Semantic Classification of Nouns

Rita Alexandra Correia Soares Amaro Policarpo

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisors: Doutor Nuno João Neves Mamede

Doutor Jorge Manuel Evangelista Baptista

May 2015

ACKNOWLEDGEMENTS

At first I would like to thank to my parents for their support, backing and incentive through all the years of my graduation, and mostly for being there in moments of struggle, and never letting me give up on this personal goal.

I would also like to thank to my advisors, professor Nuno Mamede and professor Jorge Baptista, for their dedication, incentive and collaboration on helping me develop this work.

At last, I would like to thank to my colleague Gonçalo Suissas for his patience e availability to help me whenever he could, his contribution was also important to develop this work.

Lisboa, May 14, 2015

Rita Alexandra Correia Soares Amaro Policarpo

RESUMO

Esta dissertação apresenta vários métodos que podem ser utilizados para obter a classificação semântica de substantivos, testando a aplicabilidade de um desses algoritmos, através do estudo da qualidade dos resultados obtidos.

Recorrendo ao uso de uma técnica de aprendizagem automática, co-training, espera-se que o sistema permita aumentar o número de substantivos portugueses semanticamente classificados no léxico de um sistema de Processamento de Língua Natural. Este algoritmo recebe como dados de entrada um conjunto de nomes previamente classificados, designados de *sementes*, rotulados de acordo com um conjunto existente de categorias semânticas, e realiza uma extensa pesquisa cíclica num corpus de treino que visa obter frases que contenham essas palavras-sementes e de seguida comparar essas frases com o restante corpus, a fim de extrair frases de estrutura semelhante que contenham outras palavras que se encaixem no mesmo contexto da palavra-semente, analisando para isso a estrutura sintática das frases, nomeadamente as dependências entre os seus constituintes. Assim, serão retiradas conclusões acerca de novos substantivos que devem receber como classificação o rótulo da categoria semântica da palavra-semente com a qual se assemelham em termos do contexto, isto é, estrutura sintática da frase em que ocorrem.

As propostas de classificação do algoritmo são então apresentadas sob a forma de uma listagem de substantivos identificados como pertencentes à categoria semântica escolhida pelo utilizador. Estas propostas ficam sujeitas a aprovação de um utilizador quanto à sua correção, permitindo, assim, a ampliação da base de dados de substantivos portugueses semanticamente classificados, depois de aprovados. Foi desenvolvida uma interface gráfica para facilitar a interação do utilizador com a aplicação, a fim de permitir a configuração dos parâmetros de execução, bem como da visualização dos resultados finais, sob a forma de propostas de classificação, permitindo que o utilizador avalie a sua correção.

ABSTRACT

This dissertation presents several methods that can be used to achieve a semantic classification of nouns, testing the applicability of one of these algorithms, co-training, through the study of the quality of the results obtained.

Using a machine learning technique, co-training, we expect the system to increase the number of Portuguese nouns semantically classified in the lexicon of a Natural Language Processing System. This algorithm receives as input a set of names previously classified, called *seeds*, labeled in accordance with an existing set of semantic categories, and it performs an extensive cyclic search on a training corpus that aims to obtain sentences containing such seed-words, and next compares these sentences with the remaining sentences in the corpus in order to extract other sentences with matching structure that contain other words that fit the same word-seed context, by analyzing the syntactic structure of the sentences, namely the dependencies. This way, conclusions arise about new nouns that must receive as classification label the semantic category of the word seed with which they resemble in terms of the context, as in, the sentence where they occur.

The proposed classifications of the algorithm are then presented as a list of nouns identified as belonging to the semantic category selected by the user. These proposals are subject to approval by a user, allowing the expansion of the Portuguese nouns semantically classified database, if approved. A graphic interface was developed to facilitate the interaction between user and application, in order to configure its execution parameters, and to visualize the final results obtained, as proposals for classification, allowing the user to evaluate their correction.

PALAVRAS-CHAVE
KEYWORDS

PALAVRAS-CHAVE

Processamento de Língua Natural

Classificação Semântica de Nomes

Aprendizagem automática

Co-Training

KEYWORDS

Natural Language Processing

Nouns Semantic Classification

Machine Learning

Co-Training

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ACRONYMS LIST

WSD – Word Sense Disambiguation	
ML – Machine Learning	

WSD – Word Sense Disambiguation

ML – Machine Learning

NLP – Natural Language Processing

POS – Part of Speech

STRING – Statistical and Rule-Based Natural Language Processing Chain

XIP – Xerox Incremental Parser

NLTK – Natural Language Toolkit

ME – Maximum Entropy

MT – Machine Translation

ANN – Artificial Neural Networks

BP – Back propagation

SVM – Support Vector Machines

EM – Expectation Maximization

KNN – K-Nearest Neighbors

1 INTRODUCTION

1.1 MOTIVATION

This work focus on the semantic classification of nouns, aiming to expand the existing database of Portuguese semantically classified nouns using a machine learning algorithm, co-training, chosen from a set of appropriate algorithms presented further in Section 2.3*.*

Machine learning algorithms can be organized in two different types: *supervised* and *unsupervised*. These learning algorithms are distinguished and applied according to the type of input available during the training phase of the algorithm:

- **Supervised** – These methods use labelled training examples, i.e., input where the desired output is known;
- **Unsupervised** – These methods use unlabeled training examples, i.e., input where the desired output is unknown.

To these, a third intermediate method can be added:

- **Semi-supervised** – These methods combine both labelled and unlabeled examples to generate an appropriate function or classifier.

According to these definitions, we considered that a semi-supervised learning algorithm would be appropriate to solve the classification problem, due to the existence of a small set of labelled nouns versus a much larger set of unlabeled nouns, being this difference in the size of these sets the reason why the Co-Training algorithm (Blum and Mitchell, 1998) was chosen. The Co-Training algorithm is described in Section 2.3.7.

The application of the Co-Training algorithm was based on the use of training data selected from the set of 5.000 already classified Portuguese nouns and the respective contexts of those nouns obtained from the CETEMPúblico corpus (Rocha and Santos, 2000).

To understand the value of classifying nouns according to semantic tags, one must have in mind two essential concepts that are the basis of this work: *syntax* and *semantics*. While semantics is the discipline that studies the meaning of words, syntax is the discipline that studies the rules governing the formation of sentences in (natural) languages. Thus, is the part of the grammar that studies the arrangement of words in a phrase, considering their logical relation among the many possible combinations, and the different meanings that can be derived from each combination. Both these disciplines are essential to understand another concept: *lexical semantics*.

