TÉCNICO LISBOA

# Verb Sense Disambiguation

**Tiago Manuel Paulo Travanca**

Thesis to obtain the Master of Science Degree in
**Information Systems and Computer Engineering**

**June 2013**

# Acknowledgements

First, and foremost, I would like to thank my supervisor, Prof. Nuno Mamede, for guiding me through the course of this thesis. With his experience and expertise he provided me important advice that made this work possible.

I am also very grateful to my co-supervisor, Prof. Jorge Baptista, who I had the pleasure to meet and work with, for being available to hear, discuss and give his insight on several topics addressed in this dissertation. His will to push me to go further and improve this work proved to be of great value.

I can not go by without thanking Cláudio Diniz from the L$^2$F group at INESC-ID Lisboa for his availability, cooperation and will to help.

I must also mention Claude Roux, from Xerox Research Labs, who provided helpful information regarding the KiF language, used in the Machine Learning approach.

My thanks to my family, specially my parents, for supporting my choices and giving me the opportunity to pursue a higher education.

A final word of gratitude is dedicated to my friends Ana, Elsa, Fábio, Inês, Miguel and Patrícia, who, despite knowing little about the magical gnomes I work with, were always there to cheer me up with their positivism.

Lisboa, June 11, 2013
Tiago Manuel Paulo travanca

To my brother

Pedro

# Resumo

Esta dissertação aborda o problema da desambiguação de sentido de verbos em Português Europeu. Trata-se de um sub-problema da desambiguação de sentido das palavras, tarefa que consiste em determinar, de entre os diferentes significados que potencialmente uma palavra pode ter, aquele que esta apresenta num contexto particular, num texto. Para tal, é necessário dispor de um inventário dos significados das palavras. No caso do problema da desambiguação de sentido de verbos, aqui tratado, utilizou-se o inventário de significados tal como definidos por classificação léxico-sintática dos mais frequentes verbos do Português Europeu (ViPEr).

Foram consideradas duas abordagens ao problema: a primeira abordagem, baseada em regras, usa a informação lexical, sintática e semântica presente no ViPEr para determinar o significado de um verbo; a segunda abordagem utiliza aprendizagem máquina baseada num conjunto de propriedades (features) comummente usadas em tarefas semelhantes de desambiguação de sentidos.

A exactidão (*accuracy*) de 84%, definida para o limiar de referência (*baseline*), e que resulta da mera atribuição a cada instância de um verbo do sentido mais frequente para esse lema (*most frequent sense*) revelou-se um valor surpreendentemente alto e difícil de superar. Ainda assim, ambas as abordagens seguidas permitiram obter resultados acima deste valor de referência. Efectivamente, a melhor combinação dos três métodos permitiu alcançar uma exactidão de 87.2%, um ganho de 3.2% acima do valor de referência.

# Abstract

This thesis addresses the problem of Verb Sense Disambiguation (VSD) in European Portuguese. Verb Sense Disambiguation is a sub-problem of the Word Sense Disambiguation (WSD) problem, that tries to identify in which sense a polissemic word is used in a given sentence. Thus a sense inventory for each word (or lemma) must be used. For the VSD problem, this sense inventory consisted in a lexicon-syntactic classification of the most frequent verbs in European Portuguese (ViPEr).

Two approaches to VSD were considered. The first, rule-based, approach makes use of the lexical, syntactic and semantic descriptions of the verb senses present in ViPEr to determine the meaning of a verb. The second approach uses machine learning with a set of features commonly used in the WSD problem to determine the correct meaning of the target verb.

Both approaches were tested in several scenarios to determine the impact of different features and different combinations of methods. The baseline accuracy of 84%, resulting from the most frequent sense for each verb lemma, was both surprisingly high and hard to surpass. Still, both approaches provided some improvement over this value. The best combination of the two techniques and the baseline yielded an accuracy of 87.2%, a gain of 3.2% above the baseline.

# Palavras Chave
# Keywords

## *Palavras Chave*

Processamento de Língua Natural

Desambiguação de Sentido de Verbos

Desambiguação baseada em regras

Aprendizagem Automática

## *Keywords*

Natural Language Processing

Verb Sense Disambiguation

Rule-based disambiguation

Machine Learning

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **NLP** | Natural Language Processing |
| **WSD** | Word Sense Disambiguation |
| **VSD** | Verb Sense Disambiguation |
| **POS** | Part of Speech |
| **XIP** | Xerox Incremental Parser |
| **ViPEr** | Verb for European Portuguese |
| **MFS** | Most Frequent Sense |
| **ML** | Machine Learning |
| **SR** | Semantic Roles |

# 1 Introduction

## 1.1 Motivation

Nowadays, there are many applications that make use of Natural Language Processing (NLP): search engines (semantic/topic search, rather than word matching), automated speech translation, automatic summarization, among others. All these applications, however, have to deal with ambiguity. Ambiguity is the term used to describe the fact that a certain expression can be interpreted in more than one way. In NLP, and in a pipeline approach, ambiguity is present at several stages in the processing of a text or a sentence.

One type of ambiguity is related to word and sentence segmentation, since most NLP systems require word and sentence splitting. In the scope of this task, word forms can be ambiguous because, for example, punctuation might denote other things besides the end of a sentence, such as abbreviations or field separators. However, this type of ambiguity is usually solved using lists of common abbreviations and regular expressions to represent patterns such as emails or URLs, for example.

Another type of ambiguity concerns word tagging. This type of ambiguity (morphological ambiguity) happens when a word can belong to different grammatical classes[1]. For example the word *canto* (sing/corner) can be a noun or a verb as shown in the examples (1.1a) and (1.1b)):

(1.1a) *Eu canto mal* $(I\ sing\ badly)$

(1.1b) *O sofá está no canto da sala* $(The\ couch\ is\ in\ the\ corner\ of\ the\ room)$

Most of the times, processing each word individually is not enough to determine which tag should be assigned, though processing the rest of the sentence usually enables the system to determine which part-of-speech (POS) tag is the right one in that context.

After words have been tagged, the syntactic parsing starts. This NLP task can be described as the representation of the syntactic relations (or dependencies) among the constituents of a sentence. Here the problem of ambiguity also arises. Given an ambiguous sentence, there can be more than one

[1]Besides this case, words can be ambiguous for corresponding to different inflections of the same lemma. This is particularly relevant in languages, such as Portuguese, with rich inflection systems. This problem, though, will not be addressed in this dissertation.

representation of the syntactic dependencies, each corresponding to a different meaning. The difficult step here is to choose which of the representations is the correct one or, at least, the more likely to occur.

(1.2a) *O Pedro mandou-me um postal dos Açores*

$(i)$ $(Peter\ sent\ me\ a\ postcard\ from\ Azores\ )$

*SUBJ(mandou,Pedro); CDIR(mandou,postal); CINDIR(mandou,me); MOD(mandou,Açores)*

$(ii)$ $(Peter\ sent\ me\ a\ postcard\ of\ Azores)$

*SUBJ(mandou,Pedro); CDIR(mandou,postal dos Açores); CINDIR(mandou,me)*

After the syntactic parsing is finished, there is another type of ambiguity to be solved, which is semantic ambiguity. Although this is usually the last one, it tends to be the hardest to solve among all types of ambiguity mentioned. For this type of ambiguity, the sentence has already been parsed and, even if its syntactic analysis (parse tree) is unique and correct, some words may feature more than one meaning for the grammatical categories they were tagged with. Consider the following examples:

(1.3a) *O Pedro conta carneiros antes de dormir* $(Peter\ counts\ lambs\ before\ sleeping)$

(1.3b) *O Pedro conta contigo para o ajudares* $(Peter\ counts\ on\ you\ to\ help\ him)$

The examples show how the verb *to count*, used in the same position on both sentences, means *tell* on the first sentence, while on the second it stands for *rely*. Usually, this difference in meaning is associated to syntactic properties. In these examples, the meaning of *count* in (1.3a) results from its direct-transitive construction and plural object, while in (1.3b) the preposition *on* introducing the second argument and the human trait on that same argument of *count* determine the verb's sense.

As mentioned before, this semantic ambiguity is usually one of the hardest to solve and it is a major NLP problem. Work in this area of WSD normally uses the surrounding words to help disambiguate the target word. To that end, machine learning techniques and/or other algorithms are used in combination with linguistic resources such as Wordnet[2] or Framenet[3].

As an example of the importance of word sense disambiguation, let's consider the case of automated speech translation. When trying to translate a sentence, the system has to capture the sentence's correct

[2]`http://wordnet.princeton.edu/` (last accessed in December 2011).
[3]`https://framenet.icsi.berkeley.edu/fndrupal/` (last accessed in December 2011).

meaning, in order to do a correct translation. For example, consider the following two sentences:

(1.4a) *A polícia apanhou os ladrões* ($The\ police\ caught\ the\ thieves$)

(1.4b) *O Pedro apanhou a moeda do chão* ($Peter\ grabbed\ the\ coin\ from\ the\ floor$)

Both use the Portuguese verb *apanhar*. However, this verb should be translated by different English verbs (with different meanings) in each of the examples.

## 1.2 Goals

This dissertation addresses the VSD problem, a sub-problem of WSD, for the European Portuguese. It aims at developing a set of modules of a NLP system that will enable it to choose adequately the precise sense a verb features in a given sentences, from among potential, different meanings. In this context, two VSD modules are to be developed: one, based on rules, will use a set of linguistic resources, specifically built for Portuguese, namely, a lexical database for the most frequent verb constructions from the European variety of the language - ViPEr. Another module, based on machine learning, will make use of a set of features, commonly adopted for the WSD problem, to correctly guess the verb sense.

Both modules are to be integrated in the STatistical and Rule-based Natural lanGuage processing chain (STRING) system (Mamede et al. 2012), an hybrid statistical and rule-based NLP system developed by and used at Spoken Language Systems Laboratory (L$^2$F). Finally, the two developed modules will be evaluated to assess the performance of the system.

## 1.3 Document Structure

In Chapter 2 the state of art/related work is presented, focusing first on several existing lexical resources for English, followed by an overview of the most common algorithms used in the WSD task.

Afterwards, in Chapters 3 and 4 the two implemented approaches are presented, detailing several decisions taken during their development.

Throughout Chapter 5, the different evaluation scenarios used to test both approaches are presented, followed by a discussion on the results obtained in each experiment and the main conclusions that can be drawn from them.

Finally, in the last chapter (Chapter 6), an overview of the entire work is done provided and the main conclusions are presented. The Chapter ends by pointing to future work, providing possible directions for expanding and improving the modules here developed.

# 2 State of the Art

## 2.1 Lexical Resources

When trying to disambiguate word senses using external tools like WordNet and FrameNet, the success or failure of the method used is greatly influenced by the type of information that is available about words in those databases, and how that information is represented.

The following sections will briefly described how information about words is represented in WordNet, FrameNet, VerbNet and ViPEr, giving special emphasis to the verb category, which is the main focus of this thesis.

### 2.1.1 WordNet

WordNet is an online lexical database developed at Princeton University. Initially, it was only available for English but later other WordNets were developed for languages such as Turkish (Bilgin et al. 2004), Romanian (Tufi et al. 2004), French (Sagot and Fišer 2008) and Portuguese (Marrafa et al. 2011)[1].

WordNet is a database of words and collocations that is organized around synsets. A *synset* is a grouping of synonymous words and pointers that describe the relations between this synset and other synsets. Among the most important relations represented in the WordNet (Miller 1995) are synonymy, antonymy, hyperonymy/hyponymy, meronymy, troponymy and entailment, each of them used with different categories of words.

*Synonymy* is WordNet's most basic and core relation, since everything is built around synsets. According to WordNet's definition (Miller et al. 1990), two expressions are synonymous in a linguistic context C if the substitution of one for the other in C does not alter the truth value of the context. This definition of word interchangeability requires that WordNet is divided according to the major parts-of-speech, namely: nouns, verbs, adjectives and adverbs. That is, if the concepts/meanings are represented by synsets and words in that synset must be interchangeable, then words of different syntactic categories cannot be synonyms (hence, cannot form synsets), as they are not interchangeable. Naturally, a word

[1]Portuguese WordNet is developed by the University of Lisbon in partnership with Instituto Camões, but is not available to use, only to search via the website.

| Semantic Relation | Syntactic Category | Examples |
|---|---|---|
| Synonymy | N, V, Adj, Adv | pipe, tube<br>rise, ascend<br>sad, unhappy<br>rapidly, speedily |
| Antonymy | Adj, Adv, (N, V) | wet, dry<br>powerful, powerless<br>friendly, unfriendly<br>rapidly, slowly |
| Hyponymy/Hyperonymy | N | sugar, maple<br>maple, tree<br>tree, plant<br>rapidly, speedily |
| Meronymy | N | brim, hat<br>gin, martini<br>ship, fleet |
| Troponymy | V | march, walk<br>whisper, speak |
| Entailment | V | ride, drive<br>divorce, marry |

Table 2.1: Semantic Relations in WordNet (from (Miller 1995))

may belong to more than one synset and the same word form may appear in more than one part of speech (*open*, the verb and *open* the adjective).

*Antonymy* is also a symmetric relation between two words and corresponds to the reverse of the synonymy relation, as it connects words that have an opposite meaning such as *wet* and *dry* or *fast* and *slow*.

*Hyperonymy/hyponymy*, also called superordinate/subordinate, is the equivalent to the *is-a* relation in ontologies, and allows for a hierarchical organization of concepts. For example, we can have an hierarchy of concepts with the words *maple*, *tree* and *plant*, where *maple* is the more specific concept and *plant* the most general one, forming a hierarchy like *maple-tree-plant*.

*Meronymy* corresponds to the *is a part of* relation in ontologies, and enables the representation of complex concepts/objects as a composition of their parts (simpler concepts/objects). This concept is applied in WordNet to detachable objects, like a *leg*, which is a part of the *body* or to collective nouns, as shown in Table 2.1 with concepts like *ship* and *fleet*.

*Troponymy* is a relation between verbs that denotes a particular manner of doing something. This relation the homologous of the hyponymy relation used for nouns. Take the verbs *speak, whisper* and *shout* as examples. The last two (*whisper* and *shout*) denote a particular way of speaking, therefore they are connected to the verb *speak* (the more general concept) through this troponymy relation.

*Entailment* is also a relation between verbs, that has the same meaning it has in logic, i.e. for the antecedent to be true, then the consequent must also be true, such as in the case of divorce and marry presented in the Table 2.1. One knows that in order for a couple to divorce, they must have been *married* in the first place.

All these relations are encoded in WordNet as pointers between word forms or between synsets and are the basis for the organization of WordNet categories. Nouns are mainly organized in hierarchies based on the hyperonymy/hyponymy between synsets, but they also include other pointers to indicate other relations. Adjectives are structured as clusters, containing head synsets and satellite synsets. Each cluster is built around two (occasionally three) antonymous concepts denoted by the head synsets. Most head synsets also have satellite synsets that represent concepts similar to the one represented by the head synset. Adverbs are often derived from adjectives, so usually these synsets include a relation to the adjective from which the adverb is derived from. Similarly to nouns, verbs are also organized in a hierarchical way, but instead of the hyperonymy/hyponymy, this structure is based on the troponymy relation described above.

WordNet documents the verbs based on 15 different files, each file covering a semantic domain: verbs of bodily care and functions, change, cognition, communication, competition, consumption, contact, creation, emotion, motion, perception, possession, social interaction, and weather verbs. Each of these files has a "unique" set of beginners, which correspond to the top most verbs in that hierarchy, and denotes the most basic concept in that tree, which is specialized by the remaining verbs in that tree. There is however one exception: one of the files contains verbs referring to states such as *resemble* or *belong* that could not be integrated into the other files (semantic domain files). The verbs in this last file do not share semantic properties or domain with each other, except for the fact that they all refer to states.

Verbs in WordNet do not follow a decompositional analysis like other lexical resources presented in this document, but rather a relational semantic analysis model as described in (Miller et al. 1990). Relational analysis differs from the decompositional analysis approach in the sense that it does not decompose the concepts until they are irreducible. In this case, verbs are not described in terms of their components but rather taken as a concept or as an entry in the mental dictionary of a person. Although the model used in WordNet is based on the relational analysis, it still has some aspects of the decompostional model. For example, the CAUSE component is represented in WordNet by the means of a relation among verbs like teach-learn, and other features like the MANNER can also be mapped into the troponymy relation between verbs.

Besides this difference from the pure relational analysis, WordNet also incorporates the most important syntactic aspects of the verbs. Therefore, each synset has one or several syntactic frames that specify the features of the verbs belonging to the synset and indicate the kind of sentences in which they can occur. This information allows the user to search for verbs that share the same syntactic frame and then compare the semantic components of each of the verbs; or one can start with two semantically similar verb and see whether both verbs share the same syntactic properties.

WordNet has several applications in the context of NLP, namely in some WSD tasks. One example of such, is the noun disambiguation system described in (Buscaldi et al. 2004). The system makes use of the WordNet noun hierarchy, based on the hyponym/hypernym relations described in this section, to assign a noun to a synset, which corresponds to assigning a meaning to that noun. For example, given a noun to disambiguate (target) and some context words, the system first will look into WordNet synsets that can be assigned to the target noun; then for everyone of those synsets, it will check how many of the context words fall under the sub-hierarchy defined by that synset. A simplified example of this is shown in Figure 2.1 (originally presented in the paper (Buscaldi et al. 2004)).

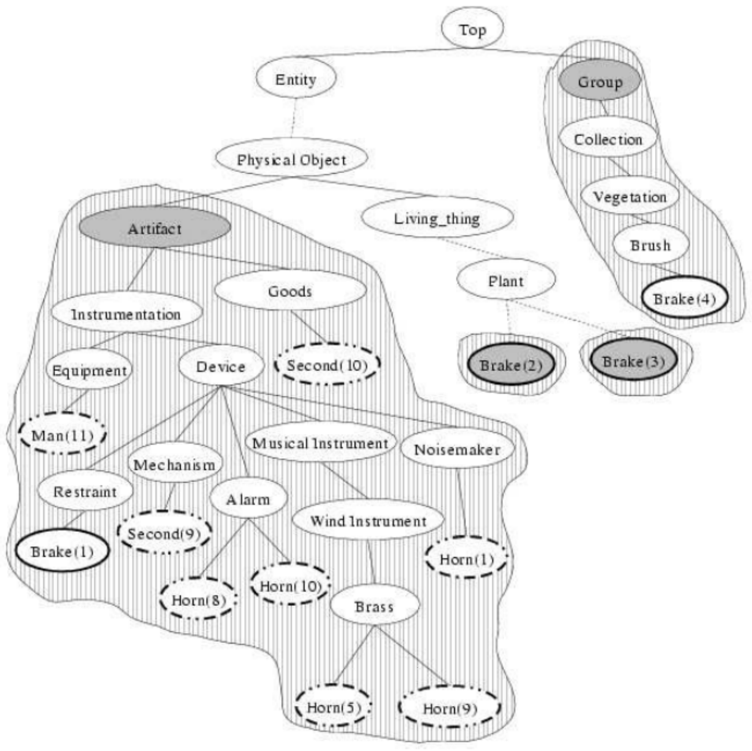

Figure 2.1: Hierarchies used in the disambiguation of *brake* with context words {*horn, man, second*} (from (Buscaldi et al. 2004))

The actual formula however is more complex than what was described here, as it takes into account parameters like the height of the sub-hierarchy, the number of expected words to fall under that sub-hierarchy. Also, another important aspect that is referenced in the paper is the fact that some senses are more frequent than others so that, to take frequency into account, a factor was also added to the original formula, giving more weight to the most frequent senses.

