INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Resolution of Place Name References in Textual Documents

Luís Miguel Rosa dos Santos

Dissertation submitted to obtain the Master Degree in
Information Systems and Computer Engineering

Jury

Chairman:	Prof. António Manuel Ferreira Rito da Silva
Supervisor:	Prof. Bruno Emanuel da Graça Martins
Co-Supervisor:	Prof. Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member:	Prof. José Alberto Rodrigues Pereira Sardinha

November 2012

Abstract

With the growing availability of large volumes of textual information on the Web, text mining techniques have been gaining a growing interest. One particularly interesting text mining problem relates to the recognition and disambiguation of place names mentioned in texts, an essential task to the analysis of textual contents from a geographical point of view.

In the context of my MSc thesis, I developed and evaluated a system which uses machine learning methods for resolving place references in text, i.e. for linking particular character strings in documents that denote locations, to the corresponding geospatial coordinates. The proposed method uses a combination of two models, in which a first learner based on Conditional Random Fields is used to tag place references in the text, and then a second learner based on a regression model is used to rank and choose from a set of possible candidate disambiguations for the place references that were initially tagged. The proposed method was evaluated on English corpora containing SpatialML annotations for the geographical references. Specifically, I measured accuracies of 0.95 and 0.55, respectively in the recognition and disambiguation of place references that are proper names in the English SpatialML corpus, and of 0.61 and 0.03, respectively in the recognition and disambiguation tasks of nominal references in the same SpatialML dataset. With the LGL dataset, the proposed method achieved accuracies of 0.61 and 0.17, respectively for the recognition and disambiguation tasks.

Keywords: Place Reference Resolution , Machine Learning , Geographic Information Retrieval

Sumário

Com a crescente da disponibilização de grandes volumes de informação textual na Web, técnicas de *text mining* têm ganho um interesse crescente. Um problema de *text mining* particularmente interessante relaciona-se com o reconhecimento e a desambiguação de nomes de locais mencionados em textos, uma tarefa essencial para a análise de conteúdos textuais, de um ponto-de-vista geográfico.

No contexto da minha tese de mestrado, desenvolvi e avaliei um sistema que usa métodos de *machine learning* na resolução de referências a locais em textos, i.e. para relacionar *strings* que denotam localizações em documentos, às correspondentes coordenadas geo-espaciais. O método proposto usa uma combinação de dois modelos, no qual um primeiro *learner* baseado em Conditional Random Fields é usado para anotar referências geoespaciais nos textos, e em que de seguida um segundo *learner*, baseado num modelo de regressão, é usado para avaliar e escolher o melhor candidato de um conjunto de possiveis candidatos de desambiguação para as referências geoespaciais que foram inicialmente anotadas. O método proposto foi avaliado em *corpora* inglês contendo anotações SpatialML para referências geográficas. Especificamente, medi uma *accuracy* de 0.95 e 0.55, respectivamente para as tarefas de reconhecimento e desambiguação para referências geoespaciais que são nomes próprios no *corpus* inglês *SpatialML*, e de 0.61 e 0.03, respectivamente para as tarefas de reconhecimento e desambiguação para referências nominais, no dataset SpatialML. No dataset LGL, o método proposto atingíu níveis de *accuracy* de 0.61 e 0.17, respectivamente para as tarefas de reconhecimento e desambiguação.

Keywords: Resolução de Referências Geoespaciais , Machine Learning , Recuperação de Informação Geográfica

Agradecimentos

Este trabalho representa para mim o fim de uma etapa da minha vida. Guardei muito conhecimento ao longo deste trajecto, que não seria possivel sem o apoio de várias pessoas às quais expresso aqui a minha enorme gratidão.

Antes de mais, agradeço à minha família, que sempre me apoiou, incentivou e ajudou a cumprir todos os meus objectivos. Agradeço especialmente aos meus pais, por tudo o que me proporcionaram desde que nasci, e que sempre me conseguiram ajudar nas mais variadas situações, particularmente por terem feito um esforço extra a cuidar da avó quando esta se encontrava doente, para que eu pudesse acabar o projecto de tese. São as pequenas coisas que ficam na vida, e vocês estarão sempre no meu coração, esteja onde estiver. Agradeço também à minha irmã Ana e ao meu cunhado Artur, que também me apoiaram bastante durante a realização deste trabalho.

Quero agradecer também a todos os meus amigos e colegas que me acompanharam no meu percurso académico, que me apoiaram, que lutaram comigo várias batalhas. Agradeço ao Sérgio Soares pela ajuda no meu trabalho. Ao Rui Valadas, Ricardo Martins, João Saúde e restante grupo pela diversão e amizade, que me ajudaram nos momentos mais difíceis. Obrigado também ao Pedro Cachaldora, Ana Karina, Cátia Dias e restante grupo da faculdade pela companhia e amizade durante todo o meu percurso na faculdade. Um obrigado muito especial à Carolina Bento e à Ana Silva, pela amizade e companhia durante a realização desta dissertação, tornando uma tarefa difícil e custosa numa tarefa divertida e bem-disposta, ao dar-me um excelente e divertido ambiente de trabalho.

Como não poderia deixar de ser, quero agradecer aos meus excelentes orientadores, o Prof. Bruno Martins e a Prof. Luísa Coheur, por todo o apoio e dedicação, que foram fundamentais para o meu progresso e empenho durante esta tarefa. Em particular, quero deixar um grande agradecimento ao Prof. Bruno Martins, pela confiança dada ao incorporar-me no projecto SInteliGIS, suportado pela Fundação para a Ciência e Tecnologia (FCT).

Gostaria também de agradecer o suporte financeiro da FCT, através da bolsa com referência

PTDC/EIA-EIA/109840/2009.

Quero também agradecer a todos os meus colegas no grupo DMIR, particularmente ao Ivo Anastácio, que sempre me apoiaram e esclareceram dúvidas que tinha, bem como pelas ideias fornecidas, que me valeram de muito para completar com mérito esta etapa da minha vida.

Por último gostaria de estender os meus agradecimentos a todos aqueles de uma forma ou de outra (fornecendo ideias e/ou criticas construtivas), foram ajudando anonimamente nas inúmeras discussões ao longo deste trabalho.

Dedico este trabalho à minha sobrinha Mariana.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Geographic information is present in almost all types of existing textual documents, from geological reports about a specific location to news feeds. Thus, it is important to study automated techniques for (i) identifying geographical information in the textual documents, (ii) associating the identified geographic information, expressed in the form of location names, to the corresponding geospatial locations over the surface of the Earth, and (iii) aggregating the extracted information for the purpose of presenting it to the users, so that the extracted information can be used to solve particular information needs.

In the past, various researchers studied different text mining approaches specific to geographic information extraction problems. The most common approaches for place reference resolution divide the problem into two separate sub-tasks, namely (i) the identification of place references in text, and (ii) the disambiguation of the recognized references. The complete resolution of place references in text presents many challenges due to the ambiguity of place names. There may be multiple locations with the same name, or multiple names for the same location. Thus, it is a core issue to address this ambiguity, by correctly identifying the locations in question. Most of the existing approaches are based on rules and dictionaries, although recently it has become popular to use Machine Learning (ML) techniques for addressing these kinds of problems.

This work proposes a supervised machine learning method for resolving spatial references in text. The proposed method uses a combination of two models, in which a first learner, based on Conditional Random Fields (CRF), is used to recognize and classify place references, and then a second learner based on a regression model is used to rank the possible candidate disambiguations for the place references that were initially tagged, using gazetteer queries for generating the candidate disambiguations.

Different configurations of the proposed method (e.g., different feature sets) were evaluated through English corpora containing SpatialML annotations for the spatial references. Results show that machine learning methods are indeed suitable for the task of resolving place references, obtaining accuracies of 0.95 for the recognition and 0.62 for the disambiguation of place references that are proper names (i.e., *NAM* references), and of 0.61 for the recognition and 0.03 for the disambiguation sub-tasks in the case of nominal references (i.e., *NOM* references) in the SpatialML dataset. With the LGL dataset, the results obtained were not so good, and we measured accuracies of 0.62 for the recognition, and of 0.17 for the disambiguation sub-tasks.

1.1 Contributions

The main objective of this research was to introduce a novel approach to the problem of resolving geospatial expressions in text, based on machine learning techniques for both the recognition and disambiguation tasks. This main objective can be divided in two important sub-goals, which correspond to the following main contributions of this research:

- The recognition of geographical references can be addressed effectively through the formalism of Conditional Random Fields (CRFs). A CRF model using a rich set of features obtained accuracy values between 0.61 and 0.95, when considering SpatialML annotated datasets that are well known in the area.
- The disambiguation of geographical references can be addressed effectively through the formalism of regression models, namely through Support Vector Machine (SVM)s or regression trees (REPTrees). A SVM model using a rich set of features obtained accuracy values between 0.14 and 0.37, while the REPTree model using the same set of features obtained accuracy values between 0.17 and 0.55, over the same datasets used for the case of recognition.

1.2 Organization of the Dissertation

The rest of the document is organized as follows:

- Chapter 2 presents the fundamental concepts required to understand the contents of this dissertation, introducing machine learning methods such as Hidden Markov Models (HMMs) and Conditional Random Fields (CRFs).

- Chapter 3 presents related work, covering both the areas of named entity recognition and geographical reference resolution. In particular, Chapter 3 details some of the previously proposed systems that address the recognition and normalization of geographical references, as given in textual documents.
- Chapter 4 presents the proposed approach for the resolution of geographical references, detailing methods for addressing the recognition and disambiguation tasks.
- Chapter 5 presents the experimental validation of the proposed machine learning approaches, detailing both the experimental methodology and the obtained results.
- Chapter 6 summarizes the main conclusions and points directions for future work.

Chapter 2

Concepts

Resolving geographical references in text is generally divided in two separate sub-tasks, namely (i) geographical reference recognition and (ii) geographical reference normalization, also commonly referred to as disambiguation.

The recognition task aims to delimit character strings corresponding to a set of different types of geographical expressions. These types can either refer to specific points/areas on Earth (e.g., *Lisbon,Portugal*, *Asian Continent*), or to locations relative to other points (e.g., *50 miles southwest of Mount Everest*). Recognizing geographical expressions in text is a particular aspect of the more general problem of Named Entity Recognition (NER), that is in itself a subtask of Information Extraction (IE) that seeks to find and classify atomic elements in text into predefined entity categories (e.g., names of locations, organizations or persons). NER problems have been studied for many years by many different researchers and with many different types of methods.

As for the geographical expression normalization task, it aims to interpret the recognized expressions in order to represent them in a standard format, e.g., through associations to latitude and longitude coordinates. The normalization task is considerably more challenging, since the interpretation of locations in text requires some sort of reasoning over the data, relating each location to other locations mentioned in the same document.

Section 2.1 details methods that are commonly used in the NER task, particularly Hidden Markov Models and Conditional Random Fields. Section 2.2 gives an overview on the evaluation methods used in entity recognition. Section 2.3 details the SpatialML annotation scheme, which was previously proposed for annotating geographical expressions. Finally, Section 2.4 summarizes the contents of this chapter.

2.1 Named Entity Recognition

Information Extraction (IE) concerns with automatically extracting structured information from non-structured text. A particularly important task in the context of IE deals with the recognition and classification of named entities. This task is commonly referred to as **Named Entity Recognition** (NER), and it has been widely studied in the Natural Language Processing (NLP) community. In the context this work, the named entities that I will be extracting from textual documents are geographic references to specific locations on the surface of the Earth.

Early IE systems were based on rules and dictionaries. According to Sebastiani (2002), rule-based systems can achieve good results, but they can also have problems in terms of portability. Rules need to be defined manually by a knowledge engineer, with the help of a domain expert for the information domain in question. If it is necessary to change rules, both the engineer(s) and the domain expert(s) will have to intervene again. If one requires porting the information extraction system to another domain, then it is necessary to bring in another domain expert, restarting the work from the ground up.

The current trend in the area relates to the usage of machine learning approaches, through the usage of algorithms which allow the learning of models for the identification and classification of entities through manually-annotated examples. The learned models can later be used to automatically identify entities in new non-annotated texts. This approach has many advantages in comparison to the usage of rule-based systems (Sebastiani, 2002). Instead of changing the rules, like in rule-based systems, one only needs to change the corpora used for model learning, by adding different annotated documents, and retrain the model. This way, the system acknowledges the new changes with less maintenance cost, being that monetary or computational.

Nadeau & Sekine (2007) stated that the currently dominant technique in NER is *supervised learning*, with models such as Hidden Markov Model (HMM)s or CRFs being commonly used. Supervised learning requires a pre-annotated set of documents for training a classifier capable of identifying entities in text. There are also *semi-supervised* approaches, where only a small degree of supervision is needed to start the training process, as well as *unsupervised learning* approaches, which use techniques that rely on lexical resources, patterns and statistics to recognize the entities.

With machine learning approaches, the NER task is usually seen as a sequence classification problem. The idea behind it is that, supposing there are F different tags available as classifications for the tokens that compose the documents, a sequence of T word tokens has up to $F \times T$ possible sequences of tags available, although some can be discarded on structural grounds. After a NER system has been prepared (i.e., after its underlying model has been trained with

an annotated corpus), the NER system analyses each of the $t \in T$ word tokens in an input sequence, and classifies each of them with the most appropriate tag $f \in F$. The analysis of each word token t is not trivial, but one aspect that facilitates the analysis is the fact that there can be clues in previous/succeeding tokens in the sequence, as well as in the *features* of each token. The major drawback of learning-based NER systems is that, if they are trained to classify in a specific domain, thus assuring good results in entities from that domain, these systems will hardly perform well on other domains.

When one wants to classify a word token as, for example, a *Location* or a *Non-Location*, there is a problem concerning the classification and the delimitation of entities. Suppose we have the phrase *Luis is going to Campo Grande* as a sequence of tokens (i.e., the *symbols*), and that we want to classify each of the tokens with the classes (i.e., the *states*) introduced earlier. When we reached the *Campo Grande* part of the observation sequence, the classification model, if well trained, would classify the symbol *Campo* and the symbol *Grande* as two locations, when the two symbols actually refer to the same location. To solve this problem, one can use the **BIO encoding**. The BIO acronym stands for *Begin*, *Inside*, and *Other*. In the previous example, using the BIO encoding, one would have three classes to classify the observation symbols: *Other*, *Begin_Location* and *Inside_Location*. With these classes, when the model classifies *Campo Grande* correctly, the symbol *Campo* will be classified with the class *Begin_Location* and *Grande* will be classified with *Inside_Location*, leaving no space for ambiguity and specifying the two symbols as one unique location.