Lexical semantics is the study of how and what the words of a language denote (Pustejovsky, 1998). The units of meaning in lexical semantics are lexical units. Thus, words can be categorized as concepts representing different kinds of entities, using categories for those lexical units, like ‘Person’, ‘Organization’ or ‘Location’, among others - these categories correspond to semantic tags. The meanings of these lexical units come from the words' individuality but also from how they relate with other linguistic elements, such as how these words relate to other words, phrases, symbols and punctuation, thus it is established by looking at its neighborhood, as in, by looking at the other words that occur in the sentence. The studies on lexical semantics are useful to solve the problem of Word Sense Disambiguation (WSD) (Lee and Ng, 2002). Word Sense Disambiguation is the task responsible for selecting the appropriate sense (meaning) to a given word in a context, where different senses potentially attributable to that word exist. These senses can be seen as the labels of a classification problem. Thus, machine learning (ML) is a natural method to solve this problem.

One of the research areas of Natural Language Processing (NLP) is related to the human-computer interaction, which is, enabling computers to derive meaning from human or natural language input. NLP tries to develop software that works with voice recognition systems varying from search engines, speech recognition applications, automatic generators of summaries, spellcheckers, among many others. For such applications, applying WSD is fundamental to solve lexical ambiguity, which is the existence of polysemous words that can express completely different things and whose appropriate sense in a given sentence is only distinguishable by the context analysis.

Although this work does not intend to solve WSD, the task of nouns semantic classification can be seen as a sub-problem of WSD, because it classifies nouns with tags representing all the possible attributable senses, supplying the contents for the disambiguation task.

1.2 Strategy

The architecture of the solution implemented follows a machine learning approach to a classification problem, based on (Bird et al., 2009). We took advantage of the tools and resources available at L^{2}F to supply the necessary data to the application. Thus, this solution is composed of four main components:

- The STRING system;
- The set of features of XIP (morphological, syntactic and semantic labels);
- The set of 5.000 already classified Portuguese nouns and their contexts obtained from a corpus;
- A co-training algorithm implementation, for the automatic learning process.

The STRING (Statistical and Rule-Based Natural Language Processing Chain) (Mamede et al., 2012), a tool developed at INESC-ID, is used for the basic processing tasks, like tokenization and text segmentation, part-of-speech (POS) tagging, morphosyntactic disambiguation and parsing (chunking and syntactic-semantic dependency extraction), being this last step developed by XIP. Together with XIP (XEROX Incremental Parser) (Ait-Mokhtar et al., 2002) and its set of defined features, the STRING system is an important processing chain available at L^{2}F and sustain the development of this solution.

The input of the solution are the noun-context pairs, which are supplied from the processing of the set of classified nouns, together with their contexts, obtained from CETEMPúblico (Rocha and Santos, 2000). These constitute the contextual features described in Section 3.2, and are processed using the STRING system described in Section 3.3.

The set of labels are part of the XIP set of features, developed for the Portuguese grammar, built at L^{2}F, and are described in Section 3.1.

The solution presents a cyclic execution, because the co-training algorithm uses the predictions for unlabeled data as new seeds that are added to the set of seeds and used to iteratively construct additional labelled training data.

This work includes the development of a simple interface for human users, who are the judges of the correction of the predictions provided by the algorithm, thus they are ultimately responsible for accepting or rejecting a prediction. The correct expansion of the semantically classified nouns database is thus responsibility of the human judge. According to this, this work is subject to human evaluation, as described later in Section 3.5.

1.3 GUIDE

This document is divided in main chapters covering the most important aspects to the development and understanding of the problem presented at this work.

Chapter 2 presents the state of art of machine learning algorithms that can be used for classification tasks similar to the one presented in this work.

The architecture of the solution is described in Chapter 3.

The concrete implementation of this project is described in Chapter 4.

Chapter 5 presents the results obtained in the classification task of this work.

At last, Chapter 6 describes the problems found during the implementation of this work along with proposed solutions that aim to improve the performance of the system, together with a brief conclusion about the applicability of the chosen algorithm to the task and contents available.

2 STATE OF ART

In this section, related works to the classification task developed in this project are presented, introducing brief descriptions of the different types of algorithms existing to achieve the same end and their reported applications in similar works of Natural Language Processing.

2.1 MACHINE LEARNING

Machine learning consists of having a computer algorithm learning from data. To clarify the definition of learning that suits the scope of the machine learning task carried out in this dissertation, the following citation applies:

"Things learn when they change their behavior in a way that makes them perform better in the future." (Witten et al., 2011).

The words 'learn' and 'train' have been used interchangeably in the literature, however, 'training' might be more appropriate for computational applications, since 'learning' has an intelligent component that training does not.

Using predefined features, ML algorithms identify patterns in the data and can therefore infer predictions. In NLP, the term machine learning is preferred, whereas in Information Technology and Business Intelligence applications the term data mining applies for the same purpose.

In machine learning, there are two kinds of learning: *supervised* and *unsupervised*. Supervised learning relies on a set of previously established senses, the labelled training data, to infer predictions. In unsupervised learning, those classes are not labelled, and therefore the algorithm must perform clustering of similar classes to find correlations, prior to disambiguation.

NLP applications evolved from rule-based systems, whose usage is currently limited to domain-specific applications, to hybrid approaches combining rule-based approaches (like ML) with statistical measures (Emms and Luz, 2011).

Rule-based systems' architecture (Stella and Chuks, 2011) include the use of an inference engine or semantic reasoner to infer information based on the interaction of input and the rule base (knowledge base), and an interpreter that executes the production system program.

Many NLP applications were subject of this paradigm shift, such as Machine Translation, POS tagging (pre-processing tasks) (Márquez et al., 2000) and Word Sense Disambiguation (Specia, 2007) (Kulkarni et al., 2008). Many of the most recent tools for NLP researchers contain plug-ins and integration to ML toolkits, like MALLET (McCallum, 2002) and the Natural Language Toolkit (NLTK) (Bird, 2009).

2.1.1 KEY CONCEPTS

To better understand the process of machine learning and its application, it is very important to understand three key concepts: *input*, *output* and the *algorithms* used for learning. When choosing an algorithm, it is important to have an intuition of what each machine learning algorithm does in practice, and its respective advantages and disadvantages.

According to (Witten et al., 2011) there are four styles of learning, or concepts, to be learned within ML applications. They are:

- **Classification Learning** – A supervised model which is provided with the actual outcome or class for each of the training examples. Classification algorithms organize instances according to these predefined classes, which means that they aim to predict the class of each instance;
- **Association** – Differs from classification, since it searches for relations between variables, usually in large data sets. Methods in association learning try to search for patterns in data in which two or more variables combined are likely to provide a given result;
- **Clustering** – An unsupervised method, where the classes are not previously defined. The algorithm groups items together automatically, according to similarities found on data;
- **Numeric Prediction** – Generates an outcome that is a numeric quantity rather than a discrete class, unlike the three previous styles of learning.