### 2.1.2 FrameNet

The FrameNet (Baker et al. 1998; Johnson et al. 2001; Ruppenhofer et al. 2006) is a lexical database that aims to represent semantic and syntactic information about English words (mostly verbs, nouns and adjectives), developed at Berkeley University. Initially, this project was only available for English, however several other universities have started similar projects in order to develop FrameNet for their own languages. Thus, there are now FrameNet databases for languages such as Japenese (Ohara et al. 2004), Swedish (Borin et al. 2009) and Spanish (Subirats and Sato 2004). Although the English FrameNet has been the first to be developed, it is not publicly available contrary to it's European equivalents.

The FrameNet database is built around semantic frames and lexical units, which belong to several semantic domains of the human knowledge. The domains covered by the FrameNet project are: HEALTH CARE, CHANCE, PERCEPTION, COMMUNICATION, TRANSACTION, TIME, SPACE, BODY (parts and functions of the body), MOTION, LIFE STAGES, SOCIAL CONTEXT, EMOTION and COGNITION.

*Semantic frames* are a schematic representation that tries to capture a concept, situation, or event along with its participants, which are called *frame elements (FEs)*. Typically, the FEs of a frame correspond to the semantic arguments of the word [TARGET] associated with that frame. An entry in the frame index has the information described in Table 2.2, and a simplified exampled is presented on Table 2.3.

| | |
|---|---|
| domain | semantic domain |
| frame | name of the frame |
| FEs | list of the FEs |
| description | brief textual description of the frame |
| examples | annotated example sentences for the frame |

Table 2.2: Frame Entry Description (from (Johnson et al. 2001))

| | |
|---|---|
| domain | Motion |
| name | Motion |
| FEs | Core: Theme, Area, Direction, Distance, Goal, Path, Source, ...<br>Non-core: Degree, Duration, Time, Speed, ... |
| description | Some entity (Theme) starts out in one place (Source) and ends up in some other place (Goal), having covered some space between the two (Path). Alternatively, the Area or Direction in which the Theme moves or the Distance of the movement may be mentioned. |
| examples | [Dust particles THEME] [floating TARGET] [about AREA] made him sneeze uncontrollably.<br>[The swarm THEME] [went TARGET] [away DIRECTION] [to the end of the hall GOAL]. |

Table 2.3: Motion Frame (adapted from the FrameNet website)

As suggested from the description above, frame elements also play a very important role in FrameNet, providing very useful semantic information. Frame elements can be seen as the participants (i.e. elements playing semantic roles) in the scene described by a particular frame.

Obviously, some words do not require all their arguments to be present. However, absent constituents can have different natures. There are participants that even if absent can be implicit. This is the case of essential frame elements, i.e., participants that are essential to the meaning of the frame. On the other hand there are those that despite being present in some sentences, play an accessory role to the scene. An example of such is the adverbial time/date *last week* in the sentence *I went to Italy last week*. In order to address this distinction, FrameNet labels the frame elements as being core or non-core.

Core frame elements can be seen as a conceptually necessary component for that particular frame, distinguishing it from other frames. For example, in the *Revenge* frame, AVENGER, PUNISHMENT, OFFENDER, INJURY and INJURED_PARTY are all core frame elements, because an avenging event necessarily includes these participants.

Non-Core frame elements correspond to optional components in a sentence. Typically, non-core frame elements introduce information to the sentence such as TIME, PLACE, MANNER, MEANS, DEGREE, among others. They do not uniquely characterize a single frame and can be instantiated in several frames, usually in the form of a PP (prepositional phrase). Recalling the example mentioned previously: *I went to Italy last week*, the time expression *last week* is an example of an optional argument (non-core frame element) in the Motion frame, to which the verb *go* belongs.

Although there is this distinction between the essential (core) and optional (non-core) elements of a frame, even the core FEs can be absent in a sentence, which adds an extra degree of difficulty to the problem of disambiguation. Again, in the example *I went to Italy last week*, and taking into consideration the Motion frame defined in FrameNet it can be seen that in this particular sentence there is the THEME *I*, the GOAL *Italy* and the optional TIME element *last week*, however it does not have other core elements such as the SOURCE, PATH, DIRECTION, DISTANCE or AREA. This happens in most cases because of an anaphoric reference, i.e., the element missing was already mentioned previously in the text, and in that case FrameNet labels the missing frame element as a Definite Null Instantiation (DNI) (see example (2.1a) below). It can also happen that a verb, which is usually transitive, is used in an intransitive way, as shown in example (2.1b) below, and, in this case, it is called an Indefinite Null Instantiation (INI). A last case of null instantiation can occur when a grammatical construction involving the target word allows for some elements to be missing, like in imperative sentences, as shown in the third sentence below. In that case the missing elements are labeled as Constructional Null Instantiation (CNI). The three sentences below (taken from (Ruppenhofer et al. 2006)) exemplify these three types of null instantiation.

(2.1a) *The result should be similar. [DNI Entity 2]*

(2.1b) *Peter rarely eats alone. [INI Ingestibles]*

(2.1c) *Cook on low heat until done. [CNI Food]*

From our daily use of language one can see that people do not usually use all the components that are referenced in the frame, in particular people tend to use some components together, just like a set. In such case it, is said that those elements belong to a CoreSet, i.e., they have a good chance to co-occur. An example of such set is the SOURCE, PATH and GOAL in the several motion frames.

Other common usage pattern is when some element requires the presence of another, or when it sometimes excludes the use of some other element, in order to make that sentence valid. These dependencies are represented through the *requires* relation and *excludes* relation respectively. Considering now the ATTACHING frame that has ITEM, GOAL and ITEMS (plural) as core elements, from the example below (taken from (Ruppenhofer et al. 2006)), it can be seen that having the ITEM frame element requires the presence of the GOAL frame element, while the sentence without the GOAL frame element is unacceptable. On the other hand, the presence of the ITEMS frame element excludes the use of both ITEM and GOAL.

(2.2a) *The robbers tied [Paul ITEM] [to his chair GOAL].*

(2.2b) *$^*$The robbers tied [Paul ITEM].*

(2.2c) *The robbers tied [his ankles ITEMS] together.*

In addition to the relations between frame elements described previously there are also similar relations between frames, such as inheritance, precedes, subframe, among others. The *subframe* relation allows to decompose a frame describing a complex event in terms of several other frames that denote smaller ones. The *precedes* relation represents the temporal relation between two frames describing different events. This relation is used usually with the subframe relation when describing a complex frame that is the result of a sequence of smaller steps in a particular order. However, it is *inheritance* the most important relation in FrameNet. Inheritance makes possible to structure frames in a hierarchy, in a similar way to what is done in ontologies with the *is-a* relation. Having this relation between frames entails that what is true for the semantics of the parent must correspond to an equal or more specific fact about the child. This includes frame element membership, relation to other frames, relations between Frame Elements and semantic types on frame elements. An example of frame inheritance is shown on Table 2.4.

| |
|---|
| frame(TRANSPORTATION)<br>frame_elements(MOVER(S), MEANS, PATH)<br>scene(MOVER(S) move along PATH by MEANS) |
| frame(DRIVING)<br>inherit(TRANSPORTATION)<br>frame_elements(DRIVER (=MOVER), VEHICLE (=MEANS), RIDER(S) (=MOVER(S)), CARGO (=MOVER(S)))<br>scenes(DRIVER starts VEHICLE, DRIVER controls VEHICLE, DRIVER stops VEHICLE) |
| frame(RIDING_1)<br>inherit(TRANSPORTATION)<br>frame_elements(RIDER(S) (=MOVER(S)), VEHICLE (=MEANS))<br>scenes(RIDER enters VEHICLE, VEHICLE carries RIDER along PATH, RIDER leaves VEHICLE) |

Table 2.4: Frame Inheritance Example (from (Baker et al. 1998))

The example in Table 2.4 shows the TRANSPORTATION frame from the Motion domain, which is described by three frame elements: MOVERS, MEANS and PATH. Subframes associated with words that denote a more specific concept inherit all these elements while having the possibility of adding new ones of their own. For example, the DRIVING frame specifies the role DRIVER as a principal MOVER, a VEHICLE as particularization of the MEANS element provided by the parent frame and CARGO or RIDERS as secondary MOVERS. On the other hand neither of the subframes specify a particular type of PATH, so they simply inherit that element from the TRANSPORTATION frame.

Besides frames and frame elements, another important piece of information represented in the FrameNet that is very useful when trying to disambiguate words are the *lexical units*. A *lexical unit (LU)* is just a pairing of a word with one sense. Each LU is connected to frames in which they are involved in. The information contained in each entry of the lexical unit index is described in Table 2.5 and Table 2.6 provides a simplified example of an entry.

| | |
|---|---|
| lemma | base form of the word |
| POS | part of speech tag |
| domain | semantic domain |
| frame | name of the frame connected to this LU |
| FrameNet definition | definition of the sense represented by this LU written by FN staff |
| FEs | list of frame elements within this LU |
| COD definition | definition from the Concise Oxford Dictionary |
| WN link | link to WordNet synset |

Table 2.5: Lexical Unit Entry Description (adapted from (Johnson et al. 2001))

Furthermore, within a lexical unit entry, each frame element, taken from the frame to which this LU is connected, is also described in terms of it's syntactic realizations, which are the combination of the

[2] A FrameNet definition was not found for this example in the FrameNet website.
The link to the corresponding WN synset was not found on the FrameNet website.

| | |
|---|---|
| lemma | ride |
| POS | verb |
| domain | motion |
| frame | ride_vehicle |
| FEs | Theme, Vehicle, Source, Goal, Purpose |
| COD definition | travel in or on (a vehicle or horse). |

Table 2.6: Ride.v (Ride_Vehicle) Lexical Unit (adapted from the FrameNet website)[2]

different phrase types and the grammatical functions that that particular frame element has on sentences collected from the British National Corpus (BNC)[3]. FrameNet includes many phrase types among the most common are noun-phrase (NP), verb-phrase(VP), prepositional phrase (PP) and subtypes of these, like VPing which denotes a gerund verb phrase or PP[by] or PP[on] that capture prepositional phrases introduced by the preposition in brackets. Grammatical functions represent the relation between the target word and the FE, and can vary depending on the class of the target word. For the verb category, the grammatical functions used in the FrameNet are External Argument (Ext), which corresponds to the subject, Object (Obj) or Dependent (Dep), for the remaining complements.

| Frame Element | Realizations |
|---|---|
| Theme | NP.Ext |
| Vehicle | DNI.– NP.Obj |
| Source | PP[from].Dep |
| Goal | PP[to].Dep |
| Purpose | VPto.Dep |

Table 2.7: Ride.v (Ride_Vehicle) Frame Element's Syntactic Realizations (adapted from the FrameNet website)

Additionally, *valence patterns* are built for that lexical unit based on the information of each frame element. This patterns consist of all the combinatorial possibilities of the frame elements (and their syntactic realizations) evidenced in sentences from the corpus. Table 2.7 illustrates an example of the syntactic realizations regarding the frame elements in the *ride.v (Ride_Vehicle)* lexical unit, and Table 2.8 shows the valence patterns for the same LU.

| Patterns | | | |
|---|---|---|---|
| Theme NP.Ext | Vehicle DNI.– | Source PP[from].Dep | Goal PP[to].Dep |
| Theme NP.Ext | Vehicle NP.Obj | Source PP[from].Dep | Purpose VPto.Dep |

Table 2.8: Ride.v (Ride_Vehicle) Valence Patterns (adapted from the FrameNet website)

[3]`http://www.natcorp.ox.ac.uk` (last accessed in December 2011)

A direct application of FrameNet to verb classification was found for Chinese. However, the majority of the work using FrameNet is more dedicated to semantic role labeling. Despite not being a direct classification of the verb class, thematic role assignment to sentence constituents implicitly leads to a better verb sense identification. The system described in (Gildea and Jurafsky 2002) uses a clustering algorithm with several features (lexical and syntactic) to automatically assign the semantic roles. Another approach presented in (Shi and Mihalcea 2004) transforms FrameNet's frames into rules, which are then used with testing examples. The following examples (presented in same paper (Shi and Mihalcea 2004)) illustrate how the role assignment is done.

Rules learned for target word *come*:

(2.3a)
$$[[ext, np, before, active, theme],$$
$$[obj, np, after, active, goal],$$
$$[comp, pp, after, active, by, mode\_of\_transportation]]$$

(2.3b)
$$[[ext, np, before, active, theme],$$
$$[obj, np, after, active, goal],$$
$$[comp, pp, after, active, from, source]]$$

Using a syntactic analyzer, the following features were extracted for each of the following sentences:

(2.4a) *I come here by train.*

$$[[ext, np, before, active], [obj, np, after, active], [comp, pp, after, active, by]]$$

(2.4b) *I come here from home.*

$$[[ext, np, before, active], [obj, np, after, active], [comp, pp, after, active, from]]$$

Comparing the extracted features with the rules that were created for the word *come*, using some scoring algorithm, it is easily seen that matching the rule (2.3a) with (2.4a) would produce a higher score than matching the (2.4a) with (2.3b). Therefore, the thematic role assigned to *train* is *mode_of_transportation*. As for the example (2.4b), its score would be greater when matched with (2.3b), than when matched with (2.3a), so in that case *home* would be assigned the role of *source*.

### 2.1.3 VerbNet

The VerbNet[4] (Kipper et al. 2000) is an online lexical database for verbs, developed at the University of Colorado, and it is currently available only for English. Although VerbNet is a hierarchical domain-independent resource, unlike other online resources such as WordNet or FrameNet, it includes mappings to these other lexical resources.

Each verb class in the VerbNet is described by thematic roles, selectional restrictions on the arguments (i.e. on the types of elements accepted as certain thematic roles) and frames. Each VerbNet class contains a set of syntactic descriptions, or syntactic frames, describing the possible realizations of the argument structure for sentence construction. Thematic roles describe how certain entity participates in the scene described by a sentence, i.e. what is the role of that entity in the action. When associating these thematic roles with word forms, restrictions can be imposed on the types of elements accepted for a certain role, such as *animate* or *human* for a thematic role of an *Agent*. Those kinds of restrictions are called semantic (or selectional) restrictions and in VerbNet they are defined based on the type tree presented in Figure 2.2.

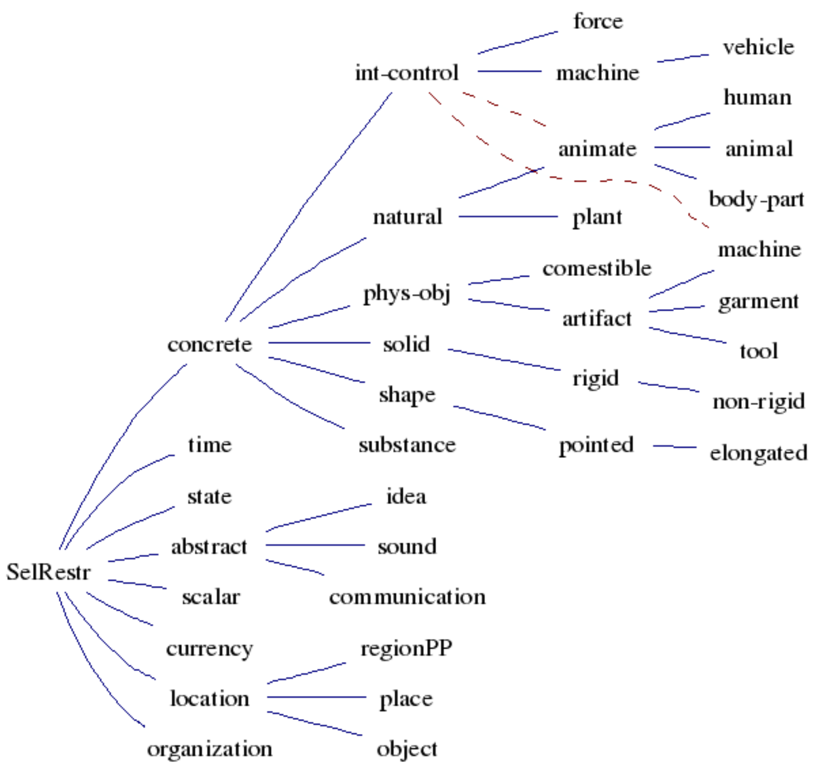

Figure 2.2: Selectional Restrictions associated with thematic roles (from VerbNet website)

[4]`http://verbs.colorado.edu/~mpalmer/projects/verbnet.html` (last accessed in December 2011)

Additional restrictions maybe be imposed to indicate the syntactic nature of the sentence's constituent that is likely to be associated with certain thematic role, such as NP or a PP. As for the semantic information in each frame, it is explicitly represented by a conjunction of predicates (in a logical form) such as 'motion', 'contact' or 'cause'. Each predicate is associated with an event variable E that allows to specify when in the event described by the frame, a certain predicate is true. An event is considered to be divided in three stages: the preparatory or *start(E)*, the culmination stage or *during(E)* and *end(E)* for the consequent stage. Table 2.9 shows an example of an entry in the VerbNet for the class Hit-18.1.

| **Class Hit-18.1** |
|---|
| Roles and Restrictions: Agent[+int_control] Patient[+concrete] Instrument[+concrete] |
| Members: bang, bash, hit, kick, ... |
| Name: Basic Transitive |
| Example: Paula hit the ball |
| Syntax: Agent V Patient |
| Semantics: cause(Agent, E) manner(during(E), directedmotion, Agent) !contact(during(E), Agent, Patient) manner(end(E), forceful, Agent) contact(end(E), Agent, Patient) |

Table 2.9: Simplified VerbNet entry (from VerbNet website)

The example describes an abstract verb class (Hit-18.1) which is connected to several word forms like *bang, bash, hit,* among others. The frame also presents the different roles of the various participants in it; in the case, the roles are: Agent, Patient and Instrument. These roles are also attached with the respective selectional restrictions, shown in square brackets in the example, which correspond to a type in the type tree presented in Figure 2.2. The frame also includes the syntactic pattern associated for sentence construction as well as logical form clauses to encode the meaning of the frame. Essentially, the logical predicates, present in the semantics field in the Table 2.9, try to encode a natural language description of such a scenario in the form of predicates. In this particular case, the Agent is considered to be the cause of the event (cause(Agent,E)), during the event the Agent is producing a motion that is directed somewhere and the Agent is also not in contact with the Patient. By the end of the scenario described by the frame, the Agent is in contact with the Patient in a forceful manner.

