that are annotated in the Contextual Information Extraction phase are parsed by their respective parsers. The extracted triplets and the related attributes such as *name*, *start_ante*, *end_ante*, *start_post*, *end_post* and *nr_days* are aggregated and stored by the *aggregateTrajectoryTriplet()* method.

Line 2: the XML Document (annotated location descriptions) is parsed and each of the parsed elements are assigned to the $XMLelement$ variable as a set of nodes.

Line 3: the triples extraction process checks for every node that one of the spatial entity annotation classes (*Coordinate*, *PlaceCoordinate* or *SpatialRelation*) exists and calls one of the *parseTrajectoryTripletCoordinate()*, *parseTrajectoryTripletPlaceCoordinate()* or *parseTrajectoryTripletSpatialRelation()* methods.

Line 4 – 10: checks if the a tag name 'Coordinate' exits in the node and calls the *parseTrajectoryTripletCoordinate()* method if the outcome is positive.

Line 11 – 17: checks if the a tag name 'PlaceCoordinate' exits in the node and calls the *parseTrajectoryTripletPlaceCoordinate()* method if the outcome is positive.

Line 18 – 24: checks if the a tag name 'SpatialRelation' exits in the node and calls the *parseTrajectoryTripletSpatialRelation()* method if the outcome is positive.

Result: set of triplets (expeditioner name, location, time) with additional attributes such as *name*, *start_ante*, *end_ante*, *start_post*, *end_post* and *nr_days*.

The *parseTrajectoryTripletCoordinate()* method parses those triplets tagged with *Coordinate* annotation class; the *parseTrajectoryTripletPlaceCoordinate()* method parses those tagged with *PlaceCoordinate*; the *parseTrajectoryTripletSpatialRelation()* method parses those with a *SpatialRelation* annotation class while the *aggregateTrajectoryTriplet()* method aggregates and stores the extracted triplets. As Figure 4.4 shows, the triplet extraction process has four parsers; *Date Parser*, *Name Parser*, *Coordinate Parser* and *Spatial Relation Parser* each of which are dedicated to the elements of a trajectory triplet (Coordinate, Person and Time). The *Date Parser* compiles the *start_date, end_date, start_ante, end_ante, start_post, end_post* and the duration (*nr_days*) of the location visit. The semantics and constraints of these attributes are discussed in sections 4.4.1 and 4.4.2. The *Coordinate Parser* and *Spatial Relation Parser* compile the *latitude/longitude* while the *Name Parser* compiles the *description id*, *place name* and *expeditioner name*.

Algorithm 2: Trajectory triplets extraction

Data: $R =$ XML document of a set of annotated location descriptions
Result: Set $\{T_1, ..., T_n\}$ such that $T_1, ..., T_n = (e, l, t)$;
where $e = expeditioner$, $l = location$ and $t = time$

```
1  begin
2      XMLDocument ⟵ XMLDocument(R);
3      for XMLelement ∈ XMLDocument do
4          if XMLelement.getElementsByTagName() = Coordinate then
5              element ⟵ parseTrajectoryTripletCoordinate(XMLelement);
6              e ⟵ element.nameParser();
7              l ⟵ element.coordinateParser();
8              t ⟵ element.dateParser();
9              T ⟵ aggregateTrajectoryTriplet(e, l, t)
10         end
11         if XMLelement.getElementsByTagName() = PlaceCoordinate then
12             element ⟵ parseTrajectoryTripletPlaceCoordinate(XMLelement);
13             e ⟵ element.nameParser();
14             l ⟵ element.coordinateParser();
15             t ⟵ element.dateParser();
16             T ⟵ aggregateTrajectoryTriplet(e, l, t)
17         end
18         if XMLelement.getElementsByTagName() = SpatialRelation then
19             element ⟵ parseTrajectoryTripletSpatialRelation(XMLelement);
20             e ⟵ element.nameParser();
21             l ⟵ element.spatialRelationParser();
22             t ⟵ element.dateParser();
23             T ⟵ aggregateTrajectoryTriplet(e, l, t)
24         end
25     end
26 end
```

4.4.1 Triplets with crisp dates

The *DateMonthYear, DateMonthListYear, DateMonthDuration and DateMonthYearDuration* annotation classes always hold crisp temporal entities. The temporal dimension of a triplet that is parsed from a location description with these annotation classes is always crisp. The temporal boundaries (*start date* and *end date*) are explicit, meaning that either the start date or both the start and end dates of a location visit are provided; for instance *14 jan 1988*. Therefore, all *start ante, end ante, start post and end post* attributes are restricted to a value `0`. After the parsing, all the compiled attributes are stored as a row in a database table. For example, given the annotated location descriptions in XML. 4.2, the records in Table 4.4 are the extracted triplets.

Constraints and rules on the attributes of triplets with crisp dates

1. *latitude, longitude* and *expeditioner*: these attributes can not be null, always the respective parser should return non-null values.

2. *start_date and end_date*: the *start_date* and *end_date* can not be null. $start\_date \leq end\_date$.

3. *start_ante*, *end_ante*, *start_post* and *end_post*: these attributes are constrained to a value 0, because in triplets with crisp date case, the temporal boundaries of a location visit are explicit.

4. *nr_days*: this attribute is constrained to a value ≥ 1.

   If $start\_date = end\_date$; then $no\_days = 1$

   If $start\_date < end\_date$; then $no\_days = (end\_date - start\_date + 1)$

4.4.2 Triplets with vague dates

The *MonthYear* annotation class always represents vague temporal entity. The temporal boundaries are not crisp. Unlike the crisp temporal boundary case, the *start ante*, *end ante*, *start post* and *end post* attributes are restricted to a value which is equal to or higher than 0. The temporal entity annotation class *MonthYear* is bounded by implicit start and end points, for instance, an expeditioner who visited a specified location in January 1922 could have started and ended the visit at any time between *1 January 1922* and *31 January 1922*. Therefore, *start_ante*, *end_ante*, *start_post* and *end_post* attributes are not necessarily 0. While parsing the vague temporal entities, the Date Parser assigns default values:

1. *start_date*: the first date of the month which is 1 for all months.

2. *end_date*: the last date of the annotated month.

3. *start_ante* and *end_post*: since the *start_date* and *end_date* are assigned the first and the last dates of the month, both the *start_ante* and *end_post* are constrained to 0.

4. *start_post* and *end_ante*: since the *start_date* and *end_date* are assigned the first and the last dates of the month, both the *start_post* and *end_ante* are constrained to a value d; where $d = (end\_date - start\_date)$.

After the parsing process is completed, the Temporal Inference (see Section 4.5) process infers the *inferred_start_date*, *inferred_end_date*, *inferred_start_ante*, *inferred_end_ante*, *inferred_star_post* and *inferred_end_post* attribute relative to other chronologically close location visits of the same expeditioner. These attributes are defined in Section 4.5. For example, given the annotated location description in XML. 4.2, the extracted triplet of a vague date looks the one in Table 4.5.

Constraints and rules on the attributes of triplets with vague dates

1. *latitude, longitude and expeditioner*: these attributes can not be null, always the respective parsers should return non-null values.

2. *start date* and *end date*: both the *start date* and *end date* can not be null. Given the *MonthYear* annotation class, the followings hold true.

   $start\_date \leq end\_date$

   $start\_date = first\ day\ of\ a\ given\ month$

   $end\_date = last\ day\ of\ a\ given\ month$

3. *start_ante* and *end_post*: these attributes are constrained to a value 0, because the *start_date* and the *end_date* are assumed to be the first and the last day of a given month, respectively.

Table 4.4 Extracted Triplets (Crisp dates)

id	name	lat	lon	expe	start_date	end_date	start_ante	end_ante	start_post	end_post	days
3059	FER...	25.25	50.33	Chro...	1910-12-21	1910-12-21	0	0	0	0	1
3059	FER...	25.25	50.33	Chro...	1911-01-04	1911-01-04	0	0	0	0	1
3059	FER...	25.25	50.33	Chro...	1911-01-11	1911-01-11	0	0	0	0	1

Table 4.5 Extracted Triplets (Vague dates)

id	name	lat	lon	expe	start_date	end_date	start_ante	end_ante	start_post	end_post	days
279	APU...	24.47	51.1	Chro...	1922-08-01	1922-08-31	0	30	30	0	31

4. *end_ante* and *start_post*: these attributes are always a value d; where $d = (end\_date - start\_date)$.

5. *nr_days*: is constrained to a sum of the days of a given month.

4.5 TEMPORAL INFERENCE

Temporal inference, in this project, is a process of inferring and interpolating a relative temporal boundary. It creates different temporal scenarios for a given location visit (the temporal entity being vague). The Temporal Inference block, as shown in Figure 4.5, communicates with the database in two ways (it fetches and stores data). This process narrows a vague temporal boundary and determines its probability for a location visit to occur in the inferred time frame. If an expeditioner visited two different locations; location X with a crisp temporal boundary (25 – 31 January 1988) and, location Y with a vague temporal boundary (January 1988), the second visit must have been started and ended between 1 January 1988 and 25 January 1988. The default start and end dates assigned by the Date Parser for January 1988 are 1/1/1988 and 1/31/1988 respectively, but after temporal inference and reasoning, the start and end dates are inferred to be 1/1/1988 and 1/25/1988 respectively, because the expeditioner has already visited another location from 1/25/1988 to 1/31/1988.

For all of the extracted and stored vague dates, assuming there are chronologically close crisp dates, the inference process infers the attributes discussed here (The *id*, *name*, *coordinate* and the *expeditioner name* remain the same):

1. *inferred_start_date*: the start date of the location visit.

2. *inferred_end_date*: the end date of the location visit.

3. *inferred_start_ante*: the number of days that the expeditioner could have started the location visit prior to the *inferred_start_date*.

4. *inferred_end_ante*: the number of days that the expeditioner could have ended the location visit prior to the *inferred_end_date*.

5. *inferred_start_post*: the number of days that the expeditioner could have started the location visit after the *inferred_start_date*.

6. *inferred_end_post*: the number of days that the expeditioner could have ended the location visit after the *inferred_end_date*.

7. *inferred_nr_days*: the number of days that the location visit took place.

8. *probability*: the probability of a location visit to occur in the inferred temporal boundary.

The Temporal inference process has three basic steps (see Algorithm 3). These steps are dicussed here in detail.

Data: the extracted triplets with vague timestamps and triplets with crisp timestamps (of the same expeditioner) upon which the relative temporal boundaries for the temporally vague triplets are inferred.

Process: the inference process discussed here is applicable only for the vague timestamps that are captured with the *MonthYear* annotation class.

Step 1: finds a parsed and stored vague date.

Step 2: finds crisp dates; the crisp dates are constrained to be of the same expeditioner, same month and year with the vague date in Step 1.

Step 3: infers relative temporal boundaries and determine their probability of occurrence for those vague dates in Step 1 relative to those crisp dates in Step 2.

Line 8 – 14: let the temporally vague triplet be VT and temporally crisp triplet be CT. If the month and year of the VT equals that of the CT, for every given VT a minimum of one or maximum of two temporal boundaries are inferred. If the given CT starts at the first day of the month or ends at the last day of the month, only one temporal boundary is inferred. If the given CT starts and ends between the first and last days of the month, two temporal boundaries are inferred. Give the VT and CT, the followings hold true. The last day of the specific month being analyzed must be taken into consideration while conducting the inference process, for the following illustrations 31 was taken as the last day of the month.

1. *If* $start\_date\ of\_CT > 1$ and $end\_date\ of\_CT < 31$
   $inferred\_start\_date\ of\ VT_1 = start\_date\ of\ VT$
   $inferred\_end\_date\ of\ VT_1 = start\_date\ of\ CT$
   $inferred\_start\_date\ of\ VT_2 = end\_date\ of\ CT$
   $inferred\_end\_date\ of\ VT_2 = end\_date\ of\ VT$
2. *If* $start\_date\ of\_CT > 1$ and $end\_date\ of\_CT = 31$
   $inferred\_start\_date\ of\ VT_1 = start\_date\ of\ VT$
   $inferred\_end\_date\ of\ VT_1 = start\_date\ of\ CT$
3. *If* $start\_date\ of\_CT = 1$ and $end\_date\ of\_CT < 31$
   $inferred\_start\_date\ of\ VT_1 = end\_date\ of\ CT$
   $inferred\_end\_date\ of\ VT_1 = end\_date\ of\ VT$

Result: the result of this algorithm is a set of triplets with inferred temporal boundaries. The inferred temporal boundaries are aggregated with the attributes that are computed on the basis of the inferred attributes and stored in the database.

Algorithm 3: Temporal inference

Data: CT (triplets with crisp dates) = Set $\{T_1, ..., T_n\}$ such that $T_1, ..., T_n = (e, l, sd, ed)$;
where $e = expeditioner$, $l = location$, $sd = start\_date$ and $ed = end\_date$

Data: VT (triplet with vague date) = (e, l, sd, ed);
where $e = expeditioner$, $l = location$, $sd = start\_date$ and $ed = end\_date$

Result: IT (triplet with inferred date) = Set $\{T_1, ..., T_m\}$ such that
$T_1, ..., T_n = (e, l, isd, ied)$;
where $e = expeditioner$, $l = location$, $isd = inferred\_start\_date$ and
$ied = inferred\_end\_date$

begin
  $expeditioner \longleftarrow VT.expeditioner$;
  $location \longleftarrow VT.location$;
  $vague\_triplet \longleftarrow VT$;
  $crisp\_triplets \longleftarrow CT$;
  $inferred\_triplet \longleftarrow NULL$;
  for $c \in crisp\_triplets$ **do**
    if $c.sd.month = vague\_triplet.sd.month$ **and** $c.sd.year = vague\_triplet.sd.year$ **then**
      $inferred\_triplet_1.isd \longleftarrow vague\_triplet.sd$;
      $inferred\_triplet_1.ied \longleftarrow c.sd$;
      $inferred\_triplet_2.isd \longleftarrow c.ed$;
      $inferred\_triplet_2.ied \longleftarrow vague\_triplet.ed$;
      $probability(visit, inferred\_time) = \frac{1}{number\ of\ inferred\ temporal\ boundaries}$
    end
    `// Add all the inferred start date and end date along with the expeditioner name and location to the IT set.`
  end
end

If a reference crisp date does not exist for a given vague date, the inference process may not be successful and the default vague temporal boundary remains as the only option. It has three main methods; (1) *findVagueDate()*, *tripletWithCrispDate()* and (2) *inferVagueDate()* (see Algorithm 3). The first method finds the vague dates from a database, in which the extracted triplets are stored, and pass them to the second method. The second method finds the reference crisp dates for each of the received vague dates and the third method infers all the related attributes. All the inferred attributes along with the *id*, *name*, *coordinate* and the *expeditioner name* are sent back the database, but as a different data model (designed to capture a triplet with inferred dates).