Figure 1 presents a sample scheme of a machine learning approach to a classification problem, based on (Bird et al., 2009).

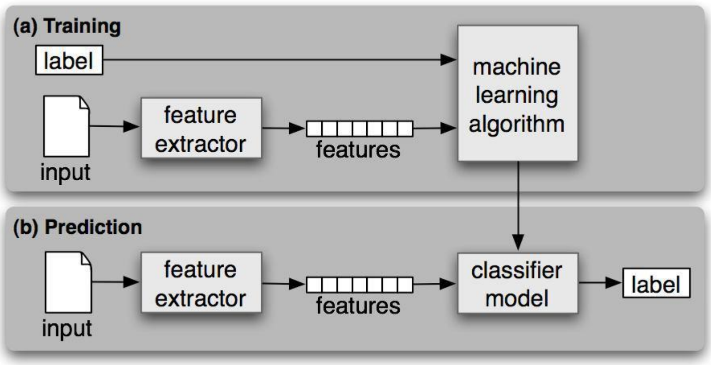


Figure 1: Training in a classification problem (Bird et al., 2009).

The model is divided in two boxes, (a) training and (b) prediction, also called testing.

In the training phase, the input data is prepared using a feature extractor module to convert raw data into features. The set of possible labels are previously defined, therefore constituting a supervised approach, and they are also provided to the ML algorithm. Then, the ML algorithm applies learning methods to the features and labels to create a model of classification that can be used for new or previously unseen data.

Once the model is created, the label prediction presented in (b) is applied, using the same features and a sample of the same dataset as the training dataset. The classifier model will determine, based on the features and attributes, which class the new test instance belongs to.

The input in ML can either take the form of concepts, instances, attributes, or of a concept description.

In a logical sequence, the definitions discussed so far can be organized as follows:

- **Concept Description** – General concept to be learned;
- **Concepts** – Individual units to be learned;
- **Instances** – Examples of the concept to be learned, given to the learner, extracted from the data and organized with attributes;
- **Attributes** – Predefined values for each instance (these can be numeric, nominal, binary, etc.).

In classification learning, a set of classified examples is presented within a concept, from which it is expected to learn to classify unseen examples. This learning is considered supervised because the outcome (class) of the training examples is known and provided for the operation of the algorithm.

The output of any ML system is known as knowledge representation, and it consists in patterns that are discovered or learned from the data by the different ML methods. Thus, the output can be presented and schematized in many ways, depending on the algorithm used.

2.1.2 ALGORITHMS AND OUTPUTS

Due to ML methods' popularity, a large number of algorithms have emerged, including: Naïve Bayes or Bayesian Classifiers, Support Vector Machines, Decision Trees (C4.5), Regression trees, Maximum Entropy, Conditional Random Fields, Co-training, K-means, Expectation Maximization (EM), Boosting, Learning Set of Rules, Neural Networks, Bayesian Networks, K-Nearest Neighbors, Yarowsky algorithm, among others.

In the next subsections some of these algorithms, along with the kind of output representation produced and their applications in state-of-art applications, will be described in more detail. For this description and presentation of outputs produced, a couple of fictitious examples are used: "the weather problem" (Witten et al., 2011) whose dataset contains instances of weather conditions that are suitable for playing, or not, some unspecific game, and "sex classification" (Strickland, 2014) whose dataset contains instances of measured human features (height, weight and foot size) that are suitable to classify a given person as a male or female.

2.1.2.1 Naïve Bayes Classifier

The Naïve Bayes Classifier is a simple probabilistic model with strong naïve independence assumptions, which is the reason why it is described in the literature as an “independent feature model”. It produces probability estimates rather than predictions. It was first proposed by (Duda and Hart, 1973) and is based on Bayes Theorem.

The key idea of the Bayes Theorem is that the probability of an event A given an event B depends not only on the relation between A and B but on the absolute probability, or occurrence, of A not concerning B, as well as the absolute probability of B not concerning A. *Equation 1* presents this assumption.

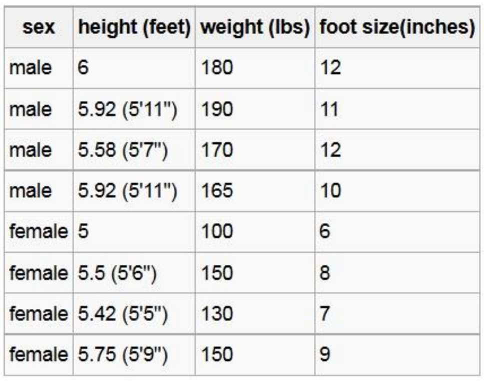
$$P(A|B) = \frac{P(B|A)\ P(A)}{P(B)}$$

Equation 1: Bayes theorem

In *Equation 1*, *P(A)* is the marginal probability of A, *P(A|B)* is the conditional probability of A given B (posterior probability), *P(B|A)* is the conditional probability of B given A (likelihood) and *P(B)* is the marginal probability of B, which acts as a normalizing constant.

Thus, a Naïve Bayes Classifier produces probabilities as output. The probabilities obtained can be used, for example, to distinguish to which classified set an unclassified sample belongs, depending on the highest probability obtained, among the probabilities of belonging to the various sets.

As an illustrating example, consider the training set from *Figure 2* that is used to classify whether a given person is a male or a female based on the measured features height, weight, and foot size.

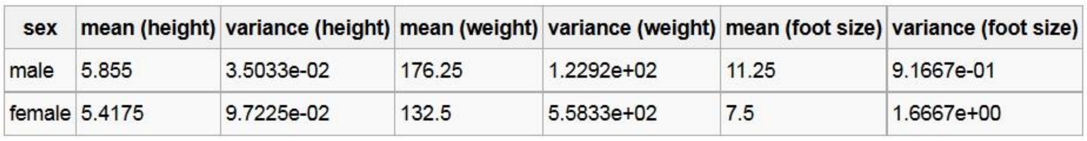


sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

Figure 2: Sample training set of the “sex classification” example.[1]

[1] Source: http://en.wikipedia.org/wiki/Naive_Bayes_classifier (Date: 30-12-2014)

From this training set, applying a Gaussian distribution assumption would result in the classifier of *Figure 3* (note that the mean is represented by µ and the variance is represented by σ^2).



