Verb Sense Classification

Gonçalo Suissas

IST – Instituto Superior Técnico
L²F – Spoken Language Systems Laboratory – INESC ID Lisboa
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
goncalo.r.suissas@tecnico.ulisboa.pt

Abstract. This document addresses the verb sense disambiguation (VSD) problem, a sub-problem of word sense disambiguation (WSD), for European Portuguese. It aims at developing a set of modules of an existing Natural Language Processing (NLP) system, which will enable it to choose adequately the precise sense that a verb features in a given sentence, from among other, potential different meanings.

This paper presents various methods used in supervised classification that can be adopted on VSD, and it discusses the main problems found for this task, briefly describing the techniques previously used to address it, as well as the new Machine Learning (ML) techniques that will be integrated in the STRING system.

These ML techniques were tested in several scenarios to determine the impact of different features. The baseline accuracy of 63.86% results from the most frequent sense (MFS) for each verb lemma in a set of 24 verbs. Among the ML techniques tested, the best method was the Naive Bayes algorithm, which achieved an accuracy of 67.71%, a gain of 3.85% above the baseline.

1 Introduction

There is a major concern in Natural Language Processing (NLP) which needs to be addressed, which is ambiguity. Ambiguity is the term used to describe that a certain word, expression or a sentence in a text could be interpreted in more than one way. Ambiguity is present at several stages of processing a sentence or a text.

Focussing on semantic ambiguity, an example of the importance of word sense disambiguation, let us consider the case of machine translation. When trying to translate a sentence, the system has to capture the sentence’s correct meaning, in order to do a correct translation. For example, consider the following two sentences:

(1.1 a) O Pedro arranjou o computador do irmão. (Peter repaired his brother’s computer.)

(1.1 b) O Pedro arranjou o livro que procuravas. (Peter found the book that you are looking for.)
Both sentences use the Portuguese verb *arranjar*. However, when translated to English, each sentence feature different verbs, corresponding to the verb's different meanings.

Our aim was to develop a set of supervised learning methods of a NLP system that will enable it to choose adequately the precise sense using a set of verb features in a given sentence, from among potential different meanings.

Section 2 features the experiments made using the Weka software package with different supervised learning methods, to see which will be integrated in the STRING (Mamede et al., 2012) system. In section 3 it will be presented the implementation in XIP (Ait-Mokhtar et al., 2002) grammar of the Naive Bayes algorithm. Later (section 4), we will present the results for Naive Bayes evaluation and how it compare with previous methods available in the system. Finally, in section 5, we will discuss the evaluation results and we will see what shall be done to improve the system’s performance.

2 Weka Experiments

In this section the different experiments using the Weka software will be presented. Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or used in other developed Java code. Using the Weka software, it was able to view the impacts of different supervised methods on the VSD problems, based on a training corpus.

The corpus was collected from the CETEM Público\(^1\) corpus (Rocha and Santos, 2000), and it contains from 200 to 500 instances for each verb lemma.

For each supervised method chosen, a set of experiments was carried out, in order to view what were the best combinations of extracted features that produced the better overall results. The features extracted for these methods can be organized in three groups, as fellow:

- **Local features**, describe the information around the target word (the verb). In the system, the context words are extracted in a window of size 3 around the target verb, which is a total of 6 tokens that are used, with their respective indexes (-3, -2, -1, +1, +2, +3). The information collected about each of the tokens was the part of speech (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 the lemma of the head word of each node in these relations were extracted, together with the respective dependency name. Several other dependencies/relations are implemented in the XIP grammar, only those of *SUBJ* (subject), *CDIR* (direct complement) and *MOD* (modifier) were considered for each 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

\(^1\) [http://www.linguateca.pt/cetempublico/](http://www.linguateca.pt/cetempublico/)
for verb arguments. 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.

In order to use the weka software package, it was necessary to transform the training data into an ARFF file (Figure 1). ARFF files have two distinct sections. The first section is the Header information, which is followed the Data information. The Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types. The other section describes the raw data observed in the training data, in this case, the features extracted from XIP and the class in ViPEr relating to a verb sense, with the missing values represented by ?.

```plaintext
@RELATION v i p e r C l a s s

@attribute TOK−3−pos {ART, NP, NOUN, PUNCT, VERB}
@attribute TOK−2−pos {NOUN, VF, PUNCT, ART, CONJ}
@attribute TOK−1−pos {ADV, PREP, CONJ, NOUN, REL, PUNCT, VCOP}
@attribute DEP−SUBJ {NOUN, ART, REL, PRON, ADJ}
@attribute class {32C, 38 L1}

@DATA
ART, NOUN, ADV, ART, 32C
ART, NOUN, ADJ, ART, 38 L1
```

Fig. 1. Example of a ARFF file used in the Weka experiments

To generate the ARFF file, a converter was implemented in order to write the features extracted from the STRING system in the required form.

