Abstract. This thesis addresses the problem of Verb Sense Disambiguation (VSD) in European Portuguese. Verb Sense Disambiguation is a subproblem of the Word Sense Disambiguation (WSD) problem, that tries to identify in which sense a polysemic 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 (MFS) 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 three techniques yielded an accuracy of 87.2%, a gain of 3.2% above the baseline.

Keywords: Natural Language Processing, Verb Sense Disambiguation, Rule-based disambiguation, Machine Learning

1 Introduction

Nowadays, Natural Language Processing (NLP) has many applications: 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, ambiguity is present at several stages in the processing of a text or a sentence, such as, tokenization, sentence-splitting, part-of-speech (POS) tagging, syntactic parsing and semantic processing.

Semantic ambiguity is usually the last to be addressed by NLP systems, and it tends to be one of the hardest to solve among all types of ambiguities 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 category they were tagged with. Consider the following examples:

(1a) I am counting the number of students
(1b) I am counting on you to help me

The examples show how the verb to count, used in the same position on both sentences, means enumerate 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 (1a) results from its direct-transitive construction and plural object, while in (1b) the preposition introducing the second argument and the human trait on that same argument of count determine the verb’s sense.

Work in the WSD area normally uses the surrounding words to help disambiguate the target word, by employing machine learning techniques and/or using other algorithms, in combination with linguistic resources such as Wordnet\(^1\) or Framenet\(^2\).

### 1.1 Goals

This work 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 be integrated in the STRING system \([1]\), an hybrid statistical and rule-based NLP system developed by and used at L\(^2\)F. Finally, the two developed modules will be evaluated to assess the performance of the system.

### 2 Related Work

#### 2.1 Lexical Resources

Four different lexical resources were compared, each of them encoding syntactic and semantic information about words in a different way. For the English, WordNet \([2]\), FrameNet \([3,4,5]\) and VerbNet \([6]\) were considered. For the European Portuguese, we considered ViPEr \([7]\), which features a classification based

\(^1\) [http://wordnet.princeton.edu/](http://wordnet.princeton.edu/)
\(^2\) [https://framenet.icsi.berkeley.edu/fndrupal/](https://framenet.icsi.berkeley.edu/fndrupal/)
on the syntactic-semantic properties of the most frequently occurring verbs. Table 1 summarizes some of the different aspects represented in each of these lexical resources.

<table>
<thead>
<tr>
<th>Lexical Resource</th>
<th>Grammatical Categories</th>
<th>Syntactic Patterns</th>
<th>Thematic Roles</th>
<th>Selectional Restrictions</th>
<th>Has Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>WordNet</td>
<td>N,V,Adj,Adv</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>FrameNet</td>
<td>N,V,Adj,Adv</td>
<td>Yes</td>
<td>Explicit</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>VerbNet</td>
<td>V</td>
<td>Yes</td>
<td>Explicit</td>
<td>Explicit</td>
<td>Yes</td>
</tr>
<tr>
<td>ViPEr</td>
<td>V</td>
<td>Yes</td>
<td>Implicit</td>
<td>Explicit</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Table 1. Lexical Resources Comparison

In the context of WSD, all these resources can be used, as they provide very useful information about the constraints 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.

Besides the resource here mentioned, there are, naturally, other known sources with linguistic information on verbal constructions of Portuguese. Among others, the most relevant (and recent) are described in [8] and [9], each developed in a different theoretical framework.

2.2 Word Sense Disambiguation Approaches

An overview of the most used techniques and features for WSD was also conducted, based on the systems evaluated at the SensEval3 [3]. The most common learning algorithms [10] used at SensEval3 are the following:

– The Naive Bayes algorithm, 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.

– The Decision List algorithm [11], 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 rule that is matched.

– The Vector Space Model algorithm, 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 the word. These centroids are then compared with vectors of features from testing examples using the cosine function.

– Support Vector Machines, the most widely used classification technique in WSD at SensEval3 [12,13,14], 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.

The most commonly used features used by the systems proposed and presented at SensEval3 can be divided as follows:

- **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-from 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 from 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.

3 Rule-based Disambiguation

The large number of verbs involved and the complexity of the rule generation task required the construction a specific module for rule generation, whose architecture is illustrated in Figure 1. The first step, *parsing*, takes as its input the lexical resource information (ViPEr) and produces a structure that is passed onto the following module, the *difference finder*. This module compares the features associated to each meaning of a polysemic verb and 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 sorted and output to be used by the parsing module of the STRING system, XIP [15].

3.1 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 intermediate evaluation.

The base features considered by the system are the selectional restrictions on the verb’s arguments (N0 to N3) and their respective prepositions, the property (*vse*), denoting intrinsically reflexive constructions (e.g. *queixar-se*) and transformational properties regarding the two most common types of passives (with auxiliary verbs *ser* and *estar*).
A new property, *vdic*, added to ViPEr, indicating which verb meanings (usually only one per lemma) allow the *verbum dicendi* construction pattern, due to its subject inversion pattern.

Afterwards, the *a Ni=the* 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.

### 3.2 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 parsing module is also responsible for producing the lexicon file used in the XIP grammar, 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 the XIP grammar evolve.

### 3.3 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.

The system compares pairs of meanings within the same lemma, requiring combinatorial generation for verbs with three or more meanings. It generates differences for each argument found different between the pair of meanings. This approach provides versatility, allowing a partial disambiguation of verbs when verbal arguments are absent, as it often occurs in real texts.
3.4 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 rule generation process, however, is not totally straightforward, as the values for the ViPEn positions do not match directly the features provided by XIP. 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 also allows for the independent development of both ViPEn and STRING.