VerbNet has had some applications for verb sense disambiguation, like described in (Brown et al. 2011) and (Abend et al. 2008). The techniques used SVM classification (Brown et al. 2011) (more on this in Section 2.2), and another learning algorithm (SNoW) (Abend et al. 2008), specifically aimed to large scale learning. Both of these works used classification methods (although with different algorithms), as well as the same training and test corpus, the SemLink corpus[5]. Several features were used, such as lexical, syntactic and semantic (more on this in Section 2.2).

[5]`http://verbs.colorado.edu/semlink/` (last accessed in 2011).

### 2.1.4 ViPEr

ViPEr (Baptista 2012) is a lexical resource that describes several syntactic and semantic information about the European Portuguese verbs. Unlike some of the other resources presented above, verbs are the only grammatical category present in ViPEr and it is available only for (European) Portuguese. This resource is dedicated to full distributional (or lexical) verbs, i.e., verbs whose meaning allows for an intensional definition of their respective construction and the distributional (semantic) constraints on their argument positions (subject and complements). A total of 6224 verb senses have been described so far, with frequency 10 or higher in the CETEMPúblico corpus[6]. The description of the remainder verbs is still on going.

As described in (Baptista 2012), the classification of each verb sense in ViPEr is done using a syntactic frame with the basic sentence constituents for Portuguese. This frame is composed of: *N0, prep1, N1, prep2, N2, prep3, N3*. The components *N0-N3* describe the verb's arguments, for a particular sense, with *N0* corresponding to the sentence's subject, and *N1, N2* and *N3* to the verb's complements. Each argument can be constrained in terms of the values it can take. An example of such restrictions are: *Hum* or *Nnhum* to denote the trait human and non-human, respectively; *Npl* for plural nouns; *QueF* for completive sentences, among others. For the arguments of certain verb senses, these constraints can be further rendered in a more precise way using semantic features such as *<instrumento>, <divindade>, <instituição>, <data>* or *<jogo>* (<instrument>, <divinity>, <institution>, <date>, <game>, respectively).

However, it is not the arguments and their restrictions alone that define a verb sense. Prepositions introducing these arguments also play a very important role, and so, they are explicitly encoded in the description of verb senses, ranging from *prep1* to *prep3*, where the index corresponds to the index of the argument they introduce (*N1-N3*). For direct (object) complements, the *Prep1* is left empty. The type of prepositions considered in ViPEr are both the simple (e.g. *a, de* (to, of)) and complex (e.g. *junto a, a favor de* (near to, in favor of)). Locative prepositions are collapsed under the value *Loc*, unless otherwise necessary.

Intrinsically reflexive verbs, i.e., verbs (or verb senses) used only with reflexive pronouns (e.g. *suicidar-se* (to commit suicide)) are marked by a feature *vse* and the pronoun is not considered an autonomous noun phrase (NP), since the verb can not be employed without the reflex pronoun (example (2.5b)), nor does it accept an accusative NP or dative PP with a noun that is not correferent to the

[6]http://www.linguateca.pt/cetempublico/

sentence's subject (example (2.5c)):

(2.5a) *O Pedro suicidou-se* (*Peter committed suicide*)

(2.5b) **O Pedro suicidou* (*Peter suicided*)

(2.5c) **O Pedro suicidou o João ao João* (*Peter suicided John to John*)

Another important aspect to take into account when describing and disambiguating verb senses is the type of passive constructions they admit. In Portuguese, there are three major types of passive sentences: the ones constructed using the auxiliary verb *ser* (example (2.6b) below), the ones with the verb *estar* (and other copula verbs) (example (2.6c) below), and a reflex construction where the direct object is raised to the subject position as well (example (2.6d)). These three types of passives constitute three fields in each verb sense entry in ViPEr and take as values a boolean flag that denotes whether or not that verb sense admits that kind of passive construction.

(2.6a) *O Pedro partiu a jarra* (active)
(*Peter broke the jar*)

(2.6b) *A jarra foi partida pelo Pedro* (passive with *ser*)
(*The jar was broken by Peter*)

(2.6c) *A jarra está partida* (passive with *estar*)
(*The jar is broken*)

(2.6d) *A jarra partiu-se* (reflexive construction)
(*The jar broke*)

Some verbs allow for the pronominalization of its PP arguments, introduced by the preposition *a* (to), by the dative pronoun *lhe* (see the examples (2.7a) and (2.7b) below). However, with other verbs that present this type of construction, this transformation is not possible, as in (2.7c), where the PP *a matemática* (at math) is not replaceable by the pronoun *lhe* (2.7d). This information is also encoded in

ViPEr as a boolean field for each of the verb senses.

(2.7a) *O Pedro escapou aos polícias.* (*Peter escaped the cops*)

(2.7b) *O Pedro escapou-lhes.* (*Peter escaped them*)

(2.7c) *O Pedro reprovou a matemática.* (*Peter failed at math*)

(2.7d) **O Pedro reprovou-lhe.* (*Peter failed it*)

Besides the cases mentioned above, some classes of verbs also have particular semantic features associated with some of their arguments. Those classes of verbs are: *psychological* verbs (class *04*), *motion* verbs (several classes), *transfer* verbs (class *36DT*), each of them with specific semantic features.

One of these features is *semantic polarity* (positive or negative), which can be used for sentiment analysis. This feature has been systematically described for *psychological* verbs. These verbs' construction involve a human entity (the experiencer), that experiences a psychological state/emotion, expressed by the verb. This emotion can be described as having *positive* or *negative* a polarity for the experiencer. For example, for sentence (2.8a) it is possible to state that *Peter* has a positive feeling about what he has done, hence the verb *orgulhar* is tagged with the *positive* feature in ViPEr to denote that semantic polarity.

(2.8a) *O Pedro orgulha-se do que conseguiu alcançar*

(*Peter is proud of what he was able to achieve*)

With *motion* verbs, on the other hand, the type of semantic information encoded in ViPEr is equivalent to thematic roles associated with the arguments those verbs take. For motion verbs, information about the locative complements, namely whether it refers to the *destination* or *source* is stored:

(2.9a) *O Pedro foi a Lisboa. (destination)*

(*Peter went to Lisbon*)

(2.9b) *O Pedro veio de Lisboa. (source)*

(*Peter came from Lisbon*)

Most of the times, a motion verb accepts only one of these values, but some can be tagged with both of the corresponding flags (e.g. transference predicates), when both of the thematic roles can be simultaneously present (see example (2.10a)).

(2.10a) *O Pedro moveu os móveis da sala (source) para o corredor (destination)*

$(Peter\ moved\ the\ furniture\ from\ the\ room\ (source)\ to\ the\ hallway\ (destination))$

*Transfer* verbs, i.e., verbs involving an object that switches owners, are described using the roles of *beneficiary* and *privative* (see examples (2.11a) and (2.11b)).

(2.11a) *O Pedro ofereceu um livro ao João (beneficiary)*

$(Peter\ offered\ a\ book\ to\ John)$

(2.11b) *O Pedro roubou um livro ao João (privative)*

$(Peter\ stole\ a\ book\ from\ John)$

Naturally, not all of this semantic information is to be used in the WSD task. For the moment, only semantic roles of locative complements of *motion* verbs will be used, to match the semantic type of locative prepositions collapsed under *Loc*.

Verbs selecting a sentential argument were encoded with the feature *QueF* in that given slot (*N0-N3*). However in some cases a verb sense only admits a particular type of completive sentence, featuring a verb in the conjunctive, the indicative or the infinitive mood. Additionally, a particular type of sentential construction, introduced by the expression *o facto de* (the fact that), and introducing a *factive* modality, was separately handled in ViPEr. All this information is present in the form of a boolean flags, one for each of the argument positions in each verb sense. This information can be helpful when disambiguating a verb sense, because if the a sentence displays that particular construction, then, only verb senses with that feature can be the correct interpretation for that instance.

Constraint co-reference between any argument NP and the argument NPs of the dependent sub-clauses are also marked explicitly in the database. Hence, verbs like *pedir* (to ask) are treated differently from *prometer* (to promise) since the zeroed subject of the infinitive sentential object can only be interpreted as N2 and N0, respectively, as indicated by the co-reference indexes in the following examples:

(2.12a) *O Pedro pediu à Ana*$_i$ *que fizesse*$_i$ *isso.*

$(Peter\ asked\ Mary\ to\ do\ this.)$

(2.12b) *O Pedro*$_i$ *prometeu à Ana que faria*$_i$ *isso.*

$(Peter\ promised\ Mary\ to\ do\ it.)$

Verb uses were classified according to their sentence structure and distributional features. 60 classes were thus construed. Some of these classes are remarkably homogeneous, both syntactically and semantically. In these cases, the argument positions of the verbs from an entire class have the same semantic role. Hence, *communication* verbs, like *dizer* (to say), were grouped in class 09, with *Agent/Speaker* subject, *Patient/Addressee* indirect object and *Theme/Content* direct object; *psychological* verbs feature *Cause* subjects and *Patient/Affected* direct object. Semantic roles designations used here are just tentative indications. The semantic role labeling (SRL) task is being dealt separately, under another dissertation project (Talhadas, in progress). Though initially not encoded explicitly in ViPEr, the semantic roles underly this classification. During the course of this dissertation, the semantic roles have been systematically encoded in ViPEr, but the information has not yet been available in time to be used by this project.

Table 2.10 shows the most recent statistics collected from the ViPEr database about the number of possible senses for each verb. Table 2.11 shows an example of the several entries created for the different meanings of the verb *apontar* (to point).

| **Number of Classes** | **Lemmas** |
|---|---|
| 1 | 4267 |
| 2 | 515 |
| 3 | 159 |
| 4 | 62 |
| 5 | 19 |
| 6 | 8 |
| 7 | 2 |
| 8 | 3 |
| 10 | 1 |
| 11 | 1 |
| **Total** | 5037 |

Table 2.10: ViPEr's verb distribution (per number of possible classes)

| Class | Example |
|---|---|
| 36DT | *O bandido apontou uma faca ao polícia.* |
| 38LD | *O Pedro apontou os números premiados num recibo do multibanco.* |
| 39 | *O Pedro apontou o João como sério candidato ao cargo.* |
| 09 | *O Pedro apontou à Joana quais os defeitos que devia corrigir.* |

Table 2.11: ViPEr's entries for the different meanings of *apontar* (to point)

### 2.1.5 Lexical Resources Comparison

In the previous sections, four different lexical resources were presented, each of them encoding syntactic and semantic information about words in a different way. Table 2.12 shows the different features represented in each of these lexical resources.

| Lexical Resource | Grammatical Categories | Syntactic Patterns | Thematic Roles | Selectional Restrictions | Has Examples |
|---|---|---|---|---|---|
| WordNet | N,V,Adj,Adv | Yes | No | No | Yes |
| FrameNet | N,V,Adj,Adv | Yes | Explicit | No | Yes |
| VerbNet | V | Yes | Explicit | Explicit | Yes |
| ViPEr | V | Yes | Implicit | Explicit | Yes |

Table 2.12: Lexical Resources Comparison

In the context of WSD all these resources can be used as they provide very useful information about the constrains that have to be satisfied when a word has a particular meaning. In the particular case of verb sense disambiguation NLP task, the most important features to be considered are formal features, such as the syntactic patterns of the sentence forms, and the distributional constraints resulting from the selectional restrictions of the verb on its arguments. These formal aspects can be used as context clues for the discovery and correct identification of a verb sense in a given sentence. ViPEr, the resource here presented for the Portuguese language, also provides additional information about the verbs and their respective sentence construction that is not represented in the other lexical resources, such as the constrains of completive sentences.

Besides the resource here described, there are, naturally, other known sources with linguistic information on verbal constructions of Portuguese. Among others, the most relevant (and recent) are described in (Borba 1990) and (Busse 1994), each developed in a different theoretical framework. The work presented in (Borba 1990) is a comprehensive description of Brazilian Portuguese verb constructions. The work breaks down the multiples verb senses and present their syntactic structure and the thematic roles associated to each argument position. However, it addresses the Brazilian variety of Portuguese, and while much must certainly be similar to the European variety, differences in detail would compromise its application to texts from the later. On the other hand, the work presented in (Busse 1994) only deals with the Portuguese variety, and produces a classification of just a little less than 2,000 verbs (lemmas). It describes the syntactic structure of verb constructions and has some information on distributional constraints. None of these dictionaries adopts a transformational approach, and thus they split the same verb sense in as much frames as syntactic structures it may present. These dictionaries seem to be available only in paper support, and though they may be in digital format, to the best of our knowledge, this is not available. Besides, these dictionaries were mostly meant for human consultation, so even if digitized, they would require an adaptation effort, which is out of the scope of this dissertation.

## 2.2 Word Sense Disambiguation Approaches

In this section, an overview of the most used techniques for WSD will be presented. This information was mostly based on systems evaluated at the SensEval3[7] conference (2004). SensEval first took place in 1998, with the purpose of gathering and evaluating state-of-the-art systems in tasks related to the semantic disambiguation of words, such as, semantic (thematic) role labelling or word sense disambiguation.

### 2.2.1 Algorithms

The majority of the systems used in the WSD task use approaches based on machine learning (Witten et al. 2011). Usually, one or more classifiers are used, and in the last case, a combination method is also used, which can vary from simply choosing the best score or by some more sophisticated combination (e.g. weighting each classifier) of the results produced by each of the classifiers. The most common learning algorithms used at SensEval3 are the following:

- *Naive Bayes* algorithm (Manning et al. 2008), which estimates the most probable sense for a given word *w* based on the prior probability of each sense and the conditional probability for each of the features in that context.
- *Decision List* algorithm (Yarowsky 1995), which builds a list of rules, ordered from the highest to the lowest weighted feature. The correct sense for the word is then determined by the first ruled that is matched.
- *Vector Space Model* algorithm (Salton et al. 1975), which considers the features of the context as binary values in a vector. In the training phase, a centroid is calculated for each possible sense of

[7] `http://www.senseval.org/` (last accessed in December 2011)

the word. These centroids are then compared with vectors of features from testing examples using the cosine function.

- *Support Vector Machines* (Cortes and Vapnik 1995), the most widely used classification technique in WSD at SensEval3 (Agirre et al. 2004; Lee et al. 2004; Escudero et al. 2004; Villarejo et al. 2004), is a classification method that finds the maximal margin hyperplane that best separates the positive from the negative examples. In the particular case of WSD, this has to be slightly tuned for multiple class classification. Usually methods like *one-against-all* are used, which lead to the creation of one classifier per class.

### 2.2.2 Features

As in every machine learning algorithm, a set of features must be defined and used in order for it to learn how to distinguish among the different possible classifications for each item. The most commonly used features among the systems proposed and presented at SensEval3 can be divided as following:

- *Collocations*: $n$-grams (usually bi-grams or tri-grams) around the target word are collected. The information stored for the $n$-grams is composed by the lemma, word-form and part-of-speech tag of each word.
- *Syntactic dependencies*: syntactic dependencies are extracted among words around the target word. The relations most commonly used are *subject, object, modifier*. However, depending on the system, other dependencies might also be extracted.
- *Surrounding context*: single words in a defined window size are extracted and used in a bag-of-words approach. Usually both the word-form and the lemma are extracted.
- *Knowledge-Based information*: some systems also make use of information such as WordNet's domains, FrameNet's syntactic patterns or annotated examples, among others.

### 2.2.3 Evaluation

In SensEval3 there were mainly two different types of tasks related to WSD: the lexical sample task and the all-words task. Both tasks were available for several languages, however the results presented here concern only the respective English tasks.

The *all-words* task consisted in sense-tagging Penn Treebank text using WordNet. The tagged words included all predicates, nouns which are heads of noun-phrase arguments to those predicates, adjectives modifying those nouns and adverbs. The texts used added up to 5000 running words, of which about 2000 were tagged. It is also important to mention that no training data is provided for this task besides what is already available publicly. The best system participating in this task scored 65.1% precision/recall.

The *lexical sample* task consisted of lexical sample style training and testing data in the Penn Tree-Bank corpus. At SensEval3 around 60 ambiguous nouns, adjectives, and verbs. WordNet was used as a sense inventory for nouns and adjectives, and WordSmyth for verbs. In this task training sets for each of the grammatical categories was provided, along with a mapping between WordSmyth and WordNet verb senses. The best system achieved 72,9% precision/recall.

# 3 Rule-Based Disambiguation

This chapter will present the rule-based approach taken on the verb sense disambiguation task. We will start by introducing STRING, which was used as a base system, and describe how the problem was modelled in Xerox Incremental Parser (XIP), one of STRING's modules. Then, in Section 3.2, we describe how the information provided by ViPEr is transformed into rules usable for the verb sense disambiguation task. This involved the creation of a rule generation module, described in that same section (Section 3.2), alongside some of the decisions made during its development. Finally, other problems found during the development are also discussed in Sections 3.2.5 and 3.3 of this chapter.

## 3.1 The STRING system

The STRING system (Mamede et al. 2012) serves as the base system for the development of this dissertation project. It already provides a functional NLP system capable of executing the major processing tasks in NLP. Figure 3.1 shows the system's architecture at the time this work was developed.