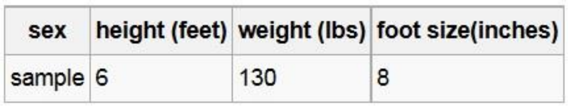
sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Figure 3: Classifier of the "sex classification" example, using a Gaussian distribution.[2]

$$f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\,\sigma^2}}$$

Equation 2: Gaussian distribution

The objective is to classify, with the provided data, the sample from *Figure 4*.



sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

Figure 4: Unclassified sample of the "sex classification" example. [2]

One must decide if this sample must be classified as male or female. For this, it is necessary to determine which posterior is greater. For these calculations, it is necessary the information from *Figure 2*, and the assumption that we have equiprobable classes. The calculations are shortly presented in *Equations 3-11*.

$$P(male) = P(female) = 0{,}5$$

$$P(height = 6|male) = \frac{1}{\sqrt{2\pi \times 3{,}5033 \times 10^{-2}}} e^{-\frac{(6-5{,}855)^2}{2\times 3{,}5033\times 10^{-2}}} \approx 1{,}5789 \times 10^{-6}$$

$$P(weight = 130|male) = \frac{1}{\sqrt{2\pi \times 1{,}2922 \times 10^{2}}} e^{-\frac{(130-176{,}25)^2}{2\times 1{,}2922\times 10^{2}}} \approx 5{,}9881 \times 10^{-6}$$

$$P(footsize = 8|male) = \frac{1}{\sqrt{2\pi \times 9{,}1667 \times 10^{-1}}} e^{-\frac{(8-11{,}25)^2}{2\times 9{,}1667\times 10^{-1}}} \approx 1{,}3112 \times 10^{-3}$$

$$P(height = 6|female) = \frac{1}{\sqrt{2\pi \times 9{,}7225 \times 10^{-2}}} e^{-\frac{(6-5{,}4175)^2}{2\times 9{,}7225\times 10^{-2}}} \approx 2{,}2346 \times 10^{-1}$$

[2] Source: http://en.wikipedia.org/wiki/Naive_Bayes_classifier (Date: 30-12-2014)

$$P(weight = 130|female) = \frac{1}{\sqrt{2\pi \times 5{,}5833 \times 10^2}} e^{-\frac{(130-132{,}5)^2}{2\times 5{,}5833\times 10^2}} \approx 1{,}6789 \times 10^{-2}$$

$$P(footsize = 8|female) = \frac{1}{\sqrt{2\pi \times 1{,}6667}} e^{-\frac{(8-7{,}5)^2}{2\times 1{,}6667}} \approx 2{,}8669 \times 10^{-1}$$

$$P(male\ denominator) = P(male) \times P(height = 6|male) \times P(weight = 130|male) \times P(footsize = 8|male)$$

$$P(female\ denominator) = P(female) \times P(height = 6|female) \times P(weight = 130|female) \times P(footsize = 8|female)$$

$$\begin{aligned} &P(male|height = 6, weight = 130, footsize = 8) \\ &= \frac{P(male) \times P(height = 6|male) \times P(weight = 130|male) \times P(footsize = 8|male)}{P(male\ denominator) + P(female\ denominator)} \\ &= \frac{0{,}5 \times 1{,}5789 \times 10^{-6} \times 5{,}9881 \times 10^{-6} \times 1{,}3112 \times 10^{-3}}{0{,}5 \times 1{,}5789 \times 10^{-6} \times 5{,}9881 \times 10^{-6} \times 1{,}3112 \times 10^{-3} + 0{,}5 \times 2{,}2346 \times 10^{-1} \times 1{,}6789 \times 10^{-2} \times 2{,}8669 \times 10^{-1}} \\ &= \frac{6{,}1984 \times 10^{-9}}{6{,}1984 \times 10^{-9} + 5{,}3778 \times 10^{-4}} \approx 1{,}1526 \times 10^{-5} \end{aligned}$$

$$\begin{aligned} &P(female|height = 6, weight = 130, footsize = 8) \\ &= \frac{P(female) \times P(height = 6|female) \times P(weight = 130|female) \times P(footsize = 8|female)}{P(male\ denominator) + P(female\ denominator)} \\ &= \frac{0{,}5 \times 2{,}2346 \times 10^{-1} \times 1{,}6789 \times 10^{-2} \times 2{,}8669 \times 10^{-1}}{0{,}5 \times 1{,}5789 \times 10^{-6} \times 5{,}9881 \times 10^{-6} \times 1{,}3112 \times 10^{-3} + 0{,}5 \times 2{,}2346 \times 10^{-1} \times 1{,}6789 \times 10^{-2} \times 2{,}8669 \times 10^{-1}} \\ &= \frac{5{,}3778 \times 10^{-4}}{6{,}1984 \times 10^{-9} + 5{,}3778 \times 10^{-4}} \approx 0{,}9999 \end{aligned}$$

Equations 3 to 11: Calculations for the classification of the sample in Figure 4 from the "sex classification" example. [3]

According to the calculations in Equations 3-11, we observe that the highest probability corresponds to the possibility of the sample belong to the female class, so it is classified as female.

In simple terms, a Naive Bayes Classifier assumes that the presence (or absence) of a particular feature of a class is unrelated to the presence (or absence) of any other feature. Even if these features depend on each other or upon the existence of the other features, a naive Bayes Classifier considers all of these properties as independent contributors to the final probability.

[3] Source: http://en.wikipedia.org/wiki/Naive_Bayes_classifier (Date: 30-12-2014)

The advantages of the Naive Bayes Classifier consist on its simplicity and over-simplified assumptions, in requiring only a small amount of training data to estimate the parameters necessary for classification, and in the fact that the classifier can be trained very efficiently in a supervised learning dataset. Naive Bayes Classifiers have worked quite well in many complex real-world situations, impressing researchers such as (Zhang, 2004):

> *"Naive Bayes is one of the most efficient and effective inductive learning algorithms for machine learning and data mining. Its competitive performance in classification is surprising, because the conditional independence assumption on which it is based, is rarely true in real-world applications."* (Zhang, 2004)

As an example of use of Naive Bayes in NLP, works like (Sebastiani, 1999) applied this algorithm for Text Categorization.

2.1.2.2 Maximum Entropy

The Maximum Entropy model (ME or Maxent) is a general purpose ML framework for estimating probability distributions from data, and has been successfully applied in various fields of research, including spatial physics, computer vision, and also NLP.

Its basic principle is that when nothing is known, the distribution should be as uniform as possible, to present the maximal entropy.

