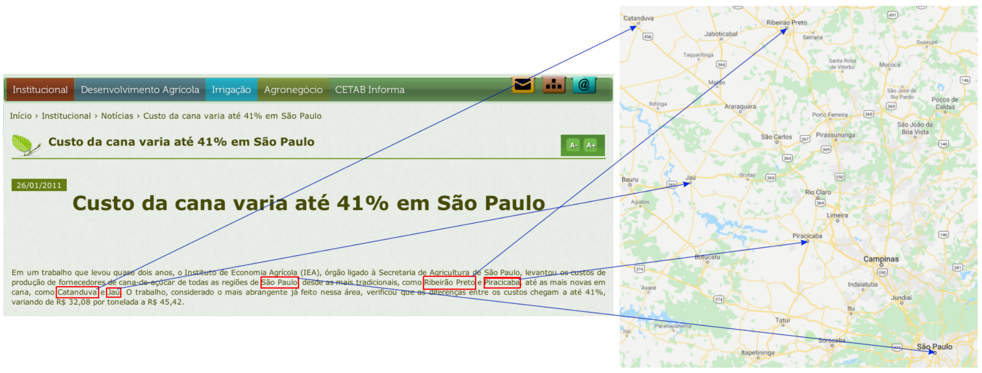


Início › Institucional › Notícias › Custo da cana varia até 41% em São Paulo

Custo da cana varia até 41% em São Paulo

26/01/2011

Custo da cana varia até 41% em São Paulo

Em um trabalho que levou quase dois anos, o Instituto de Economia Agrícola (IEA), órgão ligado à Secretaria de Agricultura de São Paulo, levantou os custos de produção de fornecedores de cana-de-açúcar de todas as regiões de São Paulo desde as mais tradicionais, como Ribeirão Preto e Piracicaba, até as mais novas em cana, como Catanduva e Jaú. O trabalho, considerado o mais abrangente já feito nessa área, verificou que as diferenças entre os custos chegam a até 41%, variando de R$ 32,08 por tonelada a R$ 45,42.

Automatic Extraction of Spatial Information From Text

Paulo Miguel de Almeida Marques
Mestrado em Ciências de Computação
Departamento de Ciências de Computação
2019

Orientador
Prof. Doutor Alípio Mário Guedes Jorge, Professor Associado, FCUP

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/_________

Acknowledgments

I would like to thank my supervisor for his patience during this process.

Special thanks to my wife for believing in me, and to my parents for always being there when I need them to.

Abstract

It is becoming more and more important to process and catalog all types of information in order to better inform decision makers, consumers and voters. Due to the high amounts of information about every topic being generated daily all over the world, automated tools need to be created to help making the accomplishment of the task feasible.

One of those tools is information retrieval from text, an automated task which aims at helping information users focus on relevant documents. One of the interesting types of extraction that can be done with documents is the detection of spatial locations that a particular text contains, as well as the most relevant one in the document as a whole.

This thesis aims to develop a model that finds Brazilian administrative locations in a series of news articles written in Portuguese, as well as the most relevant in each one of the articles. This language has had more limited research than other languages, so our study will focus on optimization of common algorithms to solve this task.

We explore the use of Conditional Random Fields, a common algorithm for the task, and attempt to find the best way to apply it in solving entity detection. We also explore whether a simple algorithm is effective in solving the problem of selecting the most relevant entity of a document. In the way of solving these problems, we also create a tool to more easily create, run and evaluate machine learning models.

Keywords: Information Extraction, Geotagging, Conditional Random Fields, Named Entity Recognition

Resumo

Está-se a tornar cada vez mais importante processar e catalogar todos os tipos de informações de forma a melhor informar os tomadores de decisão, consumidores e eleitores. Devido à grande quantidade de informação gerada no mundo inteiro diariamente sobre qualquer tópico, é necessário criar ferramentas automáticas que ajudem à realização desta tarefa tarefa.

Uma dessas ferramentas é a extração de conhecimendo em texto, uma tarefa automatizável que visa ajudar os utilizadores a concentrarem-se em documentos relevantes. Um dos tipos interessantes de extração que podem ser feitos com documentos é a detecção dos locais espaciais que um determinado texto contém, bem como o mais relevante no documento como um todo.

Esta tese visa desenvolver um modelo que encontre os locais administrativos brasileiros presentes em uma série de artigos escritos em português, bem como os mais relevantes em cada um dos artigos. Como esta linguagem tem um estudo mais limitado do que outras línguas, o nosso estudo se concentrará na otimização de algoritmos comuns para resolver esta tarefa.

Exploramos o uso de Conditional Random Fields, um algoritmo comum para a tarefa, e tentamos encontrar a melhor maneira de o aplicar na resolução da detecção de entidades. Exploramos também se um algoritmo simples é eficaz na solução do problema de seleção da entidade mais relevante de um documento. Para resolver estes problemas, criamos uma ferramenta para criar, executar e avaliar mais facilmente modelos de aprendizagem.

Contents

List of Tables

List of Figures

List of Abbreviations

CRF	Natural Language Processing
HMM	Hidden Markov Models
IBGE	Instituto Brazileiro de Geografia e Estatística
MEMM	Maximum Entropy Markov Models
NER	Named Entity Extraction
NERC	Named Entity Extraction and Classification
NLP	Natural Language Processing
POS	Part of Speech

Chapter 1

Introduction

1.1 Motivation

There are many text documents being produced on a daily basis, and it is important that these works are not quickly forgotten or lost among the huge amount of data that any person has access to. The field of automated text processing is called Natural Language Processing (NLP) [CWK+11] and is a field of application that can help deal with this by retrieving different kinds of information from these texts which can then be used in making them more useful to those who are interested, a process called Information Retrieval [S+01]. This data retrieval process can obtain different kinds of information, which can then be used for many applications, including question-answering, visualization, and data mining.

Journalistic texts are one type of text that can be useful in many different contexts, but more importantly, to help make business and political decisions [TK15], making the extraction of textual references to people, places and other entities an important first step for processing the information.

Since it is not yet possible, let alone practical, for a computer to understand words the same way a human being does, text mining presents many challenges, from how to represent the text in a computer system to how to give some meaning to its constituents [CWK+11]. Text must first be split into individual components called tokens (usually words, punctuation and numbers) to create the first (and possibly only) computer representation [SS14]. However, this representation is empty of meaning, which must come from manual human annotations (of many varieties) [MOE99], a programmed heuristic or a pre-trained model [VHPPG07]. Since full modeling of language is a very big and complicated subject that is still far from being accomplished, we will focus in a small part of this modeling.

Named entity recognition is the general name of the task related to the gathering of entities mentioned in text [NS07], where there are many ways to group and categorize each entity. In this work, we will be focusing on the detection of location entities, which can be called toponym resolution [Lei08], categorized by official administrative hierarchy. On top of this, we can try to find out the most relevant location, which could, for example, be used for aggregation purposes, although the definition of relevance is an ambiguous term that must be defined for the intended use.

1.2 Aims

The main objective of this work is to use and adapt the more adequate existing techniques and algorithms to detect locations in a series of Portuguese journalistic texts. The computational processing of the Portuguese language has been less studied than other languages, and there are less reference data [SCP19], so it is not as easy to create learning models as in English. We will be experimenting and evaluating a set of different features to be computed for the tokens in the texts in order to inform and tune the algorithms. We will also attempt to find a way to detect the most relevant entity of the text.

Our work will be tested and evaluated on a specific dataset, which consists of a series of news reports from the Brazilian company Embrapa[1], a public company administrated by the Ministério da Agricultura, Pecuária e Abastecimento (Ministry of Agriculture, Livestock and Food Supply) which had the objective of creating an industry model adequately adapted to the country [2].

1.3 Contributions

We tested of one of the most common algorithms for text datasets, Conditional Random Fields (CRF) [LMP01], for entity detection in the slightly different task of detecting only administrative locations from a single country. This algorithm can support any number of features that can be used as descriptors of words (within the memory and time computational capacities available). Our contribution is the evaluation of the suitability of a common algorithm, using an existing implementation, to an annotated dataset (with non-typical annotations), as well as the evaluation of the suitability of a set of features that can inform it.

We also examined whether a simple algorithm can be used for the detection of the more relevant location mentioned in one news article.

To aid in that endeavor, we also created a data model running and evaluation program that can run several different experiments in order to help a developer in quickly iterating over data pipelines according to test results. This allowed us to use several existing Machine

[1] https://www.embrapa.br/
[2] As translated from https://www.embrapa.br/quem-somos as of 2019/11/24

Learning libraries that are used in solving similar problems. This program could later be expanded into a more general tool to be used in solving other NLP questions.

1.4 Organization

This dissertation is organized as follows. In Chapter 2, we provide an overview of text modeling and processing, with a focus on how to solve the problem of detecting entities and Geotagging. In Chapter 3 we will discuss our approach in solving the problems described in the previous chapter and do a brief analysis of our dataset. In Chapter 4 we will talk about the implementation details of our work. Evaluation and discussion of the results will be done in Chapter 5. Finally, our concluding remarks will be presented in Chapter 6.

Chapter 2

Text processing

In this chapter, we will present required information in order to understand how text is processed in algorithms, with an emphasis on sequence models and how they can be used for the task of detecting entities in text.

2.1 How the computer sees words

A computer is a device for doing numeric computations of common number systems in a generalized way, without any intrinsic concept of what those numbers represent. Text, however, does not have the same mathematical properties and thus must have some abstracted representation in order to be processed by a computer. In everyday usage, such as in a text editor or a word processor, each potential linguistic symbol (or character) has a numeric representation and no further structure is needed. However, when trying to understand the information present in the text, it is useful to be able to recognize and use patterns to, if nothing else, minimize the computing power necessary for the task. In order to do so, one must find a way to represent the structure of the data we wish to work with.

The first way that was found to be useful [LM14] is to represent each word (and other sentence tokens, such as punctuation) as one unordered categorical number of arbitrary value. With such modeling, we can treat each token as an independent object and later attribute relationships between them as appropriate.

There are other possible representations for words, or even groups of words. If some sets of words have a specific meaning we wish to capture, we might want to consider them as different tokens with their own meaning. Perhaps those words can be grouped in groups of types (sometimes called tags) according to some criteria [HNP05]. Or we might represent each word with a numeric vector that represents some meaning with relation to the vectors other words have. The intuition behind the latter is that it is important to capture similarities, semantic relationships, and context of different words as used by human users

[BDVJ03]. By representing words in a multi-dimensional vector of arbitrary length where each word is attributed its own vector (with similar values to those of similar words), we try to capture these relationships in a latent space that cannot be captured directly. The techniques and algorithms created in [MCCD13a, MCCD13b] take advantage of the then recent advancements in Neural Networks to process a much more significant amount of words in a relatively small amount of time.

When algorithms cannot use vectors directly, we can create clusters[1] of similar words to attribute a numeric class that still caries some of the underlying meaning [HFS+17].

We must do more if we want to accurately model language, as we must also model the structural rules that a sentence must follow, such as knowing which group of words can appear before or after another group [KS63]. This assumes we are analyzing the same types of text, as grammar rules and word meaning can change according to the intended tone, audience or other contextual information that is not present in the text [You77].

Modeling sequences of tokens can be viewed and an application of Sequence Modeling [BPLA95]. In general, this modeling approach has slight differences whether one is trying to obtain information from an existence sequence or trying to generate a new one, but the theoretical mathematical reasoning remains the same - in this case, we are doing the former. Since each word being processed depends on the words surrounding it, models are based on probability theory or on approximations that are computationally similar.

2.1 Text mining for Portuguese

Working with the Portuguese language presents challenges that do not necessarily have to be dealt with in the English language, which, due to its current universality, is the language for which most studies have been done.

Of relevance to this work, it is important to mention that there are fewer reference resources for the computational processing of Portuguese, both in academic research and in annotated texts [SCP19]. For this work, we will focus on the Brazilian variant that our dataset texts are written in.

2.1 Text mining journalistic texts

Journalistic texts have characteristics that make them particularly suited to NLP tasks. They tend to be small, high-quality, and written in well structured formal language so that readers can absorb the content in a clear way that minimizes misunderstandings. Both of these limit

[1]clustering is the process of grouping data points (in this case, word vectors) with similar data points. The appropriate number of groups and the definition of similarity can vary according with the objective, but their values are usually somewhat defined by rules of thumb rather than complete optimization.

the existing variations of both words and structure, allowing for more accurate modeling [LN13, Li17].

2.2 Text pre-processing

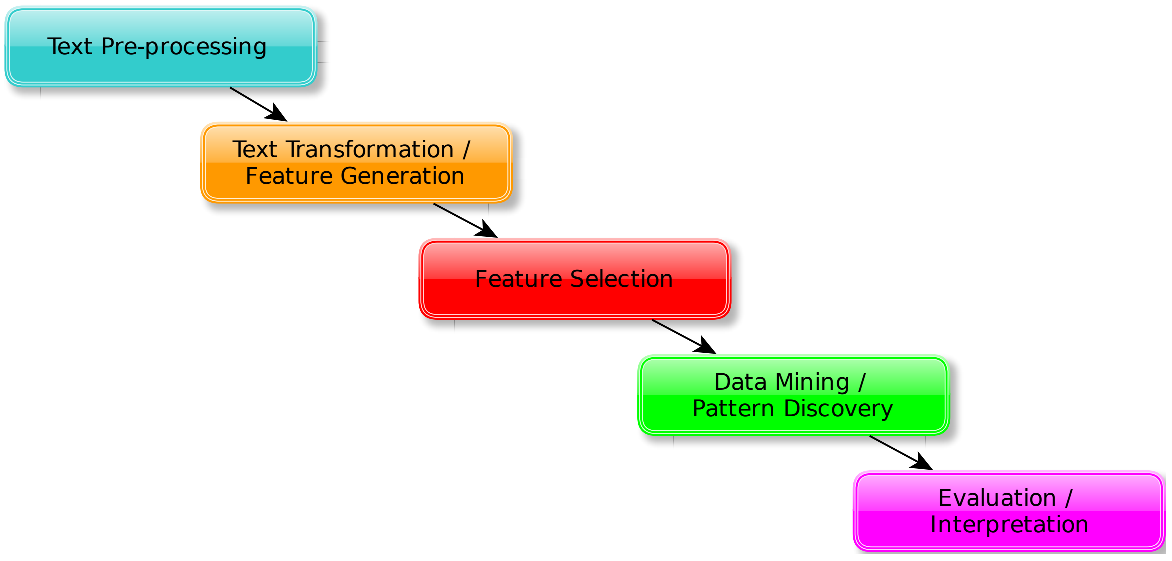

Text Pre-processing
Text Transformation / Feature Generation
Feature Selection
Data Mining / Pattern Discovery
Evaluation / Interpretation

Figure 2.1: Text Mining process

As mentioned in Section §2.1, there are steps to be followed in order to implement a language model, as can be seen in Figure 2.1. First of all, in the pre-processing stage, the documents must be stripped of all formatting (although this information could also be used as metadata), leaving only the text itself. This includes removing unnecessary or unwanted information, such as ads, formatting markers and so on.

What needs to be done afterwards depends on the task. For the case of detecting entities, the text needs to be separated into different tokens (words, numbers and punctuation). There are optional processing tasks that can be done at this point to aid in later phases, which we will take advantage of. Compound words can be separated (“da” = “de” + “a”), with the intuition that the individual components have their own meaning that can be shared between other repetitions of the same word. Secondly, in the opposite situation, when we know two words have a special meaning when occurring together (such as in a person’s name), we can join them. After that, sentences can be extracted (usually by punctuation), and, optionally, tokens can be classified according to their syntactic role in a sentence (plural, gender, verb, tense...). Some light transformation might also happen, such as stop word removal (of extremely common articles and adverbs, for example) or the replacement of words with a simplified version (such as stemming or lemmatization, see 2.2.3).

The text transformation step refers to the transformation into a data format that will later be used, while the feature selection step is where we decide what redundant or irrelevant features to remove. Data Mining is where a particular method infers a model of the provided information, which is later evaluated in the Evaluation phase. All phases of this model can inform changes to previous phases, as changes in processing are required.

2.2 Tokenizing

With the goal of separating the text into sentences and tokens, this should be the easiest task to perform, with minimal knowledge required. In theory, each sentence is separated by punctuation symbol or symbols (slightly varying by language, rarely by style) and each token by white-space (non-visible punctuation symbols), but dots might also be used by abbreviations or a sentence might contain structurally independent elements (quotations and parentheses), which changes the association between words. For example, the text "Mr. Smith said it's sunny. I really hate when it's hot outside." can be split in two sentences, "Mr. Smith said it's sunny." and "I really hate when it's hot outside.", being separated by a full stop ("."), but not when it is part of an abbreviation.

In many tools, such as the one we will be using, tokens are simply defined by regular expressions while being separated by tokens that do not fit, with exceptions for something like punctuation, which has defined sizes. For example, a word definition could be defined in a regular expression by "[:alnum:]+", which would capture all letters and numbers in the same token, but nothing else.

2.2 Part-of-Speech Tagging

This task consists in determining the grammatical function each token has in a sentence, so that an analysis can start to extract some meaning. The set of tags to assign to each token requires a substantial amount of linguistic study so that proper categories with meaningful distinctions can be attributed. There is not one objective ideal set of tags to apply to words, and non-trivial sets are not very easy to adapt from one language to the next in order to form a common basis for machine learning progress. Nevertheless, there have been attempts to normalize the classification, such as in [dMDS+14, Zem08, PDM11].