One of the expeditioners, mentioned in the Brazilian Ornithological Gazetteer, Chrostowski visited a location at position *2603/5038*. According to the extracted information, the visit was undertaken in February 1922, but the start and the dates of the visit were not explicitly provided. However, the temporal inference tool was able to come up with different temporal scenarios for this visit that may have occurred. Table 4.7 shows the records of the triplets of different location visits that were visited by the expeditioner in February 1922. Considering the crisp dates as a reference, the temporal inference tool has come up with different temporal boundaries, as shown in Table 4.6, each of which have equal probability for the visit.

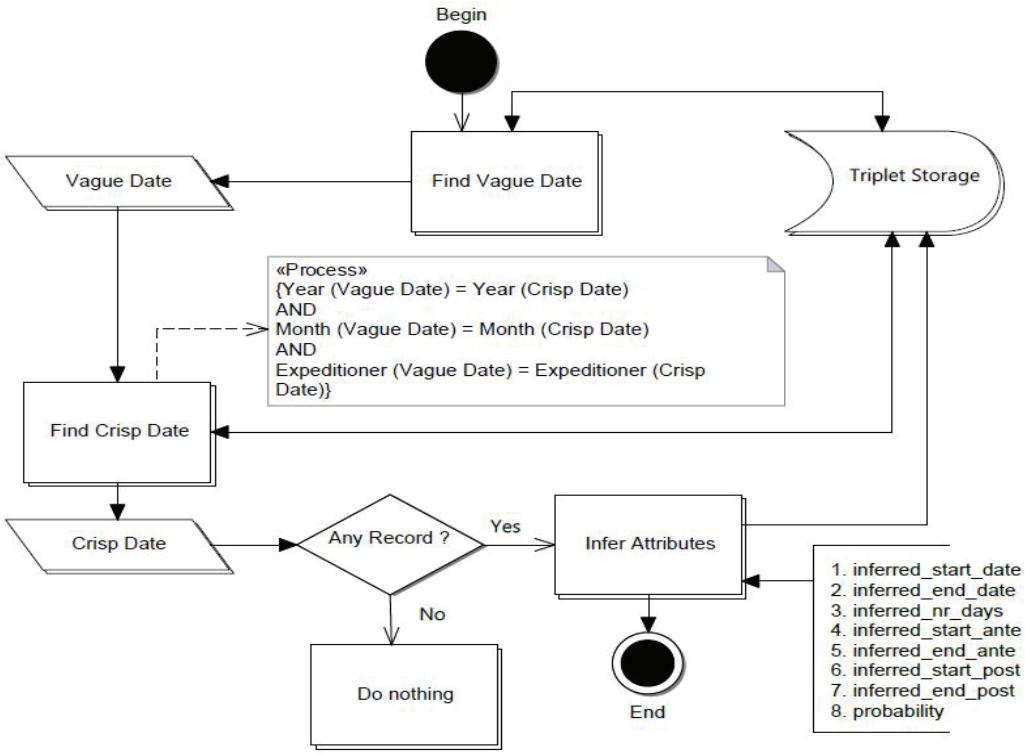


Figure 4.5: The Temporal Inference Process

Table 4.6 Triplets with inferred dates

inferred start date	inferred end date	inferred start ante	Inferred end ante	inferred start post	inferred end post	probability	days
1922-02-01	1922-02-03	0	2	2	0	0.33	3
1922-02-14	1922-02-15	0	1	1	0	0.33	2
1922-02-28	1922-02-28	0	0	0	0	0.33	1

Table 4.7 Reference Triplets used for temporal inference

No	lat	lon	Expe..	start date	end date	days	start ante	end ante	start post	end post
1	25.43	51.17	Chro..	1922-02-15	1922-02-28	14	0	0	0	0
2	25.43	51.17	Chro..	1922-02-03	1922-02-14	12	0	0	0	0

Constraints and rules on the attributes of triplets with inferred dates

1. *latitude, longitude* and *expeditioner*: these attributes can not be null, they are always equal with their respective attributes of the triplet with a vague date.

2. *inferred_start_date* and *inferred_end_date*: both can not be null. The *inferred_start_date* is always $\leq$ the *inferred_end_date*.

3. *inferred_start_ante* and *inferred_end_post*: these attributes are constrained to a value `0`, because the *inferred_start_date* and the *inferred_en_date* are inferred from two consecutive crisp dates. Given two chronologically ordered location visits (with crisp dates) for another visit with vague date, the *inferred_start_date* is the *end_date* of the preceding visit and the *inferred_end_date* is the *start_date* of the proceeding visit. Therefore, the *inferred_start_ante* and *inferred_end_post* are restricted to `0`.

4. *inferred_end_ante* and *inferred_start_post*: these attributes are always equal. Both are always a value d; where $d = (inferred\_end\_date - inferred\_start\_date)$.

5. *inferred_nr_days*: this attribute is constrained to a value ≥ 1.

   If $inferred\_start\_date = inferred\_end\_date$
   then $inferred\_no\_days = 1$

   If $inferred\_start\_date < inferred\_end\_date$
   then $inferred\_no\_days = (inferred\_end\_date - inferred\_start\_date + 1)$

6. *probability*: the probability of a location visit to occur in the inferred temporal boundary is always the same amongst inferred dates of a given location visit (see Equation 4.1).

$$probability(visit, inferred\_time) = \frac{1}{number\ of\ inferred\ temporal\ boundaries} \tag{4.1}$$

4.6 EXPEDITION TRAJECTORY STORAGE

A database was designed and implemented in Postgresql. Figure 4.6 shows the data model. The designed database stores the elements of a trajectory triplet (coordinate, person and time) in a table and enables tracking of the records amongst relations by a location description and expedition identifiers. The location description identifiers are assigned for all extracted triplets during the extraction process. The JDBC API is used as a bridge between the extraction and the storage process. It enable the automation of storing extracted triplets.

It is quite possible that a single location description describes visits of multiple expeditioners. This requires a data model that captures the triplet elements in a separate relation and create a trajectory on demand. The relation that keeps records of the extracted locations, stores a point geometry that is created from the extracted latitude and longitude. This enables, if required, further spatial analysis and manipulation of the extracted information. The designed database and its relations are discussed here:

1. *Expeditioner*: stores an expeditioner name along with the undertaken expeditions. The extracted triplets are categorized into different expedition groups on the basis of Expedition route production process (see Section 4.7).

2. *Location*: stores a location (as a point geometry)

3. *ExpeditionTrajectory*: stores a trajectory as a LineString geometry.

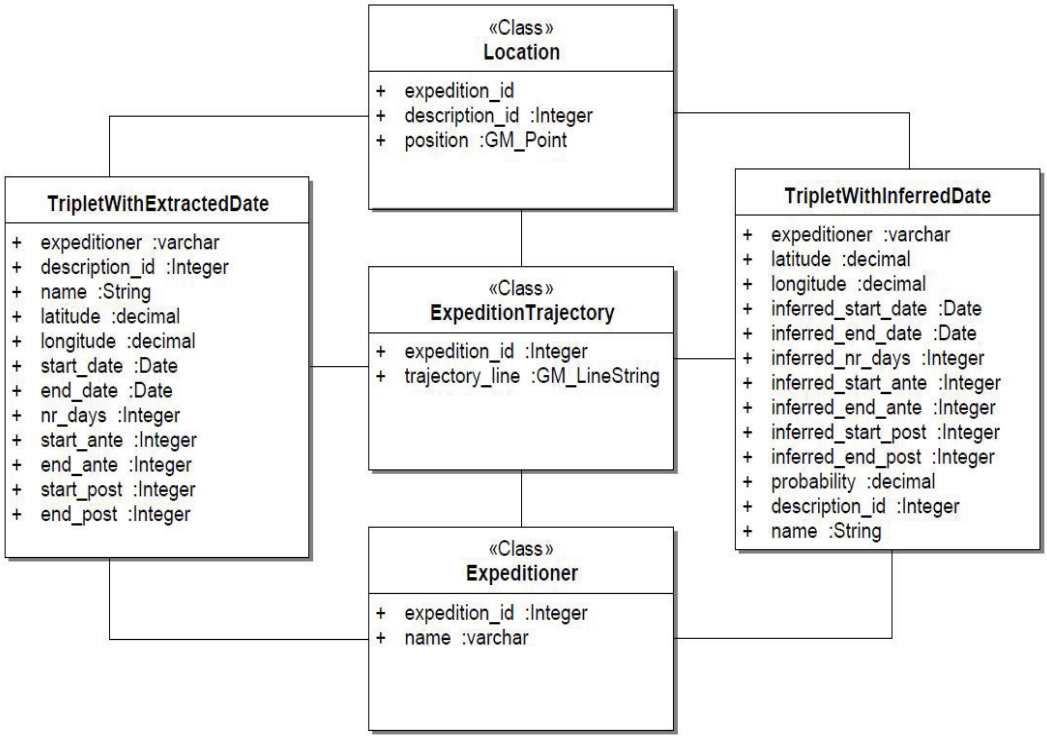


Figure 4.6: Data model for expedition trajectories

4. *TripletWithExtractedDate* : stores triplets with the extracted attributes.

5. *TripletWithInferredDate*: stores triplets with the inferred attributes.

The Expedition Trajectory Storage phase has one function, namely, *storeExpeditionTrajectory()* that produces expedition routes and stores them in the database. This method uses the *cutExpedition()* method of the Expedition Route Production phase to categorize the extracted triplets into a number of expedition groups. The *storeExpeditionTrajectory()* method brings in the triplets from the *TripletWithExtractedDate* and *TripletWithInferredDate* tables and applies the following computations.

populate the *Location* table: the *storeExpeditionTrajectory()* method populates the *Location* table with the extracted data. Each row in this table contains: (1) the point geometry that is created from the latitude and longitude columns of the *TripletWithExtractedDate* table; (2) the location description id of the text from which the latitude and longitude values that are used to produce the point geometry are extracted from and (3) the expedition id which is given by the *storeExpeditionTrajectory()* method on the fly.

populate the *ExpeditionTrajectory* and *Expeditioner* tables: the *storeExpeditionTrajectory()* populates the *ExpeditionTrajectory* and *Expeditioner* tables also. Each row in the *Expeditioner* table stores the expeditioner name and the expedition id that is given by the method. Each row of the *ExpeditionTrajectory* stores the expedition route as a LineString geometry and the expedition id that is given by the method on the fly. The LineString geometry is created from the point geometry of the *Location* table.

4.7 EXPEDITION ROUTE PRODUCTION

With all the location descriptions of a given expeditioner, the Contextual Information Extraction, Trajectory Triplet Extraction and Temporal Inference tools are adequate to parse and store the triplets of those location visits. After all the triplets of a given expeditioner are extracted and stored, a process follows to produce a trajectory line and collect all points of possible expeditions. The extracted triplets of a given expeditioner are grouped into a number of expeditions. The grouping depends on the detection a temporal gap between two location visits. The selection of the gap is preferable to be subjected to the considered expeditioner. The assumption behind constraining this gap is considering the minimum break that an expeditioner would take between two consecutive location visits. The TESS has three methods namely, *cutExpedition()*, *toKMLLine()* and *toKMLPoint()* to handle the expedition trajectory line and point production. The first method, *cutExpedition()*, finds boundary triplets between two expeditions for a given expeditioner based on a predefined temporal gap. The *cutExpedition()* method works as follows: given two triplets x and y, if the $temporal\ gap(x, y) = d$; where $temporal\ gap(x, y)$ is the number of days considered as the break between two location visits. The method goes through all the triplets of a given expeditioner and computes the $chronological\ difference(x, y)$ between consecutive triplets (see Equation 4.2). If the $chronological\ difference(x, y) \geq temporal\ gap(x, y)$; then x and y are the boundaries of two expeditions. In such a way all the boundary triplets are collected and sent to the *toKMLLine()* and *toKMLPoint()* methods to produce an expedition trajectory line and point.

The *toKMLLine()* and *toKMLPoint()* methods create Keyhole Markup Language (KML) files. These KML files are created from the stored triplets. At each moment the *cutExpedition()* method finds the boundary triplets, both the *toKMLLine()* and *toKMLPoint()* methods are called and KML point and line files are created. The created files are used as a means of visualizing an expedition; the following chapter discusses the visualization of expedition. The *toKMLLine()* creates a KML line geometry by connecting the extracted triplets of an expedition by their chronological order using the boundary triplets from *cutExpeditoin()*, while the *toKMLPoint()* creates a KML point geometry from the same triplets. Each of these KML points renders the *expeditioner name*, *place name*, *location description id*, the *start_date*, *end_date* and *nr_days* of a location visit.

4.8 QUANTITATIVE SPATIOTEMPORAL REASONING (QSTR)

QSTR is a way of testing that an extracted triplet meets all the constraints over space and time dimension. It allows to assess the reliability of a location visit and catch outliers in the spatiotemporal dimension. The underlying reasoning has the following steps:

Step 1 finds two chronologically consecutive location visit records and computes the chronological difference between them; this computation considers the end date of the first visit and the start date of the latter.

Step 2 reasons about the average required time to travel from the first point to the second; this is computed on the basis of the actual distance between the points in *Step 1* the average distance that an expeditioner could travel per a day

Step 3 compares the actual time and the computed average required time (*Step 1* and *Step 2*). Based on this comparison, a location visit can be argued to be an outlier, and be tagged as such.

The reasoning process is always dependent on constraints either in the space or time dimension or both. A constraint in space is the $average\ travel\ distance$ an expeditioner could travel per day, whereas the time constraint is an $average\ travel\ time$ for an expeditioner to travel a given distance.