Although every possible combination of extracted features was considered, some combinations were discarded because the results were not being close of those of the others experiments or the data needed to be pre-processed. The algorithms chosen for these experiments are the following:

- **ID3 algorithm** (Quinlan, 1986) which is an algorithm used to generate a decision tree from a dataset. ID3 is typically used in the machine learning and natural language processing domains.
- **Support Vector Machines** (Vapnik, 1995), which 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.
- **CART algorithm** (Breiman et al., 1984) is currently the most used technique for building decision trees (Witten et al., 2011). A strong advantage of this method consist in the fact that it can process data that has not been pre-processed yet.
- **Naive Bayes algorithm** (Manning et al., 2008), which estimates the most probable sense for a given word based on the prior probability of each sense and the conditional probability for each of the features in that context.

- **Bayes Network** is a probabilistic model that represents a set of random variables and their conditional dependencies via a directed acyclic graph (DAG). It is very useful to infer unobserved variables.

- **AdaBoost**, (Freund and Schapire, 1999) is a machine learning meta-algorithm that is used in conjunction with many other types of learning algorithms to improve their performance. The output of the other learning algorithms is combined into a weighted sum. For this experiment, it was used a decision stump algorithm as weak learner, where it is a machine learning model consisting of a one-level decision tree (Iba and Langley, 1992).

- **Decision table** is a precise method to model complicated logic. Decision tables, like if-then-else statements, associate conditions with actions to perform. The approach used in this method was the Best-first search, which is greedy algorithm which explores a graph by expanding the most promising node chosen according to a specified rule (Pearl, 1984).

- **Maximum Entropy** is used for predicting the outcome of a categorical dependent variable (i.e., a class label) based on one or more predictor variables (features), by using probability scores as the predicted values of the dependent variable.

Figure 2 presents the results of the experiments using the Machine learning algorithms mentioned above with the Weka software package.

![Figure 2](image-url)  
**Fig. 2.** The results obtain using the weka software package.
From the results, the Maximum Entropy (78.65%) and Naive Bayes (79.52%) obtain the better overall results. However, the difference between these methods and the remainder (except ID3) is small. Since the difference between these algorithms was minimal, a implementation of the Naive Bayes was decided, in order to view the its impact on the STRING system.

3 Naive Bayes Implementation

In this section the implementation of the Naive Bayes algorithm and its integration in STRING system will be described.

The Naive Bayes algorithm is based on the Bayes theorem, where every feature is assumed to be independent from the other features. The following expression presents how Naive Bayes determinates a class according to the observed features.

\[ P(C|F_1...F_n) = \frac{P(C)}{\prod_{i=1}^{n} P(F_i|C)} \]

where \( C \) is the class to be determined, and \( F_1...F_n \) the features in the instance that is processed. The probability \( P(F_i|C) \) is simple to calculate, since it is the count of times the feature \( F_i \) appears when the class is \( C \) in the training set; as well as, \( P(C) \), which is the number of instances that are labelled as \( C \).

The machine learning algorithm was implemented using the KiF language, developed by Xerox, in order to integrate with the STRING system, using a similar approach used in the existing package, MegaM (Daumé, 2004), based on Maximum Entropy Models (Berger et al., 1996).

The training phase of the model consists of for each instance it extracts the same features extracted from MegaM, however a tab separates the label from the features extracted. From then, each feature is separated from the others, where in each line on the training data has only one feature and the class labelled in that instance, separated with a tab.

When the model is created, each feature in the training data is stored in a hash table, where the key is the feature extracted and the value stored is an array containing the number of appearances of that feature for each class.

In prediction phase, the algorithm accesses the model, which contains the counts for each class of all features presented and calculates the probability for each class, according to the features seen. The most probable class returned from the algorithm is provided by the following expression:

\[ C = \arg\max P(C) \prod_{i=1}^{n} P(F_i|C) \]

where \( C \) is the class to be determined, and \( F_i \) each of the features in the instance that is processed. For features that are not present in the training data, a smoothing method was implemented. The method chosen for this implementation was the additive smoothing, which for each missing feature in the model, the probability assigned is very low. Without having an smoothing method, the probability of an missing feature would be zero, since this feature would not be
included in the model. The impact caused could leave to a greater number of incorrectly classified instances as oppose to when a smoothing method is applied.

The following expression presents the additive smoothing used in this implementation:

$$P(F_i | C) = \frac{F_i + 1}{F(C) + |F|}$$

where $F(C)$ is the number of features counted in the class $C$ and $|F|$ the number of features present in the model. This method allows to process the missing features without the algorithm returning zero, whenever such event occurs, giving the possibility to not include more training instances every time a missing feature is found.

4 Evaluation

This section will be present the results for each supervised learning method previously described.

The corpus chosen for evaluation was the Parole corpus (do Nascimento et al., 1998) which contains around 250,000 words. Each verb on the corpus had been manually annotated and then reviewed by linguists. The corpus is composed of texts from a very diverse nature (genre and topic) and its made of full texts. In this respect it is different from the training corpus, which is composed solely of journalistic text, and instead of full texts, it features extracts of one to a few sentences. The evaluation will consist of cross-validation method with 10 folds, for all methods and a comparison with the results obtained in (Travanca, 2013) will be made.