In a classification task, labelled training data is used to derive a set of constraints (represented as expected values of features or any real-valued function of an example) for the model that characterizes the class-specific expectations for the distribution.

Maximum entropy classifiers are an alternative to naive Bayes classifiers for situations where the features that act as predictors do not assume statistical independence. However, maximum entropy classifiers are slower than naive Bayes classifiers, thus they are not suitable when a very large number of classes to learn is given.

By being similar to naive Bayes classifiers, the output takes the form of probabilities as well.

There are several applications of Maximum Entropy in NLP: (Berger et al., 1996) proposes its use in an English - French Machine Translation (MT) system, (Nigam, 1999) used it for Text Classification and (Ratnaparkhi, 1996) used it for Part-Of-Speech Tagging.

2.1.2.3 Decision Trees Algorithms (C4.5)

Decision tree learning uses the divide-and-conquer strategy that recursively partitions the data to produce the tree. At the beginning, all the examples are at the root. As the tree grows, the examples are sub-divided recursively.

The general algorithm for building decision trees is, based on (Kotsiantis, 2007):

1. Check for base cases;
2. For each attribute a;
    a. Find the normalized information gain ratio from splitting on a;
3. Let a_{best} be the attribute with the highest normalized information gain;
4. Create a decision node that splits on $a_{best;}$
5. Apply recursion on the sub-lists obtained by splitting on a_{best}, and add those nodes as children of node.

One of the most famous and widely used implementations is the C4.5 algorithm, proposed by (Quinlan, 1993), a successor of the IDE3 algorithm, also invented by (Quinlan, 1986).

The algorithm C4.5 builds decision trees from a set of training data (similar to what ID3 does) using the concept of information entropy.

The training data is a set $S = \{s_1, s_2, ...\}$ of already classified samples where each sample s_i consists of a p-dimensional vector $(x_{1,i}, x_{2,i}, ...,x_{p,i})$, where the x_j represent attributes or features of the sample, as well as the class in which s_i falls.

At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other.

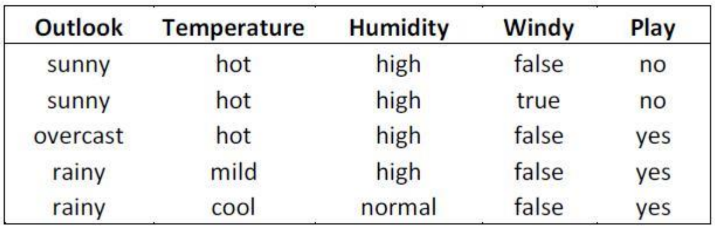
The criteria used for splitting is the normalized information gain (difference in entropy). Thus, the decision is based on the attribute with the highest normalized information gain. The C4.5 algorithm then applies recursion on the smaller sub-lists.

This algorithm has a few base cases.

- **All the samples in the list belong to the same class** – The algorithm simply creates a leaf node for the decision tree saying to choose that class;
- **None of the features provide any information gain** – The algorithm creates a decision node higher up the tree using the expected value of the class;
- **Instance of previously-unseen class encountered** – The algorithm creates a decision node higher up the tree using the expected value.

The output of a decision tree algorithm can take the form of a decision table or a decision tree. Consider "the weather problem" (Witten et al., 2011) mentioned before:

The simplest and most rudimentary form of representing the output from ML is a decision table, as represented in *Figure 5*.



Outlook	Temperature	Humidity	Windy	Play
sunny	hot	high	false	no
sunny	hot	high	true	no
overcast	hot	high	false	yes
rainy	mild	high	false	yes
rainy	cool	normal	false	yes

Figure 5: Sample decision table (Zampieri, 2010) for the weather problem (Witten et al., 2011).

The four leftmost columns are parameters extracted from the original input data, whereas the fifth column (Play) represents the predicted class from each of the instances, with the values yes or no.

For the same problem, the resulting decision tree is represented in *Figure 6*.

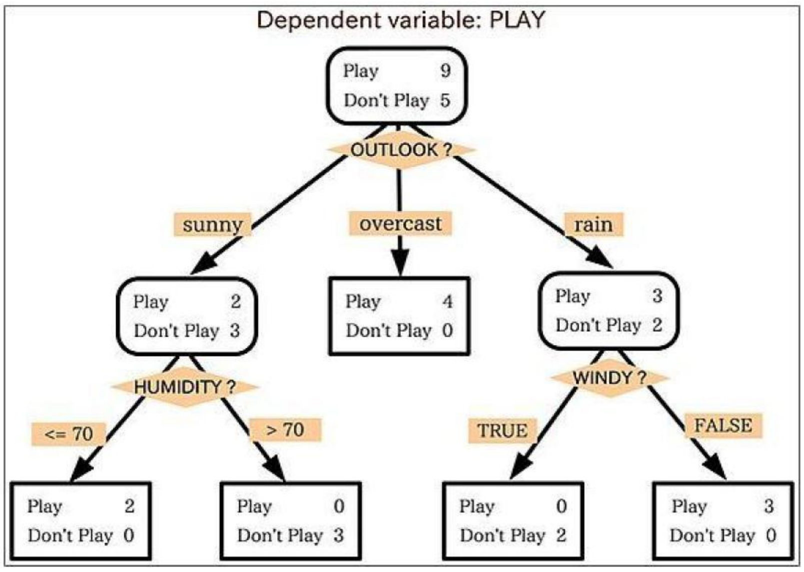


Figure 6: Sample decision tree (Zampieri, 2010) for the weather problem (Witten et al., 2011).

Decision Trees are especially useful when dealing with a limited number of variables, such as the weather problem. In cases with a larger number of variables, the tree becomes too crowded to be analyzed for predictions made by the algorithm.

Starting at the root of the tree, it represents 14 instances either labelled as play or do not play. According to the variables, the tree expands and shows the expected labels. In the second level, for example, the tree presents different probabilities regarding the two classes, if the weather outlook is sunny, rainy or overcast.

As an example of use of the C4.5 Decision Tree algorithm, works like (Jantan et al., 2009) applied this algorithm for Knowledge Discovery, in this case, Human Talent Prediction for Human

Resources Management in an organization. In (Orphanos et al., 1999) a Decision Tree algorithm was used for POS-tagging.

2.1.2.4 Neural Networks

A perceptron works as follows:

Consider that x_1 through x_n are input feature values and w_1 through w_n are connection weights/prediction vector (usually real numbers in the interval [-1,1]) then perceptron computes the sum of weighted inputs $\sum_i x_i w_i$ and output goes through an adjustable threshold: if the sum is above threshold, output is 1, else is 0.