However, the constraint in space has been used for our reasoning tool, meaning the tool requires the $average\ travel\ distance$ an expeditioner could travel per a day to assess the reliability of a given location visit.

Given the $average\ travel\ distance$ and two chronologically consecutive location visits x and y. If $start\_date(y) > end\_date(x)$, equations 4.2, 4.3 and 4.4 must hold true to say location visit y is reliable.

$$chronological\ difference(x, y) = start\_date(y) - end\_date(x) \tag{4.2}$$

$$required\ time(x, y) = \frac{distance(x, y)}{average\ travel\ distance} \tag{4.3}$$

$$required\ time(x, y) \leq chronological\ difference(x, y) \tag{4.4}$$

The QSTR has a automated method, namely, *spatioTemporalReasoning()* that assesses the reliability of the produced expedition route and creates a KML point and line files: the point file represents the outliers with different color whereas the line file represents the modified version of the assessed expedition route. The modified route excludes the outlier points and connects the remaining point with their chronological orders. This method requires the *start_date*, *end_date*, *expeditioner name* and $average\ travel\ distance$ values to be provided.

Chapter 5
Visualization and Reliability Assessment

5.1 INTRODUCTION

This chapter discusses the visualization and reliability assessment of the extracted triplets and the produced expedition routes. The preceding chapters discussed the procedures and computational requirements of the trajectory triplet extraction and storage process. In Section 5.2 the KML point and line files produced by the Expedition Route Production tool are used to visualize the produced expedition routes. We use Google Earth as a visualization tool. Section 5.3 discusses the reliability assessment of the produced expedition route. We conduct our reliability assessment in two ways: (1) Visual inspection and (2) QSTR. The assessment of an expedition route with QSTR depends on the QSTR tool. The assessment of an expedition trajectory with visual inspection compares two expedition routes visually; these routes must be of the same period and expeditioner and one of the two needs to be a reference from an external source while the second needs to be TESS-produced. An alternative to comparing the produced trajectory against a reference route is inspecting the geometrical orientation and the spatial situation of the produced expedition route visually.

5.2 EXPEDITION TRAJECTORY VISUALIZATION

Many expeditioners are mentioned in the Gazetteer that we used throughout this research project. To mention some, *Tadeusz Chrostowski (1878 – 1923)*; *Emil Heinrich Snethlage (1897 – 1939)* and *Maria Emilie Snethlage (1868 – 1929)* are among those many. The produced expedition routes of these expeditioners are illustrated in the following sections.

5.2.1 Expeditioner: Tadeusz Chrostowski

According to Wikipedia, Chrostowski conducted his first expedition in the year 1910 in Brazil by the River Iguaçu and returned to Poland in 1911; his second expedition ran from 1913 to 1915 and he returned to Poland in 1915 due to the news of the outbreak of World War I. In [20] it is mentioned that Chrostowski conducted his third expedition from 1921 to 1923. Therefore, it can be concluded that Tadeusz Chrostowski conducted three expeditions: Expedition I (1910 – 1911), Expedition II (1914 – 1915) and Expedition III (1921 – 1923). However, our system TESS produces six expeditions for the same expeditioner (see Table 5.1) with a *temporal gap* of two months between two consecutive expeditions.

The Expedition Route Production process is subject to the choice of the *temporal gap* between two consecutive expedition routes. Such is the reason behind having unequal number of produced expeditions as compared to those referred in the literature. Base on a careful inspection of the expedition routes of both sides, some expeditions could be merged conveniently. Looking at the expeditions listed in Table 5.1, one can argue that the chronological closeness of some expeditions could lead to an expedition aggregation. The six expeditions of Chrostowski produced by our system (TESS) could have been three expeditions if the *temporal gap* were set to be five month. As Table 5.1 shows, Expedition II, III and IV and Expedition V and VI are chronologically close.

Table 5.1 Expeditions of Chrostowski as produced by TESS

Expedition	Expeditioner	Start Date	End Date	No. of triplets
I	Chrostowski	1910-05-26	1911-08-26	64
II	Chrostowski	1914-01-22	1914-07-13	25
III	Chrostowski	1914-09-25	1914-12-02	7
IV	Chrostowski	1915-02-10	1915-02-10	1
V	Chrostowski	1921-08-01	1921-08-31	1
VI	Chrostowski	1922-01-01	1923-05-05	38

Based on the chronological closeness, the following aggregations are suggested to downsize the produced expedition routes:

Case 1: looking at the expeditions in Table 5.1, it can be observed that Expedition I is chronologically far from the other expeditions. To inspect the closeness with a quantified time period, the gap between the end date of the first expedition and start date of the second is from 1911/08/26 to 1914/01/22, which is not convincing to think these expeditions could have been one as the temporal gap is more than two years. This gives us a reason to keep Expedition I as it was produced.

Case 2: the *temporal gap* between Expedition II and III is about two months (1914/07/13 – 1914/09/25) while the *temporal gap* between Expedition III and IV is also about two months (1914/12/02 – 1914/1202). Moreover, the number of location visits in Expedition III and IV are fewer than Expedition II. A typical expedition is expected to involve a considerable number of location visits. Hence, it might be less convincing that Expedition IV which has only one location visit is actually an expedition. It is may be also argued that expedition teams in those days needed time to stock up again before continuing field work. As a result of the stocking up days one expedition route might be produced as two by our system. This gives us a concrete reason to aggregate Expedition II, III and IV. Considering the number of location visits and the temporal gap among these expeditions, they could be conveniently produced as one if the *temporal gap* were set to two or three months. The chronological closeness among these three expeditions leads us to an expedition route aggregation. The aggregated expedition route, therefore, merges Expedition II, III and IV and it covers the period from 1914/01/22 to 1915/02/10.

Case 3: Expedition V and VI are chronologically apart from each other by four months; the *temporal gap* is from 1921/08/31 to 1922/01/01. Expedition V has only one triplet while Expedition VI has 38. It may not be feasible to keep Expedition V while the number of triplets is one, instead the triplets of both routes can be aggregated to produce one. The aggregation gives us an expedition that covers the time period from 1921/08/01 to 1923/05/05.

In Section 3.1.1 we discussed that 'Chrostowski' is mentioned in 58 location descriptions of the Brazilian Ornithological Gazetteer. Out of these descriptions, 136 triplets were extracted and stored in a database which was designed to store triplets (see Section 4.6). Each of these stored triplets represents a location visit with three attributes (location, expeditioner and time). However,

this does not necessarily mean that each of these triplets represents a unique location, for instance, the triplets in Table 4.4 are extracted from a single description, but they represent the same location with different timestamps. As a result, given an expedition, the count of triplets may be different from the count of the distinct locations that are visited.

Expedition I: 1910 – 1911

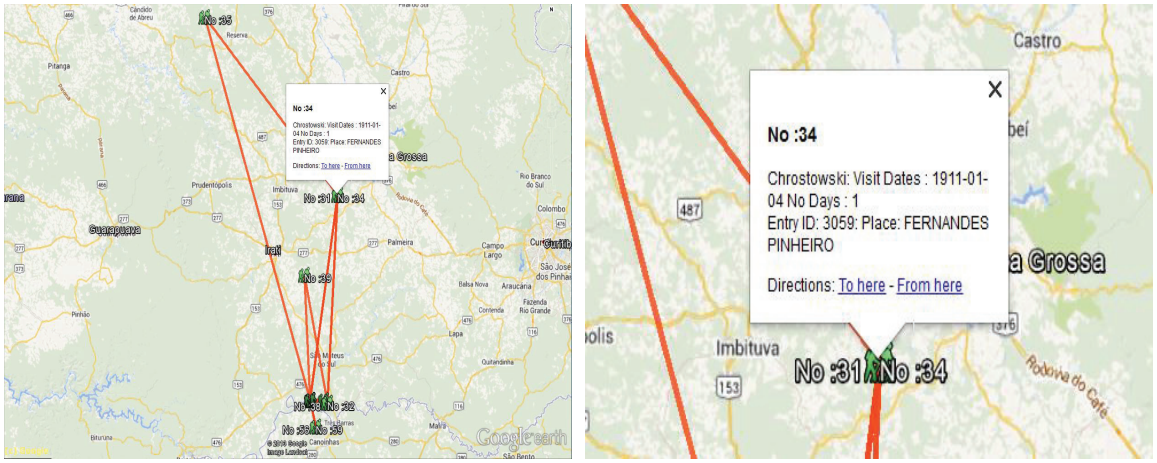
Tadeusz Chrostowski conducted his first expedition in the years 1910 and 1911. During this expedition he visited eight distinct locations. However, the number of triplets extracted for this expedition is 64 (see Table 5.1). There are two sound reasons behind the difference in count of triplets and distinct location visits: (1) a location description might have multiple timestamps while the location is unchanged, and (2) multiple visit descriptions might mention similar location. Figures 5.1a and 5.1b show the trajectory line and point of Chrostowski's first expedition. Table B.1 shows the extracted triplets of this expedition.

Expedition II: 1914 – 1915

During the period from 1914 to 1915, Chrostowski undertook his second expedition. According to the extracted information about this expedition he visited four distinct locations. Figure 5.2a shows a trajectory line connecting these four distinct locations. Since this expedition is aggregated from three expeditions, it has a total of 33 triplets. Here, it can be inferred that the expeditioner made a repetitive visit either to all or few of the distinct locations. Table B.2 shows the extracted triplets of this expedition.

Expedition III: 1921 – 1923

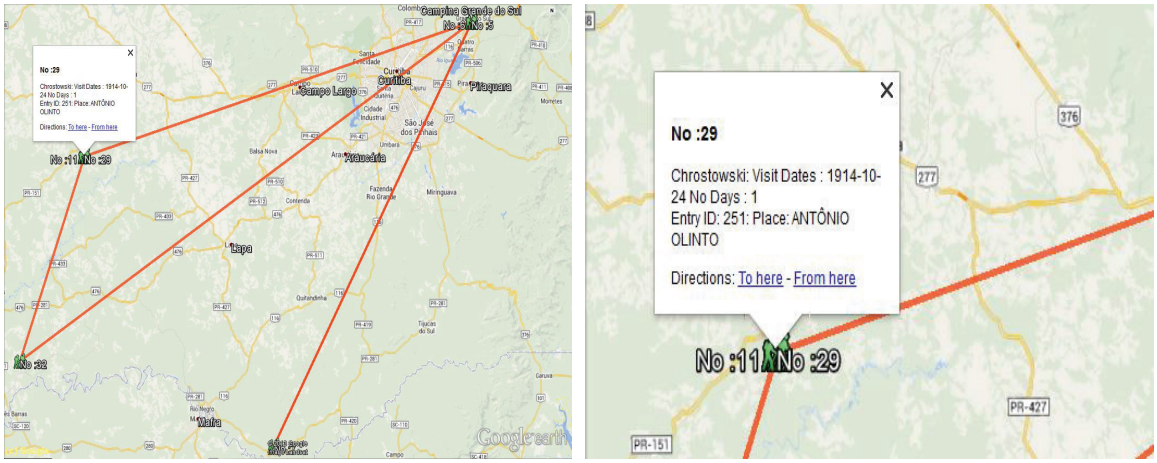
According to the extracted information from the location descriptions, Chrostowski made his longest expedition from 1921 to 1923. This expedition involve visits of 35 distinct locations while the count of triplets extracted are 39. Figure 5.3a shows the trajectory line of this expedition. This expedition is discussed in detail in Section 5.3. A reliability assessment based on the QSTR and visual inspection is conducted on this expedition in the reliability assessment section. Table B.3 shows the extracted triplets of this expedition.



(a) Expedition I Route (1910 – 1911)

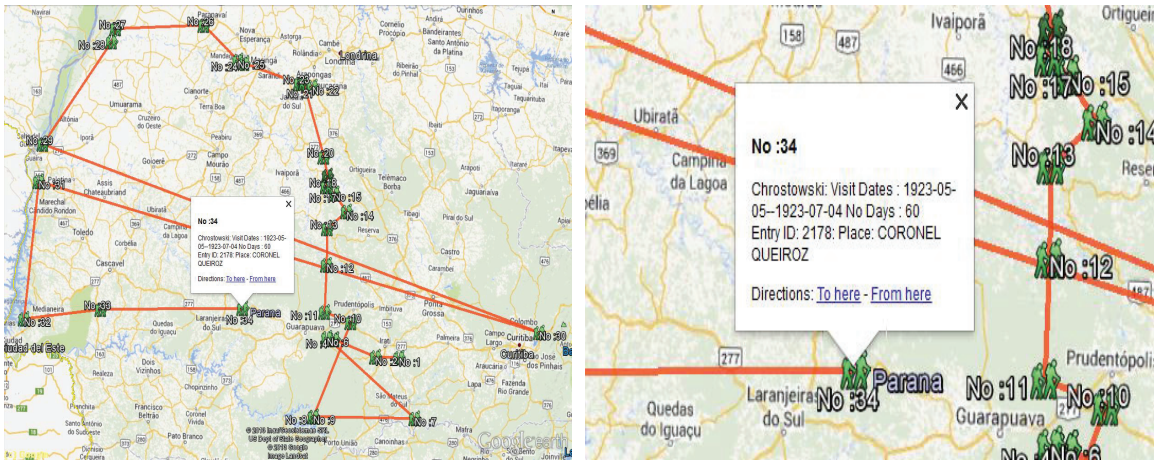
(b) Expedition I: sample triplet

Figure 5.1: Expedition route of Chrostowski: (1910 – 1911)



(a) Expedition II Route (1914 – 1915) (b) Expedition II: sample triplet

Figure 5.2: Expedition route of Chrostowski: (1914 – 1915)



(a) Expedition III Route (1921 – 1923) (b) Expedition III: sample triplet

Figure 5.3: Expedition route of Chrostowski: (1921 – 1923)

5.2.2 Expeditioners: Maria Emilie Snethlage and Emil Heinrich Snethlage

Maria Emilie Snethlage (1868 – 1929) was an ornithologist who undertook many expeditions since 1905 until her death. Emil Heinrich Snethlage (1897 – 1939) was a zoologist and ethnologist. Emil was the nephew of Maria who got influenced by his aunt's work and and which inspired him to travel with her. According to [18] Emil conducted his first expedition in the years 1923 – 1926 in association with his aunt. He started his expedition in *Maranhão* state, North Eastern Brazil. However, the two Snethlages did not travel the whole period together; from March 1924 to 1926 Emil traveled alone [18]. This brings an interesting assumption that Emil and Maria must have expedition routes that intersect someplace in the years 1923/24, since Emil traveled alone after March 1924 to 1926, the likely place to have met his aunt must be one of the places he visited in the years 1923/24. Based on the extracted information from their respective location descriptions, Maria visited the *Rio de Janeiro*, *Bahia*, *Minas Gerais* and *Espírito Santo* states in the years 1923 – 1926 while Emil traveled to places in *Maranhão*, *Ceara* and *Piaui* states.