For example, the Portuguese sentence "De acordo com o presidente da empresa, Paulo Basílio, o resultado foi beneficiado por um mercado favorável de exportação de milho e açúcar no Brasil." can be tagged as seen in Table 2.1, where nouns (N), verbs (V), punctuation (F) and so on are clearly distinguishable. These particular tags are slightly longer than usual, but they are designed with the possibility of being cut with whatever precision is needed. For example, nouns (N) can be common (C) or proper (P), have feminine (F), masculine (M) or common (C) gender and so on with plural, entity type

Word	POS Tag	Probability
De	SP	1
acordo	NCMS000	0.998155
com	SP	1
o	DA0MS0	0.950254
presidente	NCMS000	0.998938
de	SP	1
a	DA0FS0	0.675415
empresa	NCFS000	0.994778
,	Fc	1
Paulo_Basílio	NP00SP0	1
,	Fc	1
o	DA0MS0	0.950254
resultado	NCMS000	0.936047
foi	VMIS3S0	0.5
beneficiado	VMP00SM	0.721873
por	SP	1
um	DI0MS0	0.877848
mercado	NCMS000	0.997835
favorável	AQ0CS00	1
de	SP	1
exportação	NCFS000	1
de	SP	1
milho	NCMS000	1
e	CC	0.999951
açúcar	NCMS000	1
em	SP	1
o	DA0MS0	1
Brasil	NP00G00	1
.	Fp	1

Table 2.1: Sentence Tagging

(optional process) and degree information.

The final number represents how correct it believes it is. This is due to the fact that words can be used for different linguistic purposes, either syntactically or semantically, which is compounded in cases where there is a small training set or the text has mistakes. For example, in the english sentence «"I see," said the blind man as he picked up the hammer and saw.», version 4.1 of the tool *Freeling* mistakenly does not tag the word "saw" in the same manner as "hammer", tagging it as a verb instead of a noun.

Word	Stem	Lemma
university	univers	university
universities	univers	university
universal	univers	university
data	data	data
datum	datum	datum
is	is	be
are	are	be

Table 2.2: Stemming and Lemma examples, done in *NLTK* and *Freeling*, respectively

2.2 Stemming and Lemmatization

Stemming is a processing step in the pipeline that provide algorithms with a feature that represents meaning relations between words. It consists in reducing a word by cutting off a list of common prefixes and suffixes, in order do create a representation shared with similar words, such as in verb conjugations, words differing solely by quantity or qualitative adjective suffixes [NS07].

Similar to Stemming, Lemmatization is a process of reducing words into a shared common representation, but in this case it is done by a morphological analyses, which necessitates a human built dictionary of words into their lemmas. If a word was not calculated, you do not get a good answer.

In Table 2.2, we can see examples of both ideas, which are implementation dependent. We can observe some concept confusion (overstemming) that can happen in the stemming process, as “university” and “universal” do not share similar meanings. The opposite effect (understemming) happens with “data” and “datum”, or “is” and “are”.

2.3 Existing Work

Machine learning consists of the study, within scientific principles, of algorithms in order to train statistical or quasi-statistical models of complex tasks that are, to some extent, repetitive and have need of unattended automation. This can happen because of the existence of data patterns in the training data that allow for the parametrization and tuning of those models, despite the fact that manually creating an algorithm would be a very complex and time-consuming human task [HKP11].

Machine learning makes use of several fields of mathematical knowledge, such as statistics and optimization, and is being used in many fields, from content filtering to disease detection or self-driving cars. In simple terms, it is the task of discovering parameters for mathematical equations that best fit available data, usually with some amount of tuning of hyper-parameters that, with the exception of some heuristics to cut the solution space, can only be optimized by brute force.

Supervised learning, one group of the techniques for training a model, is used when someone wants to have an algorithm that can discern the relation between input variables and an expected result. In it, part of that available data, called training data, is mathematically used to automatically find an optimal (or sufficiently close, by some criteria) model that minimizes some error metric between the model and the data, often while allowing for the possibility of some generalization to non-seen information. The remaining part, around a quarter to half of the total, is used to evaluate how well the model generalizes to data it has not seen yet, as a parallel to applying the model in the real world at the end of the process. On the other hand, in unsupervised learning, we don't know the result, and thus the general idea is to find useful patterns in the data, with the intuition that related data points are represented with similar values and unrelated data is not. As the groupings do not have an easily interpretable meaning, in practice, they tend to be used as input into another process.

2.3 Entity detection

Detecting entities mentioned in a text is a good first step towards extracting or interpreting information from text. It usually means the detection of one of four types of real world entities (persons, locations, organizations and other) as a general approach [GS95], but by no means is the task limited to these sets and can be adapted according to the application.

There also does not exist a strict definition of what constitutes an entity. A pronoun, like "it" or "she", references an entity, but provides no information about it. Several adjectives, like "American" may also be considered to represent an entity (the USA, in this case), or they may not. What annotated data is available to train the model can have a big impact on the decisions that are made, since it takes a fair bit of human work to annotate.

Conceptually, and potentially also in the algorithms used, the task can be subdivided in two: entity detection, and the classification of those entities in the above mentioned types. The former is the task of detecting groups of consecutive words that refer to one entity (for example, words that look like a name), while the later is the task of guessing in which pre-defined set of types it belongs to.

In practice, the naming of the task is sometimes more precise, depending on the aim of the communication. If we want to be precise, NER is the detection of the entity, NERC is the process of NER followed by the classification of said entity [NS07].

After detecting a set of entities in a document, one possible continuation is the association of one location to the text, a process called Geotagging, which depends on resolving the ambiguity of each location, which is itself a task called Toponym Resolution [KR18].

The following subsections will describe existing work for the question we are trying to answer. Specifically, we will first mention techniques for extracting information in general, with a focus on entities, and then some techniques for extracting the most relevant location.

2.3.1.1 Human baseline

While strictly speaking these are not Machine Learning methods and were more relevant before modern computing capabilities, they serve as reference work and are still useful where the former are not useful.

Rule-coded entity detection

Nowadays, we can easily find and obtain computing power. Not long ago, however, creating and using statistical models for text was often impractical. Since word frequency tends to follow Zipf's law [Pia14] (a statistical distribution where frequency drops exponentially), having a big relative increase in available texts means that we have many unknown words. But, for a long time in the history of computers, working with many texts meant that training would be slower and it would be hard to have memory for it.

An alternative was to find regularities in texts and create patterns that were expected to match words, often expressed with the power of regular expressions [MMG99]. We could, for example, expect capitalized words that follow "Mr." to be a (male) person's name, words preceding "Corp." to be companies or "buy" preceding a product's name. Rules can also be longer, such as "person works at company" being able to detect both the person and the company. There is also the possibility of using bootstrapping techniques, such as starting with a list of pairs of entities with some relation and letting an algorithm figure out what words are used in between.

While often useful, every rule (or meta-rule) must be hand-built and checked for exceptions, potentially making them complicated to understand. Many are also inevitably domain-dependent, e.g. what make sense in an heavy industry context might be different in a food industry context.

Dictionary-based entity detection

Since we live in a computerized world, many approaches tried using the existing documented knowledge of existing entities. This knowledge can be summarized in a dictionary called a *gazetteer* [SO06], which can then be used to match words to be screened for entities. This is especially useful when there is a limited set of entities to be found, but, otherwise, a large database such as Wikipedia or Wordnet can be used to obtain a list [GDL+13].

Using a set of predefined words is a useful help for entity detection. However, it is usually not enough to rely solely in a gazetteer, since: it may not contain all the entities to be found; words might have many homonyms. E.g. the late Queen Victoria had her name given to, among others, a lake in Africa, a Canadian state and an Australian state. Those states then have universities, monuments and museums based on their state's name, to add to those present in the city of London. If we were trying to limit our search in some way, or trying to categorize them, the word "Victoria" would not be enough.

2.3.1.2 Sequence Models

Some Machine Learning problems can be dealt with by considering each event (or data point) as being solely dependent on the characteristics of that event and completely independent of all others, including ignoring the order. For example, one can predict when a bus will need to be fueled based on its past performance and its assigned services. It is also possible to model repetitive events by fixing the amount of past information to model in each sample, such as in predicting whether it will rain in a particular day by always using the last X amount of days and various other seasonal parameters as variables of that event.

Others, however, cannot (accurately), as data is not independent and the order of events is important. One cannot say whether a sentence is a compliment without analyzing the position of the words; similarly, one cannot generate a sentence with each word as independent.

In this work we assume to be always working with text for simplicity, but it is important to mention that the concepts that we will present next are applicable to other domains.

In theory, it would be optimal to model a text sequence with words (and other tokens like punctuation and numbers, all called words here for simplicity) w_i of total length n as

$$p(start, w_1, w_2, \ldots, w_n, stop) = \prod_{i=1}^{n+1} \gamma(w_i | w_1, w_2, \ldots, w_{i-1}) \tag{2.1}$$

where p is the probability of the sentence according to some function γ, and $start$ and $stop$ are markers [KN93]. In simple terms, the probability of a sequence of words would be the product of a distribution function of the probability of a word appearing after all the words that are before in the sentence. However, there are a couple practical problems with this language model: one is that each different sequence must have its probability modeled, as each has different parameters; the other is that it is impossible to gather all possible sentences, as any non-modeled one does not have any value. The opposite would be to consider each word as independent,

$$p(start, w_1, w_2, \ldots, w_n, stop) = \prod_{i=1}^{n+1} \gamma(w_i) \tag{2.2}$$

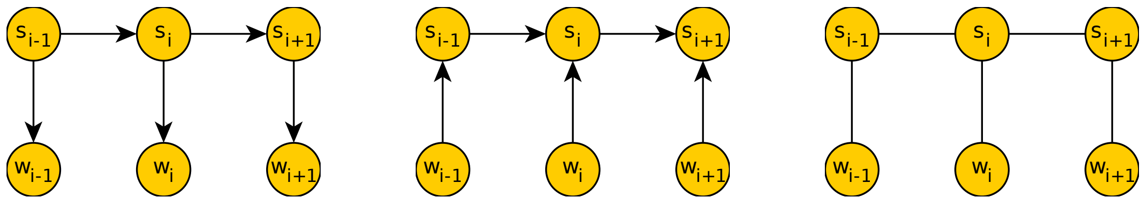
which would be must simpler and open, but would have no properties over sentence structure.

Markov Models [BJM83] are the happy medium where you define what balance between the two extremes of all or none is appropriate by choosing how many words you look back (so, $\prod_{i=1}^{n+1} \gamma(w_i|w_{i-l}, w_{i-l+1} \ldots, w_{i-1})$). The disadvantage is that the probabilities are not as straightforward to calculate, but it is a well studied problem.

However, it would be very useful to use language models for more than sentence modeling. We would also like to extract the information present in the text, and we must start with information about each word in sequence with information dependent of other words. What this means in practice is exemplified in 2.2.2, where each word is tagged with a syntactic class, but it can be something else, like whether it is part of a name. This can be modeled mathematically in what is called Hidden Markov Models (HMM) [dP98] by changing the equation to

$$p(start, s_1, w_1, s_2, w_2, \ldots, w_n, w_n, stop) = \prod_{i=1}^{n+1} \eta(w_i|s_i) \cdot \gamma(s_i|s_{i-m}, \ldots, s_{i-1}) \tag{2.3}$$

where s_i represents some hidden state over the word w_i.



(a) Hidden Markov Models (b) Maximum Entropy Markov Model (c) Conditional Random Fields

Figure 2.2: A graphical representation of the flow of information. Note that HMMs are the only model that can generate word sequences and CRFs are the only model where information can flow in every direction.

In this case, the words are observed and we would like the most probable states over the whole sequence. In practice, this is most efficiently solved by the *Viterbi* dynamic programming algorithm [Vit09]. In a simple explanation, it calculates the possible transitions from start to finish, with some paths stopping due to impossibility, and then walks backwards picking the transition with highest probability. For training with labeled data, statistics tells us that we should use the maximum likelihood principle,

$$\langle \hat{\gamma}, \hat{\eta} \rangle \leftarrow \underset{\langle \gamma, \eta \rangle}{arg\,max}\, p(s, w \mid \gamma, \eta) \tag{2.4}$$

with each state's transition and emission distributions being able to be estimated independently, and thus very quickly.

In the end, though, HMMs are very useful models that still classify one word at a time with limited feature information, which makes them reliant on the words being in the training data. They also assume a linear relation between probabilities, which limits the expressive power of the equations. While HMM can and have been adapted for sequence prediction (e.g. topic segmentation, POS tagging and information extraction [MMS99]), the other problems remain. This happens, in essence, because it is a generative model, that is, it models the generation of text and must thus limit the dependencies in order to be tractable.

Two successful discriminative models appeared in the evolution of trying to improve on these issues, Maximum Entropy Markov Models (MEMM) [MFP00] and Conditional Random Fields (CRF) [LMP01]. They both model the probability in a log-linear distribution,

$$p(y|x, w) = \frac{exp(w \cdot f(x, y))}{Z(x, w)} \tag{2.5}$$

with x, y representing the input and output sequences respectively, $f(x, y)$ representing d features, $w \in \mathbb{R}^{\mathrm{d}}$ being the parameters of the model (called weights), and $Z(x, w)$ being a normalizer, $Z(x, w) = \sum_{z \in Y^*} exp(w \cdot f(x, z))$. They differ in how they estimate the weights: MEMM assumes $p(y|x, w)$ decomposes, reducing the problem to multi-class logistic regression, while CRF does the same with $f(x, y)$, estimating the log-likelihood of the data, a more complex optimization problem.

Potential features to be represented in f for each word x_i are any indicator (binary variable) that can potentially indicate whether that word is a good candidate for a label, including peeking at close words. This will generate a very-high dimensional feature space for which the learning algorithm will learn weights that give the proper importance to relevant features.

As mentioned previously, there are many possible ways to computationally represent (or model) text, let alone represent all the extra information about it through whatever layers of software are needed to solve a particular problem. With regards to this work and the models described above, words themselves are represented as categorical numbers (independent and with no particular order) in HMMs, and as how many other features are represented in the particular implementation of MEMM and CRF.

The property of a word being an entity or not can be expressed in a feature whose value was described in [Mar95], where non-entities are marked with O (for other), B-x (beginning of entity of type x) or I-x (intermediate and final words of that entity). The separation between B and I allows the easy separation between consecutive different entities of the same type. A similar scheme is called BILOU, that can sometimes provide better results [RR09]. The differences are the additions of tagging the final word as L-x and single word entities as U-x.

2.3 Geotagging text

Geographic information can have an important role in filtering and aggregating information [MMBSV09], which has created the need of research into how to extract that information. In order to do sociolinguistic studies, for example, it is useful to be able to extract a single location as being the most important of a document [MM17]. This location can be abstracted as the geographic region that best encapsulates all the contained locations, which is usually done with the use of geographic coordinates in order to avoid ambiguity. This task has been a subject of much recent research, although there are problems comparing methodologies and results.

A common approach to this task is the detection of as many existing entity locations as possible, using the previous section's methods or others. Afterwards, it is necessary to disambiguate the found location mentions, as there can be many places with the same name. For example, “New York” and “São Paulo” are both states and municipalities in the US and Brazil, respectably.

In order to disambiguate location references, there are a few useful heuristics to use [LSNL02]:

- **One referent per discourse**: a location name that is ambiguous is likely to refer to the same real life place when mentioned several times per context (e.g. a news article);
- **Related referents per discourse**: multiple mentioned locations are likely to be spatially related, which makes it probably that they are close together;
- **Default senses**: important places are more likely to be mentioned (e.g. Paris in France is more likely to be mentioned than Paris in Texas), so an ambiguous mention can be assigned a default reference.

2.4 Evaluation

Ideally, a machine learning model should be evaluated automatically by the the same pipeline that processes the data. This provides consistency between the evaluations while

Token	Reference	Predicted
in	O	O
New	B-LOCATION	B-LOCATION
York	I-LOCATION	I-LOCATION
.	O	O

(a) Predicted an existing entity

Token	Reference	Predicted
a	O	O
Beautiful	O	B-LOCATION
New	O	I-LOCATION
World	O	I-LOCATION

(b) Predicted a non-existent entity

Token	Reference	Predicted
in	O	O
São	B-LOCATION	O
Paulo	I-LOCATION	O
,	O	O

(c) Missed an entity

Table 2.3: Basic prediction cases, where the B-LOCATION annotation denotes the first word of a location and I-LOCATION the others

providing them as soon as possible. However, this necessitates the availability of annotated data, although, in same cases, there are abstract methods of scoring solutions (such as the value of generated words vectors).

For a classification model, such as saying whether a word is part of an entity, there are many metrics that can be used. The simplest is to calculate the accuracy metric by dividing the count of correctly tagged text tokens over the full token count. However, while it might be useful to use for tracking development progress, it is not usable for the evaluation entity recognition, since most words are not part of an entity, which leads to very high (>90%) scores even for bad models. Secondly, it does not properly penalize missing words in multi-word entities, as it would still allow full partial score. The latter might make sense in particular scenarios with particular words (such as a teacher grading a student might do, for example, mistaking "Ms". and "Mss."), but this must evaluated on a case by case basis.

In Table 2.3, we can see the basic scenarios that can happen when detecting an entity word by word. This classification scheme allows for the use of the normal classification measures of true positives (good positive prediction, scenario a), true negatives (good negative prediction, when both tags are O), false positives (bad positive prediction, scenario b) and false negatives (bad negative prediction, , scenario c). For the particular problem of NER, at some academic conferences [SdM05], correctness is only attributed when all tokens, and only those tokens, are at their expected position for each entity (so all other mismatches not present in Table 2.3 would be a miss). As above, this can also be adapted according to real-life requirements.