Uses of the perceptron algorithm to learn from a batch of training instances chose to run the algorithm repeatedly through the training set until a prediction vector that is correct on all of the training set is found. This becomes a prediction rule, which is then used for predicting the labels on the test set.

Perceptrons are only able to classify linearly separable sets of instances, which means, cases where it is possible to draw a straight line or plane separating the input instances into their correct categories. When this cannot be done, learning will never reach a point where all instances are properly classified. Artificial Neural Networks (ANN) were created in an attempt to solve this problem.

For an overview of existing work in Artificial Neural Networks, consult (Zhang, 2000).

A multilayer neural network's output consists of a large number of units (neurons) joined together in a pattern of connections, as exemplified in *Figure 7*.

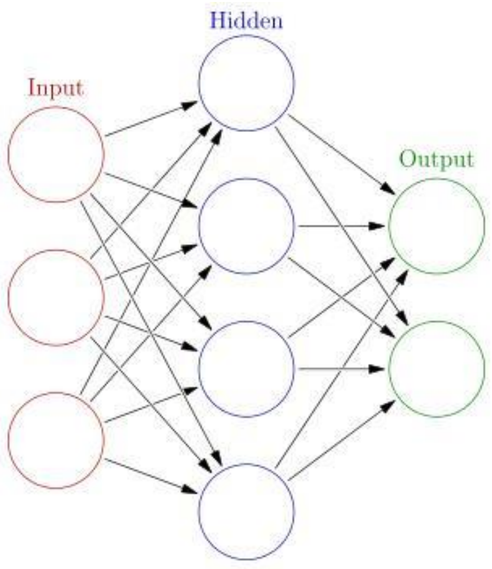


Figure 7: Multilayer ANN. [4]

[4] Source: http://en.wikipedia.org/wiki/Artificial_neural_network (Date: 19-12-2013)

Units, in a net, can be categorized in one of the three following classes:

- **Input** – Units that receive information to be processed;
- **Output** – Units that hold the results of the processing;
- **Hidden** – Units in between.

Feed-forward ANNs are one-way networks, where the signal can only travel from input to output. At first, the network is trained on a set of paired data to determine the mapping from input to output. Then, the weights of the connections between neurons are fixed and the network is used to determine the classifications of a new set of data.

The current values of the weights are thus determinant to the behavior of an ANN.

Initially, the weights of the net to be trained are set to random values, and instances of the training set are repeatedly exposed to the net. To the input units are assigned the values for the input of an instance, and the output of the net is compared with the desired output for this instance. Then, all the weights in the net are adjusted slightly in the direction that would bring the output values of the net closer to the values for the desired output.

The most well-known and widely used learning algorithm to estimate the values of the weights is the Back Propagation (BP) algorithm.

The greatest problem of neural networks is being too slow, and thus having a much longer training time than a decision tree, for example.

Works like (Socher et al., 2013) used this algorithm for the creation of Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank.

2.1.2.5 K-Nearest Neighbor

Nearest neighbor algorithms belong to the category of Instance-Based Algorithms, under the class of Statistical Methods.

Instance-based algorithms are lazy-learning algorithms, because they first perform the classification leaving the induction or generalization process delayed.

K-Nearest Neighbor (kNN) assumes that within a dataset, instances with similar properties will exist in close proximity. If some instances are tagged with a classification label, then the value of the label of an unclassified instance can be determined by observing the class of its nearest neighbors. The kNN locates the *k* nearest instances to the query instance and determines its class by identifying the single most frequent class label.

Instances can be considered as points within an n-dimensional instance space where each of the n-dimensions corresponds to one of the n-features that are used to describe an instance. The relative distance (determined by a distance metric) between instances is more significant than the absolute position of the instances within the space.

Ideally, the distance metric used must minimize the distance between two similarly classified instances, while maximizing the distance between instances of different classes.

Figure 8 presents an example of usage of a kNN classifier.

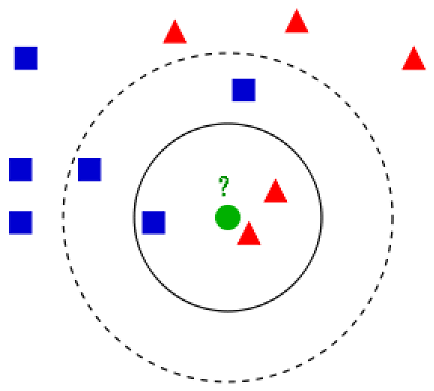


Figure 8: Example of kNN classification. [5]

Considering *Figure 8*, the green circle, that corresponds to the test sample should be classified either to the first class of blue squares or to the second class of red triangles, according to the amount of instances of each class existing in its proximity. For example:

- For *k* = *3* (represented as the solid line circle) it is assigned to the second class of red triangles, because there are 2 triangles and only 1 square inside the inner circle;
- For *k* = *5* (dashed line circle) it is assigned to the first class of blue squares, because there are 3 squares and only 2 triangles inside the outer circle.

The disadvantages of the kNN include large storage requirements, sensitiveness to the similarity function used to compare instances, lack of principled way to choose *k*, except through cross-validation computationally-expensive technique, and large computational time for classification.

Works like (Toker and Kirmemis, n.d.) applied this algorithm for Text Categorization.

2.1.2.6 Support Vector Machines

Support Vector Machines (SVM), also known as Support Vector Networks, are supervised learning methods and algorithms used to analyze data and recognize patterns that can be used for classification and regression analysis.

[5] Source: http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm (Date: 27-12-2013)

The basic SVM is a non-probabilistic binary linear classifier, because it takes a set of input data and for each given input it predicts which of two possible classes forms the output.

The input data consists in a set of training examples, where each example must be marked as belonging to one of two categories. The algorithm then builds a model that assigns new examples into one, and only one, of those categories.

In an SVM model, examples are represented as points in space, mapped so that the examples of each of the categories are divided by a gap that is as wide as possible. The new examples are then mapped into that same space and the prediction about their category is based on the observation of which side of the gap they fall on: this is what determines the category to where they belong.

In other words, it is a method for creating a classification function that tries to find an hyper-surface in the space of possible inputs that splits the positive examples from the negative examples for each category.

Figure 9 presents an example of the output of a SVM classifier, represented by the points in space, and the choice of the best hyper plane that splits the two sets, in this case.