One of the location visit descriptions (see Appendix B.4) mentioned that Maria and Emil visited a place around *"coast of northwestern Maranhao"* in the years 1923/24. This vary description gives a clue about the possible location visit that might be an intersection point between the expedition routes of these expeditioners. Another location description (see Apeendix B.3) mentioned that Emil visited *"northwestern of Maranhão"* in the years 1923/24. The attributive intersection between these two descriptions gives as a preliminary proof that *"northwestern of Maranhão"* is the place that these expeditioners could have traveled together. Having this assumption, we produced two expedition routes for each of these expeditioners.

The produced expedition routes are visually analyzed to support our assumption on the attributive and temporal intersection of the expeditions. Figure 5.4a and 5.4b show the expedition routes of Emil and Maria, Figure 5.5a shows these two expeditions together and Figure 5.5b shows the possible intersection point of these expedition routes. The expedition route of Maria runs from the southeastern of Brazil (*Rio de Janeiro*) to the northeastern of Brazil (*Maranhão*) and intersects with the expediton route of Emil and gets back to the southeastern of Brazil (*Espírito Santo*), see Figure 5.5a and 5.5b. From the going and backing of the expedition route of Maria, we might infer a story that Emil was welcomed by his aunt and after some period they traveled to the places of their interest. Here, we can think of two scenarios about the two expedition routes intersecting at starting location visit of Emil's expedition:

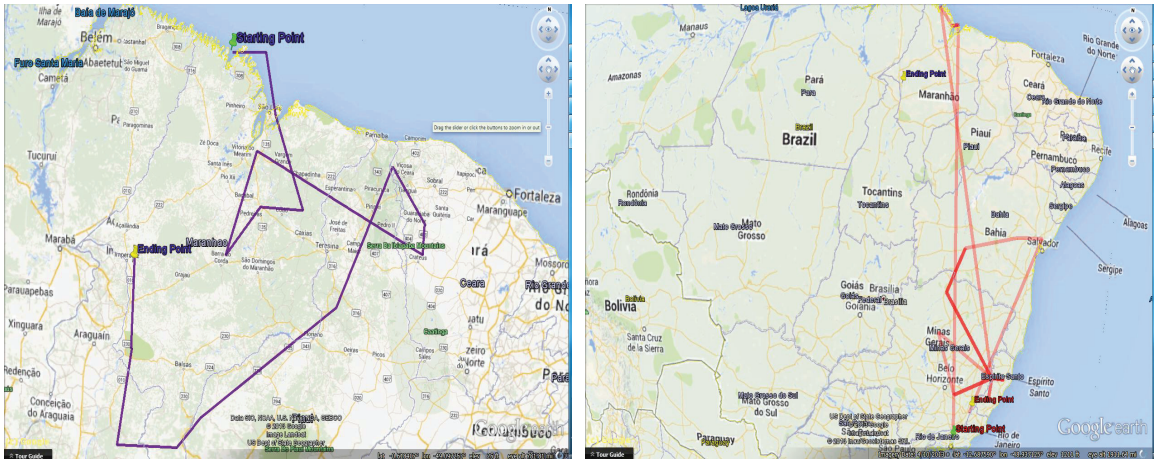
Scenario one: these two expeditioners visited the intersecting point at different time, and what we see in Figure 5.5a and 5.5b is only an intersection in space without any temporal proof that the expeditioners actually met at that intersecting point.

Scenario two: the expeditioners actually met at the intersecting point. Therefore, to either support or go against these two scenarios we need a temporal coincidence. Given the fact that these expeditioners met in 1923/24 and Emil traveled alone after March 1924 [18], we need at least two triplets at the intersecting point, one from each expedition route, with either identical or close temporal marks.

Figure 5.6a and 5.6b show two triplets one from each expedition. The temporal closeness between these triplets is in favor of *Scenario two*, which is a sound reason to believe that Maria and Emil actually met in *"northwestern of Maranhão"*. Since the intersection between the expedition routes of the expeditioners is in space and time, our story about the two expeditioners meeting in *"northwestern of Maranhão"* in the years 1923/24 is true and appears supported by a fact.

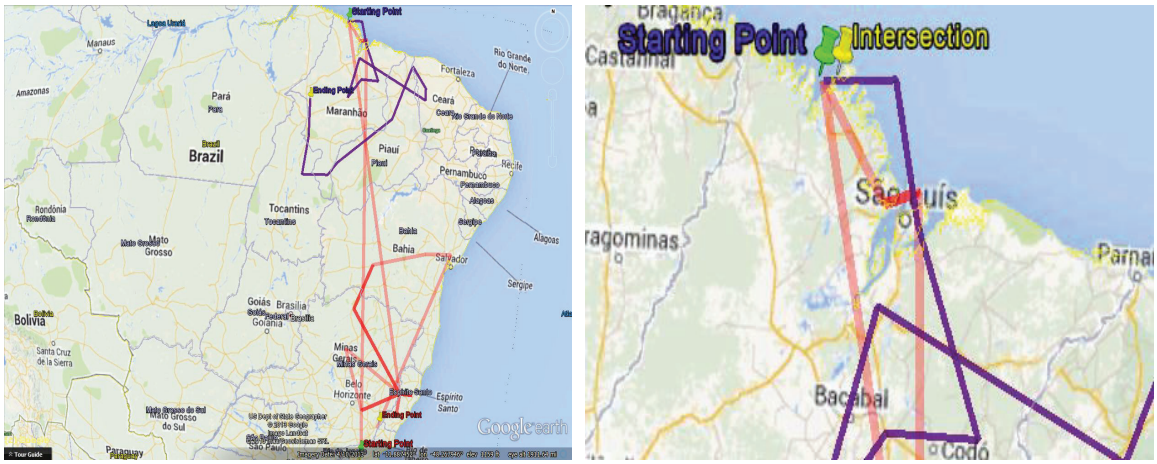
5.3 EXPEDITION TRAJECTORY RELIABILITY ASSESSMENT

To regard an expedition route as reliable, the respective extracted triplets must be assessed. Assessing such a reliability is conducted in two ways. We discussed in Section 1.3 that the reliability of a produced expedition route can be assessed either by visually inspecting its resemblance with an external source expedition or by testing it against the constrained values in space and time. In Section 5.2.1 we produced three expedition routes for Tadeusz Chrostowski. Since an expedition route from external source is required to assess the reliability by means of visual inspection, we peaked the third expedition of Chrostowski (Expedition III : 1921 – 1923), for there is a manually prepared route of this expedition by Jaczewski in 1925. The same expedition is assessed using the QSTR tool, which checks each pair of constitutive location visit triplets against the constrained values in space and time.



(a) Emil's Expediton Route (b) Maria's Expediton Route

Figure 5.4: Expedition of Maria and Emil (1923 – 1926)



(a) M.E. Snethlage and E.H. Snethlage (b) Intersection between expeditions

Figure 5.5: Maria's and Emil's expedition routes spatial intersection.

5.3.1 Reliability assessment using the QSTR

The third expedition route of Chrostowski (Expedition III: 1921 – 1923) is shown in Figure 5.3a. We used our QSTR tool to assess the reliability of this expedition route. When conducting the assessment, the *average travel distance* was constrained to 30 Km a day (see Section 4.8). After running the QSTR algorithm, the route through points 29, 30 and 31 was suggested as unreliable (see Figure 5.7a). Figure 5.7a shows a line that runs from northwestern edge to the southeastern and again runs back to northwestern edge of the Figure. This line starts at point 29, goes to point 30 and returns to point 31. The distance from point 29 to 30 is about 535 Km whereas the distance from point 30 to 31 is another 531 Km. The expeditioner visited these points as follows: point 29 (1923/01/23 – 1923/02/26), point 30 (1923/01/25) and point 31 (1923/02/27 – 1923/03/16); see Figure 5.7b, 5.8a and 5.8b, respectively. Assuming the average distance which could be covered is 30 Km a day, the expeditioner must have traveled for 18 days from point 29 to 30 and another 18 days from point 30 to 31. However, the story we see in the extracted information is that the

(a) Location visit by M.E. Snethlage (b) Location visit by E.H. Snethlage

Figure 5.6: Maria's and Emil's expedition routes temporal and attributive intersection.

expeditioner traveled through these points in a single day, which is very unlikely to have happened back in 1923. Therefore, we need to do some inspection on the route through these location visits.

Route one (from point 29 to 30): according to the extracted information the expeditioner stayed at point 29 from 1923/02/27 to 1923/03/16. These are crisp and explicit temporal boundaries specifying the period of a visit made to point 29. If any other location visit shares the same period with point 29, either of the two must have been wrong. The period of the visit at point 30 is 1923/01/25, which has a temporal intersection with point 29. This temporal intersection in addition to the traveled distance inconsistency, points us to a suggestive view on regarding either of the two points as an outlier.

Route two (from point 30 to 31): unlike the route through points 29 and 30, this route is free of any temporal intersection with either point 29 or 30. From the period of the stay made at point 31, 1923/02/27 – 1923/03/16, we can infer the chronological closeness between points 29 and 31; the expeditioner made his visit to point 31 immediately after point 29. Assuming this to hold true, traveling from point 29 to point 30 and then to point 31 is still doubtful, because even if the visit at point 30 and 31 was on the same day, it is impossible to have a total distance of 1066 Km traveled in a single day. Having these doubts on the extracted route via points 29, 30 and 31, the following scenarios are informative clues towards the raised reliability questions.

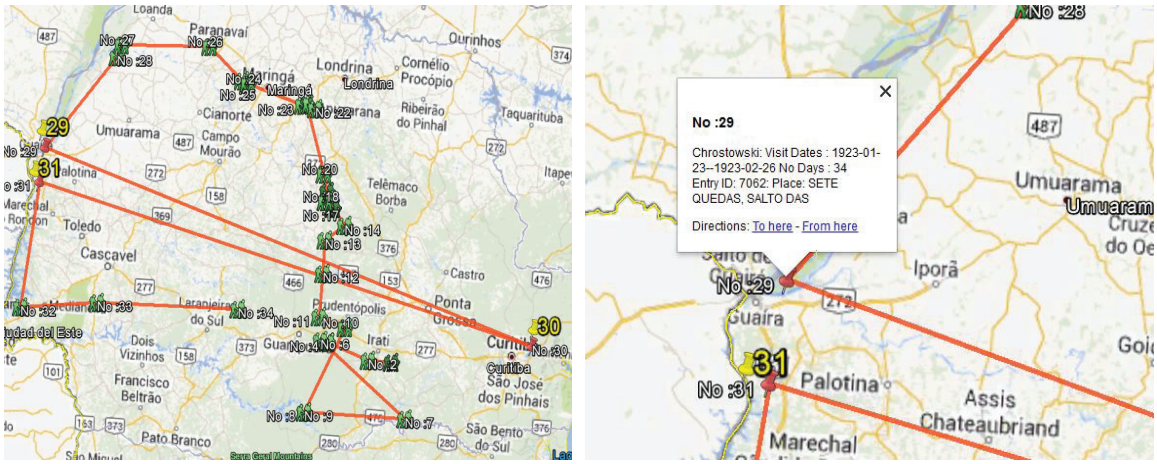
Scenario one: the reason behind the untrustworthiness of the route could be that the triplets are extracted from other expeditioner location visit description. During the Contextual Information Extraction and Trajectory Triplet Extraction process, the respective tools might have parsed wrong information about the expedition route under assessment.

Scenario two: the extracted information might actually be of the right expeditioner, but the description could be of another expedition, for instance, instead of 1923/01/25 the visit date might have be 1921/01/25.

Scenario three: the extracted information might be of the right time and expeditioner, but the problem could be the extracted location. In this case if point 30 was near to point 29 and 31, we could have believed that the route via this points is reliable.

If the reliability issue of the route under assessment is caused as mentioned in *scenario two*, the right solution seems to check if other expedition route has an attributive, temporal and spatial intersection at the very outlier point of the assessed expedition. Figure 5.9a shows the three expedition routes of Chrostowski and Figure 5.9b shows the intersection between Expedition III (red line) and Expedition II (blue line). Looking at the point descriptions of Figure 5.8a and 5.9b we can infer that these two points are extracted from identical location description and they share an identical location. Hence, the reason that point 30 of the assessed expedition route is an outlier must be either by extracting the triplets from wrong location description or having a misreported description. To support this claim we have to look at the description, from which the triplets are extracted. Appendix B.5 shows this visit description. According to this description, the outlier point is extracted correctly; *"25 Jan. 1923"* is of course there. In the same description there is a phrase that reads *".....although it was not mentioned by Chrostowski"*; here we have to be suspicious about the credibility of *"25 Jan. 1923"*. Therefore, the author of this description might have made an attributive misreporting. Assuming point 29 was completely an outlier and excluding it from the assessed expedition route, a modified expedition route that connects points 29 and 30 is produced. Figure 5.10a and 5.10b show the original and the modified expedition routes. This modified route is used in Section 5.3.2 to conduct a visual comparison with a reference expedition route of same period and expeditioner.

The method we developed in the QSTR phase (see Section 4.8), namely, *spatioTemporalReasoning()* was used to assess the reliability of the Chrostowski's third expedition route that was produced by the Expedition Route Production tool. The modified version of the produced route was reproduced by this automated reliability assessment tool.