From these basic measures, we can get more summarized metrics, such as Precision and Recall. Precision is defined as the fraction of correct positive classification predictions over all positive predictions, or $precision = \frac{TP}{TP+FN}$, while Recall can be defined as the ratio between the former and the expected number of positively classified instances, $recall = \frac{TP}{TP+TN}$ [Pow11]. However, using either of these metrics for evaluation can be problematic, as they can give better scores to degenerate methods: a method that predicts one correct case and nothing else has 100% Prediction, while a perfect score on Recall can be achieved by classifying all documents as true.

As a way to balance both requirements, a commonly used metric is F_β, defined as $F_\beta =$

$(1+\beta^2)\cdot\frac{2\cdot precision\cdot recall}{\beta^2\cdot precision+recall}$, with β often being equal to one, giving the harmonic mean between both metrics.

There are a few more things to consider, however [WFHP16]. It is not very interesting to have a model that only learns the data it is given, as in real applications one would want to keep applying the model to other data where we do not know the answer. To that end, data is split into (at least) two groups, a training set and a test set, in a technique called *Holdout*, so that we can estimate how well it would to against unseen data. This then leads to the question of how this split should be made, as, ideally, we would want both groups to be big. It has been found splitting it randomly in about 2/3 for training and 1/3 for testing tends to have good results[2].

However, you can end up being unlucky and not having enough good discriminating examples in one of the sets, which is fairly likely to happen when you have a small amount of samples. A simple idea is to repeat this process several times, which has been standardized in a technique called *Cross-Validation*. With it, you choose a number of approximately equally-sized partitions to do (folds), and then train and test the model by leaving a different fold out the same number of times. In practice, 10 folds has been found to be a good number for good results.

[2]assuming no dependencies between both groups, which is not always possible

Chapter 3

Location detection and extraction approach

In this chapter we describe our approach for the tasks of detecting entities in text and finding the most important of them based on knowledge presented in the previous chapter. We start each case by describing how to build a model, then how to apply it to our dataset, which we will describe, and finally how to evaluate it.

3.1 Location Detection Model

From the sequence models presented in 2.3.1.2, it is apparent that HMMs are problematic for many tasks. There are two problems with this approach: many tasks, such as NER, benefit from rich representations, since they allow an algorithm to infer qualities from unseen words; the second is that the text is modeled as a generative joint probability, not in a conditional problem where the text is the input [Mcc03].

MEMMs are still fairly simple to understand and computationally train. However, they still suffer from the *label bias* problem that arises from only comparing transitions between labels in a localized way [PPTT13]. In essence, the fewer states are valid in a transition, the more likely it is to be chosen as correct, as more weight gets distributed for later transitions. CRFs were designed to get around this problem by using the joint probability of the entire sequence (it learns the part within the exponential), thus it is able to trade weights to different states and features [LMP01]. For a small dataset, these effects are more pronounced, as word co-occurrences are rarer, and the computing costs of CRFs become much less relevant (and significantly less pronounced then at the time when they were created). For those reasons, we decided to focus on exploring the possibilities of CRFs for solving our problem of detecting entities in texts.

3.1 Building the Model

After extracting raw texts from a dataset, which depends on their origin, we can move into doing pre-processing tasks. Since NER is a very knowledge intensive task [RR09], we will try many features that might be useful in helping the algorithm to discriminate words according to their context, letting it figure out the appropriate weights for each one. Due to the high number of them and the impracticality of testing every variation, we will group them by intent and similarity.

It is important to note that the features accepted by the algorithm are categorical and the algorithm learns their weights while establishing no relationship between them. For example, there is no relation between a category that is true when the first letter is capitalized and one that is true when the second letter is capitalized, all that matters is whether it is present or not.

We will first tokenize the text, separating words and punctuation into separate tokens on which we can calculate attribute features. These features will be attributed names that will accompany the description„ so they can be exemplified on Table 3.1. With each individual token, we will start obtaining features and associating them with each token, starting with the simplest ones:

word.lower lower-case version of word,

word.len token length in visible symbols,

word.[-n:], word.[:n] prefixed and suffixes (first and last letters) of the word, with n being between 1 and 3 characters.

These shall be the default features for every variation of the algorithm, to serve as a baseline for further additions.

We shall then experiment with adding many of the syntactic word annotations that *Freeling* provides, since we will use it for POS-tagging where this information is provided. However, for brevity of information, we shall not show them on the example. This grouping will be called SYN_TAGS and its constituents are:

word.degree degree (superlative/evaluative),

word.gender gender (male/female/neutral),

word.plural whether the word is plural,

word.possessive whether is it a possessive verb,

word.number to what counts the word applies (plural/singular/invariant),

word.person (first/second/third) person verb,

word.pronoun pronoun casing,

word.mood mood (indicative/subjunctive/imperative/past participle/gerund/infinitive),

word.tense verb tense,

word.puncttype punctuation type,

word.openclose whether it is an open or closing punctuation mark (e.g. "(" vs. ")"),

word.polite is honorific.

We will then create features to indicate something about the token in the group INDICATIVE:

word.isnumber whether the token is a number,

word.ispunct whether the token is a number or punctuation,

word.istitle whether the token is title-cased (like the first word in a sentence or in most names),

word.isupper whether the token is upper-cased,

word.hashyphen whether the token has a hyphen,

tag.POS part-of-speech tag attributed by the chunking process done by *Freeling*[1],

word.BOS whether the token is the first of the sentence,

word.EOS whether the token is the last in the sentence.

After that, we will try to introduce some features that provide transformative generalizations of words to provide human word knowledge, and call the group TRANSFORMATIVE:

word.stem token stem,

word.lemma token lemma,

[1]Freeling does POS tagging with the EAGLES tag set, a proposal that aimed at creating common ground for European languages.

word.pattern token pattern, a simplified representation of token structure where tokens of the same type are compressed into a common representation. What we have considered as types are letters, either capitalized or not, numbers, punctuation, quotation symbols, parentheses and similar, and other. It is best understood in examples: "Paulo" becomes "Aa+", "2019" becomes "9+" and "Papel-moeda" becomes "Aa+- a+". It is partially intended to avoid creating many features that indicate word characteristics and inspired by summarized patterns described in [NS07].

We will create popularity ranking functions according to usage count in the training set, with the intent of helping the algorithm detect stop words (words that are usually used as glue and do not carry much distinctive meaning, such as, in English, "the", "this", "is" or "a") at various levels. The RANKING group is called:

word.logrank logarithmic rank of the token, ordered by frequency in the test set,

word.centrank centesimal rank of the token.

We will add a feature based on word vectors to try and take advantage of the capability of this representation in capturing semantic and syntactic similarities between words in group W2V, similar to [Sie15]:

word.w2v_cluster word2vec cluster number, which are pre-calculated clusters from word vector representations. Since this representation needs a lot of training data to properly model tokens, we have downloaded existing clusters from `http://nilc.icmc.usp.br/embeddings`, arbitrarily choosing the 600 dimension CBOW version. We then created 1000 clusters by using the MiniBatchKMeans method in *sklearn* and saved the result, which will be looked up on demand to create the feature for the token. This was based on the work done in [HFS+17].

We will then add a feature, to indicate whether the word is present in a list of words used in locations, as described in [RBB03]. Since our goal is to find a well defined and small group of locations, we expect this do be a simple way to help the algorithm.

word.gazetteer whether the word is present in the list of words in the expected entities.

At the same step, we shall process the annotations of the dataset and mark the words corresponding to those locations, as this is necessary in training the model (it is not presented in other situations):

word.bio IOB or BILOU entity annotation, depending on the model version.

An example of these features for a sample sentence can be seen in Table 3.1. As this is a sequence model, it is necessary for a token to have annotations of surrounding tokens as well. We opted for small symmetrical windows of 2 and 4, as these seem to be sufficient to model dependencies in this task, and this is marked in the feature name as a prefix of a number of -n to n.

Example Sentence	*Há*	*uma*	*pressão*	*muito*	*grande*	*de*	*o*	*mercado*	*,*	*assegurou*	*Kardec*	*.*
0:word.lower	há	uma	pressão	muito	grande	de	o	mercado	,	assegurou	kardec	.
0:word.len	2	3	7	5	6	2	1	7	1	9	6	1
0:word.[-3:]	Há	uma	são	ito	nde	de	o	ado	,	rou	dec	.
0:word.[-2:]	Há	ma	ão	to	de	de	o	do	,	ou	ec	.
0:word.[-1:]	á	a	o	o	e	e	o	o	,	u	c	.
0:word.[:1]	H	u	p	m	g	d	o	m	,	a	K	.
0:word.[:2]	Há	um	pr	mu	gr	de	o	me	,	as	Ka	.
0:word.[:3]	Há	uma	pre	mui	gra	de	o	mer	,	ass	Kar	.
0:word.isnumber	0	0	0	0	0	0	0	0	0	0	0	0
0:word.ispunct	0	0	0	0	0	0	0	0	1	0	0	1
0:word.istitle	1	0	0	0	0	0	0	0	0	0	1	0
0:word.isupper	0	0	0	0	0	0	0	0	0	0	0	0
0:word.hashyphen	0	0	0	0	0	0	0	0	0	0	0	0
0:tag.POS	VM	DI	NC	RG	AQ	SP	DA	NC	Fc	VM	NP	Fp
0:word.BOS	1	0	0	0	0	0	0	0	0	0	0	0
0:word.EOS	0	0	0	0	0	0	0	0	0	0	0	1
0:word.stem	há	uma	pressã	muit	grand	de	o	merc	,	assegur	kardec	.
0:word.lemma	haver	um	pressão	muito	grande	de	o	mercado	,	assegurar	kardec	.
0:word.pattern	Aa	a+	a+	a+	a+	a+	a	a+	.	a+	Aa+	.
0:word.logrank	6	3	-1	4	5	0	1	3	1	-1	-1	1
0:word.centcrank	8	0	20	0	1	0	0	0	0	0	0	0
0:word.w2v_cluster	-1	634	724	589	621	597	591	129	634	40	-1	634
0:word.gazetteer	0	0	0	0	1	0	0	0	0	0	0	0
0:word.bio	O	O	O	O	O	O	O	O	O	O	O	O

Table 3.1: Example features for each word in a sentence.

Sequences of these calculated features, (i.e. sentences transformed into a list composed of groups of these features) can then serve as input for the CRF algorithm to create our learning model, on which we will use almost all default hyper-parameters, or to use it after training. The non-default hyper-parameters are “feature.possible_states” and “feature.possible_transitions”, which are usually disabled in order to significantly decrease training time at the cost of a small performance decrease. The former setting allows the algorithm to generalize information over states that do not exist in the training data, while the later allows it to generalize over unseen transitions.

Note that only the features are supplied, as the text is represented in other ways making it redundant, and also that the provided name and order of features should not matter, as long as they remain consistent between the different usages of the model.

3.1 Application

We will apply our model to an existing dataset, a group of news in XHTML[2] format provided by *Embrapa*. In order to use our created model, a dataset needs to be transformed into a list of sentences grouped in a list of documents, as each sentence will separately be used for model learning. For this dataset, each document will have its sentences extracted by first finding paragraphs with an XML processor provided with an XPATH[3] expression, "body/font/p". This process returns text lines with extra residual spaces at the edges that need to be removed, due to the rules of the original format.

This particular dataset also needs to be UTF normalized (at any point after obtaining the text), as it does not use the modern universal format and also has some non-printable characters that can confuse the algorithm by creating different tokens. In particular, we will use NFKC normalization, to ensure different representations of the same character remain consistent, and then remove characters U+0094, U+0093, U+0096 and U+0092.

These paragraphs can then be joined in a single paragraph with all sentences put together, as there is no further use of paragraphs, at which point we can begin the process described in the previous section, starting with tokenization.

When we are building or evaluating the model, we also need to obtain the annotations. Usually, these are marked on the words in which they appear, as that allows for the most precision in understanding where the reference is and to what exact entity it refers to ("São Paulo" is both a city and a state, for example). However, we were only given a list of locations per news item in an Excel spreadsheet, which makes it impossible to properly attribute the correct location. Because of this, we will be forced to do a literal search for the words of each entry in the annotation file (after taking into account that it does not have accents) and attributing the highest region level of that name present in the annotation to the corpus words. It also means cannot generalize to words that are related to a location, such as considering "Paulista" (which can refer to the residents of "São Paulo"). The lack of generalization allied with dealing with solely one type of entity should both worsen the expected performance; on the other hand, we only want to detect a specific list of entities, which should make the solution perform better.

After applying the model to the dataset, we will get a list of IOB (or BILOU cases, depending on the training), one for each token in the provided text. An entity is then any sequences of tokens where the corresponding (i.e. with the same position) return sequence starts with value "B-LOCATION" and is optionally followed by values of "I-LOCATION" (for BILOU, the sequence must end with a "L-LOCATION" and the "U-LOCATION" value marks a single word entity).

[2]an HTML version that is compliant with the XML encoding format
[3]XPATH is a language for expressing the location of elements in XML

News	State	Meso Region	Micro Region	Municipality
41737	Goias	Sul Goiano	Quirinopolis	Quirinopolis
	Parana	Norte Central Paranaense	Porecatu	Prado Ferreira
	Goias	Sul Goiano	Meia Ponte	Bom Jesus de Goias
41739	-	-	-	-
41740	-	-	-	-
41746	Ceara	Norte Cearense	Baturite	Baturite
	Mato Grosso do Sul	Sudoeste de Mato Grosso do Sul	Dourados	Dourados
41748	Goias	Sul Goiano	Quirinopolis	Quirinopolis
41749	-	-	-	-
	-	-	-	-
41750	-	-	-	-
41751	-	-	-	-
41752	-	-	-	-
41753	Mato Grosso do Sul	Sudoeste de Mato Grosso do Sul	Dourados	Rio Brilhante
	Mato Grosso do Sul	Sudoeste de Mato Grosso do Sul	Dourados	Nova Alvorada do Sul
	Mato Grosso do Sul	-	-	-
41754	Sao Paulo	Ribeirao Preto	Ribeirao Preto	Ribeirao Preto

(a) Small sample of the annotations

State	Meso Region	Micro Region	Municipality
Bahia	Salvador	Alagoinhas	Acajutiba
Bahia	Salvador	Alagoinhas	Alagoinhas
Goiás	Rio Verde	Rio Verde	Porteirão
Goiás	Rio Verde	Rio Verde	Rio Verde

(b) Small sample of territorial divisions

Table 3.2: Annotations and reference

In order to apply the model to another input in order to detect entities, we should do similar normalization to ensure a clean text, and then start at the previous section's tokenization process as well.

3.1.2.1 Dataset Analysis

The data available from *Embrapa* consists of 281 annotated industrial news stories, of which 143 contain entities that were deemed relevant in the annotation process. These entities are the ones that represented (from 1989 to 2017) the administrative regions of Brazil in several detail levels: State, Meso Region, Micro Region and Municipality. The annotations also present other levels that are not of interest to this work: Macro Region, Region (both of which are not usually directly mentioned in the dataset) and Factory name, as it is too specific. In Table 3.2a, we can see the format of available annotations, while on Table 3.2b we can observe a partial list of the official administrative divisions as obtained from the Instituto Brazileiro de Geografia e Estatística (or IBGE, literally Brazilian Institute of Geography and Statistics) website. Note that the annotations are expected to be done from the level of the specific mention leftward, with one line per mention.

We count 1144 individual annotated locations, with an average of 4.847 per news story (standard deviation of 7.384), which becomes 940 after removing name repetitions per annotation (average 3.983, std 6.085). When applied to the text, we can find 524 locations in total (average 2.22, std 3.328), with 916 tokens marked as part of an entity (average 3.881, std 6.467) out of a total of 118516 (average 502.186, std 272.742). We miss 739 entities present in the annotations, although a few clearly errors ("ia" or "andia" do not exist).

The most prevalent locations are “São Paulo” (97 presences), “Ribeirão Preto” (32), “Mato Grosso do Sul” (30), “Goiás” (28), “Pernambuco” (26) and “Mato Grosso” (24). The most missed annotations are “Piracicaba” (78), “Campinas” (56), “São Paulo” (44) and “Ribeirão Preto” (32).

Clearly, some re-annotation work should be done to take the most value out of the dataset. But, even then, we shall see that it is possible to extract value out of it.

We would also like to describe some examples of problems that exist within the annotations. We did not do a full examination, as that is normally outside the scope of a dataset user. In the case of one news article, we only have a single line of annotation, but both the state and the municipality are mentioned. For another article, several municipalities are mentioned (“Franca”, “Cristais Paulista”, “Itirapú”, “Patrocínio Paulista”, “Pedregulho”, “Restinha”, “Ribeirão Corrente”, “Rifaina” and “São Josẽ da Bela Vista”), but we only have one annotation for the micro region. We noticed one instance of swapped news, and one instance where the locations were not present.

3.1 Evaluation

In order to assess the value of our feature groups described previously, we will generate various models of different combinations, adding and removing groups as performance increases or decreases. We will also test both IOB and BILOU annotation types, to see if there is an advantage to either of them, and we will test window values of 2 and 4 (i.e. each token will have the same set of features for each of their surrounding tokens, within the window distance).