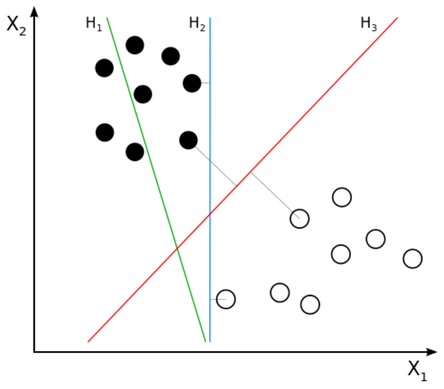


Figure 9: Sample SVM classifier points in space and possible hyper planes. [6]

Considering *Figure 9*, hyper plane *H1* does not separate the classes, so it is not a valid choice. Hyper plane *H2* separates the classes, but only with a small margin - as mentioned before, the gap must be as wide as possible! Thus, hyper plane *H3* separates them with the maximum margin, and represents the best choice.

Works like (Nulty, 2007) tested an SVM application with the Sequential Minimal Optimization method proposed by (Platt, 1998), provided by the Weka software package, to label modifier-noun compounds with a semantic relation, that was used as input the web frequencies for phrases containing the modifier, noun, and a prepositional joining term.

[6] Source: http://en.wikipedia.org/wiki/Support_vector_machine (Date: 27-12-2013)

The use of SVM for text classification was presented in (Joachims, 2002). Works like (Giménez and Márquez, 2004) present SVMTool: a general POS tagger generator based on Support Vector Machines.

2.1.2.7 Co-Training Algorithm

Co-training (Blum and Mitchell, 1998) is a ML algorithm suitable for situations when there are two distinct sets with significant size difference: a small set of labelled data and a large set of unlabeled data. It is mostly used for text mining and search engines.

It consists on a semi-supervised learning technique that enforces the existence of two views of the data to describe each example. These views are feature sets that provide different, complementary information about the instance. They must preferably be conditionally independent and each one must be individually sufficient to predict the class of an instance.

The algorithm works by first learning a separate classifier for each view using any labelled examples, which constitute the initial seeds set. Then, each prediction of each classifier on the unlabeled data becomes a new seed that is added to the set and used to iteratively construct additional labelled training data.

Co-training produces rules as output. These rules can be either classification or association rules.

Classification rules are structured with an antecedent and a consequent (also known as a conclusion) that gives the class(es) that apply to instances covered by that rule. As an example, for the weather problem (Witten et al., 2011):

- *If* outlook = sunny *and* humidity = high *then* play = no
- *If* outlook = rainy *and* windy = true *then* play = no
- *If* outlook = overcast *then* play = yes

Association rules are derived from the classification rules, but they can predict any attribute and not just the class. Consider the following example, for the same problem as before:

- *If* temperature = cool *then* humidity = normal

Works like (Mota, 2009) use this algorithm for Named Entity Recognition.

2.1.2.8 Yarowsky Algorithm

Yarowsky algorithm (Yarowsky, 1993) is an unsupervised learning algorithm for WSD that is based on two properties of human language: "one sense per collocation" and "one sense per discourse". It means that words tend to exhibit only one sense in most given discourses and in a given collocation, as concluded from observation. The term *collocation* follows its traditional dictionary definition:

“appearing in the same location; a juxtaposition of words”. Here, no idiomatic or non-compositional interpretation is implied.

The process starts by identifying examples of a given polysemous word in a large raw corpus, and storing all the relevant sentences as lines. Then, it is necessary to identify, for each possible sense of a word, a relatively small number of training examples representative of that sense. This can be accomplished by hand tagging a subset of the training sentences. However, to simplify this laborious procedure, one can simply identify a small number of seed collocations representative of each sense, and then tag all the training examples containing the seed collocates with the seed's sense label. The remainder of the examples (typically 85-98%) constitute an untagged residual.

The selection of the seed words must be careful so that the distinction of the possible senses is accurate and productive.

Collocations’ strength is inversely proportional to the distance to the target word: the effect weakens with distance. Seed words appearing in the most reliable collocational relations with the target word will be chosen, according to the criteria given in (Yarowsky, 1993), which means that words in a predicate argument relation will have a stronger effect than others with arbitrary associations, even at the same distance to the target word. Thus, a collocation word can have several collocational relations with the target word, resulting in different rankings or even different classifications.

Alternatively, a single defining collocate (obtained from WordNet, for example) for each class can be identified and contexts containing one of these defining words will constitute the seeds. The frequency of the words will determine the seed collocations’ representative: words that occur near the target word in great frequency are selected. However, this approach is not fully automatic since it needs an human judge to decide which word will be selected for each target word's sense.

Other reliable collocations are identified using a decision list algorithm that calculates the probability *P(Sense|Collocation)*, and ranks the list by the log-likelihood ratio, applying a smoothing algorithm to avoid 0 values. Then, the new resulting classifier is applied to the whole sample set. The examples in the residual that are labelled as either A or B with a probability above a reasonable threshold are added to the seed sets. The two steps (decision-list plus adding) are applied iteratively.

The senses of A or B grow as more newly-learned collocations are added to the seed sets, while the original residual will shrink. However, the probability of these collocations must remain above the threshold to stay in the seed sets, otherwise they are returned to the residual for later classification.

The algorithm iterates until no more reliable collocations are found, and the “one sense per discourse” property is used for error correction. Also, the class-inclusion threshold needs to be randomly altered to avoid strong collocates becoming indicators for the wrong class, and for the same reason the width of the context window needs to be increased after intermediate convergence.

The final decision list containing the target words with the most reliable collocations at the top of the list, replacing the original seeds, is obtained when the algorithm converges on a stable residual set. This decision list consists in the output of the algorithm.

After this, the sense labels and probabilities must be used to tag the original raw corpus, and the final decision list is applied to the new data: the classification uses the collocation with the highest rank in the list.

The results shown in (Yarowsky, 1993) reveal that accurate WSD can be achieved without the cost of a large sense-tagged training corpus. Despite being an unsupervised algorithm, it presented nearly equal or better performance when comparing to other learning algorithms, either unsupervised (achieves 96.7% vs. 92.2%) or supervised (achieves 95.5% vs. 96.1%), actually outperforming supervised methods when the one-sense-per-discourse constraint is used (achieves 96.5% vs. 96.1%).

2.1.3 CROSS-VALIDATION

Evaluation is essential to determine the accuracy of any learning method. As it was previously mentioned, any ML method needs data for training and test. The most common way to do this is to split the data into two sets, usually around 70% for training and 30% for testing.

Although this distribution is commonly used for large datasets, it presents a challenge for smaller datasets and it might lead to problems of representativeness of the training or testing data. Therefore, it is necessary to ensure that random sampling is done in a way that guarantees that each class in the data set is properly represented in both the training and test sets.