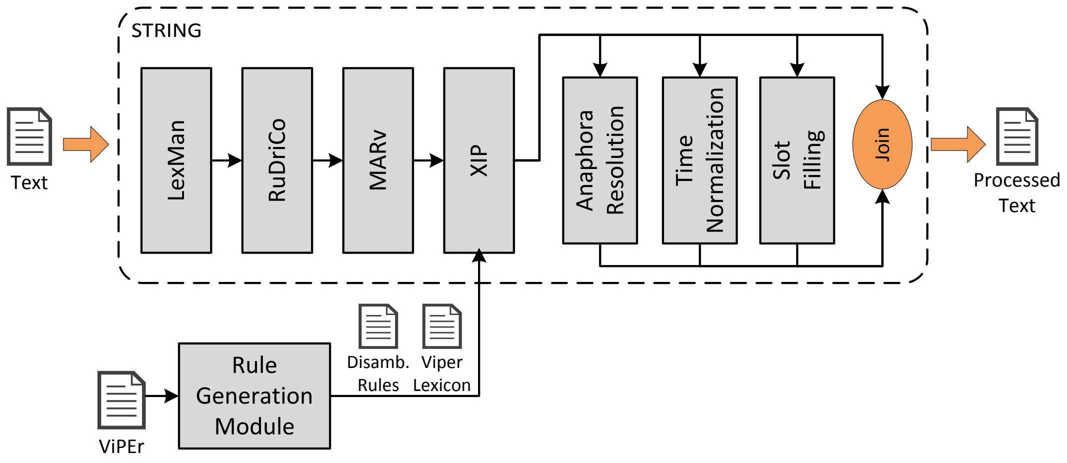

Figure 3.1: STRING Architecture with the new Rule Generation Module

Firstly, the lexical analyzer, LexMan (Vicente 2013), splits the input text into words and assigns the correct POS tag, as well as other morphosyntactic features such as the gender, number and tense.

LexMan is able to identify, among other, simple and compound words, abbreviations, emails, URLs, punctuation and other symbols. This module also splits the input into sentences.

Then, RuDriCo (Diniz 2010), a rule-based converter, executes a series of rules to solve contractions, and also identifies compounds words and joins them as a single token.

Before moving to the syntactic parsing, a statistical POS disambiguator (MARv) (Ribeiro 2003) is applied, choosing the most likely POS tag for each word. The classification model used by MARv is trained on a 250k words Portuguese corpus. This corpus contains texts from books, journals, magazines, among other, making it heterogeneous.

Next in the processing chain comes the XIP module, which is responsible for the syntactic parsing and dependency extraction. It will be described in more detail in Section 3.1.1.

Finally, after XIP, the post-processing modules are executed to solve specific tasks, such as anaphora resolution (Nobre 2011), time expressions normalization (Maurício 2011) and slot filling (Carapinha 2013).

### 3.1.1 XIP Module

XIP is a rule-base parser originally developed by Xerox (Ait-Mokhtar et al. 2002), and whose Portuguese grammars have been developed by L$^2$F in collaboration with Xerox. The XIP grammar uses a set of lexicon files to add syntactic and semantic features to the output of the previous modules. The XIP module parses the input text, dividing it into *chunks* (basic phrases like *NP* (noun phrase) or *PP* (prepositional phrase)). Chunking rules also identify the *head* of each chunk, which is used to build dependencies. Finally, dependency rules are applied to extract *dependencies* between the *heads* of chunks. These rules extract syntactic relations such as *subject* or *direct complement*, but they can also be used to create new n-ary dependencies representing named entities or time expressions, or to identify semantic roles and events. One other important aspect of this module is that rules do not have an explicit priority. Instead, priority is implicitly defined by the order the rules are presented in the several files that compose the dependency module of the XIP grammar.

#### 3.1.1.1 XIP Features

XIP uses features to represent some syntactic and semantic properties of words and nodes. For example, a word tagged with a POS tag of *noun* will have the corresponding feature in its node; or a person name will have the semantic trait *human*. Generally, every feature is binary, allowing only two possible values: present (true) or absent (false), however some features can take multiple values such as the *lemma* feature, which takes the lemma of the word as its value. Additionally, XIP must have all its features declared in the respective grammar file. Two major different types of features can be distinguished: *dependency* features and *node* features.

*Dependency features*, as the name suggests, can only be assigned to *dependencies*. An example of such features are the `completive` or `sentential` features, which denote completive and other sentential sub-clauses, respectively. These features can be assigned, for example, to a `CDIR` (direct complement) dependency. Sentence (3.1a) illustrates an example of such features.

(3.1a) *O Pedro disse que ia ao cinema*

```
CDIR_POST_SENTENTIAL(disse,ia)
```

*Node features*, on the other hand, are assignable only to nodes, and can convey different types of information. There are features related to the syntactic function of the node, for example the `cop` feature, that denotes a verb with a copulative function. More semantically related features, such as the `human` feature, can also be assigned to nodes. The example below shows a simplified version of the properties assigned by XIP to the words *Pedro* (Peter) and *está* (is), and their respective nodes (in capitals).

(3.2a) *O Pedro está feliz* (*Peter is happy*)

```
NOUNPedro[proper:+,people:+,individual:+,human:+,masc:+,sg:+,noun:+]
VERBestá[durativo:+,cop:+,pres:+,ind:+,3p:+,sg:+,verb:+]
```

The initial assignment of the various features to their corresponding nodes is done through the lexicon rules, which are the first set of rules to be executed by XIP. Then, during the other steps, features can be used as constraints, or new features can be assigned to nodes if certain conditions are satisfied (provided that they were declared in the feature file).

#### 3.1.1.2 Dependency Rules

XIP executes several types of rules, the last being the dependency rules. *Dependency rules*, as the name suggests, are responsible for extracting dependencies between the chunk heads, namely syntactic relations such as *subject* or *direct complement*. Besides these most basic relations, dependency rules are also used to build other relations that represent, for example, named entities or temporal expressions. The general format a dependency rule is illustrated below.

```
| PP#1[inquote:~]?*,#2[last] |
if (~HEAD(?,#1))
HEAD(#2,#1)
```

A dependency rule is composed of three parts: *structural conditions*, *dependency conditions* and *actions*, which are executed in that order.

The *structural conditions*, delimited by the vertical pipe character ('|'), are used to perform verifications such as the presence and value of certain features or the type of a node. Some variables can also be declared in this step, which are bound to the corresponding node, allowing it to be referenced in the dependency conditions. Naturally, if structural conditions of a rule are not satisfied, none of the remaining parts of the rule is executed. The example above declares one variable `#1` and makes sure it only binds to a `PP` chunk that it is not in quotes (`inquote` feature). A second variable `#2` is also declared, which will be bound to the last node (`last` feature) of the structure composing the PP node.

The *dependency conditions* are normally used to verify the existence of dependencies between the nodes bound to the variables previously declared and other nodes. Usually the verification of the dependency existence is followed by other constraints, such as the type of nodes allowed or the features required by a node in the dependency. In the example, the rule verifies that no `HEAD` dependency was yet extracted between the node `#1` and any other node.

*Actions*, as the name indicates, are the actions taken by the system after all the conditions are considered true. Actions consist in feature deletion, dependency creation or modification. Small actions, like feature deletion, can also be specified in the structural dependency part of the rule. Although the specification of these actions is different, they are still applied only if all the conditions, both structural and dependency, are verified. In this case, if all the previous restrictions are verified, the `HEAD` dependency is extracted between the node `#2` and node `#1` (node `#2` is the head of chunk `#1`).

Because dependency rules are the last to be executed by XIP, they were the type of rules used for the verb sense disambiguation task.

### 3.1.2 Problem Modelling in STRING

The approach to the verb disambiguation task was slightly different from the systems presented in Chapter 2. Instances start with all their possible classes, to which the automatically generated rules are applied, removing the wrong classes for that instance. This allows for partial disambiguation of some instances where some of the rules are not applicable, for example, due to absent verb arguments.

We modelled each verb class present in ViPEr as a XIP feature of type node so that they can be assigned to the corresponding verb nodes in sentences. The feature file used by the XIP grammar was manually augmented to include these new features. The lexicon file containing the initial assignments of the ViPEr classes to each of the lemmas is automatically produced by the rule generation system (see Section 3.2). Consequently, each verb will have a number of additional features corresponding to the number of classes it belongs to. Recalling the example of verb *apontar* (to point) (Section 2.1.4) and its ViPEr entries (Table 2.11), each sentence with that verb would have the corresponding verb node assigned four features: *36DT, 38LD, 39* and *9*, each corresponding to one of its possible meanings.

The usage of a lexicon file to automatically assign the ViPEr classes to all the verb instances with a certain lemma led to some wrong assignments, namely to auxiliary verbs, such as copulative, temporal, modal or aspectual verbs. To address this issue, the feature *markviper* was created. This new feature is not assigned using a lexicon file. Instead, some rules were manually crafted to assign it only to the correct verb instances (full verbs). Rules were also built to removed the wrongly assigned ViPEr classes from the auxiliary verb nodes.

Then, to disambiguate each verb instance, XIP dependency rules were used. These rules, contrary to the previous, were not crafted manually as the number would be too large to be humanly feasible and maintainable, considering the constant evolution of both ViPEr and the XIP Portuguese grammar. A rule generation module was developed for this purpose, which produces rules similar to the dependency rules already used in XIP. The example below shows a rule produced by the rule generation module.

```
|#1[markviper,lemma:abater,14,04,04=~]|
if( CDIR(#1,?[sem-cur]) | CDIR(#1,?[sem-currency]) | CDIR(#1,?[sem-mon]) )
~
```

The general structure of the rule is similar to the example presented in Section 3.1.1.2, with the major difference being that the action performed by the rule is only one: the deletion of a ViPEr class feature, denoted by `04=~`. The structural conditions assure that the rule is only applied to verbs with the `markviper` feature and with a certain lemma (`abater`). The instance must also have at least the two classes distinguished by that rule (`14,04`). Then, the dependency conditions verify the syntactic and selectional restrictions provided in ViPEr, which act as the major constraints for the verb addressed in that rule. In the example, the rule is only triggered if there is a direct complement (`CDIR`) dependency linking the verb (node `#1`) and another node that has the features `sem-cur, sem-currency` or `sem-mon`. This rule is thus able to disambiguate the two following instances of verb abater (depress/discount):

(3.3a) *A notícia abateu o Pedro* ($The\ news\ depressed\ Peter$)

(3.3b) *O Pedro abateu 20 euros ao preço da fruta* ($Peter\ discounted$ 20 $euros\ to\ the\ price\ of\ the\ fruit$)

## 3.2 Rule Generation Module

Figure 3.2 shows the internal architecture of the Rule Generation module. Each sub-module corresponds to a major step taken during the rule generation process. The first step, *parsing*, takes as its input the lexical resource information and produces a structure that is passed onto the following module. Second in the processing chain comes the *difference finder* module. This module is responsible for taking the result of the parsing step and comparing the features associated to each meaning of a polysemic verb. As a result, it produces a structure that represents the differences between those meanings. The last step, the *rule generation*, takes the differences produced by the previous step and transforms them into rules, which are then ordered and output by the system. The disambiguation rules and lexicon are then added to the XIP Portuguese grammar and used by STRING.

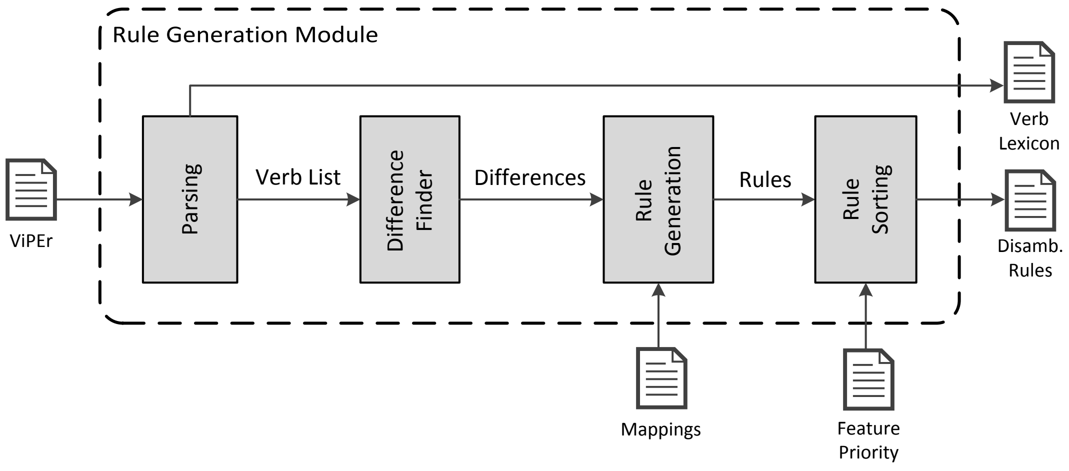

Figure 3.2: Rule Generation Module Architecture

In the following sections, each sub-module is described in more detail. The errors and warnings reported by each sub-module are covered in a dedicated section (see Section 3.2.6). A full example will also be presented throughout the sections to illustrate each step. The examples will use the verb *abanar*, whose ViPEr entries are shown in a simplified form in Table 3.1.

| Class | N0 | Prep1 | N1 | Prep2 | N2 | Pass-ser | Pass-estar | vse | vdic |
|---|---|---|---|---|---|---|---|---|---|
| 32C | Hum, Nnr | - | nHum | - | - | + | - | - | - |
| 32CL | Hum | - | sem-an | - | - | - | - | - | - |

Table 3.1: *Abanar* ViPEr entries

### 3.2.1 Features

Before building the rule generation system it was necessary to define which of the features available in ViPEr should be considered during the parsing and consequently used in rule generation. The features presented in this section correspond to the final version of the rule generation system, however they were integrated in the system incrementally after several steps of evaluation (see Section 5.3).

The base features considered by the system are the selectional restrictions on the verb's arguments (*N0* to *N3*) and their respective prepositions, along with the property (*vse*), denoting intrinsically reflexive constructions.

The set of features was then augmented with a new property, *vdic*, which had to be created in ViPEr. This new feature indicates which verb meanings (usually only one per lemma) allow the *verbum dicendi* construction pattern.

Afterwards, the *a Ni=lhe* feature was systematically annotated and added to the set of features used by the system. This property indicates the possibility of a dative pronominalization of some of the verb arguments, for which special rules had to be generated.

Some transformational properties regarding passive constructions were integrated in the system, leading to rules that capture sentences in the passive voice built with the auxiliary verbs *ser* and *estar*.

### 3.2.2 Parsing

As the first step in the rule generation process, the parsing module starts by processing the ViPEr file and building a structure that represents each verb as a set of meanings. In turn, each meaning is represented as a collection of features, described in ViPEr, and their possible values. The attributes considered as features during the parsing step correspond to the different arguments a verb can select, noted in ViPEr as *N0* to *N3* and their corresponding prepositions, *Prep1* to *Prep3*. Besides the basic distributional attributes, other features were also considered during this step, namely the intrinsically reflexive property (*vse*) and the dative pronounning (*aNi=lhe*).

The parsing module is also responsible for producing the lexicon file used in the XIP grammar (see Section 3.1.1), which contains information about the lemmas and each of the possible classes that each lemma can belong to. This enables the system to be regularly updated at the same time as both ViPEr and XIP evolve.

### 3.2.3 Difference Generation

After the parsing is complete, the system passes the processed information to the next module: the *Difference Finder* module. Here, each verb is visited and the differences among the meanings of the verb are generated, which are then used to make the disambiguation rules.

During the development of this particular module, several decisions had to be made regarding the difference generation process, which directly influenced the output of the system. In the paragraphs below we present the available options alongside their positive and negative aspects, followed by the decision made for each option.

The first major decision that had to be made concerned rule generation for verbs with more than two different meanings, since for verbs with two meanings, the comparison was straightforward. To address this problem, two solutions were considered:

1. Generating a difference by doing one meaning against all meanings for each lemma (requires finding a unique set of differences for each verb meaning)
2. Generating a difference between pairs of meanings within the same lemma (requires combinatorial generation for verbs with three or more meanings)

The most important benefit provided by the first approach is the small number of rules that are generated. By doing a one-against-all strategy, the number of differences and, consequently, of rules thus produced are directly proportional to the number of meanings a verb has. This gets more relevant the higher the number of meanings per verb becomes. However, the major problem with this approach is finding a set values for a certain argument that is unique to one meaning, which might not always be possible. Even when it is possible to find such a unique set of values, only a reduced number argument positions will be able to satisfy that restriction. This leads to rules with very specific constraints, which may not always be applicable. For example, if a verb argument is absent, a very common phenomenon in real texts, and if the rule tests that particular argument, then the rule would not be applied and the system would not be able to disambiguate that instance at all.

Regarding the second approach, the most positive aspect is the flexibility it offers, because it does not consider all the meanings in every difference/rule. This approach compares verb meanings pairwise, so that each difference and rule only addresses two meanings at a time. This eliminates the need to find an argument with a unique set of value across all the remaining meanings, giving the possibility to distinguish each pair of meanings using different properties. Consequently, with this approach, verbs can be partially disambiguated. Considering the same example as before, if some verb argument is absent, rules that tested the absent argument would still not be applied, however, rules that tested other arguments could still be applied. This potentially reduces the number of meanings assignable to that instance even if the disambiguation is not completely achieved. On the other hand, this approach requires a higher number of differences and rules to be generated in order to cover all the possible combinations of meanings, which leads the number of differences and rules generated for each verb to be much higher than the number of meanings the verb has.

Decision was made to follow the second method, as it offered more versatility by allowing a partial disambiguation of verbs when verb arguments are absent, as it often occurs in real texts. Still, this approach is able to fully disambiguate sentences for the cases where the first method would be applicable, i.e., where no arguments are absent.

After deciding how to compare the verb meanings for the same lemma, another major decision influenced the system's output: which attributes should be taken into account when generating the differences? Again, two solutions could be envisaged:

1. Generating a difference with all the different attributes between the pair of meanings
2. Generating a difference for each argument found different between the pair of meanings

This problem was relatively similar to the previous one. Choosing to generate a difference with all attributes (first scenario) would be a way of reducing the number of rules generated, as each rule would test all the attributes contained in the difference. However, this would enforce a very rigid structure in the sentence that could easily make this type of rules not to be applied due to very common phenomena such as absent verb arguments or sentence constituent's reordering, resulting in no disambiguation.

In contrast, adopting the second approach would further increase the number of rules generated, but again, would allow for further partial disambiguation and it also would deal with the phenomena mentioned above. So, once more, the second option was adopted, as it was more likely to reduce the ambiguity in a sentence, even if only partially, because there would be a larger set of differences (and rules) concerning the same pair of verb meanings, but each verb argument restriction would be verified independently from the rest.

As a result both decisions, the system ended up producing a number of rules that was much higher than the number of meanings each verb had. This resulted in some overhead in XIP by the exponential number of rules used. However, the impact was minimized using some auxiliary features and putting more constraints in the rule header, as explained in more detail in Section 3.2.4.

Regarding the example of verb *abanar*, the following differences are generated: `Diff-N1` and `Diff-Pass-ser`, corresponding to the positions in which differences can be found. A `Diff-N0` is not generated because of the presence of the *Nnr* (non-restricted noun) feature. This indicates that almost any noun can occur in this position (subject), for the distributional constraints of this position are weak (corresponding to a "cause" semantic role) and therefore it should not be used for disambiguation purposes.