The evaluation of these differently trained CRF models will be done by applying the F1 measure on each entity mention over a cross-validation of 10 folds. As we have only one entity type, it seems appropriate to be strict and only consider a true detection when an entity is found in its correct place, and either a false negative or false positive otherwise (see Section §2.4). The cross-validation folds will be determined by equally attributing all of the sentences in the available annotated data, first by cleaning it as described in 3.1.2, and then creating the features as described in 3.1.1. The Precision, Recall and F1 metrics will then be averaged over those runs to provide a comparable score between the models.

In order to provide a comparison with simpler solutions, we decided to compare against four alternate models. Our baseline will be doing entity detection by simply using the Gazetteer with a list of words with which to mark words as entities. The other alternatives consist of the use of some existing generic entity detection tools supporting Portuguese with an added word entity filter based on the same gazetteer. Since they detect far more types of entities, this will be a necessary step to more accurately compare the results.

3.2 Geotagging

A common approach to geotagging a document with the most relevant location is to find the biggest location that encompasses the regions mentioned in the text [AMC09], although machine learning approaches have also become popular [Lei16]. Since we do not have annotated data for this step, we will not do a learning approach and instead experiment and manually evaluate a simple approach inspired by [MMBSV09]. This paper discusses a joint task of discovering entities and assigning them a relevance score at the same time, while we will instead use the results of the models created for the tasks.

Our geotagging model will need the same preparation work of tokenization that the entity detection methods need for application, as it will rely on them to find entities in the input. It will then run the different models to classify the tokens as belonging to an entity or not, but taking a positive response from any of them as true. The rationale for this choice is that our trained model and the more generic tools that will be used for our baseline will have different strengths in entity detection. After this classification process, we can extract a list of entity names and a small text window of their surrounding tokens.

Relying on the heuristics presented in 2.3.2, we will attribute to each different entity name a single geographical location equivalent to the highest hierarchical level in the administrative region list in which that name exists. In case a name exists in more than a single sub-tree at the same level, we will select the one with the most subdivisions (partially as a surrogate for population count), as it is likely to be more relevant.

After having this information, we can select from the list the location that is highest in administrative level. In case of a tie, the algorithm will try to find if there is an upper level that contains them and assign it as the most relevant location. If one does not exist, it will disambiguate by selecting the appropriate entity at that level that does the most grouping of the other entities.

Since we do not have annotations, we will do a manual evaluation on whether we consider this model to be adequate in performing the task.

3.3 Evaluation tool

It is often useful to separate the automated processing of a Machine Learning approach in different layers, both as a way of separating the mental understanding of different concepts and as a necessity of being able to sequentially use different pre-existing tools that support different information formats. In addition, having some automatic evaluation functionality is better than doing it manually, saving intermediate processing steps saves time, and so

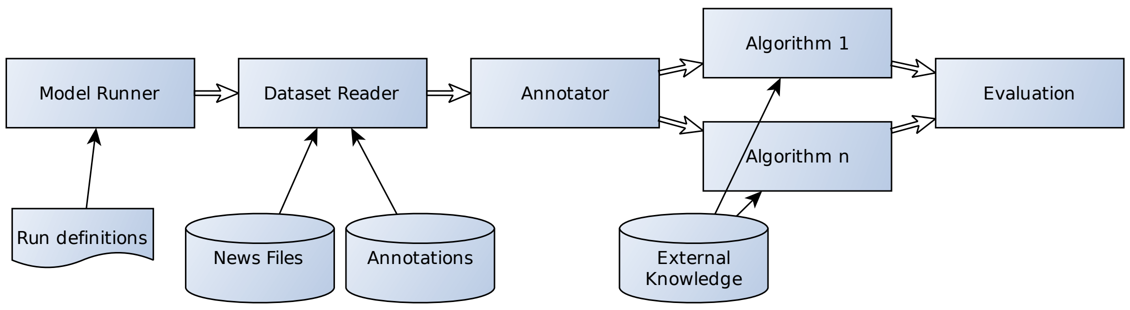


Figure 3.1: Logical Architecture of the Application

on. However, tools to manage such functionality pipelines did not conform to our needs of constant iteration of experimental code to evaluate the possibility space.

We decided to create a model running tool, uncreatively named *modelrunner*, which essentially has the job of reading a file detailing what tasks (both pre-processing tasks and model tasks) to run and the parameters they need, running them in order of their dependencies, and calculating and comparing metrics to determine which one has a better solution for a problem.

In order to obtain an answer to our problem, we needed to separate our process in the way shown in Figure 3.1. We start with a layer to read the definitions of what we want to do in a particular process, a layer that can read the files containing the news articles and the annotations, which then applies the annotations to the text from the articles. We then need a layer that further annotates the text with syntactical information (see 2.2.2). We can then finally add the layers for different models to be evaluated, with them having their own evaluation function that is later aggregated and summarized. All these layers need to be configurable with both meta-parameters and features we would like them to have, so that we can try different variants and obtain the best tuning possible. It would also have been possible to split this process further, like adding features outside the model, but we decided we did not need to. In order to avoid repeating pre-processing steps, since the available data does not tend to change while models are being designed, we have implemented functionality that allows for saving calculations at the end of a step, as well as a way to cache intermediate calculations (not shown, as they would overburden the image). Users of the former cannot always rely on the generic implementation, since some existing modules have their own functionality that can be leveraged for other uses, but they can store enough information to load them later.

Chapter 4

Implementation

This work was made with the use of the *Python* programming language, constantly kept up-to-date (up to version 3.7.3 as of this document). This language was chosen due to ease of use and its vast amount of available libraries for Data Mining and Machine Learning, while allowing quick iteration of ideas. While the language design necessitates some sacrifice of speed [CLM05], the necessary libraries where our processing will take the most time are written as high-performance modules, and the rest of the code can potentially be written fast enough to offset the execution time, as the models take a long time to learn the data.

Since we wanted to continually iterate and evaluate different variations of parameters and features, we decided to properly encapsulate the division of labor in several program modules in order to focus on one part of the work at a time. This program, *modelrunner*, is capable of reading a configuration file detailing the wanted pre-processing steps and model implementation parameters (including lists of alternatives). The result of any step is automatically saved and loaded, to avoid re-running processed tasks, and this behavior is automatically ignored if any task parameter changes.

The configuration file is an extended .INI file, and is expected to specify needed parameters, such as what data files to read, what tags to annotate, what previous task has the input and how to evaluate and compare the models.

4.1 External program dependencies

To evaluate our chosen model, there is a very good open source library, *CRFsuite* [Oka07], for labeling sequential data that implements various variants of the algorithm, which is fast (according to the author's measurements) while being easy and flexible to use. We decided to use it along with its *Python* bindings, present in the module *python-crfsuite*[1]. It should be noted that having a single entity type in the model might have decreased the capability of

[1] version 0.9.6, available at https://github.com/scrapinghub/python-crfsuite

the solution, as the model cannot be as discriminative with its transitions.

The functionalities of creating the necessary features are sometimes included together with entity recognition and even further analysis in the same tool (a complete program or a program library) with a coherent architecture. Here, we are only interested in special forms of entity recognition, and we have left the preparation tasks to existing software, namely to *Freeling*[23]. This is an open source library (with a sample program) that does many of the tasks mentioned in this thesis, as well as others. It is maintained by an academic research group and has support for many languages, making it suitable for our case. *NLTK*[4] is another known library for *NLP* tasks that is commonly used, but it has less functionality for the Portuguese language, so we will only used the reference parts of the library.

We decided that we should also experiment with some other syntactic markers that our chosen pre-processing tool provides, since they came for free (other than further processing time cost for training models). These include: degree (superlative/evaluative), gender (male/female/neutral), is plural, is possessive verb, number (plural/singular/invariant), (first/second/third) person, pronoun casing, mood (indicative/subjunctive/imperative/past participle/gerund/infinitive), (verb) tense and punctuation type[5].

For the implementation of cache functionality to store intermediate results, we relied on the capabilities of the *SQLite* relational database[6], a freely available (released as public domain) database system that was created to fill the role of a small, quick and safe data storage programming library.

4.2 Helper modules

These represent the part of the work that does not depend on the rest, meaning that its constituent modules could be used for work unrelated to our tool.

4.2 corpus

While numeric data has an easy and representation in a matrix format, which has been optimized in several libraries for both sparse and dense data, there is no defined best way to store text. *NLTK* has been a commonly used library that uses (and popularized) tuples (immutable lists) of words along with related information (POS and NER tags, stems,

[2]version 4.1, available at http://nlp.lsi.upc.edu/freeling/node/1

[3]at the time of the start of this work, this was the most generic tool available with modules for the Portuguese language, with good reference score. At the time of this writing, spaCy should be evaluated for new works, as it is a very high quality tool that is very easy to work with. However, it seems to not have as much detailed work for the language yet.

[4]version 3.4.5, available at https://www.nltk.org/

[5]https://freeling-user-manual.readthedocs.io/en/v4.1/tagsets/tagset-pt/

[6]https://www.sqlite.org/index.html, we use the version included with our *Python* version

lemmas, etc.). However, this representation leaves the work of remembering the structure of texts and what auxiliary information is available to the programmer, which presents maintenance difficulty because of the need transmit unstructured information.

We decided to create a small model that would store our texts in a structured way that would be more easily verified for structural correctness and crashed our program faster in case of errors, so they could be corrected with less time in analyses. Hopefully, the usage of classes for text representation will allow users to more easily switch the internals in case of future need while keeping existing work functional.

We created independent classes for sentences, documents and corpus, as tokens and tags can be represented as native strings. Sentences is the most interesting class, and where we deal with storing tokens and tags separately. We have added functionality to do some error checking for tag consistency and to extract the information in several formats. The latter two classes are mostly a simple storage data structure for the lower levels, with the methods being aggregators of their information. The exception is the addition of functionality to split the sentences randomly in groups, as that is usually needed to feed machine learning models.

There was some effort spent in optimizing speed and memory with use of the slot[7] and generator[8] functionality in *Python*. Despite the usefulness of this module, we will recommend a future different approach in Chapter 6.

4.2 custom_freeling

Freeling is a C++ library for natural language processing that, as mentioned in 2.2, allows for the creation of linguistic features that can supply models with more information. At the beginning of the execution of this work, there was no *Python* interface to the library, so we needed to create our own way of interacting with it. Since we did not intend to create an optimal solution, we decide to use the sample executable provided with the library. While we intended the implementation to be generic, the tool relies on definitions of linguistic parameters for its internal models and we only implemented the reference to our studied language.

Our class starts by defining the necessary files that are provided for the usage of the Portuguese language, as well as defining friendlier names for the returned features, since the optimizations afforded by shorter representations were not needed. The main functionality

[7]normal class variables access is slightly inefficient in the language, as they involve a dictionary lookup. We can get a small speedup by declaring the ones we will use as *slots*

[8]a generator is a function that does not compute the final result when called, but one value at a time to be consumed. This can allow savings of time (or, at least, displacement) and memory (as all the inputs and all the final values, especially of a group, might not be using memory at the same time). However, these savings can be tricky to obtain if there is not an easy way to express the whole data pipeline.

is then the transformation of both unstructured text (for tokenization) and of our text format (for tokenized text) into free-flowing text that can be read by an external program, the storing of invocation parameters in a temporary file[9], and the reverse transformation back into our text format. Since the same program is used for both tokenization and other pre-processing, we avoid issues with different definitions of tokenizing rules. The later transformation depends on the task, as they return different information, and is made a little trickier by the fact that we want to re-encode the returned information into a user-friendly format. The class then provides several methods that allow for different invocations according to the annotations required by the user.

Freeling has since started to include a direct and optimized interface to the underlying library, and, at the same time, OS updates have led our system to take a significant amount of type in calling the program. While taking advantage of the new interface would be advantageous, it is also a very low level interface that is cumbersome to use. We believe a hybrid approach that provides our more direct function calls with a quicker and more efficient interface would be best, especially in an application aimed at multi-language support.

4.2 gazetteer

A gazetteer is a dictionary of entity information that is expected to be useful for tasks working with entities. For our level of modeling, the gazetteer solely stores a list of words that appear on entities, potentially lower-cased and/or filtered by frequency, and then allows checking whether the word is present or not.

4.2 sqlite_cache

Most of the output of the program is stored in separate files that can be analyzed by the end-user or reused later in the processing (and some libraries use their own saving functionality, which would make it redundant). However, it was important to cut down the processing time of our external tagging system as we programmed and tested new features, since testing for correctness was easier to verify with a small amount of input that did not need to be pre-processed again. This module allowed us to fairly transparently store the needed results.

4.2 text_helpers

The small text_helpers model is intended to help in the creation of features that can be used by models.

[9]the program, analyze, can receive command line arguments for some functionality, but not all. In practice, there is not much reason to use two different methods of doing so.

4.2.5.1 unigrams

A unigram is a single item from a sequence. This could be a token or a character, and we are working on a token level. This implementation in particular is used as a text file reader to fill the gazetteer and to gather some information about word distribution, namely count, frequency and rank or a particular token.

4.2.5.2 word_featurizer

The bulk of our work in entity detection itself was placed in this module, which is responsible for creating the necessary features for each token present in text of the provided news articles, which were described in 3.1.1. These features are created for the current surrounding tokens, within a user-defined window, as surrounding words are expected to have an impact on models.

4.3 modelrunner

The modelrunner is the main module of the programming part of this work, and the one that gets invoked first when the program runs. It is tasked with the job of creating appropriate versions of requested models and executing them, while informing the user on the progress of the processing.

Specifically, it starts by reading a configuration file (that can be changed on the command line) that specifies what the program should do and sets up the requested logging. It then goes through each section in the configuration and starts the requested tasks, requesting the creation of variations for each parameter with a list and passing along parameterized input and parameters. Parameters can be specified on a global section of the file, which can be useful to define logging and evaluation for all tasks.

Each model returns a list of metrics (as many as they want), which are then gathered, summarized and compared. Optionally, the results can be saved to a .CSV file.

4.4 tasks

We decided to call the acts of both pre-processing and model-processing together as the generic name tasks, as they have some similar behavior. It is the job of this module to calculate variants of requested tasks in case of need, as well as their creation. Task creation involves some sanity checks on parameters and filling in default values in case of need.

BaseTask implements the basic functionality of a task, which is to do the initial interpretation of configuration (for universal parameters) and sets up logging as caching as requested.

4.4 processors

Processors is a module that groups the various processing tasks that do not produce results, such as obtaining or pre-processing data.

4.4.1.1 baseprocessor

The baseprocessor module provides generic functionality needed by different processing tasks of the same type (i.e. all text processors need common functionality, but XML readers do things differently than HDFS readers). It currently separates functionality in BaseProcessor, BaseTextProcessor, CorpusReader and XMLReader.

BaseProcessor Implements the basic functionality, in this case saving and loading of the models and their results;

BaseTextProcessor Built on top of BaseProcessor, it currently implement no functionality on its own, but is intended to eventually serve as a basis of text-based tasks;

CorpusReader Built on top of BaseTextProcessor, it represents a class that reads files that contain or are part of a dataset. It implements a function do to tokenizing with custom_freeling, as this is a task that is expected to be the first pre-processing task;

XMLReader Built on top of CorpusReader, this class transforms a set of XML files composing a dataset into a new tokenized dataset for use with algorithms. It receives a list of file patterns from user configuration, transforms it into a list of files and extracts the text found in user-defined elements. The file list can optionally be filtered, as in our dataset, not all files may be annotated. The extracted text is then passed through normalization steps intended to eliminate invalid UTF characters and extraneous whitespace (like the one that exists in XHTML files, since it would be irrelevant for a web browser), and finally transformed into our corpus data structures.

4.4.1.2 EmbrapaReader

This module is built on top of XMLReader and is used to annotate our particular dataset, and its additions are all related to importing the annotations. These are read from an XLS file and attributed to the appropriate news article by the number present in the first column. After having the contents of the annotations and the contents of the news articles, the former are applied to the latter by searching for the same sequence of words after removing accents, as these are not present in the annotations. They can be applied in either the IOB or the BILOU format.

4.4.1.3 FreelingProcessor

The FreelingProcessor is another class that is built on top of BaseTextProcessor, and it serves as an interface between the generic custom_freeling and the feature creation needs of our entity detection algorithm. It can be used to add the syntactic and semantic features described in that model to the dataset in order to influence algorithmic performance. Since the processing takes a long time, it also uses the caching functionality to avoid reprocessing the same information.

4.4 models

The models module groups our implementation of algorithms to facilitate de creation of trained models from provided data.

4.4.2.1 BaseModel

The BaseModel provides the basic infrastructure on which algorithms rely. It sets up evaluation parameters according to the configuration, saves and loads model versions, aggregates performance metrics and initiates model training as many times as required by the evaluation method. It currently supports cross-validation and holdout training methods, as well as a “none” type for evaluating external tools. Metric aggregation is currently used for the cross-validation methods, where it is important to obtain the means of the fold results.

4.4.2.2 BaseTextModel

This class builds on top of BaseModel, adding functionality that is necessary for training text models. It implements splitting functionality necessary for cross-validation (both by sentences and by documents), stop word detection (via the *NLTK Python* package), cluster embeddings creation (via the word2vec module of the *gensim Python* package and the *clusters* module of the *sklearn Python* package) and entity detection task evaluation. The later is done as described in Section §2.4, with only a perfect match between the annotation and the prediction being counted as correct (both in the words classified by the method as well as the class). Other than Precision, Recall and F1, tag accuracy is also calculated.