In a recent survey, Krovetz *et al.* (2011) evaluated three different NER systems. The systems evaluated were a NER system developed in Stanford using CRFs, a system developed in the University of Illinois using a regularized averaged perceptron, and the BBN IdentiFinder[1] system which uses HMMs, considering 3 different tags, namely i) person, ii) organization and iii) location. The corpus used to compare the systems consisted of 425 million words. Although the NER task has a reported state-of-the-art accuracy rate between 85% and 95%, what the authors found was that these systems made many mistakes, such as classifying in the same document an entity as a person, and later as an organization, or classifying *The Web* as a person. They also realized that the agreement rate between systems ranged from 34% to barely over 50%. These ratios depend on the systems being compared (in the author's study, systems were compared in pairs of two), and on the tag that is being analyzed in agreement terms. The major contribution from this work was a unit test to apply to a NER system, in order to discover mistakes in the classification assessment. The authors also concluded that the high accuracy rates that have been reported were based on inadequate testing methods, and that although the systems can

[1] `http://www.bbn.com/technology/speech/identifinder`

agree on a tag for a given entity, the chosen tag may be the wrong one.

Over the years, several machine learning methods have been used to address NER problems. This dissertation describes Support Vector Machines, Hidden Markov Models and Conditional Random Fields, the most used techniques to adress this problem, in the next three sub-sections.

2.1.1 Support Vector Machines

According to Moguerza & Muñoz (2006), a SVM is a model for the disambiguation of data, developed in the mid 90's by Vapnik (1995). The objective of an SVM is to find a N-dimensional *hyperplane* (i.e., a decision boundary surface) which optimally separates the given data. The distance from the hyperplane to the closest data of each type is called the *margin of separation*. The larger the margin is to the closest data of each type, the easier it is for the system to disambiguate unseen data. The distance between the closest data of the two types is called the *maximal margin*.

Sometimes, specially in the case where the data is non-linearly separable, it is easier to separate the data if the *feature* space is of an higher-dimensionality than the input space. A *kernel function* is a real-valued function that does just that, transforming the input data of a certain dimensionality into a *feature* space of a higher-dimensionality than the input space.

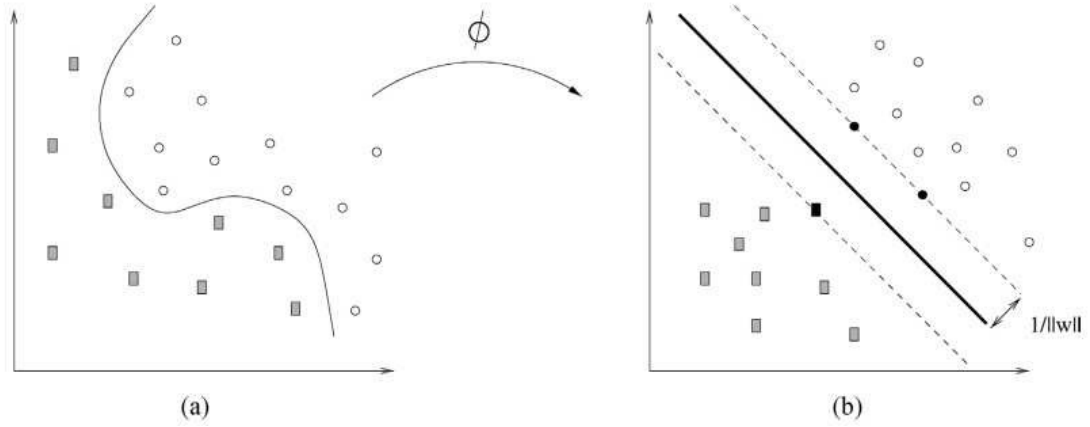

Figure 2.1: Transformation of data from the input space (a) into a high-dimensionality feature space (b).

Considering a classification problem where the discriminant function is non-linear, like in Figure 2.1a, suppose one has a mapping into a *feature space* in such a way that the data has become linearly separable, like in Figure 2.1b. The SVM looks for the hyperplane that lies furthermost from both classes, which is known as the optimal (i.e., maximum) margin hyperplane.

As a classifier, after a model is learned using the training data, the SVM takes a new data object,

Feature	Description
Context word feature	Preceding and following words of a particular word token
Word suffix	Suffix of a particular word token
Word prefix	Prefix of a particular word token
Named entity information	NER tags of previous word tokens
First word	If a particular word token is the first in a sentence
Digit *features*	If a particular word token has digits
Word frequency	Frequency of a particular word token
Word lenght	If the lenght of a particular word token is less than 3
POS information	Part-Of-Speech (POS) tag of a particular word token

Table 2.1: *Features* used in an Indian NER system by Ekbal & Bandyopadhyay (2010).

transforms it into the *feature* space, and analyses in what side of the hyperplane does the new data get placed, classifying it accordingly.

Ekbal & Bandyopadhyay (2010) used SVMs on NER problems, considering the Hindi and Bengali languages. Although SVMs can only perform binary classification directly, they can be extended to multiclass problems through the *one-vs-one* or the *one-vs-all* strategies. The authors specified that their system used an SVM model which separates data into two categories, but they did not specify how the multiclass problem was addressed. It is assumed that the authors used the *one-vs-one* strategy, where every classifier (i.e., one for each class) assigned the candidate being analyzed to one of two classes, using a max-wins voting strategy where the vote for the assigned class is increased by one vote, and finally the class with most votes determines the instance classification. Using the *features* in Table 2.1, they were able to classify individual word tokens as belonging to one of four classes, namely i) person names, ii) location names, iii) organization names, and iv) miscellaneous names.

2.1.2 Hidden Markov Models

A *Hidden Markov Model*, according to Rabiner & Juang (2003), is a doubly stochastic process, consisting of an underlying stochastic process that is not directly observable (i.e., the *hidden* nodes) but that can be seen in action through another set of stochastic processes that produce a sequence of observed symbols. HMMs can model various systems that have the Markovian property, which states that the conditional probability of the next future state depends only on the current state. Sequence classification problems, such as NER or Part-Of-Speech (POS) tagging, are two example applications for HMMs.

As an example of this model, let us assume that we have a system in which the *output* is a sequence of words from a given vocabulary. Each of these words was generated with basis on different *tags*, which we cannot see directly. The tags represent different classes for the words

(e.g., if a word is a location name or not). When modeling this system with an HMM, one would have i) ***states*** that generate the observation symbols, and are not visible to the observer (i.e., the tags), and ii) ***symbols***, which constitute the output of the system, which is visible to the observer (i.e., the words). In this model, there are a finite number of *states* N. In each instant of time t, a new state is introduced according to a transition probabilistic distribution which depends on a previous state (i.e., there is a Markovian property). After each transition is made, an observation symbol is produced.

To create an HMM one needs three probability sets, namely i) the state transition probabilities (here referred to as **A**), ii) the output observation probabilities (here referred to as **B**), and iii) the probability of a given state beginning the sequence (known as π). These probabilities describe a HMM, which is represented as $\lambda = (A, B, \pi)$. In Figure 2.2 one can see the representation of a directed graphical model for an HMM.

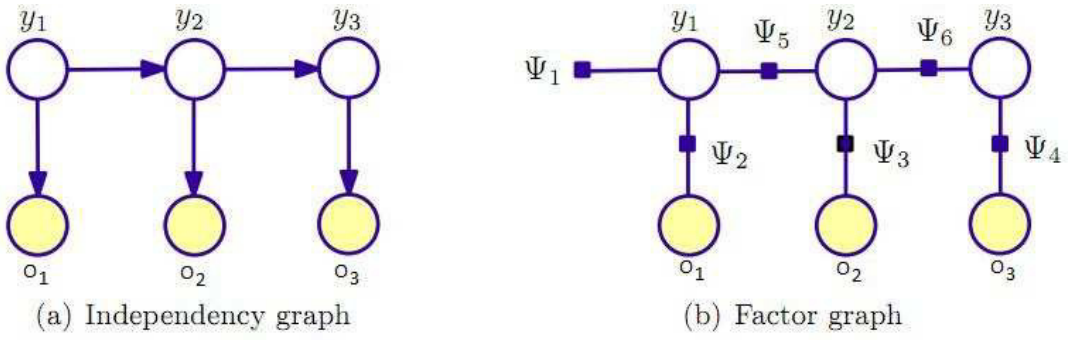

Figure 2.2: An undirected graphical model for an HMM (adapted from Klinger & Tomanek (2007)).

Given an annotated document, with the observation symbols classified with the set of states considered for a particular application domain, one can calculate the probabilities needed to define the HMM (A, B and π). Let us assume for this example that we are analyzing a phrase classified as in Figure 2.3.

O	O	O	B_Location	I_Location	O
Luís	went	to	Campo	Grande	.

Figure 2.3: Example sentence annotated with BIO tags

At first, one needs to calculate the counts of words per state, i.e. the words classified as being generated from a certain state. With this we can compute $P(o_t|i_k)$, i.e. the probability of observing o_t knowing it came from state i_k. This probability can defined as $P(o|i_k) = \frac{count(o_t|i_k)}{\sum count(O|i_k)}$, i.e. one counts the number of times that o appeared tagged with state i_k, and divides this count by

$P(row\|column)$	Other	Begin_Loc	Inside_Loc
Luís	1/4	0	0
Went	1/4	0	0
To	1/4	0	0
Campo	0	1	0
Grande	0	0	1
.	1/4	0	0

Table 2.2: Probabilities of symbol generation, in the example given in Figure 2.3.

$P(row\|column)$	Other	Begin_Loc	Inside_Loc
Other	2/3	0	1/3
Begin_Loc	1	0	0
Inside_Loc	0	1	0

Table 2.3: Probabilities of succeeding states, in the example given in Figure 2.3.

the number of observations for every o in that state. This way one computes B, i.e. the probabilities of observing each observation symbol in each state available. For the previous example, this is represented in Table 2.2.

Next, one counts the number of times a state has occurred following another state. With this we can compute $P(i_k|i_u)$, i.e. the probability a state i_k occurred after a state i_u. This probability can be defined as $P(i_k|i_u) = \frac{count(i_k|i_u)}{count(i_k|I)}$, i.e. the number of times one transitions to state i_k from state i_u divided by the number of times one transitions to state i_k from every other state available. This way one computes A, i.e. the state transition probabilities. In the example above, the transition probabilities are represented in Table 2.3.

Notice that some of these probabilities are zero, due to the corresponding event never occurring in the training observation sequence.

Finally, one counts the number of times a determined state initializes an observation sequence. With this we can compute $P(i_k| < s >)$, i.e. the probability of a given state initializing the observation sequence. This probability can be defined as $P(i_k| < s >) = \frac{count(i_k|<s>)}{count(<s>)}$, i.e. one counts the number of times a given state initializes an observation sequence, and divides it by the the number of observation sequences in the training data. This way one computes π, i.e. the state initialization probabilities. In the example above, since we only have one observation sequence, the initialization probabilities would be $P(Other| < s >) = 1$, and every other initialization probability would be equal to zero. This would again present a problem, due to the fact that we may have other observation sequences that can initiate with a location, and with this probability distribution we would never recognize it as such.

To solve the *zero-probability* problem, one can use smoothing techniques. A specific technique, called **Laplace smoothing**, consists of adding one to the count in the numerator part of the

fraction, and N (e.g., number of states available) to the count in the denominator part of the fraction, when calculating the probability. This way, one can assure that the probability of a certain state (which has not been observed initializing any observation sequence) initializing the observation sequence would not be zero.

After the model is defined, it is important to know how to solve two different problems, namely i) given an observation sequence *O*, and the model λ, how to know the probability of observing that sequence, i.e. $P(O|\lambda)$, and ii) given an observation sequence *O*, how to know which is the optimal state sequence that could have generated the sequence. The second problem is the most important in the context of NER.

When one wants to find the probability of an observation sequence *O*, given a model λ, the most straightforward way of calculating $P(O|\lambda)$ is through enumerating every possible state sequence with a lenght equal to the number of observations. This would result in Equation 2.1, where I represents the set of possible states.

$$P(O, I|\lambda) = \sum_{allI} P(O|I, \lambda) \times P(I|\lambda) \tag{2.1}$$

This means that one can calculate the probability of any state sequence generating the observation sequence, which is defined by the probability of observing such observation sequence with a specific state sequence.

To calculate the probability of O, one needs to sum the joint probability of observations and states, over all possible state sequences. To facilitate the calculation of $P(O|\lambda)$, one can use the **forward procedure**, which uses dynamic programming to calculate $P(I_k|O_{1:k})$, i.e. the probability of being in state I_k after generating all symbols in the sequence up to $O_{i,k}$, assuming that k can take any position from the beginning to the end of the observation sequence. After applying the forward procedure, we can sum the probabilities for all possible states in the last position of the sequence, thus obtaining the probability of O.

By calculating $argmax(P(I|O))$, one calculates the best state sequence that could have generated the observation sequence. This is done by using the **Viterbi Algorithm** (Forney, 1973). For the purpose of explaining the algorithm, let us assume, as an example, that we want to know the best state sequence for the observation sequence *Located in Lisbon*, and that the available states are *Local* and *Non-local*. The Viterbi algorithm consists of four steps, namely i) initialization, ii) recursion, iii) termination, and iv) path backtracking. The initialization step consists of, as the name implies, initializing the variables used in the algorithm, through the conditions which are stated in Equations 2.2 and 2.3.

$$\delta_1(i) = \pi_i b_i(o_1), 1 \leq i \leq N \tag{2.2}$$

$$\Psi_1(i) = 0 \tag{2.3}$$

In the formulas, $\delta_1(i)$ represents the probability of state i having generated the first word of the observation sequence, which we can get from $\pi_i b_i(O_1)$. In the example, $\delta_1(i)$ is the probability of state i (1 = *Local*, 2 = *Non-local*) generating the symbol *Located*. $\Psi_1(i)$ represents the most probable state that generated the previous observation symbol, which is undefined in the initialization step because we are observing the first observation symbol.