Moreover, each difference does not store only the different values for the corresponding feature but the entire verb meanings structures regarding in that difference. The main reason for this concerns the rule generation involving verb arguments introduced by prepositions in their conditions, which will be explained in more detail in Section 3.2.4.

### 3.2.4 Rule Generation Module

The rule generation module is the last step in the rule generation process. Here, differences are transformed into rules, with each difference usually generating two rules, one for each meaning encapsulated in that difference.

The process starts by getting the possible values for the verb argument regarding the type of difference being processed. After getting each set of values (one for each meanings in that difference), the common values in both sets are removed, resulting in two sets that have no overlapping values. Then, based on these remaining values, the rules are generated.

However, for differences regarding verb arguments that are introduced by prepositions, additional information about their respective prepositions is also added to the rule. This information is added because the verb argument will map onto the XIP dependency `MOD`, which is a very generic dependency[1]. The additional preposition constraints prevent the rule from being applied in wrong scenarios. Since the verb argument can often be introduced by more than one preposition, these rules require a combination of the values from the verb argument with the respective preposition values. The example below illustrates two examples of rules, without and with prepositional restrictions, respectively.

```
|#1[markviper,lemma:abalar,04,35LD,35LD=~]|
if( CDIR(#1,?) )
~

|#1[markviper,lemma:abalar,35LD,04,04=~]|
if( (MOD(#1,#2) & PREPD(#2,#3[preploc])) |
    (MOD(#1,#2) & PREPD(#2,#3[lemma:"de"])) |
    (MOD(#1,#2) & PREPD(#2,#3[lemma:"para"])) )
~
```

The rule generation process, however, is not totally straightforward, as the semantic restriction values for the ViPEr positions do not match directly the features provided by XIP (see Section 3.1.1.1). Further increasing the problem is the fact that one ViPEr value can map onto multiple XIP features, and each XIP feature can be a *dependency* or *node* feature type (Section 3.1.1.1). In fact, many of the concepts used in ViPEr map to several features in XIP, as this last one uses more fine-grained features. For example, the *Hum* concept used in ViPEr has a broader scope than the *human* feature in XIP. The

[1]In the XIP Portuguese grammar, the `MOD[ifier]` dependency links a constituent to the verb (or other constituent). Only essential arguments are represented by the `COMPL[ement]` dependency, which requires Verb Sense Disambiguation (VSD). Once the verb has been disambiguated, valid arguments of that construction are transformed from `MOD` to `COMPL`. In general, only direct (`CDIR`) and indirect (`CINDIR`) complements are identified prior to VSD.

first includes human individuals, collective organizations, institutions, among other, while the latter is only used for human individuals.

To solve this problem, an additional configuration file is used: *mappings*. This solution provides flexibility to the system, by having, in a declarative way, the correspondences of the lexical resource properties and the NLP system features. This allows for the independent development of both ViPEr and STRING. Additionally, it also enables non-computer scientists collaborators of Instituto de Engenharia de Sistemas e Computadores - Investigação e Desenvolvimento (INESC-ID) Lisboa to easily augment the rules in order to consider a newly implemented feature.

| ViPEr Value | Mapping Type | XIP Feature |
|---|---|---|
| n0 | dependency | SUBJ |
| direct-transitive-n1 | dependency | CDIR |
| indirect-transitive-n1 | dependency | MOD |
| Hum | node-feature | UMB-Hum |
| nHum | node-feature | ˜ Hum |
| QueF | dependency-feature | sentential,completiv,inf |
| Npl | node-feature | pl,sem-group-of-things |

Table 3.2: Mappings File Example

The *mappings* file is a text file with three columns of properties separated by tabs, as exemplified in Table 3.2. The first column (leftmost), corresponds to the ViPEr selectional restrictions' value found in the different verb arguments. The middle column indicates whether that property should be tested in a *dependency* (*dependency-feature*) or in a *node* (*node-feature*). It is important to confuse *dependency-feature* with the mapping type *dependency*, the latter indicates that a mapping corresponds to a dependency in XIP and not a feature to be verified. Finally, in the rightmost column, the corresponding XIP value is listed (or values, separated by a comma, in the case of multiple mappings). For the non-human (nhum) feature present in ViPEr, a special mapping was required, as there was no equivalent in XIP. Since non-human is the opposite of the human feature, it was decided that *nHum* would be true if all the human related features were absent, and false otherwise.

The end result, that is, the rules produced after the execution of this module regarding the example of verb *abanar*, are presented below.

```
|#1[markviper,lemma:abanar,32C,32CL,32CL=~]|
if( VLINK(#2[cop,lemma:"ser"],#1[pastpart]) )
~

|#1[markviper,lemma:abanar,32CL,32C,32C=~]|
if( CDIR(#1,?[UMB-Anatomical]) )
~

|#1[markviper,lemma:abanar,32C,32CL,32CL=~]|
if( CDIR(#1,?[UMB-Human:~]) )
~
```

### 3.2.5 Rules Priority

After testing the rules several times, it became evident the need for an additional sorting step before the system prints out the disambiguation rules. The need for this new processing step is directly related to the mapping of the *nHum* feature (see Section 3.2.4 and the way rule priority is implemented in XIP, in Section 3.1.1.

To better illustrate the problem, consider the example sentence (3.4a) and two rules produced for two possible meanings of the verb *abanar* (to shake a drink/to wag the tail). The first rule tests the *nHum* (non-human) feature and the second the *sem-an* (anaotmical body-part) feature.

(3.4a) *O cão abanou a cauda* (*The dog wagged its tail*)

```
|#1[markviper,lemma:abanar,32C,32CL,32CL=~]|
if( CDIR(#1,?[UMB-Human:~]) )
~

|#1[markviper,lemma:abanar,32CL,32C,32C=~]|
if( CDIR(#1,?[UMB-Anatomical]) )
~
```

When generating these rules, if they were not reordered, it would be impossible for the system to correctly guess the class of the verb in that sentence, as the word *cauda* (tail) has the *part-of-the-body* semantic feature and it does not have the *human* feature. This results in a validation of the first rule and the incorrect elimination of the class *32CL*. However, if we give a higher priority to the semantic features and a lower one to the *nHum* property, and then rearrange the rules accordingly (hence the order of these two rules order would be reversed), the system would then be able to guess the correct class.

After studying the ViPEr properties and their mappings, an order was established. The main criteria used for ordering the features and, consequently, the rules they generate, was based on the impact/strength of a restriction in determining the correct class (similar to the information gain concept used in decision tree algorithms). In the ordering step, after rule generation, the system started using a new input file, containing the priorities established.

The first properties to be tested are related to structural patterns in sentences, followed by the several semantic selectional restrictions. The order must have the structural patterns first, because the annotations present in ViPEr are only valid for the normal active form of the verb. The *vdic* feature is the first one to be tested because it corresponds to a very specific type of construction (*dicendi* verbs), and usually, only one verb meaning per lemma allows that type of construction.

Then, the passive constructions are tested, because they have the verb arguments rearranged in a different order, with the direct complement of the active sentence taking the subject's position.

The list of priorities goes on with dative pronouns and reflexive pronouns. The dative case has to be tested before the reflexive because in the latter one, the absence of a pronoun with a reflexive trait leads to the elimination of the intrinsically reflexive classes. This would create a similar problem to the *nHum* problem presented above. To avoid having the dative pronouns fall under that testing (the absence of a clitic with the reflexive feature), they were prioritized over the reflexive.

After testing the structural and other specific properties, all the other selectional restrictions are tested. The priorities for these start with semantic features that occupy the column *sem-X* on ViPEr. These are usually very strong restrictions, such as body-parts, currency, or even word lemmas or surface restrictions, thus they are the first of the selectional restriction to be tested.

After the *sem-X* restrictions the *Nloc* (locative nouns) are tested, followed by the *Npl* (plurals and collectives), *Hum* (Human) and *QueF* (completive and sentential sentences).

Before testing the *nHum*, there was the need to create a special priority for cases where only the preposition is different, in which the rules do not impose any restriction on the argument. This is also tested before the *nHum* property to avoid the incorrect triggering of the rules.

The last semantic selectional restrictions to be tested is the *nHum*. As already exemplified before, the mapping of the *nHum* proved to be a problem, so it was considered best to have this feature as the last one to be tested. In this way, the probability of having a rule wrongly triggered due to the *nHum* feature is minimized.

### 3.2.6 Error Reporting

One side objective that became increasingly relevant as the project developed, was the help it could provide to the development of both ViPEr and STRING. As a consequence of the crescent awareness of its importance, several error detection and reporting functions were developed in each of the previously presented modules.

For the first step (*Parsing*), a *Parsing Validator* was developed. As the name suggests, this module is responsible for doing some validations of the data provided by ViPEr. This module has two main objectives: to avoid errors in the subsequent modules; and to aid in the development of the lexical resource (ViPEr). This module outputs several errors related to the parsing, for example missing annotations in some features or duplicate entries.

Besides the error validation module, the team developing ViPEr was also interested in statistical information about the lexical resource, therefore, a *Statistics Generator* module was also developed. This module goes through the structure obtained from parsing ViPEr and collects data regarding the distribution of lemmas, ViPEr classes and number of meanings. After its execution the collected information is made available in a text file.

In the second step (*Difference Finding*), some error/debug information is also generated in order to help the development of ViPEr. For example, when two different meanings for the same verb (lemma) have no apparent differences, but were expected to have, the module generates a warning message that is stored. In the end, all the warnings are reported for a possible revision in ViPEr, as there might be a mistake in the lexical resource annotation.

In the last step of the system (*Rule Generation*), the error/debug information generated concerns the absence of needed mappings. During the rule generation step, if a mapping is required to build some rule but it is missing in the *mappings* file, instead of generating the rule, a warning is reported. These warnings may signal one of three possible situations:

- Features already implemented by XIP, but for some reason are missing in the mappings file;
- Features that are not yet implemented in XIP, so, they are in fact missing;
- Incorrect annotation of some properties in ViPEr.

This particular step, again, aims to help the development of ViPEr, by reporting possible annotation errors. However, in this case it also aids in the expansion and improvement of the STRING system, by reporting features needed for the disambiguation rules but that are not yet implemented in XIP.

## 3.3 Verb Meaning Filtering

Analysing the results of the rule-based approach showed that the verb meaning distribution of a lemma was biased towards one of its meanings. Doing a full counting on verb occurrences and their classes revealed that some classes never appeared in the corpus. Sentences (3.5a) and (3.5b) illustrate the example of verb *dizer* (say/match), with around 850 instances, which is never used in a *35S* manner.

(3.5a) *O Pedro disse que ia ao cinema* (dizer/say, class: 09)

$(Peter\ said\ he\ was\ going\ to\ the\ cinema)$

(3.5b) *Os sapatos dizem com a camisa* (dizer/match, class: 35S)

$(The\ shoes\ match\ with\ the\ shirt)$

In many cases verbs were tagged with one of these non-frequent classes because the sentence structure was too complicated, resulting in a wrong syntactic analysis by XIP, which consequently lead the rules to a wrong disambiguation. Considering again the example of verb dizer, in the first evaluation, from the 839 instances, only 559 (66.63%) were correctly disambiguated.

This suggested that the original problem could be simplified by ignoring some of the verb meanings during the rule generation process, as they were not likely to occur and only added more complexity to the task when processing real texts. This type of approach was already been adopted in other parts of the STRING system, namely in the part-of-speech tagging, with the main objective of maximizing the system's accuracy in real text processing, the main purpose of the system.

However, the infrequent classes were not eliminated from ViPEr, maintaining in the lexical resource all the possible constructions for each lemma. Instead, a new property was added to sign that it could be useful to filter a certain, non-frequent, verb meaning. This allows the the rule generation system to use, or remove, the filter whenever required, by simply considering or discarding that property value.

The assignment of the filtering property was not done automatically but based on a list of verb meaning classes that was built based on the verb meaning distribution collected from the corpus. This list considered all verbs with at least 50 instances, whose classes did not reach a threshold of at least 2% of the verbs's total instances (see Table 3.3). This list was then reviewed by the linguists that build the lexical resource and the information was added.

| **Lemma** | **Total Instances** | **Verb Meaning** | **Meaning Occurrences** |
|---:|---:|---:|---:|
| dizer | 849 | 35S | 0 |
| encontrar | 131 | 32R | 0 |
| entrar | 103 | 40 | 2 |
| falar | 207 | 35LD | 0 |
| falar | 207 | 35R | 2 |
| falar | 207 | 42S | 0 |

Table 3.3: Low occurrence verb meanings sample

One of the major downsides of this approach is the possible loss of some precision for the system, as it stops considering some classes as possible classifications for a certain verb lemma. However, considering them would impact the disambiguation in a lot more instances. The trade-off obtained for the the filter list presented above was considered acceptable.

# 4 Machine Learning Disambiguation

The previous chapter presented the rule-based approach adopted for the verb sense disambiguation task. However, given the restrictive nature of rule-based approaches, a complementary module, based on machine learning, was developed. This chapter will explain how the module was implemented, followed by a description of the training corpus. Finally, the features used to describe the instances will be presented.

## 4.1 Architecture

Figure 4.1 illustrates a typical architecture used for supervised classification, which is divided into two major steps: training and prediction, and composed of three major modules: the feature extraction, the machine learning algorithm and the classification module (other modules like pre-processing may also be included before the feature extraction).

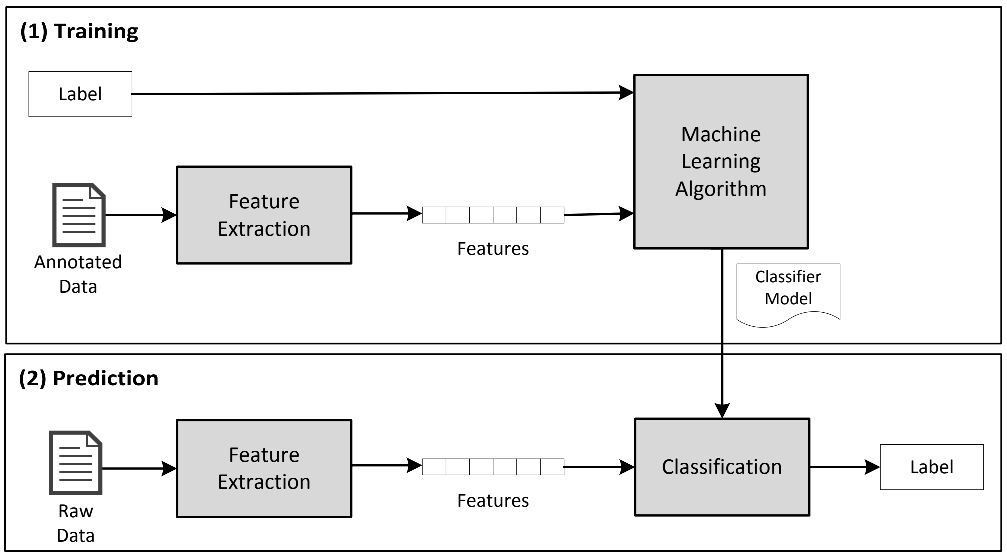

Figure 4.1: Typical Supervised Machine Learning Architecture (adapted from (Bird et al. 2009))

During the training phase, the feature extraction module is responsible for transforming the raw data into a set of features used to describe that data. These features are then passed onto the machine learning algorithm, alongside their labels, to build a model. Usually, this step is performed offline.

In the prediction phase, the same feature extraction module is used on the unlabelled data to get the respective features. These features are then passed to the classification module, which uses the previously built model to predict the label for the new data instances.

In the implementation of a supervised classification method for this project, one major requirement was that the classification module stayed inside XIP, so that rules could be built afterwards for identification of verb complements, semantic role labelling, among other NLP tasks. To address this issue, both feature extraction and classification modules had to be implemented inside the XIP module, that is, inside STRING, using the KiF language, developed by Xerox. The machine learning algorithm module was not implemented from scratch, like the previous two modules, as the implementation of machine learning algorithms was out the scope of this thesis. Instead, an existing package, MegaM (Daumé 2004), based on maximum entropy models (Berger et al. 1996), was used. The architecture of the implemented supervised classification approach is presented in Figure 4.2.

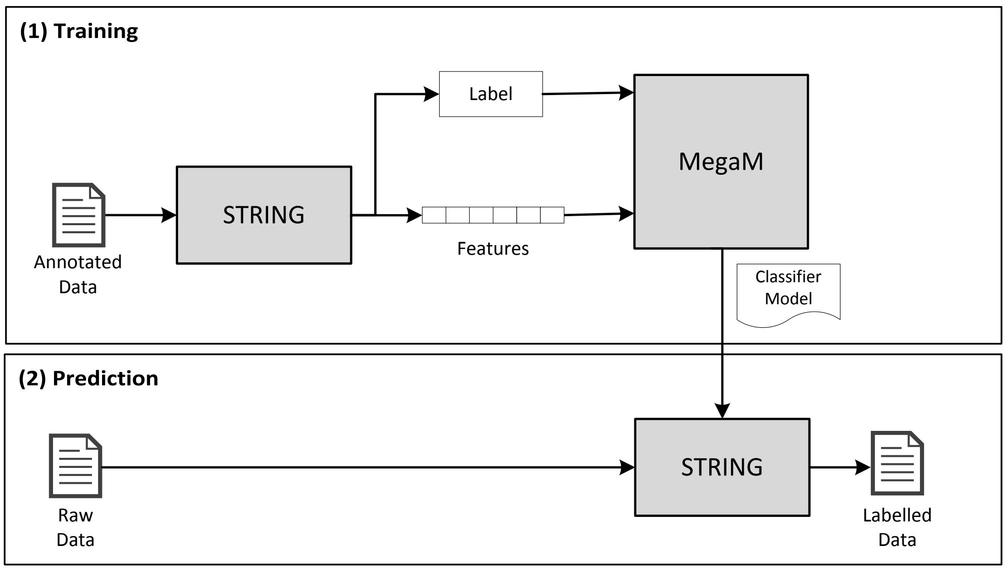

Figure 4.2: Supervised Machine Learning Architecture for VSD using STRING

In the training step, the feature extraction module inside STRING is responsible for extracting the features that are to be used to describe the target instances. The features are then passed, along with their respective reference class, to MegaM. This ML package builds a model, which is used by the classification module (inside STRING) in the prediction phase. The features used to describe each instance will be covered in Section 4.3.