4.4.2.3 CrfModel

The CrfModel is a BaseTextModel that can train and apply a CRF model according to user specifications. It uses the word_featurizer module to create the necessary features for both train and test sets, as well as any other text where we might want to apply a trained model to. In order to do the training and classification, it iterates though the input's sentences, extracts the labels and creates the list of categorical features that can be fed to the CRF algorithm, which is instanced from the *crfsuite Python* package. It can also pass optional hyper-parameters to the package to alter its training process.

4.4.2.4 ExternalModel, FlModel and SpacyModel

These are classes that were created to facilitate the comparison between externally created models as a baseline and our own. ExternalModel can interface with a command line tool by passing and receiving text, converting from the corpus format as input and to an annotation class list as output. FlModel and SpacyModel (for custom_freeling and *spaCy*) use the direct *Python* interface to communicate, but the latter must still do a conversion by replacing the tokenizer.

4.4.2.5 GeoTaggingModel

The GeoTaggingModel implements our geotagging method, as described in Section §3.2. It does this by creating instances of entity detection classes: the best version of our trained CRF models, a *spaCy* model and a *Polyglot* model. It then classifies all sentences by using the 3 taggers, accepting a positive classification of any of them as an entity detection. A list of entities with default senses is then extracted from these classifications and the highest level administrative entity that aggregates the highest population is selected as the most important of the text.

Chapter 5

Results

5.1 Entity Detection

In this chapter, we present the results of our testing methodology, defined in 3.1.3, as applied to our dataset, obtained with running the experiments with our tool. Unfortunately, we did not run a significance test to establish the complete relevance of the results.

On table 5.1, we can observe the results of our baseline approaches to solving the problem. What we call Baseline is an approach that consists in detecting entities by detecting the presence of words that exist in the list of the types of entities that we expect to find (i.e. the regions explained in the dataset description). The *Polyglot*, *spaCy* and *Freeling* approaches consist in the application of our normalized text to existing NER tools of the same name[1], which are then filtered to consist only of locations. As these tools are much wider in application, we then filter the words present in the results by words present in the same entity list. As can be expected (since these are well developed and specialized, but generic, tools), most of these approaches do a good job of finding all the references (high recall), but they also detect too much that was not annotated. An example is the detection of “Universidade de São Paulo” (University of São Paulo), which was not annotated, but is incorrectly found (as “São Paulo”, after filtering). The results of *Freeling* further suggest that its NER functionality for Portuguese can be improved.

On Table 5.2, we can observe the results of training several variations of our model, as detailed before, and on Figure 5.1 we can see the evolution of the F1 metric as we train new models with more feature groups. We can see that a window size of 4 confuses the algorithm for both annotation types and worsens every result significantly. We believe that this variant might create too many features for the algorithm to converge adequately with the default parameters. Along with that, we can see that in almost every case the BILOU

[1] https://pypi.org/project/polyglot/ (version 16.7.4), https://spacy.io/ (version 2.0.16), http://nlp.lsi.upc.edu/freeling/ (version 4.1)

Model	Precision	Recall	F1
Baseline	16.00	88.51	27.10
Freeling	37.06	59.89	45.79
Polyglot	24.05	92.28	38.16
spaCy	36.03	77.21	**49.13**

Table 5.1: Baseline Models

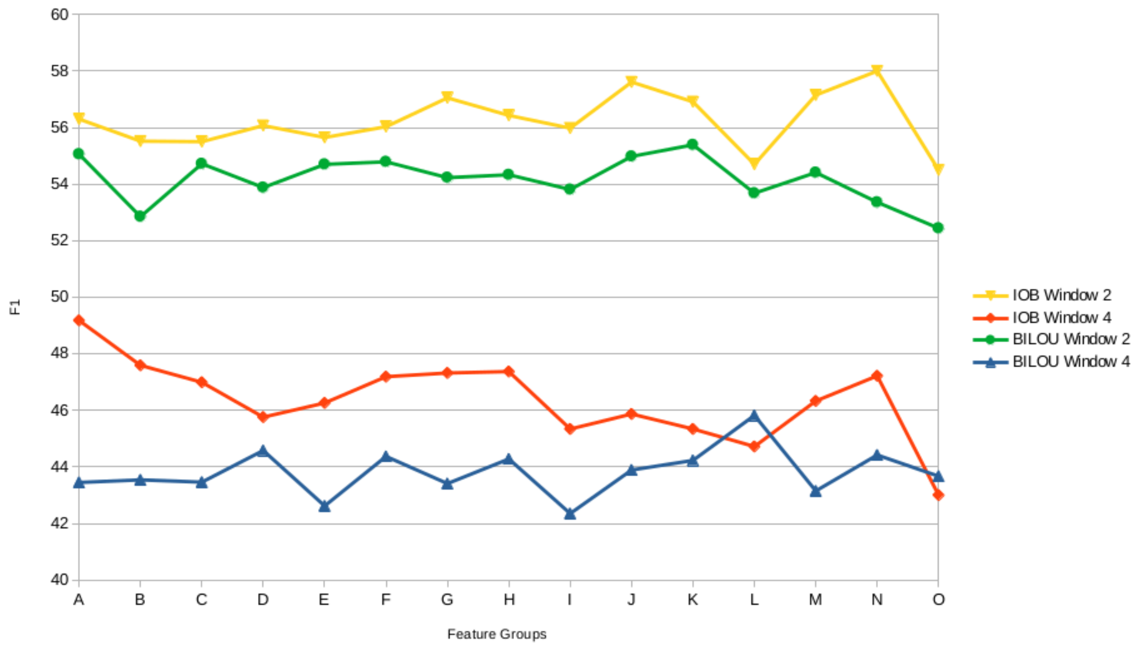


Figure 5.1: Chart of F1 results obtained by training variations of the model and applying it to our dataset. The letter representing the model can be referenced on table 5.2

encoding performs worse. Our interpretation of this is that using extra classes for a small dataset may lead to few samples of some types of entity annotations. This can usually be improved by creating similar synthetic variations, but it would require its own study on this dataset.

As to the analysis of the feature groups, the analysis is more complicated, as the result changes of adding features are not similar across our size and encoding variants. Looking solely at the best set of results, there is a gradual improvement of results as features are introduced, with some exceptions. We believe the problem can potentially be an excess of features as well. Regardless, the improved results measured in the best models are encouraging for the appropriateness of our approach.

In order to try an obtain some further improvement on these results, we decided to try some small variants of the best models. Firstly, as in [HFS+17], there is no particular reason to expect the size of our word vector clusters or the dimensionality of the word vectors themselves to be close to optimal, which would be too expensive to calculate. Nevertheless, we also experimented varying both these values in the best models, first by obtaining other trained vectors from the same origin and then creating a different number of clusters.

Model	Precision	Recall	F1	Chart Reference
IOBw2	74.91±2.63	45.68±6.96	56.30±5.28	A
IOBw2+SYN_TAGS	75.19±3.84	44.41±5.48	55.52±3.79	B
IOBw2+INDICATIVE	75.63±2.96	44.40±6.89	55.50±5.43	C
IOBw2+INDICATIVE+SYN_TAGS	73.82±3.96	46.24±9.69	56.07±7.01	D
IOBw2+INDICATIVE+TRANSFORMATIVE	76.54±2.17	44.26±7.52	55.65±5.86	E
IOBw2+INDICATIVE+TRANSFORMATIVE+SYN_TAGS	74.71±2.32	45.70±9.02	56.03±6.31	F
IOBw2+INDICATIVE+TRANSFORMATIVE+RANKING	76.84±2.83	46.27±8.90	57.05±7.41	G
IOBw2+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS	75.23±3.45	46.03±8.76	56.43±5.81	H
IOBw2+INDICATIVE+TRANSFORMATIVE+RANKING+W2V	75.96±3.61	44.86±6.17	55.98±4.02	I
IOBw2+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+W2V	72.96±4.75	48.02±6.43	**57.61**±4.56	J
IOBw2+INDICATIVE+TRANSFORMATIVE+GAZETTEER	75.91±2.51	45.79±2.51	56.91±3.96	K
IOBw2+INDICATIVE+TRANSFORMATIVE+GAZETTEER+SYN_TAGS	74.73±1.45	44.35±11.07	54.70±9.81	L
IOBw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.	74.80±2.89	46.96±9.12	**57.15**±7.20	M
IOBw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS	76.06±2.08	47.75±9.23	**58.00**±6.55	N
IOBw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+W2V	74.61±2.22	43.69±8.82	54.50±6.71	O
IOBw4	72.14±3.91	37.49±5.82	49.18±5.79	A
IOBw4+SYN_TAGS	72.41±4.03	35.88±7.16	47.59±6.11	B
IOBw4+INDICATIVE	72.36±5.97	35.37±7.83	46.99±7.55	C
IOBw4+INDICATIVE+SYN_TAGS	72.51±5.56	34.13±9.21	45.76±9.94	D
IOBw4+INDICATIVE+TRANSFORMATIVE	70.91±3.33	34.85±7.81	46.26±7.28	E
IOBw4+INDICATIVE+TRANSFORMATIVE+SYN_TAGS	72.38±3.45	35.45±5.99	47.19±5.43	F
IOBw4+INDICATIVE+TRANSFORMATIVE+RANKING	73.13±2.98	35.56±7.85	47.32±7.27	G
IOBw4+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS	73.15±4.73	35.65±7.52	47.37±6.60	H
IOBw4+INDICATIVE+TRANSFORMATIVE+RANKING+W2V	72.22±5.23	33.70±8.41	45.34±8.49	I
IOBw4+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+W2V	72.64±3.26	34.42±9.63	45.87±8.40	J
IOBw4+INDICATIVE+TRANSFORMATIVE+GAZETTEER	73.11±3.27	33.04±8.80	44.72±9.66	K
IOBw4+INDICATIVE+TRANSFORMATIVE+GAZETTEER+SYN_TAGS	72.34±5.43	34.49±7.98	46.33±8.38	L
IOBw4+INDICATIVE+TRANSFORMATIVE+RANKING+GAZETTEER	73.39±3.82	35.75±10.04	47.22±9.37	M
IOBw4+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS	74.31±4.02	35.53±7.06	47.54±6.74	N
IOBw4+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+W2V	71.05±3.98	32.18±12.07	43.00±11.95	O

(a) Results with IOB Encoding for windows of size 2 and 4

Model	Precision	Recall	F1	Chart Reference
BILOUw2	75.81±2.57	43.44±4.38	55.06±3.33	A
BILOUw2+SYN_TAGS	74.78±5.66	41.73±8.66	52.85±7.28	B
BILOUw2+INDICATIVE	75.72±1.81	43.31±6.82	54.72±5.69	C
BILOUw2+INDICATIVE+SYN_TAGS	75.75±1.96	42.21±6.28	53.88±4.76	D
BILOUw2+INDICATIVE+TRANSFORMATIVE	76.72±2.42	42.87±5.96	54.70±4.64	E
BILOUw2+INDICATIVE+TRANSFORMATIVE+SYN_TAGS	75.75±2.80	43.38±6.60	54.79±5.17	F
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANKING	74.33±5.45	43.06±7.84	54.23±6.93	G
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS	75.55±2.07	43.18±9.28	54.33±7.94	H
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANKING+W2V	75.45±2.77	42.33±7.95	53.81±6.80	I
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+W2V	75.04±2.39	43.83±7.38	54.98±6.36	J
BILOUw2+INDICATIVE+TRANSFORMATIVE+GAZETTEER	77.22±2.88	43.39±4.35	**55.39**±3.63	K
BILOUw2+INDICATIVE+TRANSFORMATIVE+GAZETTEER+SYN_TAGS	75.07±2.94	42.34±7.30	53.68±6.07	L
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZETTEER	75.65±2.47	43.16±8.41	54.41±6.91	M
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS	75.07±2.60	41.88±7.41	53.36±6.06	N
BILOUw2+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+W2V	73.71±3.22	41.18±7.18	52.44±6.32	O
BILOUw4	74.37±3.33	30.82±3.18	43.45±3.07	A
BILOUw4+SYN_TAGS	74.19±4.10	31.50±7.44	43.54±6.01	B
BILOUw4+INDICATIVE	74.37±4.98	31.23±8.25	43.46±8.67	C
BILOUw4+INDICATIVE+SYN_TAGS	74.75±3.59	32.10±6.39	44.58±6.28	D
BILOUw4+INDICATIVE+TRANSFORMATIVE	74.85±2.87	30.42±7.93	42.62±7.90	E
BILOUw4+INDICATIVE+TRANSFORMATIVE+SYN_TAGS	73.95±4.79	31.91±5.39	44.37±5.52	F
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANKING	73.27±4.17	30.92±2.74	43.41±2.96	G
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS	74.40±5.62	31.97±7.35	44.28±8.08	H
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANKING+W2V	74.85±3.93	30.13±7.65	42.35±7.88	I
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+W2V	72.73±4.36	33.08±8.51	43.89±8.97	J
BILOUw4+INDICATIVE+TRANSFORMATIVE+GAZETTEER	75.99±3.35	31.92±7.99	44.23±8.17	K
BILOUw4+INDICATIVE+TRANSFORMATIVE+GAZETTEER+SYN_TAGS	74.84±5.20	33.39±5.93	45.82±5.85	L
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANK.+GAZETTEER	73.10±3.65	31.07±7.27	43.15±6.48	M
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS	74.88±3.03	32.03±6.49	44.42±6.23	N
BILOUw4+INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+W2V	73.22±3.61	31.31±5.48	43.67±5.86	O

(b) Results with BILOU Encoding for windows of size 2 and 4

Table 5.2: Results of the model with default algorithm parameters. In the model names, IOB and BILOU refer the training encoding and w2/w4 to the window size. Chart reference indicates the corresponding line in the chart

Model	**Precision**	**Recall**	**F1**
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v600_500	75.01±2.67	45.87±7.81	56.40±5.35
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v600_1000	73.88±2.77	45.23±9.85	55.46±8.42
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v600_1500	73.79±3.22	45.16±6.34	55.70±4.93
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v300_500	75.06±2.59	46.99±8.42	**57.37**±6.51
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v300_1000	73.62±3.14	44.88±9.06	55.24±7.35
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v300_1500	74.30±2.39	46.35±10.57	56.32±8.54
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v600_500	73.60±3.59	42.99±9.16	53.57±7.55
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v600_1000	74.24±2.47	44.15±8.50	54.88±6.55
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v600_1500	75.00±2.50	47.09±6.72	**57.56**±5.46
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v300_500	75.27±4.49	43.98±9.01	54.81±7.59
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v300_1000	74.69±2.33	43.75±6.87	54.83±5.59
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v300_1500	75.48±1.51	45.36±7.21	56.37±5.98
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v600_500	75.20±2.00	45.41±4.47	56.49±3.58
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v600_1000	74.75±2.15	45.26±4.81	56.24±3.73
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v600_1500	74.11±2.43	45.50±8.78	55.95±7.01
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v300_500	74.92±2.58	46.61±7.72	57.02±5.55
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v300_1000	74.46±3.20	45.98±8.46	56.24±5.95
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v300_1500	74.81±3.94	48.27±6.91	**58.28**±5.24

(a) Results of using different word vector clusters on the best models

Model	Precision	Recall	F1
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+W2V	75.48±2.91	46.58±8.36	57.03±6.27
INDICATIVE+TRANSFORMATIVE+RANKING+GAZETTEER	75.81±2.72	46.42±3.57	57.47±2.66
INDICATIVE+TRANSFORMATIVE+RANKING+GAZETTEER+SYN_TAGS	75.70±2.37	48.21±6.74	**58.53**±5.04

(b) Results of trying different annotations on the best models

Model	Precision	Recall	F1
INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v300_500	74.90±3.22	45.03±9.01	55.65±6.94
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+w2v600_1500	75.88±3.09	45.28±4.61	56.72±3.14
INDICATIVE+TRANSFORMATIVE+RANK.+GAZET.+SYN_TAGS+w2v300_1500	74.37±3.96	46.40±6.89	**57.15**±6.21

(c) Results of joining previous optimizations

Table 5.3: Result optimization experiments

The results can be seen in Table 5.3a, where the w2vX_Y label represents Y clusters of word2vec vectors with X dimensionality. We can clearly see that having different clusters can have a big difference. This was even more clear when we noticed model "INDICATIVE+TRANSFORMATIVE+RANKING+SYN_TAGS+w2v600_1000" was supposed to be the same model as the "unoptimized" version (reference J). The explanation of the difference can be explained by noting that we created new clusters with a non-deterministic (by default) implementation of the KMeans algorithm - using the same input does not guarantee the same output, even when provided in the same order.

At the same time, we also decided to do another experiment to improve results. We noticed that sometimes locations such as "Bom Jesus do Araguaia" can be referred to in the appropriate context as simply "Bom Jesus". Although this is a change to the dataset, we thought it would be interesting to evaluate the best models with this change, and we show the results in Table 5.3b. Results suggest this is a potential improvement, although it might increase ambiguity when disambiguating entities.

Finally, we decided to join both experiments. As Table 5.3c shows, this did not work so well. Our theory is that the clusters are probably not reliable enough and might confuse the training algorithm.