The recursion step is a little more complex. It consists on iterating over the observation sequence, calculating the most probable state that could have generated each of the observation symbols. This is stated in Equations 2.4 and 2.5.

$$\delta_t(j) = max_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}]b_j(o_t); 2 \leq t \leq T; 1 \leq j \leq N \tag{2.4}$$

$$\Psi_t(j) = argmax_{1 \leq i \leq N}[\delta_{t-1}(i)a_{ij}] \tag{2.5}$$

From these equations we can see that the probability of state j generating symbol o_t is equal to transition from the most probable previous state (the one that most probably generated o_{t-1}) and multiplying it with the probability of observing the actual symbol o_t in the current state. One can see that $\delta_{t-1}(i)$ depends on the δ of previous symbols, hence the recursion, implying that the calculations of the probability of state j observing the current symbol is affected by the most probable state sequence that observed the previous symbols of the observation sequence. $\Psi_t(j)$ contains the identifier (i.e., j) of the most probable state that could have generated the current symbol o_t. The Ψ variables will be helpful in the path backtracking step.

The termination step happens when one reaches the end of the observation sequence. It consists basically of the recursion step for the last observation symbol.

$$P^* = max_{1 \leq i \leq N}[\delta_T(i)] \tag{2.6}$$

$$i_T^* = argmax_{1 \leq i \leq N}[\delta_T(i)] \tag{2.7}$$

From Equations 2.6 and 2.7, one knows that the probability of observing the observation sequence is P^*, which takes into account the most probable path until the end of the sentence, where it calculates δ_T of all states i and chooses the most probable one. The variable i^* contains the identifiers of all the states in the most probable state sequence.

At last, the algorithm executes the path backtracking step, which consists on recursively iterating over i_t^*, which contains the state identifiers of the most probable state sequence.

$$i_t^* = \Psi_{t+1}(i_{t+1}^*); t = T-1, T-2, ...1 \tag{2.8}$$

This means that, from the end of the sequence to the beginning (i.e., from the last observed symbol to the first), one enumerates the most probable state that could have generated each of the observation symbols, ending with the most probable state sequence that could have generated the observation sequence. The Viterbi algorithm is similar in implementation to the Forward-Backward dynamic programing procedure, but instead of summing the probability of previous states in the calculation of the probability $P(o_t|i_n)$, one uses a maximization operation, counting only the most probable state i at each iteration step.

2.1.3 Conditional Random Fields

According to Lafferty *et al.* (2001), CRFs are a framework for the construction of probabilistic models to segment and label data given in sequence. According to Sutton & McCallum (2011), CRFs are essentially a way of combining the advantages of discriminative classification and graphical modeling, combining the ability to compactly model multivariate sequential data with the ability to leverage on a large number of input *features* for prediction. A CRF can loosely be understood as a generalization of an HMM that makes the constant transition probabilities into arbitrary functions that vary across the positions in the sequence of hidden states, depending on the input sequence.

Suppose we have an observation sequence O as an input, and assume each observation symbol o from O is a vector of dimension *f*. We want to find the probability of classifying each symbol o with a class y from a set Y of classes. This translates into a conditional probability $P(Y|O)$. A CRF is basically a conditional probability associated with an undirected graphical model (Klinger & Tomanek, 2007). When one uses local *features* to calculate a probability $P(Y, O)$, as in HMMs, one is forced to calculate $P(O)$, which can contain complex dependencies between *features*, making this a very expensive procedure. By calculating $P(Y|O)$, which is enough for classification problems, and since this is a conditional probability, one does not need to calculate $P(O)$, enabling the possibility of using richer, global *features*.

Suppose one wants to classify the words in a phrase into one of the *F* classes available. One can use the *Linear-Chain CRF*, where the CRF is structured in a linear chain. In Figure 2.4 one can see an undirected graphical model representing a Linear-Chain CRF.

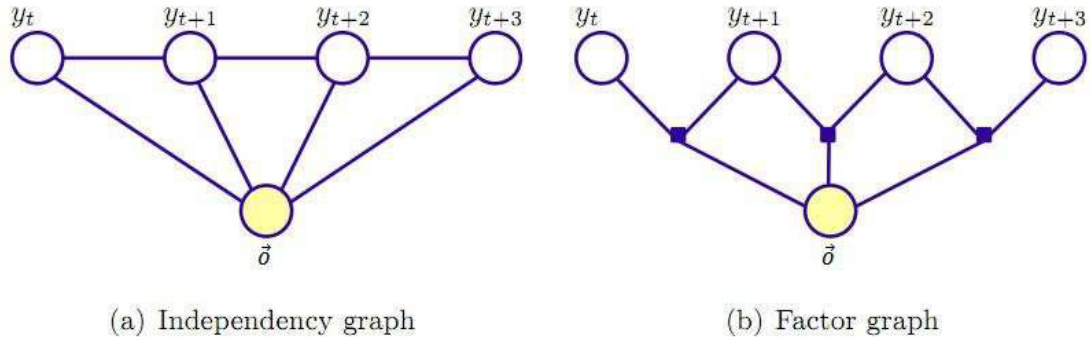

Figure 2.4: Undirected graphical model representing a Linear-Chain CRF (Klinger & Tomanek, 2007)

In a sequence tagging problem, such as POS tagging or NER, CRFs carry out logistic regression over the possible sequences of tags. One can model the conditional distribution $P(Y|O)$ as shown in Equation 2.9:

$$P_\lambda(\mathbf{y}|\mathbf{o}) = \frac{1}{Z_\lambda(\mathbf{o})} \times \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\left(y_{j-1}, y_j, \mathbf{o}, j\right)\right) \tag{2.9}$$

Index *j* is needed because a label sequence is considered instead of a single label to be predicted, and it specifies the position on the input sequence for o. The parameters λ_i are a set of real-valued weights, typically estimated from the labeled training data by maximizing the data likelihood function through stochastic gradient descent, which are not dependent on position *j*. The parameter Z is a normalization constant, and is defined in Equation 2.10. The summation over Y is performed to get a feasible probability.

$$Z_\lambda(\mathbf{o}) = \sum_{\mathbf{y}\in Y} \exp\left(\sum_{j=1}^{n}\sum_{i=1}^{m} \lambda_i f_i\left(y_{j-1}, y_j, \mathbf{o}, j\right)\right) \tag{2.10}$$

A linear-chain CRF has many applications similar to those of HMMs, although it is more flexible than the latter. A CRF can be understood as a generalization of HMMs, where the constant transition probabilities of the latter are replaced by arbitrary functions which vary throughout the positions in the sequence of hidden states (*labels*), depending on the input sequence. When training a CRF model, one needs the training data to be annotated with the *features* that are going to be considered. The y_i classes that characterize the observation sequence are structured, in order to form a chain, where each y_i that depends on o is defined by the set of feature functions. In terms of inference over a CRF model, one can use an adaptation of the Viterbi algorithm, described in the previous section. For more information about CRFs, please refer to the original

paper by Lafferty *et al.* (2001).

2.2 Resolving Place References in Text

Extracting geographical information from texts is a non-trivial task, not just because of the complexity of NER, explained in the previous section, but also because of the many ways in which a location can be referenced in text. There are essentially two different types of ambiguity problems with the usage of geographical references in textual documents, namely (i) geo/geo ambiguity and (ii) geo/non-geo ambiguity.

The geo/geo ambiguity problem happens when, for example, two cities in different locations have the same name (e.g., the *United States of America* alone has 14 cities named *Lisbon*). The geo/non-geo ambiguity happens when a name can not only be referencing a city or location, but some other kind of entity, like a person. For example, *Lisbon* can be referencing a city in Portugal, or a person's name. These kinds of disambiguation problems have to be addressed by evaluating the context of the text where the reference appears.

Locations can also be absolute (such as *Rome*, or *Los Angeles, CA*) or relative (such as *30 km east of Lisbon* or *2 miles down the road of Essex*), thus making it difficult to find a normalized way of recognizing the exact location being referenced. To help standardize the manual annotation process, Mani *et al.* (2008) developed an annotation scheme called *SpatialML*. This annotation scheme's main goal is to mark places mentioned in text and map them to data from gazetteers and other databases. This is done by a set of labels which covers tasks such as marking a location (i.e., a SpatialML tag named *PLACE*), marking relative information (i.e., a tag named *SIGNAL*) or describing the exact path to the referenced location (i.e., a tag named *PATH*). The authors of the SpatialML scheme also developed an automated method for annotating texts with SpatialML tags. The authors report results ranging from 0.72 to 0.82 in terms of the F_1 measure for the recognition of location references, and of 0.93 in terms of F_1 measure for the disambiguation of the recognized locations.

According to Leidner (2008), there are some fundamental heuristics that one can find useful to adapt when working with problems involving spatial reference resolution. The core idea behind these heuristics is that when ambiguous place names are used in conversation or in text, it is usually clear to both the author and the reader what specific referent is intended, either by a *speaker* providing more identifying information about a location that the person listening is believed to be unfamiliar with, or by linguistic context clues. In his work, the heuristics were grouped into various topics, according to their way of addressing the toponym recognition problem. Here

we survey the most important of these heuristics, taking inspiration on Leidner's work and also on some other previous works in the area.

In terms of **Spatial Distance**, the considered heuristics are presented below.

- **Spatial/Geometric minimality** assumes that, when there is more than one toponym mentioned in some span of text, the smallest region that can group the whole set is the one that gives them their interpretation, which can be used to resolve referential ambiguity by proximity. For example, if we find the reference *Paris* in a text, we can assume it is referent to either France's *Paris* or USA's Texas' *Paris*, but if in that same text we find references to *Lyon*, *Nice* and *Bordeux* then, *by proximity*, we can assume that the *Paris* referenced is the one in France, because this way we would be disambiguating according to *the minimal region that englobes all other references*.

- **Focus on geographic area** assumes that, if the source of the documents has some geographic focus (e.g., the source of the documents is a local newspaper of a certain city), all place reference candidates that lie outside of this focus region should be discarded. A geographic focus can also be computed for a single document. Considering a set of unambiguous place references in the document, all the place reference candidates that lie outside this focus are discarded. For example, for a static focus scenario, a system that is dedicated to the analysis of Portuguese news would have a gazetteer with locations outside Portugal filtered out, so one can increase the precision of the system, as far as locations inside Portugal are concerned. For the dynamic focus scenario, assume one wants to find *Springfield*. By analyzing a document whose text concerns with a non-USA territory, one can dynamically filter the American *Springfield* and reduce the list of possible referents.

- **Distance to unambiguous textual neighbors** assumes that a toponym is surrounded by a space of unambiguous toponyms, in either side of the discourse. One assigns the referent that is geographically closest to all of the other referents for the toponym in question. The usefulness of this heuristic is inversely proportional to the size of the gazetteer, and it is not applicable in a situation where there are no unambiguous toponyms.

- **Discard off-threshold** can be seen as the dynamic version of *Focus on geographic area* heuristic. One computes the centroid for the toponyms mentioned in the document, and eliminates all candidate referents whose distance are more than 2 standard deviations away from this centroid.

In terms of the considered **Textual Heuristics**, they are as follows:

- ***Contained-in* qualifier** assumes that, if a system recognizes a pattern in the text (such as t_1, t_2, t_3 or $t_1\ in\ t_2$), and there is exactly one candidate referent of t_1 which is contained in a referent for t_2, then one assigns that reference to t_1, and should proceed accordingly to resolve t_2 and t_3, should they exist. For example, if a text contains the expression *Lisbon, Portugal*, and there is exactly one candidate referent to *Lisbon* that is *inside* one referent for *Portugal*, then one assigns that referent to *Lisbon*, and continues (recursively) to resolve the other toponyms in that expression (in the case there are more than two).

- **Superordinate mention** is essentially a *long-distance*, text-wise, version of the previous heuristic. If a toponym is to be resolved, and if a second toponym, which refers to a country, appears elsewhere in the same document, and it is believed that a referent of the first toponym is located in the country of the second toponym, then one assigns the first toponym to the referent in that country. For example, if in a text there is a toponym *Lisbon* and elsewhere in the same document there is a toponym *Portugal*, which is a country, and it is believed that *Lisbon* is inside the country *Portugal*, then we assign that referent to the toponym *Lisbon*.

- **One-referent-per-discourse** assumes that a place name mentioned in a discourse refers to the same location throughout the same discourse, which means, algorithmically, that a resolved toponym can be seen to propagate it's interpretation to other instances of the same toponym in the same discourse or segment. For example, if a text contains the references *London* and *UK*, we assume that, throughout that discourse, *London* will always reference England's *London*.

- **Frequency weighting** assumes that one gives higher importance to more frequent toponyms in a text. This *meta-heuristic* can be applied with other heuristics when comparing multiple toponyms, especially when avoiding ties between them.

- **Feature type disambiguator** assumes that, when a pattern is recognized between a toponym and a term which indicates a feature type (*city*, *capital*, *country*, and so on), one should eliminate the candidate referents that are known *not* to be of the feature type encountered. For example, if one finds *city of Scotland*, one can eliminate the UK country referent.

- **Textual-Spatial Correlation** assumes that textual proximity is strongly positively correlated with spatial proximity. This can be applied on toponym-toponym relations, but also on toponym-term relations, such as finding the toponym *Calcutta* and the term *Theresa* close to each other, which leads one to believe the *Calcutta* mentioned is the one in India, due to the implicit meaning behind the term *Mother Theresa*. For example, if we find *Paris*

and *France*, or *Paris* and *Versailles* in close textual proximity with each other, then one assigns France's *Paris* to the toponym.

In terms of the considered **Candidate Heuristics**, they are as follows:

- **Assign unambiguous referent** assumes that all referents are assigned to toponyms where there is exactly one candidate, i.e. there is no ambiguity between references in the text. This should be the first disambiguation step to be taken in this kind of systems.

- **Largest population** assumes that the referent with the largest population size is the most probable to be the correct one. For example, when one finds *Paris* in a discourse, one can assume that it means *Paris, France*, which has a larger population than *Paris, TX, USA*.