4.1 Comparison with previous results

In this section the results between the verbs processed in (Travanca, 2013) and with the training data used for this dissertation will be presented. This comparison is aimed to view if the changes in both the training data and the corpus would led an impact on the system’s overall results.

These changes include the addition of more instances, the correction of some of the verbs classified and grammatical compounds, which did not leave an large impact on the system’s overall results.

Figure 3 presents a comparison between the results obtained from the rule-disambiguation system with the training data used for this dissertation and the results obtained from (Travanca, 2013).

The difference between the results is minimal for most of the verbs lemmas used in (Travanca, 2013). However, the verb lemma resolver was where the difference was considerable. The results using the rule-disambiguation system and the MFS classifier provided inconclusive results, as the accuracy improved for some verbs while decreased for others.

Another comparison made between the results of (Travanca, 2013) was in the supervised learning methods used in that work. For this comparison it will be
used the bias feature both enabled and disabled while evaluating the machine learning method integrated in STRING, presented in Figure 4. The bias feature is calculated during the training step, which indicates the deviation of the model towards each class.

Although changes were made in the corpus, the difference achieved in the results is minimal. The results are similar to the previous obtained by (Travanca, 2013), where ler was the only verb lemma with a considerable difference in the accuracy of the Maximum Entropy algorithm without the bias feature.

4.2 Naive Bayes experiments

In this section a comparison between the experiments made in the Naive Bayes algorithm will be presented.

In this algorithm, the impact of the dependencies between nodes extracted from XIP when building the features of the processed sentence during the prediction phase was tested. A comparison of storing or not these dependencies was made in order to view which had the better overall results.

Figure 5 presents the comparison between the models obtained by naive bayes and maximum entropy algorithms.

Storing these dependencies give the better overall results. However when not storing these dependencies, in some verbs (aprender, assinalar, confrontar and ver) it was achieved the same or slightly higher accuracy than storing the dependencies between nodes extracted from XIP. When applying this experiment on maximum entropy models, the results did not improve and proved to be worse than using the dependencies extracted from XIP.

Fig. 3. Comparison using the rules-disambiguation system.
4.3 Comparison

To understand better the impact of each method on the evaluation corpus, it is necessary to view the average accuracy for all verbs lemmas tested, in order to view which achieved the better overall results.

Figure 6 presents the average of the results between all methods integrated in STRING.

Comparing these results the naive bayes algorithm with XIP dependencies stored (67.71%) achieved a slightly better result than the maximum entropy without the bias feature (67.01%) and an improvement of 3.85% above the baseline (63.86% accuracy). However if the verb contornar that contains only 2 instances on the evaluation corpus, where the maximum entropy disambiguated correctly all instances as oppose to the naive bayes which disambiguate half, was excluded form the experiments then the difference between these algorithms will be increased. This would give an increase in accuracy on the naive bayes algorithm with XIP dependencies stored (68.48%) and a decrease from the maximum entropy (65.57%), which would be surpassed slightly by the naive bayes algorithm without the XIP dependencies stored (65.59%).

5 Conclusion and Future Work

Using the Weka software package was possible to infer that both maximum entropy and naive bayes achieved the better overall results, however all approaches except the ID3 algorithm achieved similar results. Therefore it was decided a implementation of the naive bayes algorithm in the STRING system.

When applying the naive bayes algorithm in the system, an improvement of 3.85% above the baseline was achieved with storing the dependencies in XIP.
Fig. 5. Comparison between naive bayes and maximum entropy methods.
during the prediction phase. Although, the results were worse without these dependencies stored, it still offered an improvement of 1.08% above the baseline using this linguistic and computational simplification.

From the 24 verbs used in this dissertation, the difference between the two machine learning approaches is 0.6% with the naive bayes algorithm achieving the higher accuracy. In general, the usage of both modules yielded better results and the system was able to achieve a final score of 67.7% accuracy, an improvement of 3.85% above the baseline (63.86% accuracy).

To improve the system’s results, since the training corpus for this thesis is solely composed of journalistic text, when applying the training data as evaluation corpus in the evaluation using the Weka software package, the ML approaches achieved higher accuracy than using the evaluation corpus, it would be interesting to use a different approach in the training data, to check if there is an improvement in the system’s overall results.

Another suggestion regarding the ML approach is to try out and possibly include a meta-classifier. This classifier would use each method integrated in STRING as weak classifiers, where each one these classifiers would be weighted, since each one of the approaches still have room for improvements. For weak classifiers, it would be considered both naive bayes and maximum entropy algorithms, as well as the rule-based disambiguation system.

Finally including other methods of supervised learning such as SVMS or Bayes Net, where these algorithms achieved high accuracy in the evaluation using the Weka software package, could enhance the meta-classifier’s overall results, since there would be a larger number of weak classifiers with similar accuracy.