Another possibility, as hinted before, is the variation of training hyper-parameters. As there are many parameters that can be changed and they have a wide range of possible values, we experimented only a few combinations on the full model, with the results that can be

Model	**Precision**	**Recall**	**F1**
feature.minfreq=6.0|c1=0.1|c2=0.2|epsilon=1e-05	72.44±3.93	49.51±7.35	58.40±4.99
feature.minfreq=6.0|c1=0.1|c2=0.2|epsilon=0.5e-05	72.38±1.71	50.93±8.44	59.39±6.15
feature.minfreq=6.0|c1=0.1|c2=0.3|epsilon=1e-05	71.47±3.65	51.37±12.06	58.66±9.18
feature.minfreq=6.0|c1=0.1|c2=0.3|epsilon=0.5e-05	73.12±3.95	51.51±8.25	59.91±5.46
feature.minfreq=6.0|c1=0.1|c2=0.5|epsilon=1e-05	72.29±3.48	50.06±5.63	58.89±3.65
feature.minfreq=6.0|c1=0.1|c2=0.5|epsilon=0.5e-05	72.32±3.41	49.76±8.85	58.35±5.88
feature.minfreq=6.0|c1=0.2|c2=0.2|epsilon=1e-05	72.40±2.17	53.08±5.36	**61.02**±3.06
feature.minfreq=6.0|c1=0.2|c2=0.2|epsilon=0.5e-05	71.77±2.43	51.74±10.04	59.45±5.87
feature.minfreq=6.0|c1=0.2|c2=0.3|epsilon=1e-05	72.90±2.52	50.35±7.35	59.10±5.23
feature.minfreq=6.0|c1=0.2|c2=0.3|epsilon=0.5e-05	71.20±3.96	52.06±5.81	59.79±3.28
feature.minfreq=6.0|c1=0.2|c2=0.5|epsilon=1e-05	72.89±2.86	47.71±10.25	56.83±8.87
feature.minfreq=6.0|c1=0.2|c2=0.5|epsilon=0.5e-05	72.35±2.79	50.67±7.44	59.28±5.27
feature.minfreq=4.0|c1=0.1|c2=0.2|epsilon=1e-05	71.71±2.69	51.71±7.16	59.82±5.00
feature.minfreq=4.0|c1=0.1|c2=0.2|epsilon=0.5e-05	72.35±4.07	51.47±7.81	59.64±5.13
feature.minfreq=4.0|c1=0.1|c2=0.3|epsilon=1e-05	73.34±4.35	50.68±6.62	59.51±3.87
feature.minfreq=4.0|c1=0.1|c2=0.3|epsilon=0.5e-05	72.50±3.59	52.01±10.96	59.72±7.00
feature.minfreq=4.0|c1=0.1|c2=0.5|epsilon=1e-05	73.75±1.78	50.37±7.00	59.51±4.58
feature.minfreq=4.0|c1=0.1|c2=0.5|epsilon=0.5e-05	73.24±3.90	51.62±8.24	60.13±5.65
feature.minfreq=4.0|c1=0.2|c2=0.2|epsilon=1e-05	71.59±3.97	52.70±8.49	60.23±5.61
feature.minfreq=4.0|c1=0.2|c2=0.2|epsilon=0.5e-05	73.16±2.90	52.14±6.96	60.49±4.41
feature.minfreq=4.0|c1=0.2|c2=0.3|epsilon=1e-05	73.44±2.68	50.69±7.40	59.60±4.72
feature.minfreq=4.0|c1=0.2|c2=0.3|epsilon=0.5e-05	71.77±2.58	49.98±3.23	58.83±2.14
feature.minfreq=4.0|c1=0.2|c2=0.5|epsilon=1e-05	72.57±3.61	51.22±6.92	59.65±4.13
feature.minfreq=4.0|c1=0.2|c2=0.5|epsilon=0.5e-05	72.76±4.41	48.96±8.69	57.96±6.02

Table 5.4: Results of training the best model with different hyper-parameters

examined on Table 5.4. The changes were chosen by trying coarse changes on samples of the dataset, which is not an optimal procedure and does not provide the best results, but the latter suffered consistent effects from the changes.

For reference, the altered (independent) parameters are:

feature.minfreq (default 0) ignore features that are present less times than this value;

c1 (default 0) coefficient for L1 regularization optimization training algorithm. Regularization is a way of generalizing the model, with increased L1 lowering the value of low-weight features and lowered L2 value decreasing the error penalty [Bis06];

c2 (default 1) coefficient for L2 regularization;

epsilon (default 1e-5) the minimum error difference for the algorithm to consider it has appropriately trained.

We can conclude that there is definitely room for experimentation here as well, although perfect optimization is costly. We believe that the notable improvements with significant dimensionality reduction hyper-parameters indicate that the algorithm may be having trouble dealing with special cases in training, whether due to the small size of the dataset or to problematic annotations.

5.2 Geotagging

Since we have no provided annotations for this task, we must do a manual evaluation of some of the news articles, with results presented in Table 5.5 and the news articles in the appendix. The model uses the best model of Table **??** as well as *spaCy* and *Polyglot* to detect entities.

The first annotated article presents us with a conundrum of what relevance really means: it refers to the importance to the state of a new factory in one of its constituent municipalities (which is considered as a Micro Region by the algorithm) and describes the details of the factory. However, with the common definition of location relevance as being the one that best encloses the more important locations, we can agree with our algorithms conclusion that selecting the state is the correct choice.

Line 4 of the results table presents an interesting case. “Ribeirão Preto” and “Campinas” are both Meso Regions of both states of “São Paulo” and “Minas Gerais”, but of the rest, only “Lorena” is in “São Paulo”. In truth, the news is about neither, but about a new book on the state of the industry and its technology that was edited in “Ribeirão Preto” in “São Paulo”. The algorithm fails.

In news 7, the algorithm selects the highest region, the Meso Region, which does not encompass the Micro Regions mentioned. This would be solved if the State entity had been found, but it signals that the approach may be too simplistic.

In line 8, we have many locations in the state, with a rogue municipality from the state of “Pernambuco”. In reality, this is an abbreviation of the Meso Region that does not affect the result.

Line 9 has an example of a news item that has no solution. It is a news item about production cost increases in several states, which makes sense.

Finally, news article 10 has 4 locations attributed as referring to the state of “São Paulo”, 1 to the state of “Bahia”, 2 in “Pernambuco”, 1 in “Rio Grande do Sul” and 1 in “Piauí”. The news item is referring to the Macro Region level, which we are not modeling. We think that attributing it to São Paulo is appropriate. It should be mentioned that without domain knowledge, we can’t tell whether “Ribeirão” or “Pedra” are abbreviations, or to what physical location “São Martinho (Pradópolis)” or “Pedra (Serrana)” refer to, as the parts in parentheses are not hierarchically related at these levels.

States	**Meso Regions**	**Micro Regions**	**Municipalities**	**Result**
Goiás	Boa Vista	Bom Jesus, Quirinópolis	Esmeralda	State Goiás
Rio de Janeiro				State Rio De Janeiro
Mato Grosso	Dourados		Laguna	State Mato Grosso
	Ribeirão Preto, Campinas, Itajubá	Viçosa	Lorena	State Minas Gerais
Mato Groso do Sul	Campo Grande		Rio Brilhante, Nova Alvorada	State Mato Grosso do Sul
São Paulo	Ribeirão Preto		Ipiranga, Ribeirão	State São Paulo
	Ribeirão Preto	Piracicaba, Catanduva, Jaú		Meso Region Ribeirão Preto
São Paulo	Ribeirão Preto	Piracicaba, Catanduva, Araçatuba,...	Ribeirão	State São Paulo
Rondônia, São Paulo, Minas Gerais, Paraná, Acre, ...				
São Paulo	Ribeirão Preto	Franca, Barretos	Ribeirão, Central, Cravinhos, São Martinho, Pradópolis, Pedra, ...	State São Paulo

Table 5.5: Qualitative evaluation of Geotagging

Chapter 6

Conclusions and Improvements

The main goals of this thesis were reasonably achieved, as we were able to achieve better results than naive solutions for detecting specific locations. We also had promising results with a simple algorithm in finding the most relevant location for a news item, results which are helped by the limits in found locations. We think we helped prove that it is possible to use machine learning approaches to solving these types of problems in the Portuguese language, even with a small dataset. We cannot do comparisons as we do not know of a tool that does exactly the same thing.

After establishing a baseline modeling with CRFs, we managed to show that there are further significant improvements that can be made with careful study of features and hyper-parameters. Finally, we created a tool that can hopefully be helpful in doing similar machine learning work in the future.

One thing that could have been done was the integration of other datasets in the training process, which should have helped achieve better results. Since these are hard to acquire and can be provided with very different types of annotation, it would also be interesting to explore how to create synthetic variations of existing text in order to achieve better algorithmic training.

Other than that, further feature experimentation could have also been done. The Geotagging process could also have had better attention, even if NER is a complicated task by itself.

6.1 Possible Improvements

There are many improvements that can be done to all levels of this work. With regards to the entity detection model, it should prove fruitful to fragment some of our groups to see if we have an excess of features. It should also be worthwhile to find a way to provide better word embedding clusters, perhaps even including hierarchal features similar to Brown clusters.

On the Geotagging problem, a good development would be to include geographic, population and/or economic information into the discrimination and aggregation process, which are features that have proven to help in getting good results. In addition, we feel the need of introducing a scoring method that can override spurious higher level mentions.

For both of the tasks, it will be important to re-annotate the news articles, as their are currently very confusing and not always accurate. Having annotations at the word level would help in getting much more performing models. Another reason to re-annotate would be to keep the dataset up to date by taking into account recent administrative changes.

As to the modeling tool, there is much that can be improved, as it is somewhat built specifically to accomplish this task. The most important change, however, is to better leverage existing *Python* models in 3 ways:

- replace our command line calls to the *Freeling* sample program with direct calls through the now included module, which would allow for both faster results and better functionality;
- replace our corpus structure internals with an implementation based on *spaCy*, or maybe even use it internally. It is much better designed for efficiency and functionality as well;
- it would make sense to make the tool provide an easy way to create *sklearn*-based modules, as it is a very popular library to do machine learning with and several other tools provide compatible interfaces. On top of that, we could avoid having our own implementation of metrics and evaluation methods, which are potentially for buggy due to having less developers.

Bibliography

[AMC09] Ivo Anastácio, Bruno Martins, and Pável Calado. A Comparison of Different Approaches for Assigning Geographic Scopes to Documents. *INForum*, 2006:285–296, 2009.

[BDVJ03] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A Neural Probabilistic Language Model. *The Journal of Machine Learning Research*, 3:1137–1155, 2003.

[Bis06] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[BJM83] Lalit R Bahl, Frederick Jelinek, and Robert L Mercer. A maximum likelihood approach to continuous speech recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (2):179–190, 1983.

[BPLA95] Frédéric Bimbot, Roberto Pieraccini, Esther Levin, and Bishnu Atal. Variable-length sequence modeling: multigrams. *IEEE Signal Processing Letters*, 2(6):111–113, 1995.

[CLM05] Xing Cai, Hans Petter Langtangen, and Halvard Moe. On the performance of the python programming language for serial and parallel scientific computations. *Scientific Programming*, 13(1):31–56, 2005.

[CWK+11] Ronan Collobert, Jason Weston, Pavel Kuksa, León Bottou, Michael Karlen, and Koray Kavukcuoglu. Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research*, 12:2461–2505, 2011.

[dMDS+14] Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. Universal Stanford dependencies: A cross-linguistic typology. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 4585–4592, Reykjavik, Iceland, May 2014. European Languages Resources Association (ELRA).

[dP98] J.A. du Preez. Efficient training of high-order hidden markov models using first-order representations. *Computer Speech & Language*, 12(1):23 – 39, 1998.

[GDL$^+$13] Abhishek Gattani, AnHai Doan, Digvijay S. Lamba, Nikesh Garera, Mitul Tiwari, Xiaoyong Chai, Sanjib Das, Sri Subramaniam, Anand Rajaraman, and Venky Harinarayan. Entity extraction, linking, classification, and tagging for social media. *Proceedings of the VLDB Endowment*, 6(11):1126–1137, aug 2013.

[GS95] Ralph Grishman and Beth Sundheim. Design of the muc-6 evaluation. In *Proceedings of the 6th Conference on Message Understanding*, MUC6 '95, pages 1–11, Stroudsburg, PA, USA, 1995. Association for Computational Linguistics.

[HFS$^+$17] Nathan Hartmann, Erick Fonseca, Christopher Shulby, Marcos Treviso, Jessica Rodrigues, and Sandra Aluisio. Portuguese word embeddings: Evaluating on word analogies and natural language tasks, 2017.

[HKP11] Jiawei Han, Micheline Kamber, and Jian Pei. Data mining concepts and techniques third edition. *The Morgan Kaufmann Series in Data Management Systems*, pages 83–124, 2011.

[HNP05] Andreas Hotho, Andreas Nürnberger, and Gerhard Paaß. A brief survey of text mining. In *Ldv Forum*, volume 20, pages 19–62. Citeseer, 2005.

[KN93] Reinhard Kneser and Hermann Ney. Forming word classes by statistical clustering for statistical language modelling. In *Contributions to Quantitative Linguistics*, pages 221–226. Springer, 1993.

[KR18] Ehsan Kamalloo and Davood Rafiei. A coherent unsupervised model for toponym resolution. In *Proceedings of the 2018 World Wide Web Conference*, pages 1287–1296. International World Wide Web Conferences Steering Committee, 2018.

[KS63] Sheldon Klein and Robert F. Simmons. A computational approach to grammatical coding of english words. *J. ACM*, 10(3):334–347, July 1963.

[Lei08] Jochen L Leidner. *Toponym resolution in text: Annotation, evaluation and applications of spatial grounding of place names*. Universal-Publishers, 2008.

[Lei16] Jochen L Leidner. Georeferencing: From texts to maps. *International Encyclopedia of Geography: People, the Earth, Environment and Technology: People, the Earth, Environment and Technology*, pages 1–10, 2016.

[Li17] Junyi Li. From discourse structure to text specificity: Studies of coherence preferences. 2017.

[LM14] Qv Le and Tomas Mikolov. Distributed Representations of Sentences and Documents. *International Conference on Machine Learning - ICML 2014*, 32:1188–1196, 2014.

[LMP01] John Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, 8(June):282–289, 2001.

[LN13] Annie Louis and Ani Nenkova. A corpus of science journalism for analyzing writing quality. *Dialogue & Discourse*, 4(2):87–117, 2013.

[LSNL02] Huifeng Li, Rohini K Srihari, Cheng Niu, and Wei Li. Location normalization for information extraction. In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics, 2002.

[Mar95] Mitchell P Marcus. Text Chunking using Transformation-Based Learning 1 Introduction. *third Workshop on Very Large Corpora*, pages 82–94, 1995.

[Mcc03] Andrew Mccallum. Early Results for Named Entity Recognition with Conditional Random Fields, Feature Induction and Web-Enhanced Lexicons. *Conference on Computational Natural Language Learning.*, pages 1–4, 2003.

[MCCD13a] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. *Nips*, pages 1–9, 2013.

[MCCD13b] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. *arXiv:1301.3781*, jan 2013.

[MFP00] Andrew McCallum, Dayne Freitag, and Fernando C. N. Pereira. Maximum entropy markov models for information extraction and segmentation. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 591–598, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[MM17] Fernando Melo and Bruno Martins. Automated geocoding of textual documents: A survey of current approaches. *Transactions in GIS*, 21(1):3–38, 2017.

[MMBSV09] Bruno Martins, H Manguinhas, José Borbinha, and Willington Libardo Siabato Vaca. A geo-temporal information extraction service for processing descriptive metadata in digital libraries. 2009.

[MMG99] Andrei Mikheev, Marc Moens, and Claire Grover. Named Entity recognition without gazetteers. In *Proceedings of the ninth conference on European chapter of the Association for Computational Linguistics* -, page 1, Morristown, NJ, USA, 1999. Association for Computational Linguistics.

[MMS99] Christopher D Manning, Christopher D Manning, and Hinrich Schütze. *Foundations of statistical natural language processing*. MIT press, 1999.

[MOE99] Ruslan Mitkov, Constantin Orasan, and Richard Evans. The importance of annotated corpora for nlp: the cases of anaphora resolution and clause splitting, 1999.

[NS07] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Lingvisticae Investigationes*, 30(1):3–26, 2007.

[Oka07] Naoaki Okazaki. CRFsuite: a fast implementation of Conditional Random Fields (CRFs), 2007.

[PDM11] Slav Petrov, Dipanjan Das, and Ryan T. McDonald. A universal part-of-speech tagset. *CoRR*, abs/1104.2086, 2011.

[Pia14] Steven T. Piantadosi. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21(5):1112–1130, oct 2014.

[Pow11] David Martin Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. 2011.

[PPTT13] Le-Hong Phuong, Xuan-Hieu Phan, and Tran The Trung. On the effect of the label bias problem in part-of-speech tagging. 11 2013.

[RBB03] Erik Rauch, Michael Bukatin, and Kenneth Baker. A confidence-based framework for disambiguating geographic terms. In *Proceedings of the HLT-NAACL 2003 Workshop on Analysis of Geographic References - Volume 1*, HLT-NAACL-GEOREF '03, pages 50–54, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.

[RR09] Lev Ratinov and Dan Roth. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado, June 2009. Association for Computational Linguistics.

[S$^+$01] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[SCP19] Marco Antonio Sobrevilla Cabezudo and Thiago Pardo. Natural language generation: Recently learned lessons, directions for semantic representation-based approaches, and the case of Brazilian Portuguese language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 81–88, Florence, Italy, July 2019. Association for Computational Linguistics.

[SdM05] Erik F. Tjong Kim Sang and Fien de Meulder. Introduction to the CoNLL-2003 Shared Task: Language-Independent Named Entity Recognition. *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2005.