(a) Unreliable Route (b) Triplet of the visit at point 29

Figure 5.7: QSTR based reliability assessment of Chrostowski's Expedition III: unreliable route and the attribute of the first outlier point

5.3.2 Reliability assessment using the visual Inspection

The modified version of the third expedition of Chrostowski is visually compared with a reference expedition route map. As mentioned in [20], the third expedition route map was produced by Jaczewski in 1925. We conducted a visual comparison between Jaczewski's map and the one produced by our system (TESS). The objective of this comparison is to visually confirm if the general geometry and spatial situation of these two expedition routes are similar. A geometrical

(a) Triplet of the visit at point 30 (b) Triplet of the visit at point 31

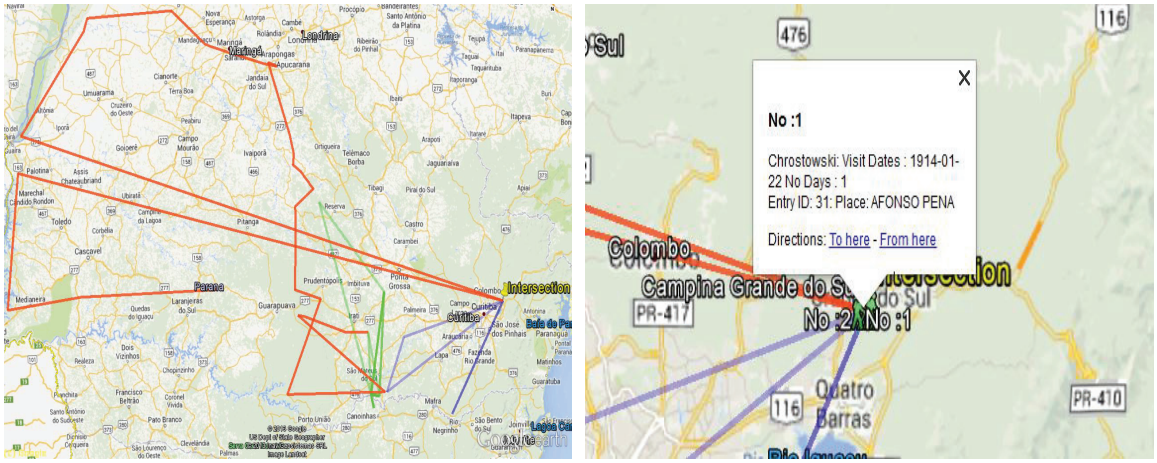
Figure 5.8: QSTR based reliability assessment of Chrostowski's Expedition III: the attribute of the second and the third outlier point

and spatial situation similarity is central to assessing the reliability of an expedition route through visual inspection. Figure 5.12a shows the reference expedition route map. The route in this map has been colored with a blue line to enhance visibility. Figure 5.10b shows the same expedition route as produced by our system (TESS).

A *Straight-lined route* is a route produced by connecting the location visit points with a straight line while a *Road-lined route* is a route produced by connecting the location visit points through currently available road network (as extant in 2014). The *Road-lined route* of Chrostowski's third expedition was manually produced using the Google Earth way finding tool. To assess the reliability of the produced expedition, we conducted a visual comparison between the *reference route* and the *Straight-lined* and *Road-lined* versions of the produced expedition route.

1. *Straight-lined route against reference route*: a straight-lined route connects two location visit points with a straight line. Figure 5.10b shows the *Straight-lined route* of Chrostowski's third expedition. Geometrically speaking this *Straight-lined route* and the reference route (see Figure 5.12a) have considerable similarity. Both of these routes have big irregularly shaped loops situated to the western part of their respective Figures and a small loop situated to the southeastern part of the Figures. The bigger loop in both of these expedition routes is bordered by *Matto Grasso* in the northwest, *Paraguay* in the most part of southwest and *Argentina* in the southwest; whereas the smaller loop is bordered by *Santa Catarina* in the southeast. Based on the geometrical and spatial situation similarity of the *Straight-lined route* and the *reference route*, the expedition route produced by our system is regarded as reliable.

2. *Road-lined route against reference route*: even if the produced expedition route is regarded as reliable, comparing the *Road-lined* version of the same expedition against the *reference route* increases the credibility of the produced route. Figure 5.12b shows the *Road-lined* version of Chrostowski's third expedition. As this very figure shows the spatial situation of the *Road-lined route* is still similar with both of the *reference route* and the *straight-lined route*. However, the shape is more close to the *reference route* than the *straight-lined route* is to the *reference route*. The similarity of the geometry between the *Road-lined route* and the *reference route*, in addition to the spatial situation, gives us a concrete reason to confirm the reliability of the produced route, conveniently. Figures 5.12b and 5.12a show that the

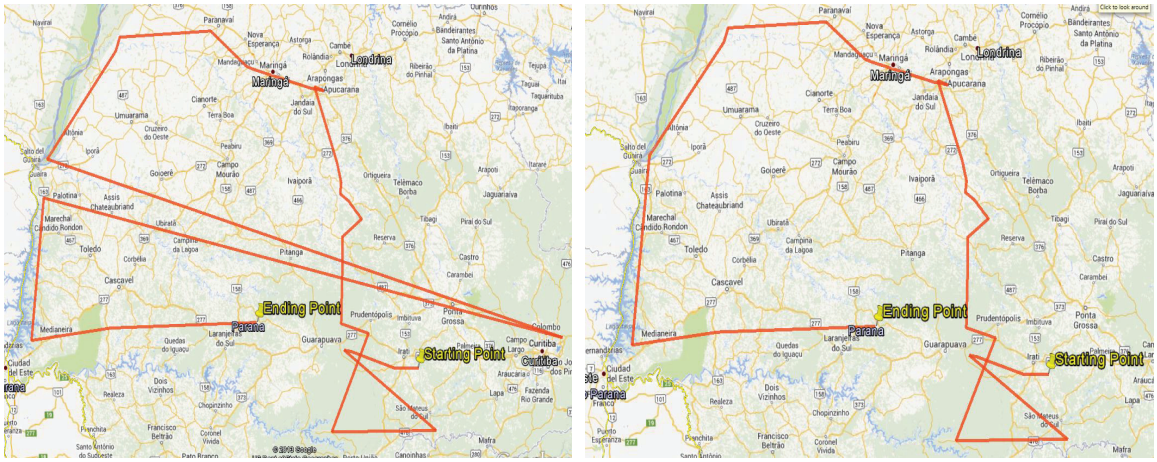


(a) Expedition routes of Chrostowski (b) The intersection point between routes

Figure 5.9: QSTR based reliability assessment of Chrostowski's Expedition III: intersection between Expedition II and III routes

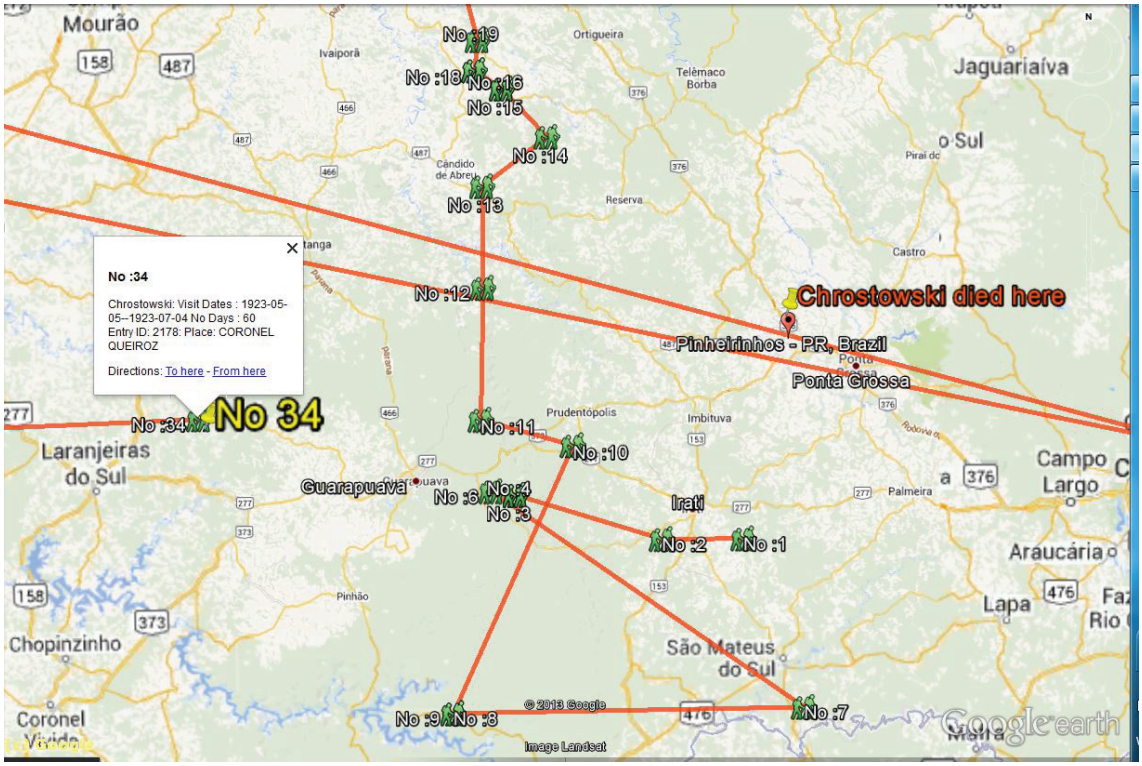
geometry and spatial situation of the *Road-lined* and the *reference* route resemble more than the *straight-lined* and the *reference* route do. Now, we have much more concrete reason to regard the produced expedition route as reliable.

According to the literature [19], Chrostowski died on April 4, 1923. We discussed in Section 5.2.1 that Chrostowski conducted his third expedition in the years 1921 – 1923. The location description of his last visit of the third expedition is shown in Appendix B.6. According to this description he stayed at the last visit site from May 5, 1923 to July 4, 1923, which contradicts the fact he died prior to this visit. Figure 5.11a shows the last site he visited. Here, we wonder how could he made his last visit a month after his death. The only explanation to this contradiction is misreporting the location description, inferred to be his last visit.

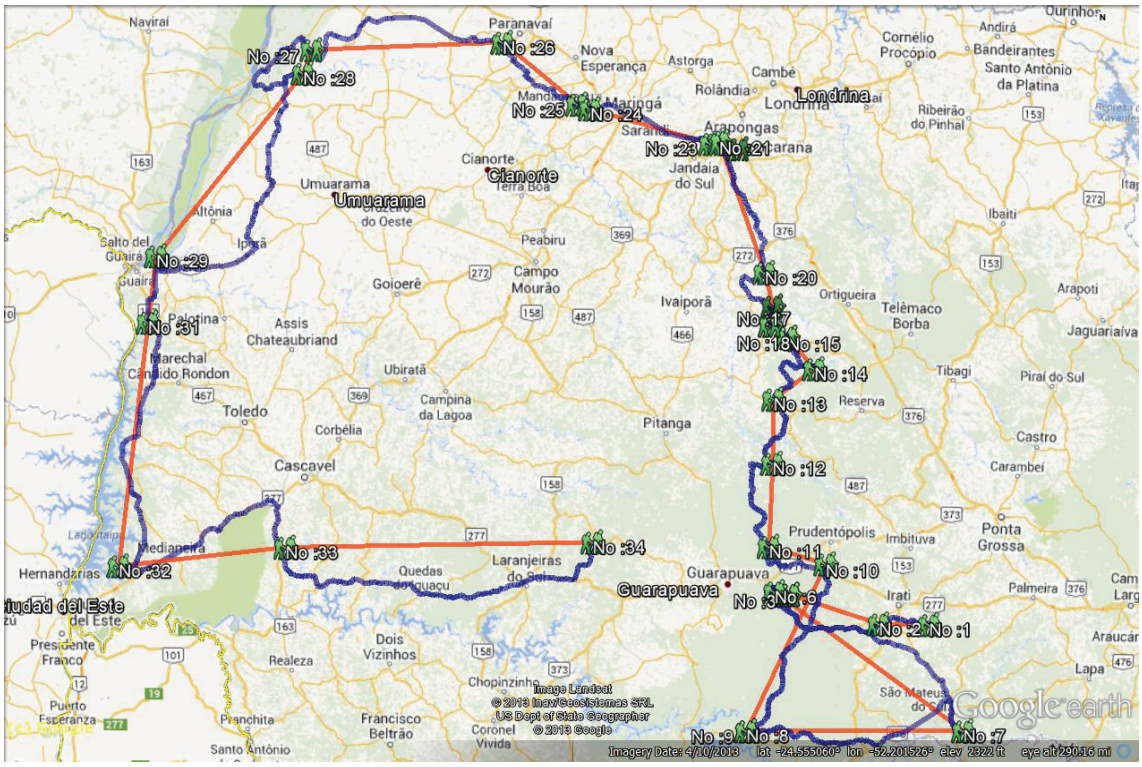


(a) The route before QSTR based assessment (Straight-lined) (b) The route after QSTR based assessment (Straight-lined)

Figure 5.10: Visual inspection based reliability assessment of Chrostowski's Expedition III: before and after the route was assessed by the QSTR tool