3.5 Rules Priority

After testing the rules, 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 non-human distributional feature. Since XIP does not have a feature to designate non-human objects, the nHum ViPEn value was mapped as the absence of any of the human-related features of XIP. Consequently, nouns with semantic traits such as location or body-part, are included in this mapping of the nHum feature. Therefore, these more specific distributional features must have precedence over the nHum, so that the rules correctly disambiguate a given verb instance.

After studying the ViPEn properties and their mappings, an order was established, with the first properties to be tested being related to structural patterns in sentences, followed by the several semantic selectional restrictions. 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).

4 Machine Learning Disambiguation

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 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 [16], based on maximum entropy models [17], was used. The architecture of the implemented supervised classification approach is presented in Figure 2.
4.1 Training Corpus

The approach adopted for the machine learning module consists in building a classifier per lemma. Therefore, only some of the verbs would be addressed by this approach in this project, due to the time required for manual annotation of the training data. 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.

Samples of examples collected and annotated were split into partitions of 50 examples, each encompassing all word forms (inflected 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).\(^4\)

4.2 Features

The features to be used by the machine learning module follow the commonly used features presented in section 2.2. 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

\(^4\) At the time of evaluation, only 250 instances were available for the verbs pensar and saber.
respective indexes (-3, -2, -1, +1, +2, +3). The information collected about each of the tokens was the POS tag and lemma.

**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 ViPEx, for example, **human**, **location**, **body-part** currency, among others.

Some additional features that do not fit exactly into any of those categories were also used. These features concern the concept of **verba dicendi** construction and information regarding the case of clitic of pronouns.

5 Evaluation

5.1 Methodology

To evaluate the two approaches presented in the previous sections a manually annotated corpus had to be built and validated. The corpus used to train the statistical POS tagger, Marv [18], was chosen for this evaluation. This corpus contains around 250k words with, 38,827 verbs manually, from which 21,368 (about 55%) are full verbs. From the full verb constructions, 13,030 correspond to ambiguous verbs, but due to errors in other STRING modules, 839 of these instances could not be classified by any of the methods developed. Thus, there is a ceiling of 12,191 correctly disambiguated instances in the results, which was considered as 100%.

Since the goal is to measure the correctness of the results produced by the system against a reference value, accuracy was adopted as the evaluation measure for this VSD task. In this case, it can be defined as the number of correctly disambiguated instances over the total number of ambiguous verb instances in the evaluation corpus processable by the system (ceiling).

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.
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. The accuracy of 84% achieve by the technique was both surprisingly high and hard to surpass. This high baseline mark is related to the conservative approach underlying the ViPEr classification, leading to a lower number of classes when compared to other lexical resources.

5.3 Rules+MFS

In this experiment, a combination of the rule-based approach and the MFS classifier (baseline) was tested. In this case, the MFS classifier was applied after the rule-based module to the remaining non-fully disambiguated instances. Three different scenarios were tested, each regarding a different grained combination of the two methods: (i) the disambiguation rules were applied to all the verbs, regardless of their number of senses by lemma; (ii) rule-based disambiguation was only applied to highly ambiguous verbs (with 8 or more meanings); (iii) rules were only applied to the verbs showing improvement with this technique. Results of each scenario are shown in Table 2.

<table>
<thead>
<tr>
<th>Experiment</th>
<th>Correctly Disamb.</th>
<th>Wrongly Disamb.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>84.00</td>
<td>16.00</td>
</tr>
<tr>
<td>Scenario (i)</td>
<td>79.15</td>
<td>20.85</td>
</tr>
<tr>
<td>Scenario (ii)</td>
<td>84.41</td>
<td>15.59</td>
</tr>
<tr>
<td>Scenario (iii)</td>
<td>86.46</td>
<td>13.54</td>
</tr>
</tbody>
</table>

Table 2. Rules+MFS Accuracy

5.4 Machine Learning Disambiguation

Among the different scenarios tested in ML, one included the use of bias, a special feature calculated by the system during the training step, which indicates the deviation of the model towards each class. Comparing these two ML approaches with all the previously presented methods (Figure 3) showed that, whenever a verb has a high MFS, it is difficult for another approach to surpass it, and only ML-Bias can equal the MFS accuracy. All verbs with low MFS exhibited an improvement in the results (using ML-NoBias) when compared to the MFS. Still, for the verb pensar, ML-NoBias 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 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.

5.5 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 5.3.

Figure 4 shows a comparison between the previous results and the ones obtained in this experiment\(^5\). Globally, adding ML as a complementary technique

\(^5\) The MFS and Rules+MFS values are same of Figure 3
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.

This experiment showed that rules and ML were not very complementary, and that choosing ML alone 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.

5.6 Rules+MFS+ML

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. This last experiment obtained the best overall results, settling a new maximum accuracy of 87.2% for the system, improving 3.2% against the baseline.

Finally, a performance test was conducted to assess the impact of the newly added modules to the global performance of STRING. This test revealed a minimal increase in the global time required for processing (around +1.8 seconds for 5,000 sentences).

6 Conclusions

This work addressed the problem of verb sense disambiguation for the European Portuguese. Two approaches were considered when tackling this problem: a ruled-based, and a machine learning disambiguation.

The results obtained from the baseline were surprisingly high (about 84% accuracy) and proved to be hard to surpass. The integration of both modules provided better results to the system, achieving a final score of 87.2% accuracy, an improvement of 3.2% above baseline.

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.

References