[Sie15] Scharolta Katharina Sien. Adapting word2vec to Named Entity Recognition. *Proceedings of the 20th Nordic Conference of Computational Linguistics (NODALIDA 2015)*, (Nodalida):239–243, 2015.

[SO06] Andrew Smith and Miles Osborne. Using gazetteers in discriminative information extraction. *Proceedings of the Tenth Conference on Computational Natural Language Learning, CoNLL-X*, (June):133–140, 2006.

[SS14] Vikram Singh and Balwinder Saini. An effective tokenization algorithm for information retrieval systems. 2014.

[TK15] Maria Touri and Nelya Koteyko. Using corpus linguistic software in the extraction of news frames: towards a dynamic process of frame analysis in journalistic texts. *International Journal of Social Research Methodology*, 18(6):601–616, 2015.

[VHPPG07] Gerhard Van Huyssteen, Martin Puttkammer, Suléne Pilon, and Handré Groenewald. Using machine learning to annotate data for nlp tasks semi-automatically. 09 2007.

[Vit09] A. J. Viterbi. Viterbi algorithm. *Scholarpedia*, 4(1):6246, 2009. revision #91930.

[WFHP16] Ian H Witten, Eibe Frank, Mark A Hall, and Christopher J Pal. *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, 2016.

[You77] Robert Young. Text understanding: a survey. *American Journal of Computational Linguistics*, 1977.

[Zem08] Daniel Zeman. Reusable tagset conversion using tagset drivers. In *LREC 2008*, 2008.

Appendix

News samples for qualitative evaluation of the Geotagging task

News Article 1

O governador Marconi Perillo e o presidente da Goiás Fomento, José Taveira Rocha, assinaram ontem novo contrato de financiamento de incentivos fiscais com a Usina Boa Vista S/A. O empréstimo foi aprovado pela Secretaria de Indústria e Comércio (SIC), no valor de R$ 4,2 bilhões, e tem por objetivo ampliar a implementação da unidade industrial no município de Quirinópolis, no Sudoeste do Estado.

A solenidade foi realizada no Salão Verde do Palácio das Esmeraldas e contou com a participação do presidente do grupo, Fábio Venturelli, e dos secretários da Indústria e Comércio, Alexandre Baldy; Casa Civil, Vilmar Rocha (DEM); e Meio Ambiente, Leonardo Vilela (PSDB).

O financiamento é destinado à Nova Fronteira Bioenergia S/A, formada pel a sociedade entre o Grupo São Martinho e a Petrobras Biocombustíveis, que atua na produção de etanol na região Centro-Oeste. As atuações em Goiás se concentram no projeto Greenfield SMBJ Agroindustrial S/A, no município de Bom Jesus, e na Usina Boa Vista, em Quirinópolis.

O projeto prevê investimentos totais de R$ 985 milhões para a produção de álcool e energia na unidade e vai gerar 3,4 mil empregos diretos e mais de 10 mil indiretos. Com o contrato assinado, a empresa vai contar com o financiamento, através de incentivos fiscais, na ordem de R$ 4,2 bilhões para os próximos dez anos.

Presidente do Sindicato da Indústria de Fabricação de Etanol do Estado de Goiás (Sifaeg), André Luiz Rocha aponta o investimento como um fator de reconhecimento da modernização econômica do Estado. A presença e consolidação da Usina Boa Vista em Goiás são bastante representativas. É uma empresa moderna, produtora de energia de forma sustentável, e a parceria com a Nova Fronteira Bioenergia demonstra a preocupação com o meio ambiente e a visão do Estado como novo centro produtor de energia, enfatiza.

Outro aspecto lembrado por André Luiz é a cidade de Quirinópolis como ponta de lança dos investimentos no setor sucroalcooleiro em Goiás. A cidade do Sudoeste Goiano foi uma das que mais cresceram no Estado e o projeto é visto com bons olhos, por gerar renda e empregos. Com a geração de emprego e renda, o ciclo produtivo da cidade é ampliado. Setor hoteleiro e a revenda de tratores, que já vem crescendo, podem crescer ainda mais, afirma.

Capacidade

A joint-venture prevê ainda investimentos de R$ 985 milhões em atividades agrícolas e industriais, de forma que a Usina Boa Vista alcance a capacidade de moagem de 7 milhões de toneladas de cana por ano. Atualmente, o empreendimento possui 1.600 empregados e o processamento atinge 2 milhões de toneladas de cana, colhidas 100% de forma mecanizada e sem queima.

Com suas caldeiras de alta pressão, a usina também gera energia elétrica e comercializa a produção excedente.

Com o novo contrato, o Programa Produzir atinge a marca de R$ 80 milhões em investimentos contratados, para projetos de médias e grandes empresas presentes em 103 municípios goianos.

News Article 2

O atual Código Florestal deve e precisa se modificado, sim!.

O atual texto permite o desmatamento;o projeto, "Relatório Rebelo" o proíbe por 5 anos e depois disso, vincula a abertura de novas áreas ao resultado do zoneamento ecológico-econômico, zoneamento este que deverá ser executado TÉCNICAMENTE, sem ingerencia politica ou ideológica.

O problema legislativo não reside na Amazônia, onde há mais de 73% da área pública e destinada ao meio ambiente (UCs, terras indígenas, devolutas etc...), mas onde o progresso do país se fez, onde ele cresceu. Essa é a parte do território onde o projeto se concentra.Estudos, inclusive da WWF, mostram o efeito econômico da recomposição de áreas exploradas quanto a legislação permitida. Isso significaria algo em torno de 2/3 do PIB anual do país (1,5 trilhões de reais). Recompostas,tais áreas não produzirão emprego e renda, reduzindo o PIB anual em 76 bilhões e a arrecadação tributária entre 22 a 26 bilhões de reais por ano, eternamente. Nos próximos 30 anos a redução será entre 650 a 700 bilhões. Os municípios abaixo de 10 mil habitantes entrarão em colapso nesse contexto,à troco de nada!.

A agropecuária é ponta de cadeia. A venda de serviços e produtos para ela produz riqueza aos demais setores dentro da matriz insumo-produto, (W.leontief), numa proporção de quase R$2,00 para cada R$1,00 da agropecuária.

Os empregos nas montadoras, petroquímicas, transportes, serviços etc...serão os primeiros a sofrer redução.

Por isso, a mudança não se destina a abrandar a abertura de novas áreas, mas evitar o colapso do que já existe desde o século passado. Não sairá de graça. Haverá compensações de área dentro do mesmo bioma. O Estado poderá fazer as reservas coletivas para a criação de grandes áreas nativas.

O espaço das ciências foi garantido no projeto; o zoneamento ecológico-econômico, repito, feito técnicamente, sem viéz politico e/ou ideológico,o plano de bacias, dentre outros estudos, são expressamente citados.

Mesmo na pequena propriedade, onde há vegetação nativa, ela não poderá ser suprimida. Nesses casos, serão emitidas CRA cotas de reserva ambiental, para compensação das exigências das propriedades maiores.

Pela atual legislação, apenas 25% do país poderia ser aproveitado para a agropecuária, aí incluso as terras degradadas. Busca-se apenas manter esse percentual em 40%, ou seja, mesmo com as mudanças, a legislação proibiria a abertura de aproximadamente 60% do território nacional. Onde falta o bom senso: na alteração ou na manutenção?.

Ao longo do ano de 2010 foram realizadas 68 audiências públicas e inúmeras reuniões e palestras em todo o país onde se discutiu com a sociedade, os anacronismos embutidos no inaplicável Código Florestal, código este que se tornou ao longo dos anos, após inúmera "modificações impostas", sem nenhuma discussão, um verdadeiro "frankstein" legal, para não falar, arredondando, nas 16.000 leis, portarias, normas e outros quejandos "legais" que tentam "proteger" o Meio Ambiente em nosso país. Basta usar um mínimo de bom senso e observar qual foi o resultado prático disto até o presente momento.

É justamente por isto que as necessárias reformas do Código Florestal devem serem feitas para que se tenha um ordenamento legal que possa ser posto em prática,o que não acontece com a atual formatação do código.

Quanto ao desastre ocorrido na região serrana do Estado do Rio de Janeiro e a colocação em manchete na FSP "Revisão do Código Florestal pode legalizar área de risco e ampliar chance de tragédia", nada mais longe da realidade, quem se der ao trabalho de ler o relatório do Dep. Aldo Rebelo, não achará em nenhum lugar esta colocação, há sim, colocações voltadas as ZONAS RURAIS utilizadas para a produção agropecuária, basta notar que cada cidade tem sua própria legislação de ocupação do solo urbano. Basta cumpri-lá e no caso de seu não cumprimento, que se responsabilize as autoridades incompetentes!,aliás,observo que nunca vi isto ocorrer.

Sabemos que o que ocorre em zonas urbanas, é bem diferente do que acontece em zonas rurais, basta ver onde acontecem estes desastres.

É importante notar que autoridades varias,ambientalistas em geral,daqui e do exterior, criam as maiores dificuldades para com o trabalho do produtor rural brasileiro quanto ao plantio técnico de culturas, por exemplo, o café, macieiras, parreirais em morros, o arroz em várzeas, como é feito à séculos em todo o mundo, porem o criminoso "plantio de seres humanos" nestes locais de sabido alto risco, à vista de todos, ano apos ano, catástrofe apos catástrofe, logo serão esquecidas por estas mesmas instituições, até que se repitam novamente. Hoje e cada vez mais, dado ao crescimento populacional, o grande problema ambiental do planeta está nas cidades e em seus entornos e não no campo.

Insisto que o Código Florestal precisa sim ser adaptado à realidade atual para que possa ser viabilizado e aplicado, e esta discussão deve ter um viéz técnico e não emocional/político/ideológico como tentam fazer os contrários, "misturando" fatos totalmente distintos,como ocorreu agora.

Que se restabeleça a verdade e o bom senso, é o que pedimos.

Atenciosamente

José Augusto Baldassari

News Article 3

Durante cerimônia realizada nesta segunda-feira (24/01) no Sesi de Dourados, o presidente da Fiems, Sérgio Longen, entregou ao Cetec Senai Dourados a microdestilaria didática de etanol para atender a demanda de formação de profissionais do setor sucroenergético em Mato Grosso do Sul. Estamos buscando fomentar as potencialidades de cada região e, no caso de Dourados, o destaque vai para o setor sucroenergético. Por isso, entregamos essa planta didática para facilitar na formação de trabalhadores, disse. Sérgio Longen acrescenta que o Sistema Fiems precisa construir ações que consolidem as indústrias que estão instaladas no Estado e também para atrair aquelas que têm interesse de vir para Mato Grosso do Sul. O município de Dourados passa por um momento muito importante e por isso precisamos unir esforços para que ele se desenvolva no ritmo do resto do Estado, destacou, informando que a microdestilaria didática de etanol reproduz, em escala duas mil vezes menor que uma usina normal, todo o processo industrial de processamento e destilaria de etanol. Para o presidente da Biosul (Associação dos Produtores de Bioenergia de Mato Grosso do Sul), Roberto Hollanda Filho, o Brasil é hoje uma potência mundial na produção e no consumo de etanol e bioenergia e o Estado está aproveitando esse momento. Já temos 21 usinas em operação e mais três que devem entrar em operação ainda neste ano. Hoje temos área de 400 mil hectares plantadas com cana-de-açúcar e mais 6 milhões de hectares que podem ser destinadas para essa cultura. Os mercados interno e externo estão em crescimento e por isso é importante essa iniciativa do Sistema Fiems, analisou. Roberto Hollanda ressalta ainda que para este ano o setor sucroenergético vai gerar mais 3 mil empregos diretos e essa mão-de-obra precisa ser qualificada, chegando na hora certa essa nova ferramenta do Senai. Muito obrigado ao presidente da Fiems, Sérgio Longen, por esse presente que é a microdestilaria didática de etanol. A Biosul só pode prometer que vai retribuir esse investimento com a geração de mais empregos e também mais impostos para o Estado, reforçou. O empresário Werner

Semmelroth, da Usina Laguna, também pontuou que a falta de mão-de-obra qualificada é o principal gargalo do setor em Mato Grosso do Sul. A necessidade do setor por trabalhadores capacitados é muito grande, principalmente aqui em Dourados, que hoje é o centro de uma região que concentra 60% do setor sucroalcooleiro do Estado, fixando-se como pólo na área de tecnologia. A implantação dessa microdestilaria didática de etanol vem para somar a tudo que o setor disponibiliza hoje na questão de qualificação de profissional, disse, prevendo que a planta didática vai atender a demanda da área industrial e que será uma excelente ferramenta para o setor. Qualificação O diretor-regional do Senai, Jaime Verruck, informa que a microdestilaria será utilizada em cursos e treinamentos nas áreas de qualificação profissional, habilitação profissional, aperfeiçoamento profissional e aprendizagem industrial, além de contemplar os STT (Serviços Técnicos e Tecnológicos) do setor sucroenergético. Ele ressalta que os profissionais formados pelo Cetec Senai Dourados terão um diferencial competitivo para que possam atuar no mercado de trabalho, agregando valor aos cursos e serviços já oferecidos hoje pela instituição em todo o Estado no setor sucroenergético. O gerente do Cetec Senai Dourados, Gilberto Schaedler, destaca que, entre as facilidades, a microdestilaria de etanol dispensará as visitas técnicas nas usinas. O aluno aprenderá todo o processo produtivo do etanol da mesma forma que em uma usina convencional, mas em menor proporção, comentou, completando que a planta também poderá ser deslocada até o local de trabalho para treinamentos dos funcionários das usinas instaladas no Estado. Segundo do o instrutor dos cursos de química e açúcar e álcool do Cetec Senai Dourados, Clauber Rodrigues, a microdestilaria será inserida nos cursos disponibilizados pela entidade em Dourados, além de ser utilizada na realização de pequenos serviços técnicos solicitados pelas usinas instaladas na região. Com relação ao funcionamento da planta didática, ele explica que o processo de produção do etanol começa na moenda, primeiro componente da destilaria que recebe a cana-de-açúcar para extrair o caldo. Depois de extraído, o líquido vai para a decantação para ser separado o bagaço da cana-de-açúcar e, na sequência, o caldo é exportado até outro tanque para ser aquecido a 110º C, quando são eliminadas as bactérias e os germes presentes no caldo, detalhou o instrutor, completando que, após esse aquecimento, a matéria-prima do álcool é resfriada para dar continuidade ao processo e, a partir daí, outra bomba pega o caldo e despeja para o processo de fermentação. Quase ao fim da produção, o líquido recebe leveduras e é fermentado, quando se acrescenta o caldo da cana puro para que o conjunto seja fermentado por quatro horas. Logo em seguida ele é centrifugado e tem as leveduras separadas, sendo que nessa etapa o líquido já se transformou em vinho e entra em uma última torre para ser novamente aquecido - dessa vez a 70º C - para se obter o álcool. Todo o processo dura entre 10 e 12 horas e a microdestilaria pode produzir até 10 litros de álcool por hora, informou Clauber Rodrigues.