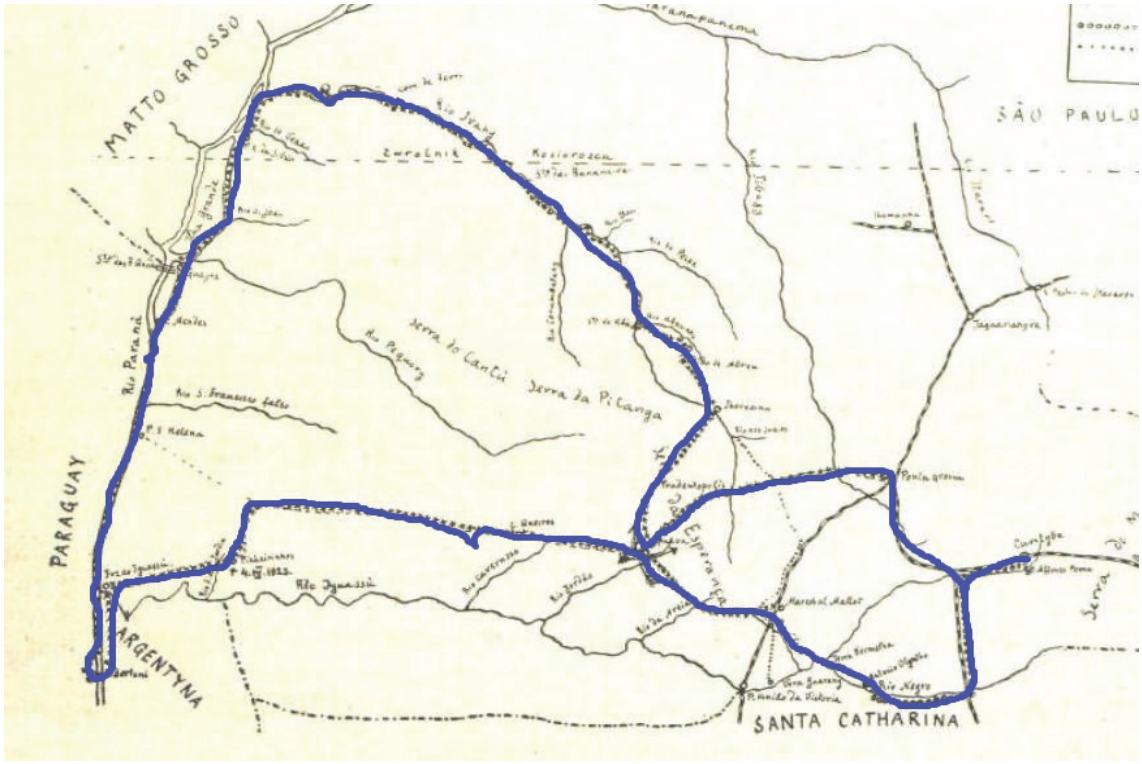


(a) The last location visit of Chrostowski

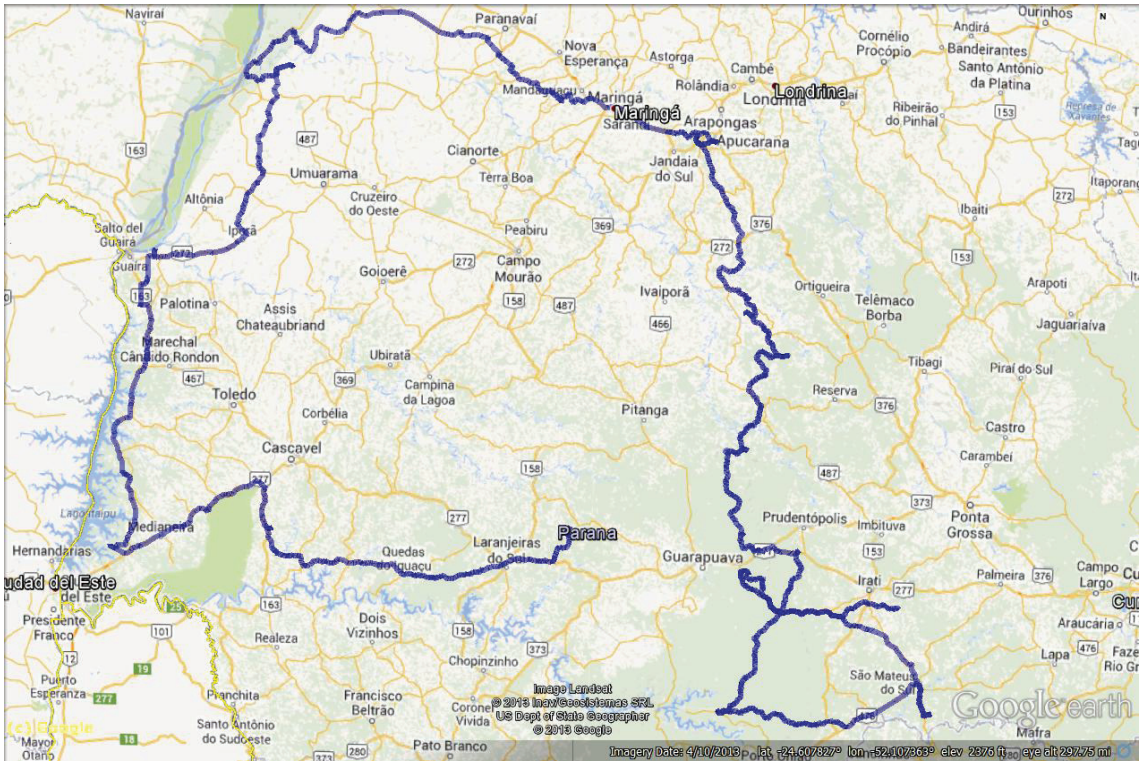


(b) Straight-lined (red) and Road-lined (blue) routes

Figure 5.11: Visual inspection based reliability assessment of Chrostowski's Expedition III: showing the last visit of Chrostowski's third expedition and the Straight-lined and Road-lined routes in combination



(a) Reference route (Source: Jaczewski 1925)



(b) Road-lined route

Figure 5.12: Visual inspection based reliability assessment of Chrostowski's Expedition III

Chapter 6

Summery, Discussion, Conclusion and Recommendation

6.1 SUMMERY

Spatial tracing, as defined in the first chapter, is a discovery of historic events that have taken place in a specified time frame and space. Historic events such as expeditions consist of multiple location visits. The spatial and historic descriptions of such location visits are likely documented in a textual form. Therefore, tracing historic expeditions spatially needs extracting the attributive, temporal and spatial context of the description of such events. To this end, we built a Trajectory Extraction and Storage System which has the capability to: extract contextual information, extract trajectory triplets, store the extracted triplets, produce expedition route from the stored triplets and assess the reliability of the produced routes via a spatiotemporal reasoning tool. Even if there is a number of gazetteers, the Brazilian Ornithological Gazetteer was used as a data source throughout this project.

6.1.1 Raw Data Extraction (RDE)

The first phase of the built system is the Raw Data Extraction which was developed as an independent Java application. This phase is used to extract the location descriptions from the database in which the Gazetteer is stored. This phase has the text tokenization and parsing tools that are used to break the location description paragraphs into sub-paragraphs, each of which contains a description for a single expeditioner. The advantage of incorporating such component in the system is to minimize the risk of mistagging entities in the Contextual Information Extraction phase. If a location description consists multiple historic descriptions and multiple expeditioners, without a way to recognize all separately, the contextual information to be extracted out of such descriptions might end up incorrect. Such risks are minimized, as much as possible, by separating multiple historic descriptions and multiple expeditioners. The Gazetteer we used for this project stores the location descriptions as paragraphs. These paragraphs use the semicolon to separate the spatial and historic descriptions and the text between multiple expeditioners. The Raw Data Extraction phase uses the semicolon to separate the spatial and historic descriptions and the text between multiple expeditioners. The separated parts of the location descriptions are concatenated with a phrased coordinate that is brought in from the same database. The concatenated part is always a description that contains a coordinate, historic description and single expeditioner. The processing time and quality of the Contextual Information Extraction process is improved by using the location descriptions from the Raw Data Extraction phase. The Contextual Information Extraction process takes less time when used over the descriptions from the Raw Data Extraction phase than when used on the original location descriptions. The Raw Data Extraction phase communicates its output with the Contextual Information Extraction phase via an XML file created on the fly.

6.1.2 Contextual Information Extraction (CIE)

Identifying and annotating an entity contained in a location description is the center of CIE. The RDE feeds its output to the CIE phase to annotate each entity of a given location description. The annotation phase is organized as two pipelines: *Entity Extraction Pipeline* and *Spatiotemporal Relationship Extraction Pipeline*. Entities such as *coordinate*, *temporal marks*, *person name* and *place name* are annotated by the Entity Extraction Pipeline, whereas the second pipeline annotates relationships such as *"12 km SE of Curitiba"* and *"sometimes in April 1923"* The Entity Extraction Pipeline has two main annotators and one preliminary annotator. The two main annotators are: Gazetteer Processing Machine and JAPE Transducer, which function with different principles. The Gazetteer Processing Machine is item list-dependent and annotates entities with a defined class if found in the item list when matching takes place. The JAPE Transducer depends on pattern-based JAPE Grammar Rules: the rules comprising a LHS and RHS. The LHS defines the pattern of an entity to be annotated whereas the RHS consists of the action to be taken when the pattern in the LHS is matched. The Text Preprocessing Machine is a preliminary annotator which supports the JAPE Transducer by chunking a location description into different POS categories. Always, the pattern definition in the LHS rule is a sequence of words, punctuation and number tokens. Unless a given location description is broken into tokens, the annotation with a JAPE Transducer will not succeed, because defining the pattern of a given entity depends on sequencing tokens. Since recognition of relationships requires recognizing the items to be related, the Spatiotemporal Relationship Extraction Pipeline depends on the Entity Extraction Pipeline. The Spatiotemporal Relationship Extraction Pipeline has two JAPE Transducers: Spatial Relationship Transducer and Temporal Relationship Transducer to recognize and annotate spatial and temporal relationships, respectively.

Annotating entities with the Gazetteer Processing Machine is more reliable than the JAPE Transducer, because the Gazetteer Processing Machine uses a distinctive list of items to match input entities against. However, it cannot be used to annotate entities with, possibly, infinite list. For instance, annotating a coordinate with the Gazetteer Processing Machine requires to list all combinations of pair of latitude and longitude. Hence, this annotator is restricted to be used for entities with a short list of items. Entities with infinite list are much better annotated at considerably lower cost with a pattern-based JAPE Transducer. The risk in using the pattern-based annotator is in the mixing of entities with rather similar patterns. Unless an explicit pattern is defined for every possible entity in a location description (though the annotated entity is not relevant), the annotation might end up with mixed entities. To overcome the risk of loosing the right spatiotemporal information, annotating every entity in a given description is a wise solution. The CIE phase of our system has such a characteristic that enabled to recognize and annotate all the focus entities explicitly. The annotated location descriptions are communicated with the Triplet Extraction and Storage phase via an XML document.

6.1.3 Triplet Extraction and Storage

An annotated location description consists of the trajectory triplets (*expeditioner name*, *location* and *time*). The *Triplet Extraction and Storage* phase is a process in which these triplets are extracted and stored in a spatial database. This phase is a crucial tool that is flavored with analytical computations on the location and time components. It has four basic parsers: Date Parser, Name Parser, Coordinate Parser and Spatial Relation Parser. The Name and Coordinate Parsers are the easiest ones with capability of parsing their respective attribute from the annotated location descriptions. The Date Parser not only parses the dates from a given description, but analyzes the parsed information to infer additional attributes such as start date, end date and duration of a

visit. The Spatial Relation Parser has semantic characteristics that incorporate the understanding of spatial relationships and infer coordinates out of the given spatial relationship. The temporal, spatial and attributive entities parsed by the Date, Name, Coordinate and Spatial Relation Parsers are aggregated to a list of triplets of *expeditioner name*, *coordinate* and *visit date*. All the parsed and aggregated triplets are stored in a database. The system we built stores the extracted triplets using the JDBC API. The JDBC facilitates the communication between the database and the main trajectory extraction system TESS. Once all the triplets of the annotated location descriptions are parsed and stored, the Temporal Inference phase infers a relative temporal boundary for the stored vague temporal marks.

6.1.4 Temporal Inference

The temporal dimension of a location visit is not always provided explicitly. There are cases in which the start and end date of a location visit are implicit. When such a case is encountered, bounds to start and end dates are subjected to a reasoning , for instance, a visit made in *"March 1923"* may have been made any time in the given period unless it is defined explicitly. The Temporal Inference phase is equipped with a reasoning tool, which infers a relative temporal boundary (start and end dates) of a location visit relative to other chronologically close visits made by the same expeditioner. Our system has a tool to infer a relative temporal boundary for a given vague temporal mark such as *"March 1923"* Along with inferring the relative boundary, the tool computes a confidence level of the occurrence of a visit in the inferred period.

6.1.5 Expedition Route Production

Only extracting the triplets of the location descriptions is not enough to visualize an expedition trajectory. All the parsed triplets must be connected in chronological order to create an expedition route. To visualize an expedition route, the specifications of the chosen visualizing environment must be followed when creating the route. Our system uses a KML file format to communicate an expedition route to our choice of visualization environment (Google Earth). The Expedition Route Production tool of TESS creates two KML files: Expedition route line (KML line) and Expedition route point (KML point). The line connects all the triplets of a given expedition route in chronological order while the points render the basic attributes of a location visit (*start date*, *end date*, *duration*, *description id*, *expeditioner name* and *place name*). When producing an expedition route, the start and end dates of an expedition are determined by the system. However, the system requires the expeditioner name and the $temporal\ gap$ between consecutive expeditions as parameters to produce the route. Once the $temporal\ gap$ is fixed, the system produces a KML line and point files for every detected expedition route of the given expeditioner.

6.1.6 Reliability Assessment

Assessing the reliability of an expedition route (produced by our system) is carried out using the QSTR tool and visual inspection. The reliability of a produced expedition route can, indirectly, appraise the quality of our system. The assessment through the QSTR tool depends on a user provided value of the *average distance* that an expeditioner could travel per day. The tool checks the distance and the temporal gap of a sub-route that connects each pair of the chronologically ordered triplets. If the distance of the sub-route exceeds the defined *average distance*, the triplets at the start and end point of the sub-route is suggested as an outlier. The reliability of an expedition route can also be assessed with a visual inspection. To assess with a visual inspection, comparison between a reference route and a produced route is an option. The comparison is to find a geometrical and spatial situation resemblance between the produced and reference route. An expedition route

that is produced by our system is reliable if the geometrical and spatial situation resemblance is high when compared against the reference route. Conducting a reliability assessment with both the QSTR tool and visual inspection increases the credibility of both the built system and the produced expedition routes.