During the prediction phase, the same feature extraction step obtains the features used to describe the instances to be classified. Then, the classification module loads the respective verb lemma model (according to the instance being classified) and computes the confidence value (probability) for all the possible verb classes for that instance. The highest confidence/probability value determines the class assigned to that verb instance. The formula used to determine the probability of each class $c$ given an instance $i$, denoted by $P(c|i)$ and presented below, is the standard used for maximum entropy models (Berger et al. 1996), where each $f_k(c,i)$ is a feature describing instance $i$ for class $c$; $\lambda_k$ is the parameter (weight) estimated in the model for that feature and $Z(i)$ is the normalizing factor used to obtain proper probability values.

$$P(c|i) = \frac{1}{Z(i)} exp\left(\sum_k \lambda_k f_k(c,i)\right)$$

$$Z(i) = \sum_c exp\left(\sum_k \lambda_k f_k(c,i)\right)$$

Another concern with the machine learning approach consisted in the number of classifiers that had to be produced, which was directly related to the amount of corpora that had to be collected and annotated.

To address this issue, the ambiguous verbs of ViPEr were studied. Noticing that some verbs shared the same set of classes, this suggested that classifiers could be produced to disambiguate between a set of classes, despite the lemma, leading to a fewer number of classifiers (that is, fewer than having one classifier per lemma). This situation would also ease the task of corpora collecting and annotation.

However, after analysing the respective restrictions for verbs showing the same ambiguity class set, which were likely to be used as features, it became apparent that this approach would not be adequate, as some verbs, belonging to the same ambiguity class set, had completely different semantic restrictions over the same argumental position. Therefore, a strategy similar to the one adopted in the rule-based disambiguation had to be adopted, building one classifier per verb lemma. The major downside of this choice is that the number of verbs targeted by the machine learning approach in this project had to be reduced, due to time required for collecting and manually annotating corpora for each lemma. Nevertheless, a careful selection of verbs was undertaken, which is explained in the next section (Section 4.2).

## 4.2 Training Corpus

As mentioned in Section 4.1, the approach adopted for the machine learning module will consider building a classifier per lemma. Therefore, only some of the verbs would be addressed by this approach in this project. The lemmas chosen to be disambiguated by the machine learning technique were: *explicar*, *falar*, *ler*, *pensar*, *resolver*, *saber* and *ver*. The main reason for choosing these verbs over the rest was the higher number of instances left to disambiguate that these lemmas exhibited after the rule-based testing. Further analysis revealed that these verbs also formed an heterogeneous group, with different MFS percentage values, different number of senses and different ambiguity classes (see Table 4.1).

| Verb | Instances | Instances Left to Disambiguate | MFS Accuracy | Classes |
|---|---|---|---|---|
| ver | 451 | 282 | 54.77 | 06,32C |
| saber | 481 | 264 | 95.84 | 06,33,34 |
| pensar | 165 | 121 | 64.85 | 06,08,16,32C,35R,36R |
| resolver | 95 | 89 | 66.32 | 06,32C |
| falar | 206 | 83 | 97.09 | 32R,35LD,35R,41,42S |
| explicar | 134 | 76 | 94.78 | 01T,09 |
| ler | 77 | 69 | 96.10 | 06,32C,32R,36DT |

Table 4.1: Machine Learning Verbs

Since the only annotated corpus available was the evaluation corpus, which did not have enough instances for all the verbs chosen, a new training corpus had to be collected and the verb instances manually annotated by linguists with their respective ViPEr class. The number of instances collected per lemma varied a lot depending on the verb frequency, however, all of the verb had at least around 7500 instances. The instances collected consist mainly from journal articles from the CETEMPúblico corpus.

From the total number of instances collected, sentences containing several word forms of the same verb lemma were filtered out. The main reason for this filtering was to facilitate the training step. The intuition backing this option is that, usually, sentences with multiple instances of the same verb are relatively infrequent, and also, that those complex sentences can be analysed into simpler, distinct sentences. Because of the verb repetition, similar verb arguments are often zeroed, thus reducing the features available for ML. Thus, the machine learning system would still learn from the other simpler sentences and be able to disambiguate the more complex ones. The average number of instances filtered corresponded to about 10% of the total instances collected.

After filtering the instances, they were split into partitions of 50 examples, each encompassing all word forms found for that lemma. These sets were then handed to a team of linguists, who manually annotated 500 examples of each lemma (10 partitions)[1]. During the annotation process the linguists also filtered some unwanted instances, mostly regarding three different scenarios: (i) the word form of the verb being classified in an instance was in fact an English word that shares the same word form;

[1]At the time of evaluation, only 250 instances were available for the verbs *pensar* and *saber*.

(ii) instances that consisted mainly of titles of creative work such as books, songs, paintings, among others, were also excluded; and (iii), the instances whose size was too small, thus not providing enough contextual information to accurately determine the verb meaning for that particular case, were also eliminated.

## 4.3 Features

The features to be used by the machine learning system follow the commonly used features presented in Chapter 2. These features can be organized into three groups, as follows:

- *Local features*, also called *contextual features*, describe the information around the target word (the verb). In the system, the context is extracted in a window of size 3 around the target verb, that is, a total of 6 tokens are used, with their respective indexes (-3, -2, -1, +1, +2, +3). The information collected about each of the tokens was the POS tag and lemma[2].

- *Syntactic features*, regarding the constituents directly depending on the verb were also used, that is, constituents that had a direct relation (i.e. XIP dependency relation) with the verb. The POS tag and lemma of the head word of each node in these relations were extracted, together with the respective dependency name. Though several other dependencies/relations are implemented in the XIP grammar, only those of *SUBJ*, *CDIR* and *MOD* were considered for the ML system.

- *Semantic features* were extracted for the head words of the nodes that had a direct relation with the verb, as these act mostly as selectional restrictions for verb arguments. The number of semantic features extracted from a single node varies, as different words can have multiple semantic traits. Each semantic trait found in the head word of the node is also accompanied by the relation name (subject, direct complement or modifier) the node had with the verb. The semantic features considered by the system are those that are also present in ViPEr, for example, human, location, body-part, animal, plant, currency, among others.

  A reverse mapping to the one performed for rule generation had to be done when extracting the semantic features. This aimed at converging the different implementations that XIP had for a semantic restriction under the same unique semantic value. For example, the semantic feature location (*Nloc* in ViPEr) is implemented using three different features in XIP (`geo`, `geo-toponym`, `umb-geo`). However, the machine learning algorithm should learn based on the fact that all three restrictions concern the same semantic restriction value (*location*), hence, the requirement for this reverse mapping.

[2]Experiments with a different window size and additional information regarding the context tokens were also performed, though none of them provided better results.

Some additional features that do no fit exactly into any of those categories were also used. These features concern the concept of *verba dicendi* construction and information regarding clitic case. The reflexive clitic case may indicate an intrinsically reflexive construction, while the dative case is used in pronominalization of indirect complements under the form of a dative pronoun. Both the *verba dicendi* and the the clitic case (dative and reflexive) features exhibited some positive impact on the results of the rule-based disambiguation system (see Section 5.3), therefore it could also be useful to include this type of information in a machine learning based approach.

# 5 Evaluation

In the previous two chapters, the two different approaches adopted for the verb sense disambiguation task were described. In this chapter, the evaluation scenarios considered for the different techniques, followed by the results obtained in each case are presented. At the end of each section, an analysis of the results achieved with these approaches is made, and some conclusions that can be drawn from these experiments are presented.

## 5.1 Evaluation Methodology

### 5.1.1 Corpus

To evaluate the two approaches presented in the previous chapters a manually annotated corpus had to be built and validated. The corpus chosen for evaluation was the one used to train the statistical POS tagger (see Section 3.1), which contains around 250k words. Each verb on the corpus had to be manually annotated and then reviewed by linguists with the aid of a validation software developed by the STRING team. Although the corpus contained around 38,827 verbs, only 21,368 (about 55%) of those verbs correspond to full verbs, as showed in Table 5.1. The full verbs distribution according to their number of meanings is presented in Table 5.2.

| | **Total** | **Full Verbs** | **Auxiliary Verbs** |
|---|---|---|---|
| **Count** | 38,827 | 21,368 | 17,459 |
| **%** | 100 | 55.03 | 44.97 |

Table 5.1: Evaluation Corpus Verb Occurrences

Before evaluation, a preliminary processing was performed to determine the amount of errors on ambiguous verbs, consequence of previous modules. Table 5.3 shows the number of errors found, divided by their type. The POS taggger wrongly tagged 94 (0.72%) verb instances and incorrectly assigned 10 (0.08%) of the lemmas, while the syntactic chunker did not recognize as full verb constructions 735 (5.64%) instances. This leads to a total of 839 (6.44%) of the instances not being classified by any of the methods presented, thus, there is a ceiling of 93.56% in the results.

| Meanings | Count | % |
|---|---|---|
| 1 | 8,338 | 39.02 |
| 2 | 6,404 | 29.97 |
| 3 | 3,352 | 15.69 |
| 4 | 1,590 | 7.44 |
| 5 | 900 | 4.21 |
| 6 | 183 | 0.86 |
| 7 | 118 | 0.55 |
| 8 | 182 | 0.85 |
| 10 | 118 | 0.55 |
| 11 | 183 | 0.86 |
| **Total** | 21,368 | 100 |
| **Total Ambiguous** | 13,030 | 60.98 |

Table 5.2: Evaluation Corpus Full Verbs Distribution

| | Total | Processed | Wrong POS | Wrong Lemma | Not identified as a full verb |
|---|---|---|---|---|---|
| **Count** | 13,030 | 12,191 | 94 | 10 | 735 |
| % | 100 | 93.56 | 0.72 | 0.08 | 5.64 |

Table 5.3: Corpus Processing Results

### 5.1.2 Measures

The goal is to measure the correctness of the results produced by the system against a reference value, that is, how many verbs were correctly classified by the system, among all the verbs that were hand-tagged in the corpus. This fits the definition of *accuracy*. The formula used for accuracy is presented below, with $n_c$ being the number of correctly disambiguated instances and $N$ the total ambiguous verb instances in the evaluation corpus processable by the system.

$$Accuracy = \frac{n_c}{N}$$

## 5.2 Baseline

Generally, a baseline is a starting point for comparison of a system's performance. For the evaluation of the VSD task, we adopted as baseline the results of using the most frequent sense (MFS) to decide the correct verb sense.

This approach can be seen as a very simple classifier based only on counting. In the training step, the system counts the occurrences of each sense for every lemma. Then, in the prediction step, it assigns the most frequent sense to every instance of that lemma. This technique does not use any type of contextual information provided by the sentence of the verb instance being classified; it simply uses counting to determine the MFS and then always classifies every instance of the same lemma with the same class.

Since there is not many annotated data using the ViPEr classification, the evaluation corpus had to be used for both training and prediction step. To ensure the results were not biased, the common 10-fold cross-validation method was used, thus the number of instances used in evaluation correspond to 1/10th of the original number of processed instances.

| | **Instances** | **Correctly Disambiguated** | **Wrongly Disambiguated** |
|---|---|---|---|
| **Average** | 1,219 | 1,024 | 195 |
| % | 100 | 84.00 | 16.00 |

Table 5.4: MFS Accuracy

The results for this experiment, presented in Table 5.4, were surprisingly higher than it was initially expected. This can be explained by the low number of classes used in the ViPEr classification when compared to other lexical resources. To better understand these results, information about the MFS of each verb lemma and its frequency was collected. As illustrated in Figure 5.1, around 25% of the corpus verbs have a MFS of 100% of their instances, and only about 15% of the corpus verbs have a MFS of 50% of less. With this information, it became evident that any initial assumptions that the verbs would have a balanced distribution of their meanings would not be adequate.

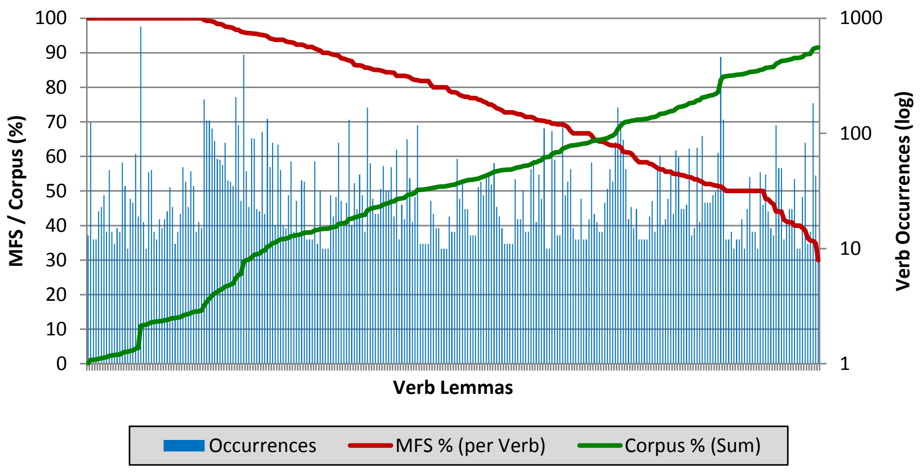

Figure 5.1: MFS distribution on evaluation corpus (verbs sorted by MFS). Only verbs with frequency 10 or higher were considered (278 (48%) of total lemmas and 11,905 (91.5%) of total instances)

## 5.3 Rule-based VSD Experiments

In the following subsections, the different testing scenarios used for the rule-based approach to VSD will be presented. Each of the scenarios was aimed to test the impact of certain features used by the rule generation module. These experiments were done iteratively and incrementally, which means that every change resulting from an experiment was included in the subsequent tests. Note that in these experiments the number of processed instances is lower than the value presented before, because the version of STRING used at the time, naturally, did not include the most recent developments.

### 5.3.1 Standard Rules

The first testing scenario used only the verbal selectional restrictions on their arguments as conditions in the disambiguation rules, which corresponds to the first set of features considered by the rule generation module during its development, as presented in Section 3.2.1. Using these features, the system generated a total of 7,328 rules and achieved the results presented in Table 5.5.

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 5,967 | 4,594 | 2,490 | 3,716 | 5,089 |
| % | 100 | 49.02 | 37.74 | 20.46 | 30.53 | 41.81 |

Table 5.5: Disambiguation rule's results

These first results present a high error rate, with around 20.5% of instances being incorrectly classified. Though rules addressed almost half the instances of the corpus (49.02%), with the majority (37.74%) being fully disambiguated just by this method. Still, a great portion of the instances were left untouched by the rules or only partially disambiguated (41.81%). Finer-grained results revealed that the class and the verb lemma with more incorrect classifications were the class *09*, with 321 instances from the 1938 instances marked with *09* in the corpus, and verb *dizer*, with 178 from the 839 instances of *dizer*.

### 5.3.2 *Verba dicendi* Constructions

The previous experiment hinted that one way to improve the system was through the correct identification of *verba dicendi* constructions (see Sections 3.2.1 and 3.2.5). The first step to address this problem consisted in improving the XIP grammar to correctly identify a broader range of *verba dicendi* constructions. Then, a property was added to ViPEr to represent whether or not a verb meaning allowed for *verba dicendi* constructions[1]. Table 5.6 shows the results obtained by the new set of rules (7,630 rules).

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 6,218 | 4,881 | 2,474 | 3,481 | 4,818 |
| % | 100 | 51.08 | 40.10 | 20.32 | 28.6 | 39.58 |

Table 5.6: Disambiguation rule's results after adding the *verba dicendi* constructions

Comparing these results with those of the previous experiment, we see that the *verba dicendi* feature had a positive impact on the number of disambiguated instances (+2%), both fully and partially, and it also slightly reduced the number of wrongly disambiguated instances (-0.14%). This indicates that *verba dicendi* seems to be important to the full disambiguation of verbs, but that the majority of the errors had a different cause.

### 5.3.3 Dative Pronouns

The third experiment consisted in the addition of the possibility for some verb argument to occur in the form of dative pronouns, which alternate with fully fledged prepositional (dative) phrases. To consider this case, a new property was systematically annotated in ViPEr by the linguists team, and then, added to the set of properties used by the rule generation system. New rules were produced by the system (7,970 rules), whose results are presented in Table 5.7.

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 6,256 | 4,943 | 2,463 | 3,454 | 4,767 |
| % | 100 | 51.39 | 40.61 | 20.23 | 28.37 | 39.16 |

Table 5.7: Disambiguation rule's results after adding dative pronouns properties

The results show a slight improvement on the number of correctly disambiguated instances (+0.31%) and a reduction on the number of incorrect classifications (-0.12%) and non-disambiguated instances (-0.23%). The dative pronouns experiment showed the same effect on the results as the *verba dicendi* feature, though in a much smaller scale. This small impact can be explained by the fact that this feature represents only a particular case of a verb meaning's construction, however frequent it may be[2].

[1]Usually, only one verb sense for a given lemma is marked as *verbum dicendi*.

[2]The use of benefactive/privative (circumstantial) dative pronouns can also result in incorrect classification, though such negative impact was not measured.

### 5.3.4 Passive Sentences

Some construction might only be distinguished by their transformational properties, for example the type of passive constructions allowed. In this experiment, we tested two types of passives, with auxiliary verbs *ser* and *estar* encoded in ViPEr. Again, these properties were added to the rule generation system and a new set of rules was generated (9,712 rules), producing new results, shown in Table 5.8.

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 6,377 | 5,050 | 2461 | 3,335 | 4,662 |
| % | 100 | 52.39 | 41.49 | 20.22 | 27.40 | 38.30 |

Table 5.8: Disambiguation rule's results after adding the passive constructions properties

The inclusion of the transformational properties regarding the two passive constructions showed some impact on the correct classification of verb instances (+1%), while reduction on the error was almost negligible (-0.01%). Further analysis showed that these results did not correctly reflect the impact this feature might have on system, as most of the instances not identified as full verb constructions (see Table 5.3) correspond to participial verb instances that are identified as adjectival phrases (AP) rather that verb phrases (VP). Therefore no conclusions about the impact of this type of properties can be drawn.

### 5.3.5 Feature Priority

During the deeper analysis performed on the previous results, it was noticed that some verbs were only being incorrectly classified because of the order implied by the rules, thus reordering the rules seemed to be a possible way of improving to the system. This experiment was aimed to test the impact of reordering the rules before they are outputted by the system, as presented in Section 3.2.5. Consequently, the number of rules stays the same as in the previous scenario.

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 6,552 | 5,227 | 2,286 | 3,335 | 4,660 |
| % | 100 | 53.82 | 42.94 | 18.78 | 27.40 | 38.28 |

Table 5.9: Disambiguation rule's results after sorting the rules

As shown in Table 5.9, reordering the rules based on the priority of the features tested proved to be a good way to reduce the number of incorrectly classified instances (-1.44%), which dropped below of the 20% error rate mark. Consequently, the number of correctly disambiguated instances increased (+1.43%). However, this change did not impacted the number of instances left untouched by the rules.