News Article 4

Cientistas de todo o mundo estão em busca de uma tecnologia para a produção em escala industrial do etanol derivado da celulose da cana-de-açúcar. Atualmente, só se pode fabricar etanol a partir da sacarose, que corresponde a um terço da biomassa da planta. O etanol celulósico permitiria aproveitar os outros dois terços, aumentando a produtividade sem alterar a área plantada. Atingir esse objetivo, no entanto, não é tarefa trivial. O conhecimento adquirido até agora na busca do etanol celulósico foi consolidado no livro Routes to Cellulosic Ethanol, que acaba de ser lançado pela editora norte-americana Springer. Reunindo textos de alguns dos principais especialistas do mundo em etanol celulósico, o livro foi editado por Marcos Buckeridge, professor do Departamento de Botânica do Instituto de Biociências (IB) da Universidade de São Paulo (USP) e membro da coordenação do Programa Fapesp de Pesquisa em Bioenergia (BIOEN), e Gustavo Goldman, professor da Faculdade de Ciências Farmacêuticas da USP, em Ribeirão Preto. O livro é um produto do 1° Simpósio sobre Etanol Celulósico, realizado em setembro de 2008 com o objetivo de definir estratégias para obtenção do etanol celulósico por meios genéticos e bioquímicos. A obra também agregou o conhecimento gerado posteriormente, com o lançamento do Instituto Nacional de Ciência e Tecnologia (INCT) do Bioetanol, coordenado por Buckeridge. "O foco principal do livro é a ciência básica originária do Bioen e do INCT, que buscam construir a plataforma de conhecimento necessária para que sejam desenvolvidas as tecnologias de etanol celulósico. No livro, procuramos consolidar o que foi conquistado até agora", disse o cientista à Agência Fapesp. Segundo Buckeridge que também é diretor científico do Laboratório Nacional de Ciência e Tecnologia do Bioetanol (CTBE), em Campinas (SP) , a obra reúne textos de cientistas japoneses, norte-americanos, mas, sobretudo, de brasileiros. "O Brasil é o maior produtor mundial de etanol de cana-de-açúcar e pioneiro no desenvolvimento de tecnologias para o uso de etanol para geração de energia. O país tem décadas de estudos acumulados sobre o tema", disse. A obra é dividida em três partes: "Bioenergia", "Parede celular das plantas, enzimas e metabolismos" e "Genética da parede celular das plantas". A primeira parte contextualiza o tema da bioenergia e as outras duas tratam de aspectos ligados a uma questão crítica para o desenvolvimento do etanol celulósico: dominar o conhecimento sobre a degradação da parede celular da planta. "Se pudermos entender a síntese das paredes celulares, os geneticistas e biólogos moleculares poderão desenvolver plantas com polissacarídeos atualmente inexistentes na cana-de-açúcar, mas que serão introduzidos a fim de facilitar a hidrólise do etanol celulósico", afirmou. BIOENERGIA NO BRASIL E NO MUNDO Na primeira parte, a obra apresenta uma visão geral sobre as rotas para o etanol celulósico. O físico José Goldemberg discute a questão da energia no mundo, a fim de mostrar a posição brasileira no contexto internacional da bioenergia. Em seguida, Buckeridge, Wanderley dos Santos e Edgardo Gómez ambos pesquisadores do CTBE discutem a produção de etanol celulósico pela via bioquímica. Luiz Nogueira, da Universidade Federal de Itajubá (Unifei), Joaquim Seabra, do CTBE, e Isaías Macedo, da Universidade Estadual de Campinas (Unicamp), finalizam a parte introdutória discutindo a produção de etanol celulósico pela via da gaseificação. ALTERNATIVAS DE DEGRADAÇÃO A segunda parte reúne os capítulos que tratam da degradação da parede celular por diferentes métodos. Takahisa Hayashi, da Universidade Agrícola de Tóquio (Japão), e Rumi Kaida, da Universidade de Kioto (Japão), escreveram sobre seus experimentos com madeira na Ásia. "Hayashi é um dos grandes pesquisadores da área no mundo. Nesses experimentos, eles utilizaram transgênicos para alterar a parede celular", disse Buckeridge. O contraponto brasileiro à experiência asiática é feito por um grupo da Escola de Engenharia de Lorena (EEL), da USP, liderado por Adriane Milagres. "Eles tratam especificamente do ataque enzimático em materiais lignocelulósicos, revelando a experiência que acumularam ao longo dos anos", contou. Um grupo da Universidade de Brasília (UnB), liderado por Edivaldo Filho, faz uma revisão sobre a atividade enzimática na degradação da parede celular. Em seguida, o artigo de Igor Polikarpov e Viviane Serpa ambos do Instituto de Física de São Carlos (IFSC), da USP, trata da estrutura das proteínas. "Polikarpov é um dos melhores especialistas brasileiros em cristalização e enzimas, entre elas as hidrolases", disse Buckeridge. Maria de Lourdes Polizeli lidera o grupo da Faculdade de Filosofia, Ciências e Letras de Ribeirão Preto (FF-CLRP), da USP, que contribuiu com um artigo sobre hidrolases de microrganismos usadas para a degradação da parede celular. "Ela tem uma das maiores coleções do mundo de enzimas e fungos bem caracterizados", disse Buckeridge. Richard Ward, também da FFCLRP-USP, é o autor de um artigo sobre engenharia de celulase para sacarificação de biomassa. "É um trabalho que eu considero espetacular: ele constrói proteínas que não existem na natureza e conta a história dessa engenharia, que já aplicou em plantas e microrganismos", disse o membro da coordenação do Bioen-Fapesp. A segunda parte é encerrada com o artigo de Goldman sobre o melhoramento genético do uso da xilose um açúcar de cinco carbonos pelas leveduras. "É um tema muito importante, envolvendo um problema crítico para o processo do etanol celulósico. No artigo, Goldman conta seus experimentos, retrata o que já existe e indica o que ainda pode ser feito", disse. BRASIL E ESTADOS UNIDOS A terceira parte tem o objetivo de traçar um paralelo entre o que está sendo feito no Brasil e nos Estados Unidos. "Em um dos capítulos, um grupo de autores norte-americanos mostra como eles estão explorando a maravilhosa diversidade do milho com uma abordagem genética muito avançada. O objetivo é chegar a uma planta ideal para a bioenergia", contou Buckeridge. Eric Lam, da Universidade de New Jersey (Estados Unidos), liderou o grupo que discutiu o desenvolvimento de "plantas inteligentes" para a produção de bioetanol. "Trata-se de uma área na qual eu e Hayashi também atuamos: buscar plantas capazes de degradar sua própria parede celular", disse. O grupo liderado por Marcelo Loureiro, da Universidade Federal de Viçosa (UFV), foi responsável por um artigo relacionado à seleção de plantas para uma conversão mais eficiente em bioetanol. "Eles coordenam um trabalho muito importante de busca de variedades que tenham modificações na parede celular para, a partir daí, produzir etanol celulósico", contou Buckeridge. O livro é encerrado com o artigo de Michel Lawton e Hemalatha Saidasan ambos da Universidade Rutgers (Estados Unidos) sobre genômica e parede celular. "Trata-se de um modelo de um musgo que tem um genoma muito pequeno e que pode servir para estudar o etanol celulósico", disse.

News Article 5

A alta no preço da cana-de-açúcar estimulou a expansão da cultura em Mato Grosso do Sul. Algumas usinas já se programam para dobrar a produção em 2011. Na Fazenda Santo Antônio, em Rio Brilhante, município que fica há 170 quilômetros de Campo Grande, todos os dias o agricultor José Francisco de Souza Neto dá uma volta pela lavoura de cana para ver como está o desenvolvimento da planta. A propriedade fica a 4,5 quilômetros de uma usina sucroalcooleira. Há quatro anos a cana de açúcar ocupa a área que antes era de soja. O produtor está tão contente com a produção que quer dobrar a área plantada. "Dois fatores básicos me fizeram sair da soja e entrar na cana: um é a distância da usina, é muito próximo à usina, que se torna economicamente muito mais viável; o segundo fator é que a cana tem mais resistência a poucas chuvas. Ela tem mais tempo para se recuperar, justificou Neto. Nas usinas é período de entressafra. A moagem da cana só retorna a partir de abril. Em uma delas, que fica em Nova Alvorada do Sul, há 150 quilômetros de Campo Grande, o que sai da chaminé é o vapor da caldeira. A indústria faz a manutenção dos equipamentos e termina as obras de ampliação da área de produção. Dois dos quatro novos tanques, com capacidade para 20 milhões de litros de etanol cada, estão quase prontos. São investimentos necessários para atingir as metas. No campo, a mesma máquina que faz a colheita, agora retira as mudas, que são plantadas em uma área recém aberta. O clima ajudando, a terra com boa produtividade e o mercado aberto fizeram a empresa olha mais longe. Esse é só o inicio do plantio da expansão da área de cana em Mato Grosso do Sul. Só a usina em Nova Alvorada vai plantar nada menos esse ano que mais 18 mil hectares de cana. A terra já está arrendada. Tudo foi feito para atingir uma produção em 2011 de quatro milhões de cana moída. Nossa expansão é muito acelerada. Moemos dois milhões em 2010. Vamos moer quatro milhões em 2011 e seis milhões em 2012, disse o gerente agrícola Luís Borges. Um levantamento feito pela Companhia Nacional de Abastecimento mostra que Mato Grosso do Sul é o terceiro estado no país que mais aumentou a produção nessa safra. Foram 47,4%. Passou de 23,3 milhões de toneladas para 34,3 milhões de toneladas. Em 2010, sete novas usinas foram construídas. No total hoje são 21 em funcionamento. O diretor da Associação dos Produtores de Bioenergia no estado, Isaias Bernardini, explicou que a demanda crescente tem incentivado a chegada de novas indústrias. Tanto devido ao consumo de etanol, com um mercado interno muito vigoroso, como o consumo de açúcar tem aumentado principalmente a nível de mercado internacional, cobrindo queda de produção de outros países tradicionais produtores, disse Bernardini. Este ano, três novas usinas devem se instalar em Mato Grosso do Sul.

News Article 6

O aumento de postos de bandeira branca em Ribeirão Preto tem contribuído para reduzir o preço médio do combustível na cidade. De acordo com a ANP (Agência Nacional de Petróleo), a cidade tem 180 postos, sendo 84 de bandeira branca. Em média, a diferença de preços entre postos com marca e os independentes é em torno de R$ 0,10. Ontem, postos com distribuidoras como Ipiranga, BR e Esso, por exemplo, comercializavam o etanol a R$ 1,699. Já nos estabelecimentos sem marca, sai entre R$ 1,55 a R$ 1,60. Segundo o presidente regional do Sincopetro (Sindicato do Comércio Varejista de Derivados de Petróleo do Estado de São Paulo), Oswaldo Manaia, a diferença no valor ocorre porque os postos sem bandeiras não pagam o valor das marcas. "O posto de gasolina paga para divulgar uma determinada bandeira. Já o valor do p roduto é o mesmo", disse. Em 2010, segundo Manaia, pelo menos quatro novos postos de bandeira branca (sem marca, não ligado a grandes distribuidoras) abriram suas portas em Ribeirão. Segundo a gerente do posto JPS (sem bandeira), no bairro Campos Elíseos, a ausência de uma marca não é entrave ao consumo. "A clientela é fixa porque o produto tem qualidade", disse.

News Article 7

É grande a diferença de custos da produção de cana-de-açúcar conforme a região e o sistema adotado pelos agricultores. A variação chega a 41%, de R$ 32,08 a tonelada até R$ 45,42, segundo extenso trabalho do Instituto de Economia Agrícola, ligado à Secretaria de Agricultura de São Paulo. A pesquisa abrangeu todas as regiões do Estado, desde as mais tradicionais, como Ribeirão Preto e Piracicaba, até as mais novas, como Catanduva e Jaú. Foram detalhados os variados sistemas de produção que surgiram para conciliar as maiores exigências ambientais e trabalhistas com a necessidade de redução de custos.

News Article 8

Em um trabalho que levou quase dois anos, o Instituto de Economia Agrícola (IEA), órgão ligado à Secretaria de Agricultura de São Paulo, levantou os custos de produção de fornecedores de cana-de-açúcar de todas as regiões de São Paulo, desde as mais tradicionais, como Ribeirão Preto e Piracicaba, até as mais novas em cana, como Catanduva e Jaú. O trabalho, considerado o mais abrangente já feito nessa área, verificou que as diferenças entre os custos chegam a até 41%, variando de R$ 32,08 por tonelada a R$ 45,42. Mais do que um ineditismo geográfico, o trabalho conseguiu incorporar os diversos sistemas de produção que se multiplicaram nos últimos anos como reação a uma demanda contraditória no setor: aliar o aumento das exigências ambientais e trabalhistas com a necessidade de reduzir custos, diz a pesquisadora do IEA, Marli Dias Mascarenhas de Oliveira. No estu do, um calhamaço de mais de 90 páginas, foram encontrados nada menos do que sete formatos de produção diferentes, que contemplam os já tradicionais, nos quais o fornecedor planta a cana e a usina colhe, e os mais inovadores, como o de compartilhamento de máquinas e mão de obra. A pesquisa calculou custos de fornecedores em todas as seis principais regiões produtoras de cana-de-açúcar do Estado: Araçatuba, Assis, Catanduva, Jaú, Piracicaba e Ribeirão Preto. No centro da questão "custo" estão a colheita e o transporte, também conhecidos como CCT (corte, carregamento e transporte). São esses itens que pesam no bolso do produtor e chegam a responder por mais de 50% do custo total da gramínea. Mas o IEA botou tudo na ponta do lápis e chegou à conclusão de que essa participação do CCT também varia bastante e fica entre 52,8% e 66% dependendo da região e do sistema de produção adotado. Entre as combinações mais competitivas está a de colheita feita pelos produtores, mas no si stema de condomínio, diz Marli. Por meio deles, os fornecedores compram máquinas e equipamentos para colher e organizam a logística para atender à área de todos os fornecedores-condôminos. Na região de Jaú, por exemplo, a colheita mecânica feita nesse regime resultou em um custo total de produção de cana de R$ 32,75. Na mesma localidade, o valor sobe para R$ 39,30 se o serviço de colheita for prestado pela usina. Esse tipo de organização, explica Marli, está mais presente nas regiões novas em cana, como Jaú e Catanduva. Nelas, a relação entre usina e fornecedor é mais recente, por isso, o comportamento é diferente, afirma a pesquisadora. Outra questão que fomentou nessa região a expansão dos condomínios foi a pouca concorrência entre usinas. Com menos indústrias para disputar a cana, os valores cobrados por serviços, como colheita mecanizada, subiram, explica Kátia Nachiluk, também autora da pesquisa. De acordo com o estudo, em Jaú, o custo da cana ao pr odutor sobe para R$ 39,30 com colheita mecânica feita pela usina e para R$ 40,01 quando esta presta serviço de colheita manual. Ambos os sistemas estão entre os valores mais altos encontrados no Estado. Já em Ribeirão Preto, quando a colheita com máquinas é feita pela usina, o custo do fornecedor por tonelada é de R$ 34,76, quase R$ 5 menos do que na mesma modalidade em Jaú. Além da relação usina-fornecedor ser mais tradicional em Ribeirão, essa diferença de preço se deve, segundo Katia, ao fato de na região haver uma concentração maior de usinas que "disputam" o fornecedor de cana, o que gera vantagens ao produtor. Outros sistemas alternativos apresentam crescimento, como a terceirização da colheita com a contratação de empresas especializadas. Em Araçatuba, o custo total da cana nessa modalidade foi também um dos mais baixos - R$ 32,08 por tonelada. O estudo do IEA estará disponível no site do instituto no dia 31 de janeiro.

News Article 9

Os valores médios do etanol hidratado subiram em postos de 19 Estados brasileiros na semana passada, de acordo com dados coletados pela Agência Nacional de Petróleo, Gás Natural e Biocombustíveis (ANP) e compilados pelo AE Taxas, da Agência Estado. O maior reajuste nos preços do etanol, de 2,26%, ocorreu nos postos de Rondônia. De acordo com os dados apurados pela ANP no Estado o preço médio do litro do etanol variou de R$ 2,032 para R$ 2,078 na semana passada. Em São Paulo, maior produtor nacional do combustível, o preço saltou 0,98%. O litro do etanol hidratado nos postos paulistas ficou em média em R$ 1,741 na última semana, ante R$ 1,724 na semana anterior. Em Minas Gerais, segundo maior produtor, o preço médio do hidratado fico u estável na semana, em R$ 1,903. No Paraná, terceiro maior produtor, o valor médio apresentou leve recuo de 0,05%, de R$ 1,809 para R$ 1,808, se comparados os mesmos períodos. Além do Estado, as cotações recuaram em apenas cinco unidades da Federação: Acre, Alagoas, Rio de Janeiro, Rio Grande do Norte e Roraima. Os valores ficaram estáveis em Minas Gerais e no Distrito Federal. A maior baixa, de 0,87%, ocorreu nos postos alagoanos. O aumento semanal foi de 0,43%, para R$ 1,864 o litro, na média brasileira, o que levou o etanol a 71,44% dos R$ 2,609 cobrado pelo litro da gasolina. Com isso, se considerada a média dos preços do País, o uso da gasolina nos postos pelo consumidor é mais vantajoso ao consumidor que possui um veículo flex (bicombustível). No Brasil, o menor preço médio registrado para o etanol foi de R$ 1,741 por litro no Estado de São Paulo. O maior preço médio foi R$ 2,369 por litro no Acre. O menor preço em um posto foi no Estado de Goiás, de R$ 1,27 por litro, e o maior também foi registrado no Acre, de R$ 2,79 por litro. A vantagem do etanol é calculada considerando que o poder calorífico do motor a álcool equivale a 70% do poder nos motores à gasolina. Ou seja, é mais vantajoso para o motorista abastecer com álcool quando o preço cobrado no posto equivale a até 70% do preço da gasolina.

News Article 10

Usineiros da região de Ribeirão Preto (interior de São Paulo) fazem lobby contra a instalação de novas usinas no nordeste do Estado. Os produtores do mais tradicional polo do setor sucroalcooleiro do país alegam principalmente escassez de matéria-prima. Além de a região já ter 56 usinas de açúcar e álcool, 12,73% do total do país, os produtores dizem que as unidades têm capacidade instalada de moagem superior à produção anual. Três usineiros ouvidos pela Folha afirmaram, reservadamente, que há empecilhos para o funcionamento de mais uma unidade. E que, para evitar concorrência, têm fechado contratos adiantados com produtores de cana, para evitar perder fornecedores e, também, desestimular o surgimento de novas usinas. A estimativa é que as quatro regiões administrativas da macrorregião (Ribeirão, Franca, Central e Barretos) tenham moído 150 milhões de toneladas de cana nesta safra, mas poderiam chegar a 200 milhões, caso não faltasse matéria-prima. Um dos usineiros, que não quis se identificar, disse que a região está saturada e que uma eventual nova usina teria que buscar cana fora. Buscar cana fora significaria deslocar caminhões a até cem quilômetros, o que reduz o lucro -o ideal é, no máximo, 70 quilômetros. O lobby atinge a CEC (Companhia Energética Cravinhos), vizinha a Ribeirão e que prevê iniciar as atividades em 2012. Em "área de influência" de gigantes como a São Martinho (Pradópolis) e a Pedra (Serrana), ela enfrenta ainda resistência de ambientalistas, por causa do aquífero Guarani. Para o empresário Maurilio Biagi Filho, é "ótimo" que a região esteja satu rada. "Não é lobby, mostra a força da região." A diretoria da CEC informou que "a cana existe" e que tem parceiros comprometidos com o projeto. Para Antonio de Padua Rodrigues, diretor-técnico da Unica (União da Indústria de Cana-de-Açúcar), a dificuldade é a concentração de unidades, mas não há veto.