6.2 DISCUSSION

6.2.1 Attaching a reference to the extracted triplets

The trajectory triplets are extracted from the *Person*, *Coordinate*, *DateMonthYear*, *DateMonthListYear*, *DateMonthDuration*, *DateMonthYearDuration* and *MonthYear* annotation classes. Even if parsing these annotation classes is enough to produce the expedition routes, parsing the annotation classes such as *JournalName* and *PublicationYearPage* creates the possibility to attach a descriptive reference to the parsed triplets. The *JournalName* and *PublicationYearPage* are one of the contextual contents in the location description. When a triplet is doubtful or unreliable, it can be crosschecked with the original publication from which the location description is derived. The crosschecking is easily performed if the publication information is attached to the extracted triplet. With the current system we can track the very location description from which a triplet is extracted. However, incorporating the publication information facilitates linking the extracted triplets and their respective descriptions to an external source.

6.2.2 Producing the expedition routes using the actual road network dataset

The Expedition Route Production phase produces expedition routes by connecting the extracted triplets in a chronological order. The triplets are connected with a straight line which creates a line that connects two points with the shortest distance possible. Such an expedition route may not resemble the actual expedition route that the expeditioners traveled. In Section 5.3.2 we conducted a geometrical and spatial situation resemblance comparison between the straight-lined and road-lined expedition routes against a reference route. According to this visual comparison the resemblance between the road-lined route and the reference route was high. Therefore, if the actual road networks datasets of those days were available we could have produced expedition routes that represent the actual routes with high resemblance.

6.2.3 Assessing the reliability of the produced expedition routes

One of the ways to assess the reliability of the extracted triplets and the produced expedition routes is via the QSTR process. This assessment depends on the constraint of the *average travel distance* that an expeditioner could travel per a day. This *average travel distance* is compared against the distance between consecutive location visits. On the basis of this comparison triplets may be identified as reliable while they are not. This happens because of two reasons:

Reason one: the *average travel distance* that is selected for the assessment may exceed the actual average distance that the expeditioners traveled those days. If this is the case some unreliable triplets may be identified as reliable since the distance between location visits (considering the extracted triplets) may be shorter than the *average travel distance*.

Reason two: the *average travel distance* is compared against the straight distance that connects two location visits. The distance that considers the actual road network between two location visits is longer than the distance that connects the same visits with a straight line. The straight distance is always shorter than the actual distance between two points in the physical world. Therefore, some triplets may appear reliable as result of comparing the straight

distance with the the *average travel distance*. Since our system produces the expedition routes by connecting the extracted triplets with a straight line, some triplets may be considered reliable wrongly.

6.3 CONCLUSION AND RECOMMENDATION

The general aim of this project was to trace historic expeditions spatially. To achieve this aim, extracting the triplets of an expedition trajectory (expeditioner name, location and time) from the gazetteer in which the location descriptions are stored was the main task. To this end, we built a *trajectory extraction* and *storage system* that extracts the attributive, spatial and temporal entities from the location descriptions, stores the extracted entities, produces an expedition route and conducts a reliability assessment on the produced expedition routes. The built system has six phases: RDE, CIE, Triplet Extraction and Storage, Temporal Inference, Expedition Route Production and Reliability Assessment.

The system was built on two different platforms. The Contextual Information Extraction phase was built on the GATE Developer which is a text processing application development platform. The remaining components were built as Java applications. To communicate the results of these two platforms we used the XML file format. However, embedding the Contextual Information Extraction phase with the rest of the components of the system automates the trajectory extraction process fully. To this end, using the GATE Embedded which allows to embed the text processing functionality in various applications is an alternative. The application we built in GATE Developer can be exported to diverse applications using the GATE Embedded.

The Brazilian Ornithological Gazetteer was used as a source for the location descriptions. To illustrate the built system three expeditioners (Tadeusz Chrostowski, Maria Emilie Snethlage and Emil Heinrich Snethlage) were chosen from the Brazilian Ornithological Gazetteer. Our system was used to extract the triplets these expeditioners from their respective location descriptions. We produced the expedition routes according to the extracted information. The reliability of one of the Chrostowski's expedition was assessed using both the QSTR tool and visual inspection-based comparison.

6.3.1 Improving the CIE process

In this research project, we presented a trajectory extraction storage system that depends on a pattern-based JAPE Grammar Rule and Gazetteer list matching to annotate different entities in a location description. The annotation quality is completely dependent on the pattern-based JAPE Grammar Rules and the Gazetteer lists. If either an explicit JAPE Grammar Rule or Gazetteer list is not defined for the target entities, the annotation process may not succeed. The method we used to recognize and extract the spatiotemporal relationships was also dependent on the pattern-based JAPE Grammar Rules. Such a rule may not capture all the possible spatiotemporal relationships. Shifting the traditional Named Entity recognition to a semantic and ontology annotation process improves the annotation and the contextual information extraction process functionality and quality.

LIST OF REFERENCES

[1] Rocio Abascal-Mena and Erick López-Ornelas. Geoinformation extraction and processing from travel narratives. *Proc. ELPUB*, pages 363–373, 2010.

[2] Elisabeth André, Guido Bosch, Gerd Herzog, and Thomas Rist. Characterizing trajectories of moving objects using natural language path descriptions. In *Proc. of the 7th ECAI*, volume 2, pages 1–8, 1986.

[3] Branimir Boguraev and Rie Kubota Ando. TimeML-Compliant text analysis for temporal reasoning. In *In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, volume 5, pages 997–1003, 2005.

[4] Nieves R Brisaboa, Miguel R Luaces, Ángeles S Places, and Diego Seco. Exploiting geographic references of documents in a geographical information retrieval system using an ontology-based index. *Geoinformatica*, 14(3):307–331, 2010.

[5] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, Valentin Tablan, Niraj Aswani, Ian Roberts, Genevieve Gorrell, Adam Funk, Angus Roberts, Danica Damljanovic, Thomas Heitz, Mark A. Greenwood, Horacio Saggion, Johann Petrak, Yaoyong Li, and Wim Peters. *Developing Language Processing Components with GATE Version 7 (a User Guide)*. University of Sheffield Department of Computer Science, 2012.

[6] Euthymios Drymonas and Dieter Pfoser. Geospatial route extraction from texts. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Data Mining for Geoinformatics*, pages 29–37. ACM, 2010.

[7] Nuno Freire, José Borbinha, Pável Calado, and Bruno Martins. A metadata geoparsing system for place name recognition and resolution in metadata records. In *Proceedings of the 11th Annual International ACM/IEEE joint Conference on Digital libraries*, pages 339–348. ACM, 2011.

[8] Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial intelligence*, 54(1):199–227, 1992.

[9] Judith Gelernter and Shilpa Balaji. An algorithm for local geoparsing of microtext. *GeoInformatica*, pages 1–33, 2013.

[10] Lluis Godo and Lluis Vila. Possibilistic temporal reasoning based on fuzzy temporal constraints. In *IJCAI*, volume 95, pages 1916–1922. Citeseer, 1995.

[11] Rocio Guillén. Geoparsing web queries. In *Advances in Multilingual and Multimodal Information Retrieval*, pages 781–785. Springer, 2008.

[12] Jiří Horák, Pavel Belaj, Igor Ivan, Peter Nemec, Jiří Ardielli, and Jan Ržička. Geoparsing of Czech RSS News and Evaluation of Its Spatial Distribution. In *Semantic Methods for Knowledge Management and Communication*, pages 353–367. Springer, 2011.

[13] Unmesh Kurup and Nicholas L Cassimatis. Quantitative spatial reasoning for general intelligence. In *Proceedings of the Third Conference on Artificial General Intelligence Conference*, pages 1–6, 2010.

[14] Nirvana Meratnia and Rolf A de By. Trajectory representation in location-based services: problems & solution. In *Web Information Systems Engineering Workshops, 2003. Proceedings. Fourth International Conference on*, pages 18–24. IEEE, 2003.

[15] Gábor Nagypál and Boris Motik. A fuzzy model for representing uncertain, subjective, and vague temporal knowledge in ontologies. In *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, pages 906–923. Springer, 2003.

[16] Raymond A. Paynter and Melvin A. Traylor. *Ornithological gazetteer of Brazil*, volume v.1. Obtainable from Bird Dept., Museum of Comparative Zoology, Harvard University, 1991.

[17] Raymond A. Paynter and Melvin A. Traylor. *Ornithological gazetteer of Brazil*, volume v.2. Obtainable from Bird Dept., Museum of Comparative Zoology, Harvard University, 1991.

[18] Dr. Rotger Michael Snethlage. Life, Expeditions, Collections and Unpublished Field Notes of Dr. Emil Heinrich Snethlage. `http://www.snethlage.info`, February 2010.

[19] FC Straube and Alberto Urben-Filho. Tadeusz Chrostowski (1878-1923): Biografia e Perfil do Patrono da Ornitologia Paranaense. *Bol. Inst. Hist. Geogr. Paraná*, 52:35–52, 2002.

[20] Fernando Costa Straube and Alberto Urben-Filho. Dicionário Geográfico das Expedições Zoológicas Polonesas ao Paraná. *Atualidades Ornitol*, 133, 2006.

[21] Dhaval Thakker, Taha Osman, and Phil Lakin. Gate JAPE grammar tutorial. *Nottingham Trent University, UK, Phil Lakin, UK, Version*, 1, 2009.

[22] Jeremy Witmer and Jugal Kalita. Extracting geospatial entities from Wikipedia. In *Semantic Computing, 2009. ICSC'09. IEEE International Conference on*, pages 450–457. IEEE, 2009.

[23] Álvaro Zubizarreta, Pablo de la Fuente, José M Cantera, Mario Arias, Jorge Cabrero, Guido García, César Llamas, and Jesús Vegas. Extracting Geographic Context from the web: Geo-Referencing in MyMose. In *Advances in Information Retrieval*, pages 554–561. Springer, 2009.

Appendix A

JAPE Grammar Rules

A.1 JAPE RULE FOR TEMPORAL ENTITY NOTATION

```
Phase:timefinder
Input: Token Lookup SpaceToken
Options: control = appelt

//Macros

Macro: DAY_ONE
({Token.kind == number,Token.category==CD, Token.length == "1"})

Macro: DAY_TWO
({Token.kind == number,Token.category==CD, Token.length == "2"})

Macro: YEAR
({Token.kind == number,Token.category==CD, Token.length == "4"})

Macro: MONTH
({Lookup.majorType=="Month"})

Macro: DAY_SE
(
(DAY_ONE|DAY_TWO)
({Token.kind==punctuation})
(DAY_ONE|DAY_TWO)
)
Macro: DAY_LONG
(
(DAY_ONE|DAY_TWO|DAY_SE)
({Token.kind==punctuation})
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
({Token.kind==punctuation})?
(DAY_ONE|DAY_TWO|DAY_SE)?
)
```

```


// Rules

Rule: ddmmyyyy
Priority:50
(
	(
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	)
	(
	(MONTH)
	({Token.kind==punctuation}|{SpaceToken})?
	({Token.kind==punctuation}|{SpaceToken})?
	(YEAR)
	)
)
:ddmmyyyy
-->
 :ddmmyyyy.DateMonthYear= {rule = "ddmmyyyy"}


Rule: ddmm
Priority:50
(
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(DAY_ONE|DAY_TWO|DAY_LONG|DAY_SE)?
	({Token.kind==punctuation}|{SpaceToken})?
	(MONTH)

```

```
)
:ddmm
-->
 :ddmm.DateMonth= {rule = "ddmm"}

Rule: dmdmy
Priority:50
(
        (
        (DAY_ONE|DAY_TWO)
        ({SpaceToken})?
        (MONTH))
        ({Token.kind==punctuation,Token.string=="-"}|{SpaceToken})?
        (
        ((DAY_ONE|DAY_TWO)?
        ({SpaceToken})
        (MONTH))
        ({SpaceToken})?
        )
        (YEAR)?

)
:dmdmy
-->
 :dmdmy.DateMonthDuration= {rule = "dmdmy"}


Rule: mmyyyy
Priority:50
(
        (MONTH)
        ({Token})?
        ({SpaceToken})?
        (YEAR)
)
:mmyyyy
-->
 :mmyyyy.MonthYear= {rule = "mmyyyy"}

Rule: ddmmyyyydu
Priority:20
(
        (
        (DAY_ONE|DAY_TWO)
        ({SpaceToken})?
        (MONTH)
        ({SpaceToken})?
        (YEAR)
        )
        ({Token.kind==punctuation,Token.string=="-"})
    (
        (DAY_ONE|DAY_TWO)
        ({SpaceToken})?
        (MONTH)
        ({SpaceToken})?
        (YEAR)
        )
)
:ddmmyyyydu
-->
```

```
 :ddmmyyyydu.DateMonthYearDuration= {rule = "ddmmyyyydu"}

 Rule: mydu
Priority:50
(
        (
        (MONTH)
        ({SpaceToken})?
        )
        ({Token.kind==punctuation,Token.string=="-"})
    (
        (MONTH)
        ({SpaceToken})?)
        (YEAR)

)
:mydu
-->
 :mydu.MonthDuration= {rule = "mydu"}

Rule: dmmy
Priority:50
(
        (
        (DAY_ONE|DAY_TWO)
        ({SpaceToken})
        (MONTH))
        ({Token.kind==punctuation,Token.string=="-"})?
        (
        (MONTH)
        ({SpaceToken})
        (YEAR)
    )
)
:dmmy
-->
 :dmmy.DateMonthMonthDuration= {rule = "dmmy"}

 Rule: unknowndate
Priority:50
(

        (MONTH|YEAR)
        ({SpaceToken})?
        (
        ({Token.kind==punctuation,Token.string=="["})
    ({Token.kind==punctuation,Token.string=="?"})
    ({Token.kind==punctuation,Token.string=="]"})
        )
)
:unknowndate
-->
 :unknowndate.DateVague= {rule = "unknowndate"}

 Rule: mmyyyymmyyyy
Priority:50
(

        (MONTH)
        ({Token})?
```

```
225         ({SpaceToken})?
226         (YEAR)
227     ({Token.kind==punctuation,Token.string=="-"})
228         (MONTH)
229         ({Token})?
230         ({SpaceToken})?
231         (YEAR)
232 )
233 :mmyyyymmyyyy
234 -->
235  :mmyyyymmyyyy.MonthYearMonthYear= {rule = "mmyyyymmyyyy"}
```

Appendix B
Location descriptions

B.1 LOCATION DESCRIPTION WITH SINGLE EXPEDITIONER

Ca. 800 m, se Paraná, 113 km SW of Curitiba [2525/4915 (USBGN)] almost on the Santa Catarina border (ICWB); Chrostowski, 3, 8, 18, 26-27 Apr., 6, 10-11, 17, 26 May, 1, 5, 9, 26, 28 June, 7, 9, 14, 18 July, 25 Sept., 4, 18, 24 Oct., 17, 24 Nov. 1914 (Chrostowski, 1921:32-38, as "Antônio Olyntho").