### 5.3.6 Verb Meaning Filtering

Finally, after some deeper analysis, it was concluded that a simplification of the task could be of some advantage considering that the system's purpose is to process real texts. A low occurrence filter was developed (see Section 3.3), and a new set of rules was generated. Because of the low occurrence filter, a fewer number of rules was generated (8,503). The results for this last experiment are presented in Table 5.10.

| Classes | Instances | Correctly Disamb. | Fully Disamb. | Wrongly Disamb. | Not Disamb. | Ambiguous Left |
|---|---|---|---|---|---|---|
| **Total** | 12,173 | 7,891 | 6,056 | 1,888 | 2,394 | 4,229 |
| % | 100 | 64.82 | 49.75 | 15.51 | 19.67 | 34.74 |

Table 5.10: Disambiguation rule's results using the low occurrence verb filter

The introduction of the low occurrence filter proved to be a major improvement regarding the reduction of the number of incorrect instances, dropping the error rate from 18.78% to 15.51%. Also, due to the reduction on the number of meanings being considered, the number of correctly disambiguated instances improved greatly (+11%), reaching almost 50% fully disambiguated instances just by applying rules. There is also relevant decreasing in the number of instances not disambiguated (-7.73%) and ambiguous instances left (-3.54%). Figure 5.2 summarizes the evolution of the results obtained in these experiments.

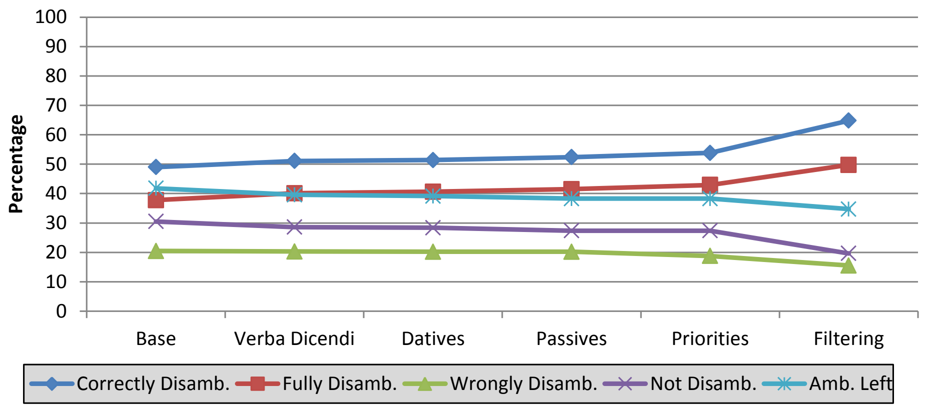

Figure 5.2: Rule-based disambiguation results summary

## 5.4 Rules + MFS

In this experiment, we tested a combination of the rule-based approach and the MFS classifier (baseline). In this case the MFS classifier was applied after the rule-based module to the remaining non-fully disambiguated instances. In other words, the MFS classifier was used to decide the verb sense of the remaining classes accorded to the verb instances still left ambiguous by the rule-based approach. The values presented in Table 5.11 result from the application of this technique to the to the whole corpus, using the same cross-validation evaluation methodology (due to the inclusion of the MFS classifier).

| | **Instances** | **Correctly Disambiguated** | **Wrongly Disambiguated** |
|---|---|---|---|
| **Average** | 1,219 | 965 | 254 |
| % | 100 | 79.15 | 20.85 |
| **Baseline (%)** | 100 | 84.00 | 16.00 |

Table 5.11: Rules+MFS Accuracy

The results obtained (Table 5.11) reveal that this combination performed worse than just applying the MFS technique (Table 5.4). However, comparing the results by the number of ambiguity senses of verbs (Figure 5.3), shows that rules+MFS performed better that just MFS alone for verbs that had a higher number of senses.

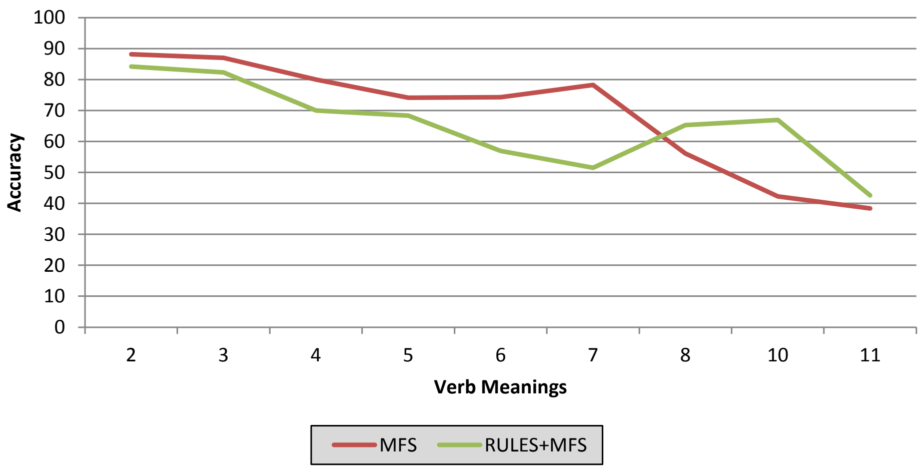

Figure 5.3: MFS vs Rules+MFS per number of ambiguity senses

Using rules to disambiguate only the highly ambiguous verbs (with 8 or more meanings) yielded better results (see Table 5.12) than the previous combination. In fact, using this finer grained combination also achieved slightly better results than the baseline (MFS) alone.

| | Instances | Correctly Disambiguated | Wrongly Disambiguated |
|---|---|---|---|
| **Average** | 1,219 | 1,029 | 190 |
| % | 100 | 84.41 | 15.59 |
| **Baseline (%)** | 100 | 84.00 | 16.00 |

Table 5.12: Rules+MFS Accuracy (using rule-based approach for highly ambiguous verbs)

Considering the marginal gain (+0.4%) in the results, a final analysis was performed. The break-down was performed by verb lemma, revealing that the improvement on results was not directly de-pendant on the number of meanings, as some verbs with 2 senses still showed improvement when using rules followed by MFS. Figure 5.4 illustrates the best accuracy achieved for each verb lemma with the respective technique.

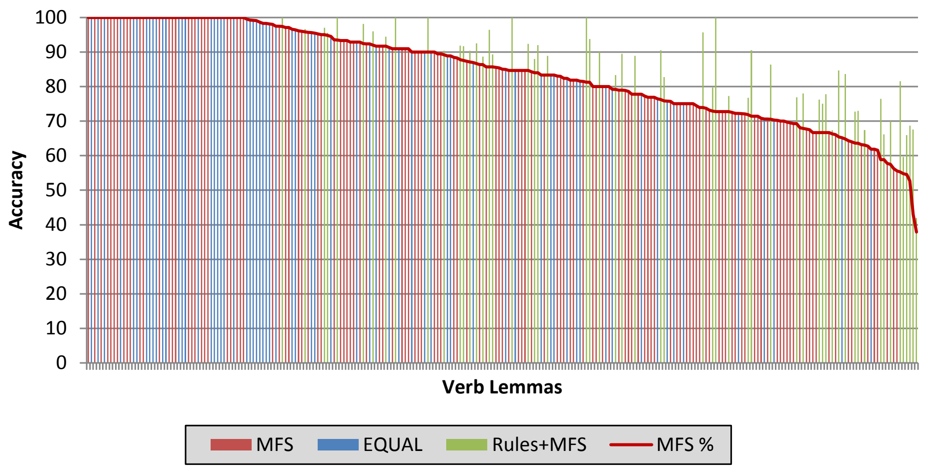

Figure 5.4: MFS vs Rules+MFS per lemma (best case, verbs sorted by MFS)

In the chart we can identify three major areas. The first, going from 100% down to 90% of MFS accuracy, shows that the combination of rules+MFS can not surpass the MFS in the majority of the verbs, however, rules still achieve an equal performance in some cases. The middle area of the chart, below 90% and above 65% MFS, is still dominated by the MFS approach, but the best approach seems more dependent on the verb lemma, as there are already several verbs whose best approach is a combination of both techniques. Finally, below 65% MFS the combination of both methods seems to outperform the baseline for almost all lemmas, however, for some particular verbs the MFS is still the best choice. The previous observations indicate a naturally expectable behaviour from verbs, though there seems to be no general tendency that can help to determine, for a given lemma, the best method to disambiguate it.

Considering the previous information, a final combination of these two techniques was performed, choosing the best scenario for every lemma. The results achieved in this last combination, presented in Table 5.13, surpassed all of the previous experiments, scoring 86.46% accuracy, an improvement of 2.4% against the baseline.

| | Instances | Correctly Disambiguated | Wrongly Disambiguated |
|---|---|---|---|
| **Average** | 1,219 | 1,054 | 165 |
| % | 100 | 86.46 | 13.54 |
| **Baseline (%)** | 100 | 84.00 | 16.00 |

Table 5.13: Rules+MFS Accuracy (best approach per verb lemma)

## 5.5 Machine Learning

In this section, several scenarios used to test the machine learning approach described in Chapter 4 will be presented, followed by the results obtained in each of them. Finally, a discussion of the results will be made, and the main findings will be highlighted.

### 5.5.1 Training Instances

This first scenario was aimed to test the impact of the size of the training corpus in results of the ML approach. The accuracy obtained for each verb lemma using the different training set sizes[3] is presented in Figure 5.5.

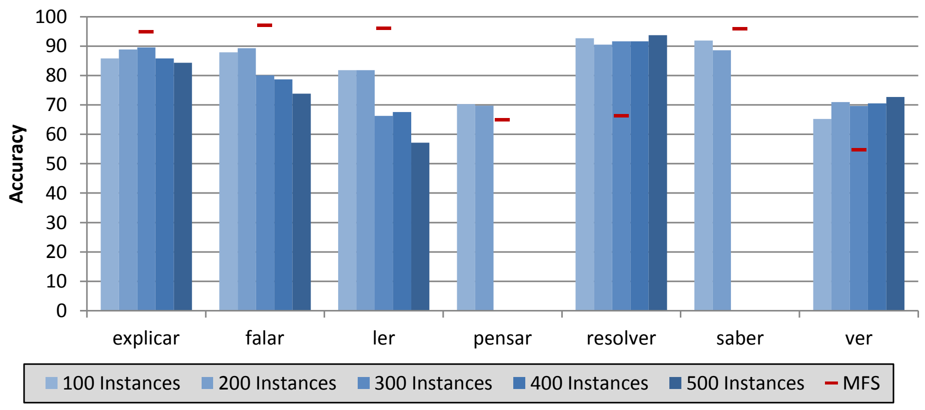

Figure 5.5: Training corpus size impact on ML Accuracy

[3]For verbs *pensar* and *saber* only 250 instances were available at the evaluation time.

From this first experiment we can conclude that whenever the ML system performs better than the MFS, an increase in the number of training instances leads to an increase in the accuracy for that lemma. On the other hand, if the ML module performs worse than the MFS, providing more training instances leads to even worse results, which suggests that the system is deviating from the MFS the more training instances are used.

### 5.5.2 Semantic Features and Window Size

In this testing scenario, semantic information was added to the feature set about the tokens in the context of the target verb, and an increase in the context window size was also tried. Both these changes aimed at compensating some possible incorrect syntactic analysis performed by the previous modules. The number of training instances used for each verb was chosen based on to the best result obtained from the previous experiment. The results for each of the modifications are presented in Figure 5.6.

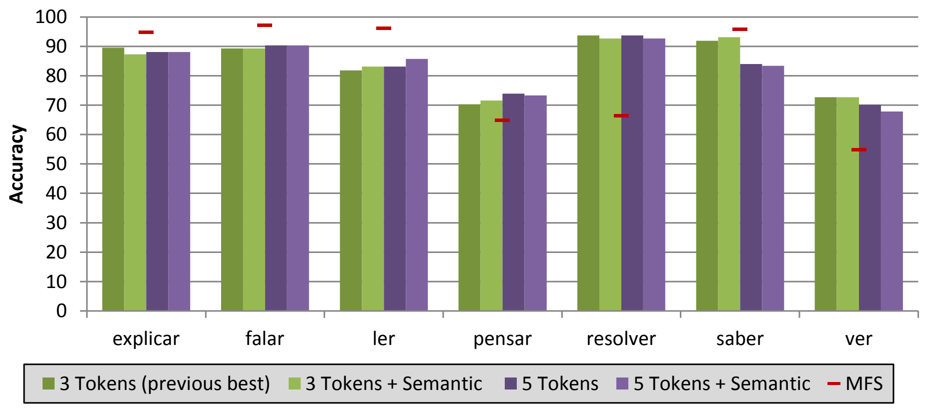

Figure 5.6: Window size and Semantic information impact on ML Accuracy

Increasing the window range to encompass more tokens yielded equal or worse results for most of the verb lemmas (*pensar* was the only verb whose results showed some improvement). Adding semantic information to the context tokens provided inconclusive results, as the accuracy improved for some verbs while it decreased for others. This experiment tested some variations to the most commonly used features in WSD, still, further exploration of other possible features might be performed (see Section 6.2).

### 5.5.3 Bias

The final modification tested for the ML module was the inclusion of the special feature *bias* calculated by the system during the training step. This feature, as the name suggests, indicates the deviation of the model towards each class. Thus, this feature can be interpreted as a kind of a-priori probability of each of the classes. The results obtained by using bias in the prediction step are presented in Figure 5.7.

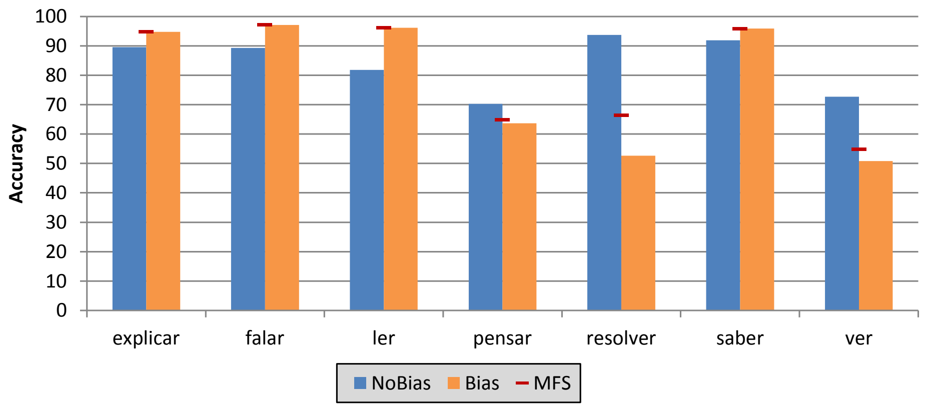

Figure 5.7: Bias impact on ML accuracy

Adding of the bias feature to the classification step in the prediction phase exhibited an expectable behaviour, increasing the accuracy of the system for verbs that have a high MFS. However, the MFS was never surpassed when using the bias feature, which leads to the conclusion that, in practice, considering the bias makes the ML approach for these verbs degenerate into a MFS classifier.

### 5.5.4 Comparison

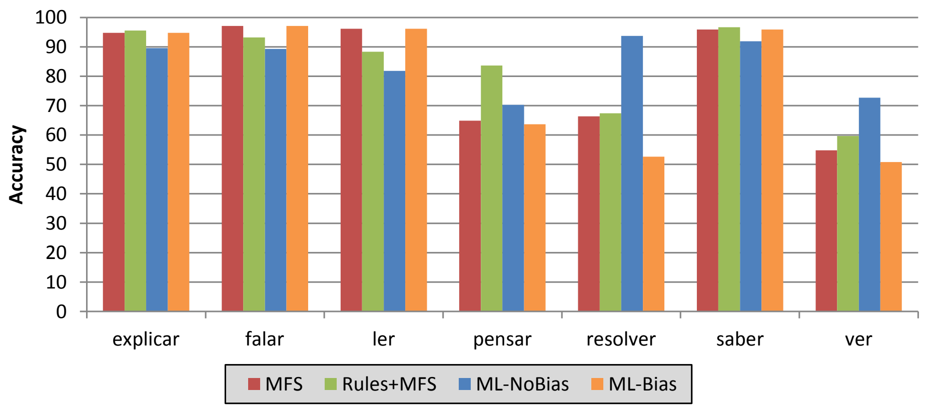

Figure 5.8: Machine Learning Comparison (ML verbs only)

Comparing this technique with all the previously presented methods (Figure 5.8) showed that, whenever a verb has a high MFS, it is difficult for another approach to surpass it. All verbs with low

MFS exhibited an improvement in the results when compared to the MFS. Still, for the verb *pensar*, ML is outperformed by the combination of rules and MFS.

Looking at the set of classes associated with each lemma, both *resolver* and *ver* share the same ambiguity set (06, 32C), while *pensar* has more senses (06, 08, 16, 32C, 35R, 36R). This suggests that ML might be not be able to deal with verbs that have many senses or that a specific set of classes is better disambiguated by this technique, though further testing encompassing more verbs should be performed to verify this finding.

## 5.6 Rules + ML

This scenario, tested how the ML performed as a complementary technique to the rule-based disambiguation. It is similar to the scenario that combined rules and MFS, presented in Section 5.4.

The configuration used in the ML module was the one that provided the best previous results. The bias feature was not used, as its results did not surpass those of MFS. Therefore, using this feature, would make this combination yield the same results as those of the Rules+MFS. The features by the ML system in this combination were those initially proposed, and presented at Section 4.3, since the changes carried out in a previous experiment (Section 5.5.2) did not provide any improvement. Therefore, in this scenario the gains to be obtained are likely to occur in one of the three verbs (*resolver*, *pensar* and *ver*) that had already exhibited better results using ML alone (ML-NoBias).

Although ML disambiguation was also applied to the non-fully disambiguated instances after the rules, the combination method used in this test was slightly different to the one used for rules + MFS. This time, the final class for those instances was determined by ML disambiguation, forgetting any disambiguation previously performed by rules. The main reason for this change was to give ML the possibility to guess least frequent classes that might have been filtered out.

Figure 5.9 shows a comparison between the previous results and the ones obtained in this experiment[4]. Globally, adding ML as a complementary technique to rules proved to be worse than just using ML for the majority of the verbs here studied. This suggests that the instances being correctly classified by the rules are a subset of the instances correctly disambiguated by ML, and that rules are incorrectly classifying the remaining instances that ML would otherwise guess correctly. Although for the majority of the verbs this combination of rules+MFS performed worse, some verbs still showed improvement when compared to the ML scenario (*explicar* and *pensar*), though the difference was minimal. In none of the cases the new accuracy values surpassed the previous best.