- **Ignore small places** is more of a simplification of the problem than actually a solution, since it reduces ambiguity, but also recall, although it can be useful on some applications. The main objective is to reduce the list of possible referents for one toponym by eliminating the referents with a small number of inhabitants. Imagine, for example, that there are 35 possible candidates for *Washington* in the USA. If one eliminates the candidates with less than a half a million inhabitants, only one candidate would be left, which is *Washington, DC*, capital of the USA.

- **Singleton capitals** assumes that if a toponym is unique in a text, i.e. it only appears once, and if one of the candidate referents is a capital, then one should choose this capital as the reference for the toponym in question. For example, if the toponym *Madrid* appears only once in a text, one assumes that it means *Madrid, Spain*, due to it being the capital of the country.

- **Prefer higher-level referents** assumes that, when a toponym can refer to various candidate referents, one chooses the referent with a *higher-level* category, i.e. we assume that a wider-range referent is the correct one. For example, if one finds *Africa* in a text, the continent Africa would be assigned to the toponym, rather than the cities *Africa, Mexico*, *Africa, Indiana, USA*, or *Africa, Ohio, USA*.

- **Default Referent** assumes the use of existing knowledge about the most salient referent in a discourse and assings referents accordingly, which means there is a direct relation to human intuitions about the salience. For example, one can assume the *Washington* reference refers to the capital city of *Washington, DC* instead of the state of *Washington*, although the latter has more inhabitants.

- **Preference order** assumes that, given a set of different gazetteers covering the same geographic area, the more gazetteers that a given toponym occurs in, the more salient it will typically be, but in the case that there are many different references (for example, one references a city and another references a country) in the gazetteer set, the most important gazetteer's reference will be chosen. This implies that, in the application, an importance ranking must exist between the gazetteers.

In terms of entity disambiguation with machine learning techniques, a regression model can be used to disambiguate the candidates of a specific toponym (Loureiro *et al.*, 2011; Martins *et al.*, 2010). The idea is to generate disambiguation candidates for a given spatial reference, calculate their set of features, and feed that information to the regression model, so that it can rank the candidates and choose the one that is most probably the best one for the reference in question.

2.3 Evaluation Methods for Entity Recognition

The evaluation of information extraction systems is usually made by comparing the results of the automatic assignment done by systems, against the manual assignments done by human experts. This evaluation can be done at the token or the chunk levels. The first method states that the evaluation is done token by token, i.e., each token is verified to see if, for example, it is part of a location or not. The second method states that the evaluation is done chunk by chunk, i.e., at the level of segments comprised of multiple tokens, and the system verifies if, for example, that chunk forms an entire entity string.

A spatial reference will be considered as correctly identified only if it is an exact match with the corresponding spatial reference given in the test collection, and correctly disambiguated only if the assigned disambiguation is an exact match with the one that is given in the human annotation. Thus, correct disambiguation can only occur when a correct recognition has first taken place, since it does not make sense to disambiguate textual expressions that are not recognized as spatial references. The measures that can be used to evaluate place reference resolution systems are Precision, Recall, the F-Measure, and Accuracy.

Precision is the ratio that represents the returned information that is relevant to the user's need, i.e. the number of correctly-identified locations, i.e., the $TruePositives$, divided by the total number of identified locations, i.e., the $TruePositives$ plus the $FalsePositives$. To calculate it, one uses the formula below.

$$P = \frac{Correctly\_IdentifiedLocations}{TotalIdentifiedLocations} \tag{2.11}$$

As an example, suppose we have an IE system that can identify locations in documents. Suppose we have a total of 9 locations in a document, and our IE system identifies 5 locations and 3 persons as locations. The Precision would be $P = \frac{5}{8} = 0.625 = 62.5\%$.

Recall is the ratio that represents the returned items that are relevant to the query that are successfully retrieved, i.e. the number of correctly-identified locations, i.e., the $TruePositives$, divided by the total number of locations that should have been identified, i.e., the $TruePositives$ plus the $FalseNegatives$. To calculate it, one uses the formula below.

$$R = \frac{Correctly\_IdentifiedLocations}{TotalExistingLocations} \tag{2.12}$$

In the example above, assuming our IE system still identified 5 locations and 3 persons as locations, from a total of 9 locations in the document, then the Recall would be $R = \frac{5}{9} \approx 0.56 = 56\%$.

For one to realistically assess an IE system's performance, one needs to combine both the Precision and Recall measures. The **F1 Score** can be interpreted as a weighted average of both precision and recall. The F1 score is helpful to accurately assess an IE system's overall performance, as it combines precision and recall. To calculate it, one uses the formula below:

$$F1 = 2 \times \frac{P \times R}{P + R} \tag{2.13}$$

In the example used above, the F1 score for the IE system that recognizes locations in documents is $F1 = 2 \times \frac{0.625 \times 0.56}{0.625 + 0.56} = \frac{0.7}{1.185} \approx 0.591 = 59.1\%$.

Finally, the most important measure for this work will be accuracy. **Accuracy**, as described by Han (2005), can be calculated as shown in Equation 2.14, which corresponds to the ratio of correct classifications in the entire analyzed corpus.

$$Accuracy = \frac{Correctly\_ClassifiedLocations}{TotalLocations} \tag{2.14}$$

2.4 Annotation Schemes for Geospatial Expressions

Over the years, several annotation schemes for marking entities in textual documents have been developed. A well-known annotation scheme for geospatial expressions is the SpatialML scheme, developed by Mani *et al.* (2008). This annotation scheme's main goal is to mark places mentioned in text and map them to data from gazetteers and other location databases. This is done by a set of tags and attributes which cover several tasks. The tags used in SpatialML are as follows:

- PLACE: This tag is used to delimit and identify a string as a location mentioned in text (e.g., *U.S* or *building*).
- LINK: This tag has two possible variations:
  - RLINK tags: Links which relate relative locations to absolute ones. It is used when, for example, one describes a *building 5 miles east of Lisbon*.
  - LINK tags: Links which relate locations to each other while recording the type of topological relation involved. It is used when, for example, one describes a *building in Aveiro*.
- SIGNAL: This tag is used to identify the distance or direction of a relative path (e.g., *5 miles east of Aveiro*).

The attributes that can be used in the PLACE tag are as follows:

- latLong: This attribute records the geospatial coordinates of a marked location.
- mod: This attribute is used to resctrict the orientation of a specific location, e.g., *southern New York* can be indicated by a *mod="S"* attribute on the markup tag.
- form: This attribute indicates the form of reference in the location mentioned, which can be a proper name (type NAM) or a nominal (type NOM).

The attributes that can be used in the RLINK tag are as follows:

- source: This attribute indicates the id of the tag which describes the starting point of a relative path.
- destination: This attribute indicates the id of the tag which describes the ending point of a relative path.
- direction: This attribute indicates the direction of a relative path.

- distance: This attribute indicates the id of the tag which describes the distance of a relative path.
- signals: This attribute indicated the ids of the SIGNAL tags which are part of the description of a relative path.

The attributes that can be used in LINK tag are as follows:

- source: This attribute indicates the id of the tag which describes the starting point of a relative path.
- target: This attribute indicates the id of the tag which describes the ending point of a relative path.
- signals: This attribute indicates the ids of the SIGNAL tags which are part of the description of a relative path.
- linkType: This attribute indicates the type of link between the source and the target. The values which it can assume are as follows:
  - Disconnection (DC): Indicates that the source has a disconnected relation with the target, e.g., *the well outside the house*.
  - External Connection (EC): Indicates that the source and target have a connection, but not a direct connection, e.g., *the border between Portugal and Spain*.
  - Equality (EQ): Indicates that the source and target are the same location, e.g., *Lisbon and Lisboa*.
  - Partial Overlap (PO): Indicates that some part of the source location is included in the target location, e.g., *Russia and Asia*.
  - IN: Indicates that the source is included in the target, e.g., *Paris, France*.

The tags that were used in this work are the PLACE tags, with a special focus on the *form* attribute for evaluating the recognition of geospatial expressions, and the *latLong* attribute for evaluating the disambiguation.

2.5 Summary

In this chapter, an overview on fundamental concepts related to information extraction and retrieval was given, together with a review on the most important concepts on the subject of place reference resolution in text.

As we can see from the contents of this chapter, machine learning methods are well suited for the problem of geospatial information resolution, constituting the basis for the development of portable and robust systems. An explanation of Support Vector Machines, Hidden Markov Models, and Conditional Random Fields was given.

Also in this chapter, we saw an explanation of the evaluation methods that are more commonly used in the entity recognition area, as well as a presentation for the annotation schemes that are more frequently used in the area of geospatial information resolution.

Chapter 3

Related Work

This section presents the most relevant previous works in the context of this dissertation. Section 3.1 presents work related to advanced techniques that can be used in sequence classification problems. Section 3.2 presents work related to the problem of recognizing place references in text, and to the problem of disambiguating these place references.

3.1 Advanced Models for Sequence Classification

Besides HMMs and CRFs, there are other advanced models for sequence classification in IE, which can help in building more accurate systems. This section presents a description of CRFs with high-order features, and of Semi-Markov CRFs. It also presents a previous work that combined CRFs and SVM models.

3.1.1 Conditional Random Fields with High-Order Features

Ye *et al.* (2009) explained how to learn and predict label sequences with CRFs using high-order *features* efficiently. When one wants to compute the most likely label sequence, one can use the Viterbi algorithm. Then one computes the marginals of labels sequences and single variables, i.e. computes $P(\boldsymbol{y}_{t-|\boldsymbol{z}|:t} = \boldsymbol{z}|\boldsymbol{o})$, where $\boldsymbol{z} \in \boldsymbol{Z}$, and where $\boldsymbol{Z}$ is the set of distinct label patterns used in the *features*, i.e. $\boldsymbol{Z} = \{\boldsymbol{z_1}, ..., \boldsymbol{z_M}\}$ with M corresponding to the number of *features*.

The authors tested if the high-order *features* were practically significant, i.e. if the changes introduced by the usage of *features* of a higher order would be beneficial to the system's performance. What the authors found was that, when using synthetic data generated using a k-Order Markov

process, the high-order indicator *features* were useful. When the observations were closely coupled with the states, the *feature* engineering on the observations was generally enough to perform well, being less important to use high-order *features*. The problem was when the coupling was not very clear (i.e., different states generated some similar observations). In these cases it becomes important to capture the label correlations, and therefore using high-order *features* is useful. In the case of a NER problem using 4 different classes, introducing 2^{nd} order *features* showed improvements in 10 out of 12 combinations, while degrading in the other 2. The average improvement was of 0.0062 in terms of the $F1$ score.

3.1.2 Semi-Markov Conditional Random Fields for Information Extraction

Sarawagi & Cohen (2004) presented inference and learning methods for semi-Markov CRFs (henceforth referred to as semi-CRFs), which are CRF models that relax the Markov assumption.

Assume we have a segmentation of the observation sequence $\boldsymbol{o}$ called $\boldsymbol{s} = \{s_1, ..., s_p\}$, where each segment $s_j = \{t_j, u_j, y_j\}$ is composed by t_j, which denotes the *start position*, u_j the *end position* and y_j the label given to all word tokens starting in t_j and ending in u_j. In a NER problem, when evaluating the sentence $\boldsymbol{x}$ = *Luis went to Campo Grande*, a correct segmentation could be $\boldsymbol{s} = \{(1,1,O), (2,2,O), (3,3,O), (4,4,BL), (5,5,IL)\}$, where the label sequence is $\boldsymbol{y} = \{O, O, O, BL, IL\}$, using BIO classes to classify word tokens as locations (note that, in this sentence, *BL* and *IL* denote *B_Location* and *I_Location*, respectively, which are the labels used in the example in Section 2.1.2). When one wants to compute the most likely label sequence, one can use a slightly changed Viterbi algorithm, where we want to find the best segmentation, i.e. $argmax_s P(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W})$, where $\boldsymbol{W}$ is a weight vector for the sum of all *segment feature functions* (i.e. *feature* functions applied to segments s). The best segmentation corresponds to the path traced in the last step of the algorithm.

The main difference from normal linear-chain CRFs to semi-CRFs is that, instead of trying to find $P(\boldsymbol{y}|\boldsymbol{x})$, in semi-CRFs we want to find $P(\boldsymbol{s}|\boldsymbol{x}, \boldsymbol{W})$, i.e. the probability of a given segment being generated given $\boldsymbol{x}$ as input and $\boldsymbol{W}$ as the *feature* functions to be considered. Also, the main difference between semi-CRFs and N-order CRFs is that semi-CRFs only consider sequences where the *same* label is assigned to all T positions, rather than all possible label combinations in segments of lenght T, leading to a faster inference.

The authors also described a learning algorithm, where the goal is to maximize the log-likelihood over a given training set. The authors experimented with NER data, comparing semi-CRFs against a baseline approach based on a linear-CRF-based model to classify entity segments

with a given label, and non-entity segments with a different label, and against a linear-CRF-based model to classify entity segments with four different labels, depending on where the word token appeared within an entity. In terms of *features*, the authors used indicators for specific words at a given position, or within a 3 word window of that position, together with word capitalization/letter patterns. The authors also implemented a number of dictionary-derived *features*, each based on a different dictionary. The results showed that semi-CRFs obtained slightly better results, outperforming both CRF variants on 8 out of 10 conditions, with a maximum difference of 0.09 in terms of the F_1-score. The authors also compared semi-CRFs with N-order CRFs by changing the dimensionality of both CRF systems to 2^{nd} and 3^{rd} order CRFs. The results showed again that semi-CRFs had better results, maxing at a difference of 0.05 F_1-score.

In the end, the authors concluded that semi-CRFs offered much of the power of a higher order CRF without the associated computational cost. They also concluded that a major advantage of semi-CRFs was the use of segment-property-measurement *features*, rather than individual elements.

3.1.3 Learning a Two-Stage SVM/CRF Sequence Classifier

Hoefel & Elkan (2008) implemented a two stage sequence classifier, where a first learner, based on SVMs, was trained to predict the label of each input sequence element, and a second learner, based on CRFs, was trained to predict the output sequence of labels, using the outputs of the previous learner as inputs. The idea was that, by combining the two learners, the results would be superior, due to the combination of both learners' advantages. The dataset used was an optical character recognition (OCR) dataset, containing 6876 words, and where each character image is labeled with one of 26 letters and has a size of 128 pixels.