B.2 LOCATION DESCRIPTION WITH MULTIPLE EXPEDITIONERS

173 m (GRB); extreme sw Paraná on Rio Paraná [3343/5817 (USBGN)] just before confluenee with Rio Iguaçu [2536/5436 (USBGN)] at junction of Brazil, Paraguay, and Argentina (MHA, as "Foz do Iguassú", ICWB); Chrostowski, et al., 18-25 Mar. 1923 (Jaczewski, 1925:348, as "Foz do Iguassú?, Sztolcman, 1926:113, as "Foz Iguassú"); Snethlage, Oct. 1928 (Snethlage, 1936:90, as "Foz do Iguassú"); Kaempfer, 16-21, 28, 30 May, 2 June 1930 (Naumburg, 1935:463, as "Foz do Iguassu", 1937:149, as "Fazenda do Iguassú," p. 192, as "Foz do Iguassu"), Partridge, Dec. 1955 (Partridge, 1961, Neotrópica, 7:26, as "Foz do Iguazú"), Zimmer, 1955, Amer. Mus. Novit., no. 1749, p. 17, as "Foz de Iguassú
).

B.3 H. SNETHLAGE'S VISIT DESCRIPTION (ONE ENTRY)

"northwestern Maranhão, SW of Turiaçu [0141/4521 (USBGN)], 40 km inland, H. Snethlage, 7-10, 13-17, 19-22 Nov., 6 Dec. 1923,4 Jan. 1924 (Hellmayr, 1929a:map, 239, 318, 319, 333, 343, 355, 363, 370, 380, 416, 419, 450; Snethlage, 1936:85, as "Alto da Alegria, Turiassu"; Ruschi, 1951b:5, as "Tury-assú, Alto da Alegria"); correct spelling unknown; apparently not the Alto Alegre farther S at 0307/4550 (USBGN)."

B.4 E. SNETHLAGE'S VISIT DESCRIPTION (ONE ENTRY)

"Sea level, on coast of northwestern Maranhão, on left side of mouth of Rio Turiaçu [0136/4519 (USBGN)] (ICWB); E. Snethlage, latter half of 1923, H. Snethlage, 3, 5-6, 8-10, 12-13, 15-19, 23-27, 29-31 Oct., 3,5-9,12-17,19-23,26-30 Nov., 4·-7, 10-15, 17, 19-20, 23, 27, 29 Dec. 1923, 5 Jan. 1924 (Snethlage, 1927d:58, as "Tury-assú"; 1936:87, 88, 91, as "Tury-assú", Hellmayr, 1929a:238ff, as "Tury-assú"; Novaes, 1947:246, as "Turiassú"; Ruschi, 1951b:2, as, "Turry-assu," pp. 5ff, as "Tury-assu"); collector?, 17 Feb. 1945 (Ruschi, 1949b:7); SIlva and Oren, sometime between 1980-1988 (Silva Oren, 1990:311); Pinto, 1944a:296, as "Turiassu.""

B.5 CHROSTOWSKI'S VISIT DESCRIPTION (ONE ENTRY)

"Ca. 900 m, on S side of Rio Iguassu [Rio Iguaçu, 2536/5436 (USBGN)], ca. 12 km SE of Curitiba [2525/4915 (USBGN)], Chrostowski, 22, 31 Jan., 11, 14, 19-20, 22 Feb., 15 Mar. 1914, 10 Feb. 1915[?], 25 Jan. 1923 (Chrostowski, 1921:31-34, as "Affonso Penna"; 1922, Ann. Zool. Mus. Polonici Hist. Nat., 1:400, as "Affonso Penna"; Sztolcman, 1926:119); description places this near São José dos Pinhais [2531/4913 (USBGN)], although it was not mentioned by Chrostowski."

B.6 THE LAST LOCAITON VISIT OF CHROSTOWSKI

"920 m, settlement in sw Paraná, near Larangeiras [Larajeiras do Sul, 2525/5225 (USBGN)], between Rio da Tapera [not located] and Rio Cantagallo [Rio Cantagalo, not located], tributaries on n side of Rio Cavernoso [2542/5225 (USBGN)], Chrostowski, 5 May-4 July 1923 (Jaczewski, 1925:350, also as "Amolafaca"); probably more correctiy as "Coronel Queirós"; possibly nowadays called Virmond [2524/5214 (USBGN)]."

Table B.1: Extracted trplets of Chrostowski's Expedition I

id	name	latitude	longitude	expeditioner	start_date	end_date	no_days	start_ante	end_ante	start_post	end_post
3585	INDIOS, RIO DOS	24.58	51.01	Chrostowski	1911-01-06		1	0	0	0	0
3059	FERNANDES PINHEIRO	25.25	50.33	Chrostowski	1910-12-21		1	0	0	0	0
3059	FERNANDES PINHEIRO	25.25	50.33	Chrostowski	1911-01-04		1	0	0	0	0
3059	FERNANDES PINHEIRO	25.25	50.33	Chrostowski	1911-01-11		1	0	0	0	0
4992	PACIENCIA, RIO	26.02	50.4	Chrostowski	1911-08-23		1	0	0	0	0
3570	IMBITUVA	25.12	50.35	Chrostowski	1910-08-01	1910-08-31	31	0	30	30	0
4313	MALLET	25.55	50.5	Chrostowski	1911-01-16		1	0	0	0	0
2020	CLARO, RIO	26.03	50.38	Chrostowski	1910-12-02		1	0	0	0	0
2020	CLARO, RIO	26.03	50.38	Chrostowski	1910-12-16		1	0	0	0	0
2020	CLARO, RIO	26.03	50.38	Chrostowski	1910-12-25		1	0	0	0	0
2020	CLARO, RIO	26.03	50.38	Chrostowski	1911-01-13		1	0	0	0	0
1974	CHAPÉU DE SOL	26.12	50.44	Chrostowski	1911-08-03		1	0	0	0	0
1974	CHAPÉU DE SOL	26.12	50.44	Chrostowski	1911-08-06		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-05-26		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-02	1910-06-04	3	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-07		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-12		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-14		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-16	1910-06-17	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-19		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-06-28	1910-06-29	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-07-06	1910-07-07	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-07-10		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-07-14		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-07-24		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-08-08	1910-08-09	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-08-17		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-08-25		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-08-27		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-09-03		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-09-08		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-09-12		1	0	0	0	0

7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-10-09	1910-10-10	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-01		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-05	1910-11-06	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-11		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-17		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-23		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-11-28		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1910-12-10		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-01-18		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-03-03		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-03-05		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-04-20		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-05-06		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-05-16		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-05-23		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-06-04		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-06-22	1911-06-23	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-07	1911-07-09	3	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-11	1911-07-12	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-14	1911-07-17	4	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-19		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-23	1911-07-24	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-26	1911-07-27	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-07-29		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-08-03		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-08-05		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-08-07	1911-08-08	2	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-08-16		1	0	0	0	0
7836	VERA GUARANI	26.02	50.47	Chrostowski	1911-08-26		1	0	0	0	0
3835	IVAÍ, RIO	26.02	50.47	Chrostowski	1910-12-29		1	0	0	0	0
6348	SANTA CRUZ DO TIMBÓ	26.02	50.47	Chrostowski	1910-12-17	1910-12-18	2	0	0	0	0
6348	SANTA CRUZ DO TIMBÓ	26.02	50.47	Chrostowski	1911-01-07	1911-01-08	2	0	0	0	0

Table B.2: Extracted trplets of Chrostowski's Expedition II

id	name	latitude	longitude	expeditioner	start_date	end_date	no_days	start_ante	end_ante	start_post	end_post
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-11-17		1	0	0	0	0
7414	TERRA VERMELHA	26	50.3	Chrostowski	1914-12-02		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-07-07		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-07-09		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-10-04		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-04-03		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-05-06		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-09-25		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-04-08		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-04-26	1914-04-27	2	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-01-22		1	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-01-31		1	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1915-02-10		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-05-17		1	0	0	0	0
6790	SÃO LOURENÇO	26.17	49.6	Chrostowski	1914-03-15		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-04-18		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-10-24		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-07-14		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-05-10	1914-05-11	2	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-02-19	1914-02-20	2	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-02-14		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-06-09		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-06-28		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-07-18		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-06-01		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-06-05		1	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-03-15		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-11-24		1	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1914-02-22		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-05-26		1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-06-26		1	0	0	0	0

31	AFONSO PENA	25.32	49.06	Chrostowski	1914-02-11	1	0	0	0	0
251	ANTÔNIO OLINTO	25.59	50.12	Chrostowski	1914-10-18	1	0	0	0	0

Table B.3: Extracted trplets of Chrostowski's Expedition III

id	name	latitude	longitude	expeditioner	start_date	end_date	no_days	start_ante	end_ante	start_post	end_post
2797	FAZENDA FERREIRA	26.01	51.36	Chrostowski	1922-03-12	1922-03-28	17	0	0	0	0
7640	UBÁ, SALTO	24.3	51.28	Chrostowski	1922-11-18		1	0	0	0	0
6238	SALVADOR	12.59	38.31	Chrostowski	1921-08-01	1921-08-31	31	0	30	30	0
2178	CORONEL QUEIROZ	25.22	52.1	Chrostowski	1923-05-05	1923-07-04	60	0	0	0	0
1747	CARA PINTADA	24.88	51.26	Chrostowski	1922-05-15	1922-06-04	20	0	0	0	0
2129	CONCÓRDIA, RIO	25.43	51.17	Chrostowski	1922-03-01	1922-03-12	12	0	0	0	0
2039	COBRE, SALTO DO	23.53	51.53	Chrostowski	1922-12-11	1922-12-19	9	0	0	0	0
6626	SÃO DOMINGOS	25.43	51.17	Chrostowski	1922-02-15	1922-02-28	14	0	0	0	0
279	APUCARANA	24.47	51.1	Chrostowski	1922-08-01	1922-08-31	31	0	30	30	0
5245	PARY, CORREDEIRA DO	23.38	52.19	Chrostowski	1923-01-04	1923-01-06	3	0	0	0	0
5508	PINHEIRINHO	25.25	53.55	Chrostowski	1923-03-28	1923-04-23	26	0	0	0	0
4827	MUTUM, ILHA DO	23.15	53.43	Chrostowski	1923-01-14	1923-01-15	2	0	0	0	0
3040	FAZENDA WISNIEWSKI	26.03	50.38	Chrostowski	1922-02-01	1922-02-28	28	0	27	27	0
4363	MANGUINHOS	22.47	41.56	Chrostowski	1922-01-01	1922-01-31	31	0	30	30	0
874	GUARAPUAVA	25.23	51.27	Chrostowski	1922-04-28	1922-05-14	17	0	0	0	0
1156	BOM, RIO	23.56	51.44	Chrostowski	1922-12-20	1922-12-22	3	0	0	0	0
31	AFONSO PENA	25.32	49.06	Chrostowski	1923-01-25		1	0	0	0	0
4313	MALLET	25.55	50.5	Chrostowski	1922-01-10	1922-02-02	23	0	0	0	0
3055	FÊNIX	23.54	51.57	Chrostowski	1922-12-23	1923-01-02	10	0	0	0	0
3064	FERRO, CORREDEIRA DO	23.12	52.54	Chrostowski	1923-01-07	1923-01-13	7	0	0	0	0
589	BANANEIRAS, SALTO DAS	23.4	52.13	Chrostowski	1923-01-03		1	0	0	0	0
3130	FOZ DO IGUAÇU	25.33	54.35	Chrostowski	1923-03-18	1923-03-25	8	0	0	0	0
3042	FAZENDA ZAWADSKI	25.43	51.17	Chrostowski	1922-02-15	1922-02-28	14	0	0	0	0
2799	FAZENDA FIRMIANO	26	50.32	Chrostowski	1922-03-01	1922-03-12	12	0	0	0	0
5485	PINDAHUBA, CACHOEIRA DE	24.08	51.31	Chrostowski	1922-11-28	1922-12-06	9	0	0	0	0
7649	UBÀZINHO, RIO	24.35	51.2	Chrostowski	1922-10-12	1922-11-20	39	0	0	0	0
402	AREIA, RIO DA	26.01	51.36	Chrostowski	1922-03-29	1922-04-12	14	0	0	0	0

426	ARIRANHA, CACHOEIRA	24.22	51.27	Chrostowski	1922-11-23	1922-11-26	4	0	0	0	0
7847	VERMELHO	24.61	51.26	Chrostowski	1922-06-06	1922-07-05	30	0	0	0	0
603	BANHADOS	25.3	51	Chrostowski	1922-04-13	1922-04-17	5	0	0	0	0
5753	PORTO XAVIER DA SILVA	23.25	53.47	Chrostowski	1923-01-15	1923-01-17	3	0	0	0	0
1629	CÂNDIDO DE ABREU	24.35	51.2	Chrostowski	1922-08-02	1922-10-11	70	0	0	0	0
7062	SETE QUEDAS, SALTO DAS	24.02	54.16	Chrostowski	1923-01-23	1923-02-26	34	0	0	0	0
2129	CONCÓRDIA, RIO	25.42	51.24	Chrostowski	1922-03-01	1922-03-12	12	0	0	0	0
7402	TERESA CRISTINA	24.48	51.07	Chrostowski	1922-07-08	1922-07-31	24	0	0	0	0
6086	RIO DE JANEIRO	22.54	43.14	Chrostowski	1922-01-01	1922-01-31	31	0	30	30	0
2020	CLARO, RIO	25.55	50.74	Chrostowski	1922-02-03	1922-02-14	12	0	0	0	0
5727	PÔRTO MENDES	24.3	54.2	Chrostowski	1923-02-27	1923-03-16	20	0	0	0	0
7640	UBÁ, SALTO	24.3	51.28	Chrostowski	1922-11-14		1	0	0	0	0