This last test showed that rules and ML were not very complementary, and that choosing ML alone

[4]The MFS and Rules+MFS values are same of Figure 5.8

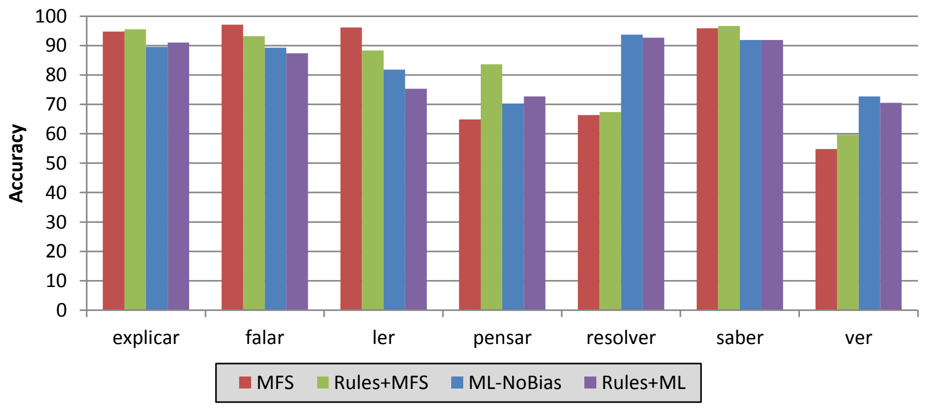

Figure 5.9: Rules + Machine Learning Comparison (ML verbs only)

provides the best improvement to the system. For the cases where Rules+ML surpasses ML alone, other approaches provide better results. Further testing of the ML scenarios should be performed to confirm these results, namely, using annotated data regarding other verb lemmas and trying out other combination of methods and other scenarios.

## 5.7 Discussion

In the previous sections the evaluation scenarios used to test the two approaches to the VSD task were presented. Globally, MFS proved to be the best technique, mostly due to the high MFS percentage exhibited by the majority of the verbs of the corpus. Nonetheless, the two developed modules still provided improvement to the system for some particular verbs.

Applying rule-based disambiguation prior to using the MFS performed worse than just using the MFS, which indicates that rules are not very good to address the problem of verb disambiguation. Though, they still proved to be useful for some verbs.

Machine learning disambiguation was able to improve the accuracy of the system for all the tested verbs with low MFS, while with verbs with high MFS this technique performed equal or worse than the MFS. Nevertheless, for one of the verbs showing improvement (*pensar* with ML-NoBias), this approach was still outperformed by the combination of rules and MFS. Further testing should also be performed for the ML approach using more verb lemmas to confirm these findings.

The combination of rules and ML did not provide any improvement to the system, which suggests that these approaches should be used in a complementary way. Thus, we can conclude that for new verbs one of the following strategies should be used: MFS, Rules+MFS or ML. However, tests did not provide conclusive evidence to determine which strategy performs best for a given verb, except the natural correlation with the low MFS frequency.

With all the previous information regarding each verb lemma, a final combination using all the techniques was done, choosing the best case for each lemma. The results obtained in this last test, reported in Table 5.14, were the best overall, settling a new maximum accuracy of 87.2% for the system, improving 3.2% against the baseline.

| | **Instances** | **Correctly Disambiguated** | **Wrongly Disambiguated** |
|---|---|---|---|
| **Average** | 1,219 | 1,063 | 156 |
| % | 100 | 87.20 | 12.80 |
| **Baseline (%)** | 100 | 84.00 | 16.00 |

Table 5.14: Accuracy obtained in the final combination with all the methods (MFS, Rules+MFS and ML using the best per verb lemma)

Finally, to assess the impact of the developed work in the global performance of STRING, a profiling test was performed, using a different corpus, available at L$^2$F for that purpose. The corpus contains around 5000 sentences collected from CETEMPúblico corpus. Table 5.15 shows the performance impact resultant from the addition of the VSD modules.

| **Modules Integrated** | **Execution Time (s)** |
|---|---|
| Base (without VSD) | 317.00 |
| MFS | 317.10 |
| Rules+MFS | 318.77 |
| Rules+MFS+ML | 318.77 |

Table 5.15: STRING performance after adding VSD modules

The most basic disambiguation method, MFS, increased the processing time by only 0.10 seconds, while the addition of rule-based disambiguation provided the biggest increase, with around 1.7 seconds for the 5,000 sentences corpus. Finally, the impact on the ML was null. However, this last result is not conclusive, as only 2 verb lemmas are disambiguated using the ML approach. Globally, the integration of all the VSD modules did not have a big performance impact on the total time required for STRING to process corpora.

# 6 Conclusions and Future Work

## 6.1 Conclusions

This dissertation addressed the problem of Verb Sense Disambiguation for European Portuguese, that is, the task of selecting the correct sense of a verb in a given sentence. Two approaches were considered when tackling this problem: a ruled-based, and a machine learning disambiguation.

For each technique, a separate module was developed. The rules-based disambiguation module is based on the lexical, syntactic and semantic information described in ViPEr, a detailed description of the most frequent verbs of European Portuguese, containing approximately 6,000 verb senses along with their distributional, structural and transformational properties. The restrictions tested by the rules correspond mainly to the semantic selectional restrictions on the argumental positions of the verb, but other features such as the prepositions selected by the verb for its argumental positions, and several transformational properties like dative pronouning, or the *verba dicendi* constructions, were also used. In its turn, the machine learning module considers the most common properties used in the WSD task, that is, features related to context, syntactic and semantic information of the target word. However, due to the time constraints required for manual annotation a large training corpora, only seven verb lemmas have been addressed by this technique.

A baseline was established, based on the Most Frequent Sense (MFS), and several scenarios were considered to test both modules, along with the baseline thus established. The results obtained from the baseline were surprisingly high (84% accuracy), and proved to be hard to surpass. This high baseline mark is related to the conservative approach adopted for the verb senses collection and the transformational theory underlying it, which significantly factorizes linguistic descriptions into clear-cut semantic distinctions. Globally, using rule-based disambiguation prior to MFS proved to be worse than just using the MFS. However, a finer-grained combination of the two allowed some improvement to the system (+2.4%). From the seven verbs considered in the machine learning experiments, three exhibited improvement in their results. Nevertheless, for one of them, the best solution still consisted in the application of rules complemented by the MFS. In general, the integration of both modules yielded better results and the system was able to achieve a final score of 87.2% accuracy, an improvement of 3.2% above the baseline.

This pioneer work in European Portuguese VSD contributed significantly to the global development of the STRING NLP chain. Having the verbs semantically disambiguated will now allow other pending tasks to be addressed also, such as the correct identification of verb complements and semantic role labelling.

This work also contributed to other areas of the STRING processing chain, by uncovering some less developed aspects of the system, namely, the correct recognition of *verba dicendi* constructions and the correct identification of the pronominal cases, which were tackled during the development of this thesis. Some other problems were found, such as, the correct identification participial constructions and the problem of reduced completive sentences and these await further development in STRING.

Besides the main contribution resulting from this thesis, that is, the verb sense disambiguation modules, other important side modules were developed that should also be mentioned here.

The API created to process the evaluation corpus makes it easier to develop any subsequent modules that need to access the information available in the corpus. This API was also used in some of the modules developed in this thesis and and will be available at L$^2$F.

The corpus validator created to help annotators find possible classification errors, provides an automated way of reviewing the entire corpus whenever the classification of a verb changed in ViPEr. The frequency counter allows for automated collection of statistical information regarding the verb meaning distribution in the corpus. The evaluation module created facilitates the result retrieval of the verb sense disambiguation task.

Finally, this thesis also contributed to the development of the basic structure of the module responsible for the assignment of the semantic roles (SR). In this preliminary phase, all linguistic features pertaining to the SR labelling task are assigned to the verb node after its disambiguation. Then, rules are currently being built (Talhadas 2012) to correctly convert this features assignment to the correct attribution of SR to all verb arguments. In this way, it will be, eventually, possible to use this semantic information to distinguish the verb arguments from other circumstantial constituents.

## 6.2 Future Work

As a pioneer work in the area of VSD for the European Portuguese, the approaches here presented still have much room for improvement. In the following, some possible extensions and future directions for each of the approaches described in this dissertation are briefly sketched.

Although the features detailed in ViPEr were well explored, as presented in Section 5.3, not all the features could be tested. Therefore, one possible extension to this approach would be the inclusion of some of the features left out, such as the two remaining types of passives, the transformational properties of symmetric constructions and the fusion operation for locative constructions.

Additionally, an improvement in the XIP grammar for the correct identification of the participial constructions as full verb constructions would allow a more conclusive testing on the passive sentences' feature tested in this thesis.

For the machine learning disambiguation experiments, only 7 verb lemmas could be tested due to the time required to collect and manually annotate corpora. Thus one proposition is the extension of the training corpus to encompass more verbs, allowing for more conclusions to be drawn regarding that technique.

Currently, the ML module only uses as syntactic features information related to nodes that are directly connected to the verb node. Consequently, information regarding the prepositions introducing the verb arguments is not being used, even though, this is one of the criteria to distinguish verb meanings in ViPEr. Nevertheless, this information on the prepositions is already being taken into consideration by the context features, if they occur in the window define by the context. Notice, however, that many verbs select direct objects and these do not involve this dependency. A possible extension of the syntactic features to include information regarding the prepositions chosen by the verb arguments could provide improvement to the system.

Another suggestion regarding the ML approach is to try out and possibly include other learning algorithms, such as those presented in Chapter 2 (SVM, Naive Bayes, among other). This addition would require another combination methodology, like weighting each classifier, either manually or using a meta-classifier.

Adding an unsupervised learning approach, side-by-side with the supervised methods, would enable us to discard the time-consuming task of manual annotating of the training instances. Naturally, the results provided by this unsupervised approach should be evaluated to test if they are equivalent to those obtained using the supervised approach. One possible unsupervised technique to be used is clustering, based on the k-means algorithm (MacQueen 1967), for example.

Finally, the feature set here used by the learning algorithm could be expanded and/or modified. The following changes may already be suggested: considering only the dependencies involving elements at a certain distance from the target verb; adding new context tokens, but this times filtered by their POS, ignoring, for example, articles, and conjunctions.

Certainly, there is still much work to be done in VSD, and the system here presented can be improved. Nevertheless, we would like to think some valid contribution has been made to the domain.

# Bibliography

Abend, O., R. Reichart, and A. Rappoport (2008). A supervised algorithm for verb disambiguation into VerbNet classes. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, Stroudsburg, PA, USA, pp. 9–16. Association for Computational Linguistics.

Agirre, E., I. Aldabe, M. Lersundi, D. Martínez, E. Pociello, and L. Uria (2004, July). The Basque Lexical-Sample Task. In R. Mihalcea and P. Edmonds (Eds.), *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 1–4. Association for Computational Linguistics.

Ait-Mokhtar, S., J. Chanod, and C. Roux (2002). Robustness beyond shallowness: incremental dependency parsing. *Natural Language Engineering 8*(2/3), 121–144.

Baker, C. F., C. J. Fillmore, and J. B. Lowe (1998). The Berkeley FrameNet Project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*, ACL '98, Stroudsburg, PA, USA, pp. 86–90. Association for Computational Linguistics.

Baptista, J. (2012, September). ViPEr: A Lexicon-Grammar of European Portuguese Verbs. In *31st International Conference on Lexis and Grammar*, Nové Hrady, Czech Republic, pp. 10–16.

Berger, A. L., S. A. D. Pietra, and V. J. D. Pietra (1996). A Maximum Entropy approach to Natural Language Processing. *Computational Linguistics 22*, 39–71.

Bilgin, O., Özlem Çetino Glu, and K. Oflazer (2004). Building a WordNet for Turkish. *Romanian Journal of Information Science and Technology 7*, 163–172.

Bird, S., E. Klein, and E. Loper (2009). *Natural Language Processing with Python*. O'Reilly Media, Inc.

Borba, F. (1990). *Dicionário Gramatical de Verbos do Português Contemporâneo do Brasil*. São Paulo: UNESP.

Borin, L., D. Dannélls, M. Forsberg, M. Toporowska Gronostaj, and D. Kokkinakis (2009). Thinking Green: Toward Swedish FrameNet++.

Brown, S. W., D. Dligach, and M. Palmer (2011). VerbNet class assignment as a WSD task. In *Proceedings of the Ninth International Conference on Computational Semantics*, IWCS '11, Stroudsburg, PA, USA, pp. 85–94. Association for Computational Linguistics.

Buscaldi, D., P. Rosso, and F. Masulli (2004, July). The upv-unige-CIAOSENSO WSD system. In R. Mihalcea and P. Edmonds (Eds.), *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 77–82. Association for Computational Linguistics.

Busse, W. (1994). *Dicionário Sintáctico de Verbos Portugueses*. Coimbra: Almedina.

Carapinha, F. (2013). Extração Automática de Conteúdos Documentais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Cortes, C. and V. Vapnik (1995). Support-vector networks. *Mach. Learn. 20*(3), 273–297.

Daumé, H. (2004, August). Notes on CG and LM-BFGS Optimization of Logistic Regression. Paper available at `http://pub.hal3.name#daume04cg-bfgs`, implementation available at `http://hal3.name/megam/`.

Diniz, C. F. P. (2010, October). Um Conversor baseado em regras de transformação declarativas. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Escudero, G., L. Màrquez, and G. Rigau (2004, July). TALP System for the English Lexical Sample Task. In R. Mihalcea and P. Edmonds (Eds.), *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 113–116. Association for Computational Linguistics.

Gildea, D. and D. Jurafsky (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics 28*(3), 245–288.

Johnson, C. R., C. J. Fillmore, E. J. Wood, J. Ruppenhofer, M. Urban, M. R. Petruck, and C. F. Baker (2001). The FrameNet Project: Tools for Lexicon Building.

Kipper, K., H. T. Dang, W. Schuler, and M. Palmer (2000). Building a Class-Based Verb Lexicon using TAGs. In *TAG+5 Fifth International Workshop on Tree Adjoining Grammars and Related Formalisms*, Paris, France.

Lee, Y. K., H. T. Ng, and T. K. Chia (2004, July). Supervised Word Sense Disambiguation with Support Vector Machines and Multiple Knowledge Sources. In R. Mihalcea and P. Edmonds (Eds.), *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 137–140. Association for Computational Linguistics.

MacQueen, J. B. (1967). Some Methods for Classification and Analysis of MultiVariate Observations. In L. M. L. Cam and J. Neyman (Eds.), *Proc. of the fifth Berkeley Symposium on Mathematical Statistics and Probability*, Volume 1, pp. 281–297. University of California Press.

Mamede, N. J., J. Baptista, C. Diniz, and V. Cabarrão (2012, April). STRING: An Hybrid Statistical and Rule-Based Natural Language Processing Chain for Portuguese. In *International Conference on Computational Processing of Portuguese (Propor 2012)*, Volume Demo Session. Paper available at `http://www.propor2012.org/demos/DemoSTRING.pdf`.

Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval*. New York, NY, USA: Cambridge University Press.

Marrafa, P., R. Amaro, and S. Mendes (2011). WordNet.PT global: extending WordNet.PT to Portuguese varieties. In *Proceedings of the First Workshop on Algorithms and Resources for Modelling of Dialects and Language Varieties*, DIALECTS '11, Stroudsburg, PA, USA, pp. 70–74. Association for Computational Linguistics.

Maurício, A. S. B. (2011, November). Identificação, Classificação e Normalização de Expressões Temporais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Miller, G. A. (1995). WordNet: A Lexical Database for English. *Communications of the ACM 38*, 39–41.

Miller, G. A., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller (1990). WordNet: An On-line Lexical Database. *International Journal of Lexicography 3*, 235–244.

Nobre, N. R. P. (2011, June). Resolução de Expressões Anafóricas. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Ohara, K., S. Fujii, S. Ishizaki, T. Ohori, H. Saito, and R. Suzuki (2004). The Japanese FrameNet Project; An Introduction. In C. J. Fillmore, M. Pinkal, C. F. Baker, and K. Erk (Eds.), *Proceedings of the Workshop on Building Lexical Resources from Semantically Annotated Corpora*, Lisbon, pp. 9–12. LREC 2004: LREC 2004.

Ribeiro, R. (2003, March). Anotação Morfossintáctica Desambiguada do Português. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Ruppenhofer, J., M. Ellsworth, M. R. Petruck, C. R. Johnson, and J. Scheffczyk (2006). *FrameNet II: Extended Theory and Practice*. Berkeley, California: International Computer Science Institute. Distributed with the FrameNet data.

Sagot, B. and D. Fišer (2008). Building a free French wordnet from multilingual resources. In *OntoLex*, Marrakech, Maroc.

Salton, G., A. Wong, and C. S. Yang (1975, November). A vector space model for automatic indexing. *Commun. ACM 18*(11), 613–620.

Shi, L. and R. Mihalcea (2004). Open Text Semantic Parsing using FrameNet and WordNet. In *Demonstration Papers at HLT-NAACL 2004*, HLT-NAACL–Demonstrations '04, Stroudsburg, PA, USA, pp. 19–22. Association for Computational Linguistics.

Subirats, C. and H. Sato (2004). Spanish FrameNet and FrameSQL. In *Proceedings of LREC 2004. Workshop on Building Lexical Resources from Semantically Annotated Corpora*, Lisbon, Portugal.

Talhadas, R. (2012, November). Automatic Semantic Role Labelling for European Portuguese. Dissertation Project.

Tufi, D., E. Barbu, V. B. Mititelu, R. Ion, and L. Bozianu (2004). The Romanian Wordnet. *Romanian Journal of Information Science and Technology*, 107–124.

Vicente, A. (2013). LexMan: um Segmentador e Analisador Morfológico com transdutores. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa.

Villarejo, L., L. Màrquez, E. Agirre, D. Martínez, B. Magnini, C. Strapparava, D. McCarthy, A. Montoyo, and A. Suárez (2004, July). The "Meaning" System on the English All-Words Task. In R. Mihalcea and P. Edmonds (Eds.), *Senseval-3: Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text*, Barcelona, Spain, pp. 253–256. Association for Computational Linguistics.

Witten, I. H., E. Frank, and M. A. Hall (2011). *Data Mining: Practical Machine Learning Tools and Techniques* (3 ed.). Amsterdam: Morgan Kaufmann.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd annual meeting on Association for Computational Linguistics*, ACL '95, Stroudsburg, PA, USA, pp. 189–196. Association for Computational Linguistics.