During the SVM training, the goal was to predict each class based on each sequence element and it's label in the training set. This was done by maximizing the separation between data points with labels in the same class, and other data points. The advantage of SVMs comes from their ability to use high-dimensional feature spaces via kernels, although an important drawback is that is is typically hard to choose the settings for an SVM that obtain optimum results, in particular, the best value for the soft-margin penalty C. The output vector of scores from this model is then used as the input attributes for the CRF classifier.

In terms of the SVM multiclass classification problem, the authors chose to approach the problem with the one-against-one technique, where each class is trained separately against each other class. The authors believed that this technique i) yielded slightly more accurate results for the OCR data, ii) yielded faster SVM training, and iii) increased the bandwidth of information passed

to the CRF.

In order to evaluate the system, the authors compared the system's accuracy in two ways, namely by measuring i) accuracy per character, and ii) accuracy per word. The system was compared with three baseline methods. The SVM model was trained with the LIBSVM package, and it was compared using three different kernels (i.e., linear, quadratic, and cubic) to a logistic regression model. The CRF model was compared to a Standard CRF model, from the CRFSGD software package. The two-stage SVM/CRF classifier is compared to the Max-Margin Markov Networks (M3N) approach, using i) quadratic and ii) cubic kernels. The evaluation was done using cross-validation in two different ways, namely i) by using a small 10% training set in each fold, and ii) by using a large 90% training set in each fold. The authors found that each model alone achieved slightly better performances than the standard model used for comparison, and that the two-stage SVM/CRF method is better than either method alone, and almost as good as the M3N method. When using small training sets, the two-stage method achieved an accuracy per character of 0.88 with a quadratic kernel (+ 0.01 than the M3N method), and an average accuracy per character per word of 0.88 with a quadratic kernel (+ 0.13 than the M3N method). When using large training sets, the two-stage method achieved an accuracy per word of 0.95 with a quadratic kernel (- 0.01 than the CGM method), and an average accuracy per character per word of 0.95 with a quadratic kernel.

The authors concluded that the two-stage approach yields the same accuracy as mathematically more complex methods, being faster and more scalable than the competitors. The main conclusion is that a margin-based approach can extract the most important information about individual data points, while a sequential approach can augment the learning process by exposing the sequentially structured nature of the problem.

3.2 Place Reference Resolution in Text

This section presents three important previous works in the subject of resolving place references, namely an automated method for annotating texts with SpatialML tags built by the authors of SpatialML, a method based on a combination of multiple heuristics by Lieberman *et al.* (2010), and a method for extracting place entities and routes from mountaineering texts, introduced by Piotrowski *et al.* (2010). The section ends with a method for predicting the geospatial coordinates to assing a given document using language models, introduced by Wing & Baldridge (2011).

3.2.1 SpatialML: Annotation Scheme, Corpora, and Tools

Mani *et al.* (2008) described both the SpatialML annotation scheme and a system architecture used to resolve place references in text using the SpatialML schema. Using a SpatialML-annotated dataset, Mani et al. trained a statistical entity tagger, using a CRF model to classify entities in a document with *PLACE* tags, and a disambiguator, using a log linear model to rank the potential candidates from a gazetteer, using both the *features* from the *PLACE* tag and the gazetteer entry. *PATH* and *LINK* taggers were also considered, in order to recognize relations between *PLACE* entities. A *PLACE* tag can either refer to proper names (e.g., *New York*, annotated as *NAM*) or to nominals (e.g., *town*, annotated as *NOM*).

The log-linear learning model for disambiguation, which achieved an F-measure score of 0.93, used the cross product of each *PLACE* tag occurrence and all applicable gazetteer candidates for that mention, constructing a *feature* vector for each combination. For each candidate G_i for a *PLACE* mention M, a weight vector for G_i is computed, normalized for all other candidates for M. This is used to compute the probability of a candidate G_i being correct for a mention M, i.e. $P(G_i|M)$. The model then finds the maximal probability, i.e. the candidate which has the highest probability $P(G_i|M)$, and chooses that candidate as the correct entity for M. The *features* used are a combination of document, gazetteer and joint *features*.

In terms of document *features*, the considered *features* are presented below.

* Document id - The document identification.
* Mention string - The word token annotated with the *PLACE* tag.
* 10-word window - A window of 10 words on each side of the *PLACE* mention.
* First word - If the *PLACE* mention is the first in the document.

In terms of gazetteer *features*, the considered *features* are presented below.

* Gazetteer id - The gazetteer identification.
* Tag type - The type of the *PLACE* tag.
* State - The state of the *PLACE* mention, if applicable.
* Country - The country of the *PLACE* mention.
* Coordinates - The geospatial coordinates of the *PLACE* mention.

Finally, in terms of joint *features*, the considered *features* are presented below.

* Number of candidates - The number of gazetteer candidates for the *PLACE* mention.

* Gazetteer relation - If the parent/sibling of the gazetteer entry is in the document *features*.

In terms of entity recognition, the system achieved an F-measure of 0.85 in tagging names (i.e., SpatialML *NAM* entities), and an F-measure of 0.72 in tagging nominals (*NOM*).

3.2.2 Geotagging Documents with Local Lexicons

Lieberman *et al.* (2010) evaluated the usage of newswire reports to enable the construction of indexes based on toponyms, through geotagging. This was done in three steps, namely i) Inferring Local Lexicons, ii) Toponym Recognition, and iii) Toponym Resolution. Lieberman et al. stated that an audience's local lexicon plays a key role in how news authors write for their audiences, which translates into the need for an automated and scalable method for the extraction of local lexicons from online news sources (including not only online newspapers, but also blogs and Twitter messages). The key characteristics to automatically infer local lexicons, considered by the authors, were i) stability, ii) proximity and iii) modesty.

Stability indicates that a local lexicon is constant across articles from its news source. This can be done by observing and analyzing toponyms from a collection of articles from a news source, thus being able to determine the local lexicon as a common geographic theme among these articles. This can be observed for both local and global lexicons. **Proximity** indicates that toponyms in a local lexicon are geographically close. If and only if the toponyms within the spatial lexicon are geographically close, then we can assume to be in the presence of a local lexicon (which implies that this property is used as a means of filtering and validating an audience's local lexicon). Finally, **modesty** indicates that a local lexicon contains a considerable, but not excessive, number of toponyms. It would be rare for a person to know only one or two local toponyms, and therefore this property highlights the notion that a person's local lexicon should at least contain several toponyms, while limited geographically. This can be enforced by specifying a minimum size to the local lexicon.

In terms of inferring local lexicons, the task is analogous to geotagging a single article, but in large scale for a news collection. This, however, presents a bootstrapping problem, since determining a local lexicon relies on the correct geotagging of individual articles in the collection, and correct geotagging relies on knowing the local lexicon. To break this dependency cycle, a geotagging process, termed *fuzzy geotagging*, is used, which does not fully resolve toponyms in a single article, but instead returns a set of possible interpretations for ambiguous toponyms. *Fuzzy geotagging* can be compared to a variant of traditional heuristic-based geotagging, where

the final *default sense assignment* of the traditional geotagging is removed, instead assigning a weight to each interpretation for the toponym in question, and summing the weights across all articles. The candidates with the largest weight across the most articles are considered as part of a potential local lexicon. If these candidates are within a geometrical polygon which delimitates the geographical limit for the local lexicon, then they are added to the local lexicon, otherwise they are left out. When the local lexicon achieves a certain minimum size limit, the lexicon is returned to the user.

The toponym recognition task is particularly challenging, due to the fact that many names of places are also common names for people and for other entities (i.e., the *geo/non-geo ambiguity* stated earlier in this paper). Lieberman et al. did not concern themselves too much with the toponym recognition performance, instead focusing on the whole geotagging system's performance, assuming the recognition as just a means to an end. To enable toponym recognition, Lieberman et al. used the GeoNames *gazetteer*, a database of geographic locations with geographic coordinates and associated metadata. The GeoNames gazetteer contained, at the time, almost 7 million entries, with a variety of metadata such as feature type, population, classification within a political geographic hierarchy, etc. Lieberman et al. made use of standard approaches for solving problems from the natural language processing domain, such as POS tagging and pre-existing NER models, from which the toponym recognition task is a part of. These approaches use statistical machine learning methods to train models from annotated language, which are used to determine the most likely sequences of parts of speech, or to extract sets of named entities. NER is a more general task than geotagging, and most NER systems sacrifice recall in favor of precision (i.e., they miss many toponyms, so that the ones that are reported are valid). NER systems are also, usually, trained on *corpora* of tagged newswire text containing only few less-prominent toponyms. Given this, the authors concluded that the toponyms in a NER training corpus serve only as a very limited gazetteer. With this in mind, Lieberman et al. used a hybrid toponym recognition technique which consisted of, given a news article, tagging each word with POS tags, collecting then all word phrases consisting of proper nouns. They also collected all phrases tagged as locations using a statistical NER system and probable toponyms using rules based on some toponym recognition heuristics, specified later in this text. They also collected frequent geographic *cue words*, such as *X-based* or *city of X*, which are strong indicators of a geographical toponym. For each of these collected phrases, the gazetteer was queried, and phrases that exist in it as toponyms were collected. This results in many more toponyms being reported than the ones that actually exist in the article, due to the ambiguity of these toponyms, as well as potentially misleading contextual geographic clues (resulting in low toponym precision). Still, at this stage, the most important thing to do is to maximize recall, since local lexicons serve

as a powerful way to filter precision errors.

In terms of toponym resolution, Lieberman et al. resolved toponyms using a set of heuristics, inspired by how humans read news articles. The heuristics were applied by the order from the list shown below (i.e., first H_1, then H_2, and so on), meaning that when a toponym can be resolved using various heuristics, the candidate suggested (from the gazetteer) by the highest ranked heuristic is the one chosen. The considered heuristics for the resolution of place references in text are as follows:

* H_1 **- Dateline**, where dateline toponyms were resolved using H_4, H_5 and H_6, and other toponyms were resolved according to the proximity to the resolved dateline. If present, it appears at the article's beginning and establishes the general geographic locality of the events described in the article, e.g., *LONDON, Ont. - A police squad (...)*.

* H_2 **- Relative Geography**, where an anchor toponym was resolved using H_1, H_4, H_5 and H_6, and other toponyms were resolved according to the proximity to the anchor toponym geographic point or region. Certain phrases in article text denote relative geography, which is language that defines a usually imprecise geographic region in terms of distance from, or proximity to, another geographic location. These regions are important because they usually target the geographic areas where the events in an article took place, e.g., *4 miles east of Athens, Texas*.

* H_3 **- Comma Group**, where a toponym group was resolved using H_5 and H_6, and geographic proximity. Comma groups are defined by the authors as lists of toponyms in articles which are of frequent occurrence, organized into concise groups when they share a common characteristic, e.g., *California, Texas and Pennsylvania*.

* H_4 **- Loc/Container**, where toponym pairs were resolved with a hierarchical containment relationship. Authors commonly provide contextual evidence for a toponym by specifying its containing toponym, in terms of a geographic hierarchy, e.g, *College Park, Maryland*.

* H_5 **- Local Lexicon**, where toponyms were recognized using geographically proximity to the local lexicon centroid. First one computes the geographic centroid of the documents' inferred local lexicon, and then one resolves the toponyms which are geographically proximate to the centroid.

* H_6 **- Global Lexicon**, where toponyms were recognized using a curated list of globally known places. The list is composed of toponyms which one regards as prominent enough to be known by audiences, regardless of their geographic location.

* H_7 **- One Sense**, where toponyms were recognized using shared names with earlier recognized toponyms. If all other heuristics fail to resolve a toponym, one assumes all instances of a repeated toponym will have the same resolution.

A particularly interesting contribution from this work is the LGL dataset containing a collection of news from smaller news sources, which can be used to assess the accuracy of toponym recognition systems.

In a following work entitled *Multifaceted Toponym Resolution in Streaming News*, Lieberman & Samet (2011) explained in detail the various steps in the toponym recognition algorithm, including the usage of entity tables, entity dictionaries, NER, and proper nouns to identify toponyms, followed by a method of toponym filtering, in order to facilitate the disambiguation phase. The toponym filtering part involves the usage of toponym refactoring, active verbs, noun adjuncts and type propagation within the entity dictionary, in order to filter out potential recognition errors. The authors concluded that this toponym recognition method is particularly suited for the streaming news domain, and that due to the ever-changing nature of the domain, it is advisable not to use only methods based on static corpora of news. The method proposed in the paper outperformed the competition in toponym recognition recall, which to the authors is the crucial method for success in place reference resolution methods.

The main difference between this work and my MSc thesis proposal is that, while Lieberman et al. used heuristics in the toponym resolution phase, the system that I developed in the context of my thesis uses a machine learning approach (i.e., a regression model) in order to disambiguate candidates.

3.2.3 Towards Mapping of Alpine Route Descriptions

Piotrowski *et al.* (2010) investigated toponym resolution on mountaineering texts, with basis on a digitalization of all journal issues and yearbooks from the journal *Die Alpen/Les Alpes/Le Alpi*, which contained mountaineering reports from 1864 until 2010. The journals that were digitalized at the time of the article composed a *corpus* of 6300 articles, a total of 20 million word forms. These journals were published in German and French versions, and the texts constitute a unique literary genre called *Alpine Literature*, which contained fiction (prose or poetry) and non-fiction work on mountaineering (e.g, actual mountaineering route descriptions), where the non-fiction work is the most important for researchers working in geographic place reference resolution. These non-fiction works contained not only a location, but how that location was reached (i.e., the route), whose automatic extraction was the primary challenge. The methods for identification and extraction of place references and route composition were not fully disclosed, but the problems

encountered were the main focus of the article. One problem was the small area that a text would cover, which meant that many small *features* would be mentioned (ridges, foot passes, etc..), which would contribute to a detailed route, but would also imply an overhead in processing. The first experiments with a naive approach based on a lexicon yielded low recall (i.e., lower than 40%), which was explained by the many different names a location could be mentioned with (in an official map, a location would have the local name, but in the reports studied the location names would appear in the language of the text, not mentioning historical spellings or the optical character recognition errors).

In terms of route extraction, the main task was to *connect the dots* once the toponyms had been resolved. The greatest problem in this task was recognizing the referents which were part of the route, and the referents that were landscape/panorama related, which did not contribute directly to the route. In this situation, the standard toponym resolution methods, based on minimum bounding rectangles and centroids, were not helpful, miscalculating the focus of the texts due to the high number of irrelevant points. There are many ways one can maximize the accuracy of the route extraction. For example, if there is a massive number of toponyms in a paragraph, and especially if the distances between them are relatively large, then this paragraph would very probably describe a landscape/panorama. But, at a sentence level, when the authors would be subjects in sentences, being the toponym the object and the verb indicates movement/position, the toponyms included in the sentence would most probably be relevant for the route. Verbs of vision would still need a further analysis, since they could appear in both cases.

In the end, the authors concluded that mountaineering accounts are a special text genre, which contains numerous geographic references, and where toponym properties and the style of the texts pose specific challenges to geocoding, particularly in the distinction between toponyms pertinent to a route description and those mentioned in panorama descriptions. The authors also concluded that preliminary experiments suggested that spatial information alone was not sufficient for this task, and that it is particularly important to explore the use of natural language processing in order to find clues to distinguish between the two types of references.

3.2.4 Supervised Document Geolocation with Geodesic Grids

Wing & Baldridge (2011) investigated the automatic geocoding of entire documents, instead of geocoding place references, and described several simple supervised methods for document geocoding, using the raw document text as evidence and a geodesic grid as a discrete representation of the Earth's surface, where discrete cells representing areas on the Earth's surface. The idea is that new documents can be geocoded to the most similar cell. The authors use four

probability distributions to represent the information in the Earth's grid representation, namely i) the distribution over the vocabulary for every cell in the grid, i.e. for every word w_j in a vocabulary V, and for every cell c_i in the grid, compute $P(w_j|c_i)$, ii) the distribution over the vocabulary in a document, i.e.for a document d_k, compute $P(w_j|d_k)$, iii) the distribution over the grid's cells, i.e. $P(c_i|w_j)$, and iv) the distribution over the cells, i.e. $P(c_i)$.

The dataset used in this work consisted of a combination of Wikipedia annotated documents and Twitter tweets. The Wikipedia corpus dump contained 3,431,722 useful articles, of which 488,269 were geotagged. A sample of 80.000 articles from this corpus was split into training, development and testing sets. The other part of the dataset consisted of geotagged tweets, containing 380.000 messages. Each user's tweets were concatenated into a document, and the location label of each document is the location of the first tweet that the user made. This part of the dataset was also split into training, development and testing categories.

In terms of representing the Earth's surface as a grid, Wing and Baldridge use a grid of cells of equal size (1º by 1º, for example), which produces variable-size regions which shrink latitudinally, becoming progressively smaller and elongated the closer they get towards the poles. This representation ignores all higher level regions, such as states, countries, etc., although it is consistent with the geocoding of both sources in the dataset. The geolocation methods that were proposed predict a cell for a given document, and the latitude-longitude of the degree-midpoint of the cell is used as the predicted location.

The authors showed that automatic identification of the location of a document, based only on the text, can be performed with high accuracy using simple supervised methods and a discrete grid representation of the Earth's surface. The most effective geolocation strategy finds the grid cell whose word distribution has the smallest KL divergence (a non-symmetric measure of the difference between two probability distributions P and Q) to the word distribution of the test document. The authors concluded that the task of identifying a single location for an entire document provides a convenient way of evaluating approaches for connecting texts with locations, but that it is not fully coherent in the context of documents that cover multiple locations. The authors acknowledged that an additional area to explore is to remove the *bag-of-words* assumption and explore the ordering between words, which would have a number of benefits, being sensitivity to multi-word toponyms (such as *London, Ontario*) an important one.

This article is important to this dissertation because of the approach used to predict the coordinates to assign to a given document, where the document in question is assigned approximate geographic coordinates through the analysis of probability distributions on it's raw text . This line of thought is important in the disambiguation step of the proposed system, as it could be used as a feature to help the candidate disambiguation process.

3.3 Summary

In this chapter, an overview of the most relevant related work focusing on the analysis of place references in text, either addressing the recognition problem or the normalization problem, was given.

In terms of work related to advanced techniques that can be used in sequence classification problems, we can see that machine learning methods are highly investigated for recognition problems, which results in various ways in how to approach said problems. Work where chain-CRF with high order features, and semi-Markov models used in sequence classification, is studied. A particular interesting work for this dissertation is the two stage sequence classifier, implemented by Hoefel & Elkan (2008), which showed that, by combining two models, one can achieve better results than each of the models alone.

It is also addressed, in this chapter, works where the approach considered consisted on the construction of indexes based on toponyms through geotagging, where a method based on a combination of multiple heuristics, a method for extracting place entities and routes from mountaineering texts, and a method for predicting the geospatial coordinates to assing a given document using language models, are explained.

These works are helpful to this dissertation in the way that they found a solid basis for the approach studied.

Chapter 4

Using Machine Learning for Geospatial Reference Resolution

Geospatial reference resolution refers to identifying and disambiguating the occurrences of location references, as given over textual documents.

In this dissertation, the SpatialML standard is followed for defining what a spatial expression is and how one should be disambiguated. However, the focus is mainly on the *latLong* attribute from the PLACE tag, ignoring the remaining SpatialML tag and attribute information.

The proposed method for resolving spatial references in text uses a combination of different models. The basic concept is to train two or more learners sequentially, with each successive learner incorporating the results of the previous one(s) in some way.

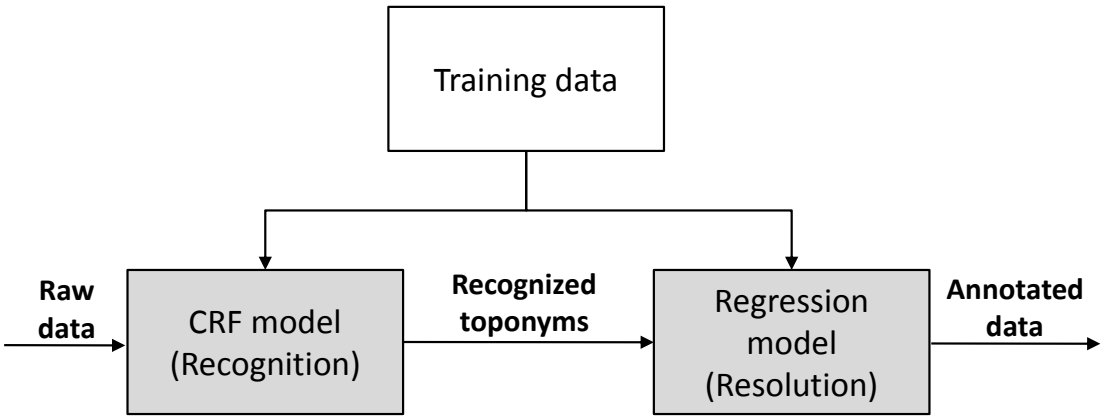

Figure 4.5: Spatial reference resolution with a combination of models.

Figure 4.5 provides an illustration for the general procedure. A first level CRF learner is used for identifying spatial references in text, based on the implementation available on the LingPipe[1] package. This tagger annotates tokens as either being part of a reference to a specific location, a part of a reference to a relative location, or not being part of a spatial reference. Afterwards, a second learner takes as input the spatial references identified in the first level, uses a full-text index to identify possible disambiguation candidates for these references, and uses a set of features to identify the most probable candidate for the reference in question. Each of these disambiguation candidates contains the values for the *latLong* attribute of the SpatialML annotation. The disambiguation module is trained to choose, from the set of candidates, the one whose geographical coordinates denoted by the *latLong* attribute have the closest distance to the correct coordinates for the spatial expression.

The rest of this chapter details the learning approaches used in both levels.

4.1 Geospatial Reference Recognition

The first level learner of the proposed method is responsible for delimiting the occurrences of geospatial references in text. This task is addressed by translating what is essentially a chunking problem (i.e., a task of discovering the chunks of text that correspond to spatial references) into a tagging problem (i.e., a classification task of assigning tags to each individual text token, according to whether they belong to a spatial reference or not), using the B-I-O encoding of chunkings as taggings. In the case of this dissertation, the chunking problem was addressed through the formalism of Conditional Random Fields (CRFs), where two different types of spatial chunks were considered:

- **Proper name locations:** These chunks correspond to spatial expressions having SpatialML annotations where the *form* attribute has the value of NAM.
- **Nominal locations:** These chunks correspond to spatial expressions having SpatialML annotations where the *form* attribute has the value of NOM.

The tagging problem involves separate B and I tags for each of the chunk types. The classification model tags words given in a sequence according to the above tags, from which the system generates the final results for the spatial reference recognition.

The identification process starts with a tokenization scheme that deterministically breaks an input text into a sequence of tokens. Tagging these tokens is a non-trivial classification problem, since

[1] http://alias-i.com/lingpipe/

if there are K different tags, then a sequence of length N has up to K^N possible sequence of tags (although some may be eliminated with basis on structural grounds). Conditional Random Fields, described in 2.1.3, offer an efficient and principled approach for addressing this sequence classification problem.

In this dissertation, LingPipe's[1] implementation of first-order chain conditional random fields (CRFs) is used, which are essentially undirected probabilistic graphical models (i.e., Markov networks), in which vertexes represent random variables and each edge represents a dependency between two variables, that are discriminatively-trained to maximize the conditional probability of a set of hidden tags y, given a set of input tokens o. LingPipe's implementation makes the first-order Markov assumption on the dependencies among y (i.e., the transitions between tags y depend only on the origin and destination).

The modeling flexibility of CRFs allows the feature functions to be complex, overlapping features of the input, without requiring additional assumptions on their interdependencies. The list of features considered in this dissertation's experiments is as follows:

- Context words: The token identity within a 1-token window of the target token. For example, in the sequence *Luís went to Campo Grande*, the word *Campo* would generate the following features:
  - *TOKEN_PREV_+prevToken*, which represents the previous token. In the example, the previous token is *to*, forming the feature *TOKEN_PREV_to*;
  - *TOKEN_+currentToken*, which represents the current token. In the example, the current token is *Campo*, forming the feature *TOKEN_Campo*;
  - *TOKEN_NEXT_+nextToken*, which represents the next token. In the example, the next token is *Grande*, forming the feature *TOKEN_NEXT_Grande*;
  - *TOKEN_PREV_CAT_+previousTokenCategory*, which represents the category given by LingPipe's IndoEuropeanTokenCategorizer class to the previous token. In the example, the previous token is *to*, forming the feature *TOKEN_PREV_CAT_LET-LOW*;
  - *TOKEN_CAT_+currentTokenCategory*, which represents the category given by LingPipe's IndoEuropeanTokenCategorizer class to the current token. In the example, the current token is *Campo*, forming the feature *TOKEN_CAT_LET-CAP*;
  - *TOKEN_NEXT_CAT_+nextTokenCategory*, which represents the category given by LingPipe's IndoEuropeanTokenCategorizer class to the next token. In the example, the next token is *Grande*, forming the feature *TOKEN_NEXT_CAT_LET-CAP*;

[1] http://alias-i.com/lingpipe/

The IndoEuropeanTokenCategorizer class from the LingPipe package, used in the context words feature generator, categorizes the token with one of the following classes:

  - NULL-TOK: Class given to a zero-length string.
  - 1-DIG: Class given to a string containing a single digit.
  - 2-DIG: Class given to a string composed of two digits.
  - 3-DIG: Class given to a string composed of three digits.
  - 4-DIG: Class given to a string composed of four digits.
  - 5+-DIG: Class given to a string composed of five or more digits.
  - DIG-LET: Class given to a string containing digits and letters.
  - DIG–: Class given to a string containing digits and hyphens.
  - DIG-/: Class given to a string containing digits and slashes.
  - DIG-,: Class given to a string containing digits and commas.
  - DIG-.: Class given to a string containing digits and periods.
  - 1-LET-UP: Class given to a string containing a single uppercase letter.
  - 1-LET-LOW: Class given to a string containing a single lowercase letter.
  - LET-UP: Class given to a string containing uppercase letters only.
  - LET-LOW: Class given to a string containing lowercase letters only.
  - LET-MIX: Class given to a string containing letters only, both upper and lowercase.
  - PUNC-: Class given to a string containing a sequence of punctuation characters.
  - OTHER: Class given to a string which does not fall in the previous categories.

- Lexicons: Whether the token appears in a list of specific words associated with spatial expressions. For example, in the sequence *Luís went to Campo Grande*, the token *Campo* would generate the following features:

  - *LNAMES*, if the current token consisted in the list. In the example, the current token is *Campo*;
  - *LNAMES_PREV*, if the previous token consisted in the list. In the example, the previous token is *to*;
  - *LNAMES_NEXT*, if the next token consisted in the list. In the example, the next token is *Grande*;

- POS tags: The Part-Of-Speech tags for the tokens, as predicted by an HMM that was trained separately for English part-of-speech tagging using the Brown corpus. For example, in the sequence *Luís went to Campo Grande*, the token *Campo* would generate the following features:
  - *POS_PREV_+PrevTag*, which represents the tag chosen by the POS model for the previous token. In the example, the previous token to be classified is *to*;
  - *POS_+Tag*, which represents the tag chosen by the POS model for the current token. In the example, the current token to be classified is *Campo*;
  - *POS_NEXT_+NextTag*, which represents the tag chosen by the POS model for the next token. In the example, the next token to be classified is *Grande*;
- Prefix-suffix: The four letter prefix and suffix of the word. For example, in the sequence *Luís went to Campo Grande*, the token *Campo* would generate the following features:
  - *SUFF_PREV_+PrevTagSuff*, which represents the suffix's first 4 letters of the previous token. In the example, the previous token is *to*, generating the feature *SUFF_PREV_to*;
  - *SUFF_+TagSuff*, which represents the suffix's first 4 letters of the current token. In the example, the current token is *Campo*, generating the feature *SUFF_ampo*;
  - *SUFF_NEXT_+NextTag*, which represents the suffix's first 4 letters of the next token. In the example, the next token is *Grande*, generating the feature *SUFF_NEXT_ande*;
  - *PREF_PREV_+NextTag*, which represents the prefix's first 4 letters of the previous token. In the example, the previous token is *to*, generating the feature *PREF_PREV_to*;
  - *PREF_+NextTag*, which represents the prefix's first 4 letters of the current token. In the example, the current token is *Campo*, generating the feature *PREF_Camp*;
  - *PREF_NEXT_+NextTag*, which represents the prefix's first 4 letters of the next token. In the example, the next token is *Grande*, generating the feature *PREF_NEXT_Gran*;

4.2 Spatial Reference Disambiguation

The proposed approach for spatial reference disambiguation involves (i) generating disambiguation candidates by using gazetteer queries, (ii) scoring possible candidates using a regression model, and (iii) selecting the highest scoring candidate. The regression objective used in the second step is based on the distance between the true geographic coordinates associated with the expression, and the geographic coordinates of the candidates.

The proposed disambiguation module uses gazetteer queries to compute, for each spatial reference, a set of possible candidate disambiguations. A candidate is an object representing a possible location for the reference being disambiguated. The gazetteer used was a Lucene index containing 393.279 Wikipedia geo-political entities, which supports queries with basis on textual similarity towards given names. The gazetteer consists of 3 indexes:

- Content: An index containing the entities' page information.
- Names: An index containing the entities' alternative names.
- Spell: An index containing information about the entities' n-gram information.

The *Content* index contains the information of the geo-political entities. In this index, one can find the following information:

- eid: The entity's integer identifier.
- name: The entity's name.
- text: The text content of the Wikipedia article of the entity.
- pop: The entity's population count.
- area: The entity's area value in km^2.
- coord: The entity's geographical coordinates.

As an example, in Figure 4.6 one can see the *Lisbon* entry in the Content index.

The *Names* table contains information about the entities' alternative names. In this table, one can find the following information:

- eid: The entity's integer identifier.
- name: The entity's name.

In this table, it is very common to find many entries with the same *eid*, but with different values in the *name* column, stating that a given geo-political entity can have many ways to be referenced. This information is used on a feature considered for representing the disambiguation candidates, which is used to help in finding the most likely candidate. As an example, Figure 4.7 shows the various alternative names in the index for *Lisbon*.

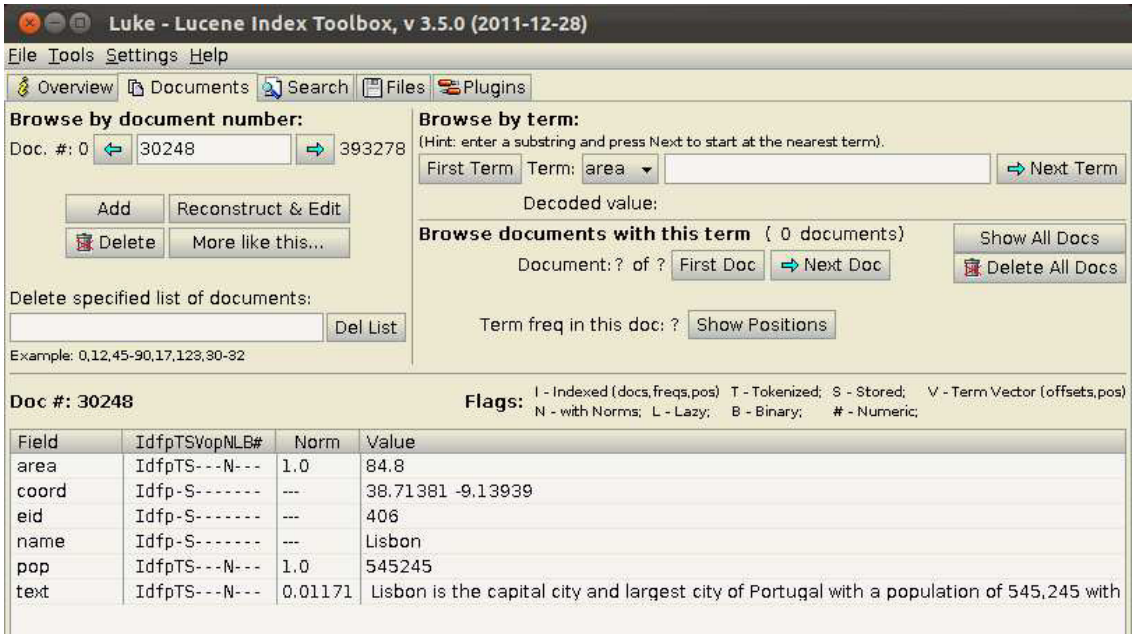

Figure 4.6: Example of an entry in the Content table of the index used.

The information displayed in Figure 4.6 and Figure 4.7 are shown through Luke[1], a development and diagnostic graphic tool, which accesses existing Lucene indexes and displays their contents.

Finally, the *Spell* table contains information about the entities' gram information. This table is very important due to the storage of the entities names into grams. When one needs to find candidates for a given reference, one queries this table to find similar spatial references to the reference given. In this table, one can find the following information:

- eid: The entity's integer identifier.
- term: The entity's name.
- gram: The entity's gram information.

An n-gram is a sequence of n items from a given sequence of text. There are two different types of n-grams, which are i) character-level n-grams, and ii) word-level n-grams. In this index, the character-level n-grams are used. As an example, the token *Lisbon* would generate the following grams:

- 1-gram: *L*, *i*, *s*, *b*, *o*, *n*.
- 2-gram: *Li*, *is*, *sb*, *bo*, *on*.

[1] `http://code.google.com/p/luke/`

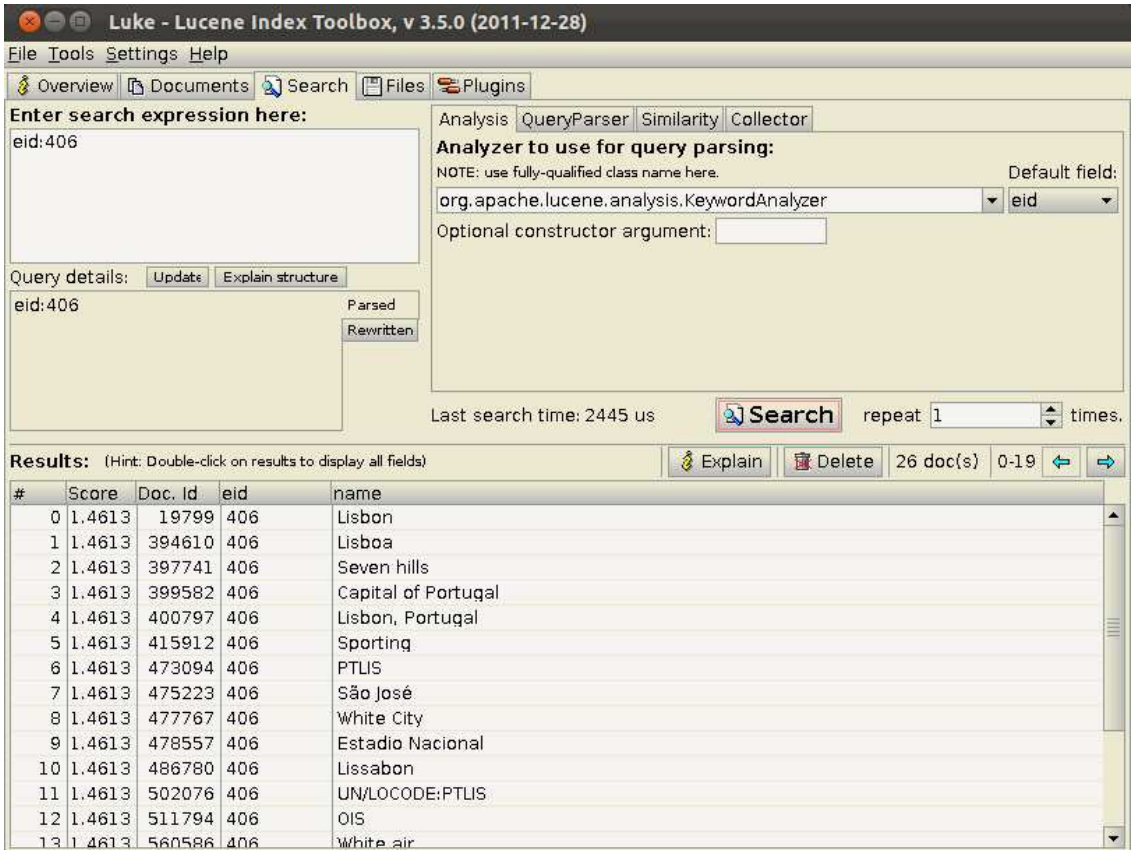

Figure 4.7: Example of an entry in the Names table of the index used.

- 3-gram: *Lis*, *isb*, *sbo*, *bon*.
- 4-gram: *Lisb*, *isbo*, *sbon*.

Given a spatial reference identified in the first level, the system queries the gazetteer for possible candidates, where each candidate contains the following information:

- eid: A number used to identify the candidate.
- name: The name of the candidate, i.e., the name of the location.
- wiki_text: The text of the Wikipedia article of the candidate.
- altNames: A vector containing the alternative names of the candidate.
- coordinates: The candidate's geographical coordinates.
- area: The candidate's area value in km^2.
- population: The candidate's population value.

In order to get these candidates, one queries the gazetteer for a maximum of 30 similar names, using the gram information on the *Spell* table. The returned candidates are based on textual similarity of both the actual candidate name or alternative names. After generating disambiguation candidates, the following set of features, based on the heuristics described in Section 2.2, is computed for the purpose of representing each candidate:

- The population count of the disambiguation candidate.
- The area in km^2 of the disambiguation candidate.
- The number of words in the candidate's text.
- The least Vincenty distance between the candidate's geographical coordinates and all spatial references' geographical coordinates in the same document. The Vincenty distance is the distance between two geographical points using Vincenty's formulae (Vincenty, 1975), assuming the WGS-84 standard for the ellipsoid used in the geodetic calculations.
- The average Vincenty distance between the candidate and all spatial references in the same document.
- The convex hull formed by the spatial references in the document.
- The concave hull formed by the spatial references in the document.
- The cosine similarity between the candidate's text and the phrase where the spatial reference occurs.
- The minimum edit distance between the disambiguation candidate's alternative names to the name of the spatial reference.
- The least Vincenty distance between the disambiguation candidate and all the previous spatial reference's candidates.

For training and evaluating the regression model, the distance between the disambiguation candidate's geographical coordinates, and the spatial reference's geographical coordinates, as annotated in the corpus through the *latLong* attribute in the SpatialML tags, is computed. The regression model attempts to estimate this distance, with basis on the considered features. More formally, let X be the training set consisting of m instance pairs $x_i = \{f_{i,1}, ..., f_{i,n}, d_i\}$, where $f_{i,j}$ is the j-th input feature of a given example i, where $1 \leq i \leq m$, and d_i is the corresponding spatial distance. The regression goal is to estimate a function $reg(x)$ with basis on the training set X that takes as input a set of instance pairs $x'_i = \{f_{i,1}, ..., f_{i,n}\}$, is as close as possible to the target values d_i for every x'_i.

One of the regression models used in this dissertation was a Support Vector Machine (SVM) regression model, available in the Weka machine learning toolkit. This implementation corresponds to the approach used by Shevade *et al.* (2000), which in turn is an improvement over the sequential minimal optimization (SMO) algorithm for training a support vector regression by Smola & Schölkopf (2004). In this dissertation's experiments, a Radial Basis Function (RBF) is used as kernel, where the parameters C and γ were set to the default values of 250007 and 0.01 respectively.

The other regression model used in this dissertation was a linear regression tree model called REPTree, also available in the Weka machine learning toolkit. This model is a fast decision tree learner, which builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning (with backfitting). *Reduced-error pruning* is a technique where, at each node in the tree, it is possible to establish the number of instances that are misclassified on a training set by propagating errors upwards from leaf nodes. This can be compared to the error rate if the node was replaced by the most common class resulting from that node. If the difference is a reduction in error, then the subtree below the node can be considered for pruning. This calculation is performed for all nodes in the tree and the one which has the highest reduced-error rate is pruned. The procedure is then iterated over the freshly pruned tree until there is no possible reduction in error rate at any node. In this dissertation's experiments, all the parameters were set to their default values, i.e., a minimum of 2 instances per leaf, no maximum tree depth, 3 folds for reduced error pruning, and a minimum numeric class variance proportion of train variance for split of $1e^{-3}$.

The final step of the proposed approach involves selecting the candidate with the least estimated distance as the result for the disambiguation.

4.3 Summary

In this section, a novel method for resolving spatial references in text using machine learning was described. This method uses machine learning in both the recognition and disambiguation tasks of the geospatial reference resolution problem. The recognition module uses a CRF model to recognize the extent of the spatial expressions present in the text, and the normalization module uses a regression model to disambiguate between many possible candidates for a given spatial reference. The regression models used in this dissertation were a SVM regression model and a regression tree model called REPTree. The generation of possible disambiguation candidates for a given geospatial reference is done through queries to a gazetteer, containing information of

various geo-political entities' Wikipedia pages. The architecture of the system, and the features used in the machine learning modules, were are also described.

Chapter 5

Experimental Validation

This section describes the empirical evaluation of the proposed place reference resolution method. This includes the details of the experimental design (i.e., the datasets and the evaluation metrics that were used to measure the quality of the results), as well as the results for the experiments that evaluated the effectiveness of the methods under study.

5.1 Evaluation Methodology

The validation methodology used is based on the standard NER evaluation setup, which involves comparing the annotations produced by the automated system against gold-standard annotations provided by human experts. As gold-standard annotated data, two English datasets were used, namely i) the SpatialML dataset and ii) the LGL dataset. Section 2.5 briefly described the SpatialML annotation standard and Table 5.4 presents a statistical characterization for these collections. Each document in these collections represents an article formatted in XML, in which SpatialML tags denote spatial expressions. For the purpose of evaluating the system, the models were trained using 80% of each dataset, and tested with the remaining 20%.

While the first dataset already used SpatialML annotations, the second dataset had to be transformed to use SpatialML annotations. This was done by using using an XQuery expression for translating between the representation formats. The XQuery expression that was created crawled through the LGL dataset in search of spatial references. If one was found, the query would replace the LGL tag surrounding the string representing a spatial reference, with a SpatialML *PLACE* tag, getting the geographical coordinates from the LGL tag that was replaced. Through an analysis of the LGL dataset, I concluded that all the spatial references encountered

	LGL	SpatialML
Articles	588	429
Words	213.446	208.389
Unique Words	18.950	16.822
Toponyms	4.793	4.783
Distinct Toponyms	1.297	915

Table 5.4: Comparison between the LGL and the SpatialML datasets

were specific references, therefore the form *NAM* was attributed to every spatial reference in the dataset. Therefore, we can only compare specific references (i.e., *NAM* references) between datasets.

Given the two-stage approach, separate measurements for the identification and disambiguation sub-tasks were used. In both cases, the results were measured with the commonly used accuracy rate, which is the ratio of correct classifications in the entire analyzed corpus. Precision is the percentage of correct references identified and/or disambiguated by the system. Recall is the percentage of references present in the test collection that are identified/disambiguated by the system. A spatial reference is correctly identified only if it is an exact match with the corresponding spatial reference given in the test collection, and correctly disambiguated only if the assigned disambiguation is within a radius of 5 km of the one that is given in the SpatialML annotation, i.e., the *latLong* attributes of the reference and the disambiguation have values which are at a maximum distance of 5 km. Thus, correct disambiguation can only occur when a correct recognition has first taken place, since it does not make sense to disambiguate textual expressions that are not recognized as spatial references.

5.2 The Obtained Results

The proposed approach was compared using different features for the CRF spatial expression identification model. Section 4.1 described the complete set of features that were considered and Table 5.5 presents the obtained results across the two different document collections, and considering the two different forms of expressions. Results are shown in terms of precision (P), recall (R), and accuracy (A).

In terms of the disambiguation performance, the proposed approach was compared using two different models, SVMs and REPTrees, as explained in Section 4.2. Section 4.2 also described the complete set of features that were considered and Table 5.6 presents the obtained results across the two different document collections, and considering the two different forms of expressions. Results are also shown in terms of precision (P), recall (R), and accuracy (A).

	SpatialML						LGL		
	NAM			NOM			NAM		
	P	R	A	P	R	A	P	R	A
Tokens	0.97	0.94	0.92	0.86	0.51	0.47	**0.85**	0.62	0.56
Tokens and Lexicons	0.97	0.95	0.92	0.86	0.53	0.49	0.84	0.71	**0.62**
Tokens and POS	0.97	0.94	0.92	0.86	0.51	0.47	**0.85**	0.62	0.56
Tokens and Pref_Suf	**0.98**	0.97	**0.95**	0.86	0.67	**0.61**	0.79	0.70	0.59
Tokens, Lexicons and POS	0.97	0.95	0.93	0.86	0.56	0.51	0.84	0.70	0.61
Tokens, Lexicons and Pref_Suf	0.97	**0.98**	**0.95**	**0.87**	**0.68**	**0.61**	0.80	0.72	0.61
Tokens, POS and Pref_Suf	**0.98**	0.97	**0.95**	0.86	0.67	**0.61**	0.79	0.70	0.59
All features	0.97	**0.98**	**0.95**	0.85	**0.68**	**0.61**	0.80	**0.73**	0.61

Table 5.5: The obtained results when comparing different CRF recognition models.

	SpatialML						LGL		
	NAM			NOM			NAM		
	P	R	A	P	R	A	P	R	A
SVMReg	0.54	0.53	0.37	**0.06**	**0.05**	**0.03**	0.26	0.24	0.14
REPTree	**0.70**	**0.71**	**0.55**	0.03	0.03	0.02	**0.30**	**0.27**	**0.17**

Table 5.6: The obtained results when comparing different disambiguation models.

The obtained results vary between 0.92-0.95 in accuracy for the recognition of *NAM* references, and between 0.47-0.61 in accuracy for *NOM* references for the SpatialML dataset, and between 0.56-0.62 in accuracy for *NAM* references for the LGL dataset.

In the disambiguation sub-task, the obtained results vary between 0.37-0.55 in accuracy for *NAM* references, and 0.02-0.03 in accuracy for *NOM* references for the SpatialML dataset, and 0.14-0.17 in accuracy for *NAM* references for the LGL dataset. In these disambiguation experiments, the recognition was performed through the CRF model considering the entire set of features.

5.3 Discussion

The results show that, as expected, the CRF tagger using a richer feature set achieves the best results. Accuracy in the LGL dataset is significantly lower than on the SpatialML dataset. This can be due to the fact that the LGL dataset has less articles than the SpatialML dataset, and that the LGL dataset has much more restricted references than the SpatialML dataset. By restricted, it is meant that the references found in the LGL dataset are references to smaller locations in the real world (this was referred in Section 3.2.2 as a local lexicon).

Token features are the most informative, both in the SpatialML and the LGL dataset, getting accuracy values between 0.47-0.92 when used alone. After these, Prefix and Suffix features seem to be the most informative in the SpatialML dataset, both for *NAM* and *NOM* references.

This can be due to the many similar prefixes and suffixes that spatial expressions can have. In the case of the LGL dataset, the most informative features seem to be the Lexicon features. Since this dataset has a smaller, more specific type of spatial references, it is possible that a reference being part of a lexicon containing location names is more discriminative than the other features.

The results show that *NOM* references have a difference of around 0.30 to *NAM* references in the recognition sub-task. This difference can be due to the more difficult task of recognizing nominal locations using only the *latLong* attribute of the SpatialML annotation scheme with the features considered for recognition.

Table 5.6 presents the obtained results for the disambiguation sub-task, showing that, overall, the proposed approach has comparable results to other state-of-the-art systems in terms of proper name references, with nominal references having poor results. It is of notice that, while previous works report results for spatial expression disambiguation separately from the recognition, the evaluation process used considered recognition failures also as failures in terms of disambiguation. Also, the 5 km threshold may be too restrictive. One can see from Table 5.6 that the REPTree disambiguation model has better results than the SVM disambiguation model.

The disambiguation results for *NOM* spatial references are very poor compared to other state-of-the-art systems, having accuracy values between 0.02 and 0.03. Through an extensive look on the nominal references in the SpatialML dataset, I concluded that these references are anaphoras, for example, *country*, *building* or *region*. This kind of references cannot be resolved without evaluating the surrounding context, since the token alone is not descriptive enough to resolve the correct geographical coordinates. Another conclusion taken from the evaluation made is that some of these references were poorly annotated, meaning that some of the locations annotated as *NOM* references, were actually proper name locations, i.e., *NAM* references. For example, *Pittsburgh* was annotated as nominal, when in fact it is a proper name location. This leads me to believe that the nominal references the system was capable of resolving were actually proper name locations poorly annotated as nominal references. Therefore, it is necessary in future work to evaluate this kind of references in a different way to that of proper name references. Also, it is necessary to expand the system to recognize more SpatialML tags, in order to make the system more robust in this aspect.

Despite the above problems, the results achieved on the recognition sub-task are of an acceptable quality, capable of being used in the subsequent processing stages of many different types of spatial information retrieval applications. The disambiguation results show that, for specific references (i.e., *NAM* references), the results achieved are comparable to other state-of-the-art systems, and are a good basis for subsequent processing stages of other types of spatial information retrieval applications. However, the results achieved on the disambiguation sub-task for

relative references (i.e., *NOM* references) were largely inferior to those of other state-of-the-art systems, making it necessary to carefully review the recognition and disambiguation process for this kind of spatial references.

Chapter 6

Conclusions

This work proposed a supervised machine learning method for resolving spatial references in text. The proposed method for resolving spatial references in text uses a combination of different models. A first level learner is used for identifying spatial references in text, based on the CRF implementation available on the LingPipe[1] package, and then a second learner, based on a regression module, takes as input the spatial references identified in the first level, uses a full-text index to identify possible disambiguation candidates for these references, and uses a set of features to identify the most probable candidate for the reference in question. Experiments with English datasets containing SpatialML annotations attested for the adequacy of the proposed approach, showing that it compares well against previous methods in terms of recognition accuracy, while the disambiguation process needs some improvements.

In the recognition sub-task, the proposed approach achieved accuracy results between 0.92 and 0.95 for *NAM* references, and between 0.47 and 0.61 for *NOM* references when using the SpatialML dataset. In the same sub-task, the proposed approach achieved accuracy values between 0.56 and 0.61 for *NAM* references when using the LGL dataset.

In the disambiguation sub-task, the proposed approach achieved accuracy results between 0.37 and 0.55 for *NAM* references, and between 0.02 and 0.03 for *NOM* references when using the SpatialML dataset. In the same sub-task, the proposed approach achieved accuracy values between 0.14 and 0.17 for *NAM* references when using the LGL dataset.

This shows that the system is well suited for the specific spatial expressions problem, serving as a good basis for future systems. It also shows that there is a lot of work to be done in the nominal geospatial expressions problem.

[1] `http://alias-i.com/lingpipe/`

6.1 Contributions

The main contribution from this work are the results from the empirical evaluation to the proposed approach in this dissertation. Also, the developed prototype used to accomplish the related experiments was made available online as an open-source package on Google Code, with the name *stemptag*[1].

6.2 Future Work

Despite these results, there are many ways to improve the current work. It would also be interesting to extend the system to handle texts in other languages, namely Portuguese, since the system currently only recognizes the English language. It is also necessary to extend the current system to use more tags and attributes of the SpatialML annotation scheme, in order to make it more complete.

Another possible direction for future work relates to the urge of a Recurrent Sliding Windows (RSW) model to help in the recognition phase (Joshi, 2003). The idea is to train a model using RSW with a Random Forest as a base classifier, with a sliding window with a one-word span over the sentences being analyzed.

Another point for future work, one could use more advanced machine learning models, namely second-order CRFs or Semi-Markov CRFs to better recognize spatial expressions, or other unsupervised and semi-supervised techniques.

Finally, it would be interesting to use this research to create an online prototype that uses this system to actively recognize and disambiguate spatial references in text submitted by the user, showing the results in the prototype or exporting them, for example, to a XML-formatted file.

[1] `http://code.google.com/p/stemptag/`

Bibliography

EKBAL, A. & BANDYOPADHYAY, S. (2010). Named Entity Recognition using Support Vector Machine: A Language Independent Approach. *International Journal of Electrical and Electronics Engineering*.

FORNEY, G.D. (1973). The viterbi algorithm. *Proceedings of the IEEE*, **61**.

HAN, J. (2005). *Data Mining: Concepts and Techniques*.

HOEFEL, G. & ELKAN, C. (2008). Learning a two-stage svm/crf sequence classifier. In *Proceedings of the 17th ACM conference on Information and knowledge management*.

JOSHI, S. (2003). *Calibrating Recurrent Sliding Window Classifiers for Sequential Supervised Learning*.

KLINGER, R. & TOMANEK, K. (2007). Classical Probabilistic Models and Conditional Random Fields. Tech. rep., Department of Computer Science, Dortmund University of Technology.

KROVETZ, B., DEANE, P. & MADNANI, N. (2011). The Web is not a PERSON, Berners-Lee is not an ORGANIZATION, and African-Americans are not LOCATIONS: An Analysis of the Performance of Named-Entity Recognition. In *Proceedings of the ACL Workshop on Multiword Expressions: From Parsing and Generation to the Real World*.

LAFFERTY, J.D., MCCALLUM, A. & PEREIRA, F.C.N. (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning.*.

LEIDNER, J.L. (2008). *Toponym Resolution in Text : Annotation, Evaluation and Applications of Spatial Grounding of Place Names*. Universal Press.

LIEBERMAN, M.D. & SAMET, H. (2011). Multifaceted toponym recognition for streaming news. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*.

LIEBERMAN, M.D., SAMET, H. & SANKARANARAYANAN, J. (2010). Geotagging with local lexicons to build indexes for textually-specified spatial data. In *Proceedings of the 26th International Conference on Data Engineering*.

LOUREIRO, V., MARTINS, B. & CALADO, P. (2011). A machine learning method for resolving temporal references in text. In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence*.

MANI, I., HITZEMAN, J., RICHER, J., HARRIS, D., QUIMBY, R. & WELLNER, B. (2008). SpatialML: Annotation scheme, corpora, and tools. In *Proceedings of the 6th International Conference on Language Resources and Evaluation*.

MARTINS, B., ANASTÁCIO, I. & CALADO, P. (2010). A machine learning approach for resolving place references in text. In *Proceedings of the 13th AGILE International Conference on Geographic Information Science*.

MOGUERZA, J. & MUÑOZ, A. (2006). Support vector machines with applications. *Statistical Science*, **21**.

NADEAU, D. & SEKINE, S. (2007). A survey of named entity recognition and classification. *Linguisticae Investigationes*, **30**.

PIOTROWSKI, M., LÄUBLI, S. & VOLK, M. (2010). Towards mapping of alpine route descriptions. In *Proceedings of the 6th Workshop on Geographic Information Retrieval*.

RABINER, L. & JUANG, B. (2003). An introduction to hidden Markov models. *ASSP Magazine, IEEE*, **3**.

SARAWAGI, S. & COHEN, W.W. (2004). Semi-markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*.

SEBASTIANI, F. (2002). Machine learning in automated text categorization. *ACM Computing Surveys*, **34**.

SHEVADE, S.K., KEERTHI, S.S., BHATTACHARYYA, C., MURTHY, K.R.K. & SMOLA, I. (2000). Improvements to the smo algorithm for svm regression. *IEEE Trans. Neural Netw*.

SMOLA, A.J. & SCHÖLKOPF, B. (2004). A tutorial on support vector regression. *Statistics and Computing*.

SUTTON, C. & MCCALLUM, A. (2011). An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, **To appear**.

VAPNIK, V.N. (1995). *The nature of statistical learning theory*.

VINCENTY, T. (1975). Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. *Survey Review*.

WING, B. & BALDRIDGE, J. (2011). Simple supervised document geolocation with geodesic grids. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*.

YE, N., LEE, W.S., CHIEU, H.L. & WU, D. (2009). Conditional random fields with high-order features for sequence labeling. In *Proceedings of the 23rd Annual Conference on Neural Information Processing Systems*.