To avoid inaccuracy of results due to data splitting, a statistical technique called cross-validation can be applied. In cross-validation, a fixed number (n) of folds or partitions of the data are assigned, and it is referred to as n-fold cross-validation.

In the case of a three-fold cross-validation, data is split into three equal partitions, two of them used for training and the last one is used for testing. The process is repeated three times to ensure that all the instances in the data set were used for training and for testing. For evaluation purposes, the average of the three iterations is calculated.

In order to predict the error rate of a learning technique given a fixed sample of data, the use of 10-fold cross-validation has become common in the ML research community:

"Extensive tests on numerous datasets, with different learning techniques, have shown that 10 is about the right number of folds to get the best estimate of error, and there is also some theoretical evidence that backs this up." (Witten et al., 2011)

3 SOLUTION

As previously introduced in Section 1.2, this project is composed of four main components:

- The STRING system;
- The set of features of XIP (morphological, syntactic and semantic labels);
- The set of 5.000 already classified Portuguese nouns and their contexts obtained from a corpus;
- A co-training algorithm implementation, for the automatic learning process of this work.

The architecture of the solution follows a machine learning approach to a classification problem, previously presented in *Figure 1* from Section 2.1.1 and based on (Bird et al., 2009).

The input of the system are the noun-context pairs, obtained from the set of already classified nouns and the CETEMPúblico corpus (Rocha and Santos, 2000). These constitute the contextual features described in Section 3.2, and are processed using the STRING system described in Section 3.3.

The set of semantic labels are part of the XIP set of features described in Section 3.1.

The co-training algorithm uses the predictions for unlabeled data as new seeds that are added to the set and used to iteratively construct additional labelled training data, thus it presents a cyclic behavior.

In the following sections, the constituent parts of this architecture are described in more detail.

3.1 XIP AND ITS SET OF FEATURES

XIP stands for XEROX Incremental Parser, and, as it was mentioned before, it 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. XIP constitutes the last module of the STRING chain system (Mamede et al., 2012) described in Section 3.3, and is able to perform several tasks, namely:

- Add lexical, syntactic and semantic information;
- Apply local grammars;
- Apply morphosyntactic disambiguation rules;
- Calculate chunks (where sequences of categories are grouped into structures, using chunking rules, to obtain the elementary phrase constituents) and syntactic-semantic dependencies (where nodes are subject to dependency rules to identify relations between them).

The fundamental data representation unit in XIP is the node. A node has a category and a set of features with its respective values. XIP assumes features to be binary values, hence the presence of a feature assumes the node word presents it.

Tree essential concepts apply when consulting the grammar of XIP:

- **Node** – represents the word;
- **Category** – represents the main category of a word (noun, adjective, verb, etc.) - note that the category is also a feature;
- **Feature** – represents any other feature, at any level, of that word:
    - **morphological**: number, gender, tense, etc.;
    - **semantic**: human, locative, time, etc.;
    - **syntactic**: noun phrase, subject, head, etc..

We use these tags to classify a noun based on its surrounding context. With this, we aim to expand the quantity of nouns available with each semantic tag chosen, which is influenced by the data available, as is discussed later in Chapter 5.

The semantic categories to choose for expansion were a delicate subject, because choosing the ones that actually contained less members would mean we would be lacking classified nouns to serve as seeds and sustain the classification task, what would result in less-reliable results, if any of minimal quality, as can be seen in the results obtained and described later in Chapter 5.

To acquire a view over the existing classified nouns and their distribution into semantic tags, an analysis process constituted one of the first tasks developed during this work, as can be seen further in Chapter 4. This analyze processes the output obtained from the STRING chain when applied to the input corpus, constituted by XIP nodes organized in a XML structured file.

Consider the following example demonstrating the interpretation of XIP entities:

```
Pedro: noun[human, individual, proper, firstname, people, sg, masc, maj]
```

This structure must be interpreted as follows:

- Node: Pedro
    - Category: Noun (**noun**)
        - Feature: Human (**human**)
        - Feature: Individual (**individual**)
        - Feature: Proper Noun (**proper**)
        - Feature: First Name (**firstname**)
        - Feature: A person's name (**people**)
        - Feature: Singular (**sg**)
        - Feature: Masculine (**masc**)
        - Feature: Spelled with an upper case initial letter (**maj**)

Where:

- **noun** – represents the POS category;
- **proper, sg, masc, maj** – represents morphological features;
- **human, individual, firstname, people** – represents semantic features.

Some of these features can be discarded from the processing of the algorithm, as can be seen further in Chapter 4.

As shown, features describe the properties of nodes. Features, by themselves, do not exist: only the value they assume is relevant. They can be instantiated, tested, or deleted. It is possible to define custom lexicon files, providing features to enrich the vocabulary.

3.2 SEEDS: CLASSIFIED NOUNS

The seed set of the co-training algorithm is constituted by elements of the already semantically classified 5.000 Portuguese nouns, together with their surrounding contexts, grouped as noun-context pairs.

The 5.000 Portuguese nouns available at L^{2}F were initially classified according to (Bick, 2006) prototypes and then manually revised. Their semantic tags are part of the hierarchical set of labels existing as features of the XIP module and described before.

These nouns have a major problem: ambiguous nouns present sets of features that were not separated into different entries; thus *banco* will assume the meaning of ‘institution’ and ‘furniture’. Therefore, for each ambiguous noun, a manual selection of the appropriate features to be used as seeds for a certain classification must be done. As an example, if we intend to classify pieces of furniture, we will select the semantic classification of ‘furniture’ from *banco* as seed, ignoring the other alternative features. If we intend to classify institutions, the seeds will include the ‘institution’ classification of *banco*. Another alternative would be rejecting the use of ambiguous nouns as seeds, however, this would lead to problems of representativeness of seeds for certain classifications, negatively influencing the results.

The STRING processing chain deals with ambiguity in a very convenient manner. To understand how ambiguity is dealt by the STRING, it is important to notice that a XML structured file (whose structure is presented in Section 4.4) resulting from the processing chain execution presents nodes in a hierarchy, where the nodes that represent single words are identified by its corresponding POS tag and contain as child nodes the features (morphological and semantic) that classify that occurrence of the word.

An ambiguous word can assume different meanings, depending on the surrounding context (remember the *banco* example given above, where *banco* is a noun with two possible meaning – corresponding to two occurrences of the same POS tag), or in the same context it can be considered to have different behaviors (for example, the word *comida* can be a verb or a noun – corresponding to two different POS tags). For such ambiguous situations, the XML file presents each possible POS tag and corresponding features as different blocks in the file, representing the different possible classifications or meanings of the word. As seen before, the different classifications can be represented by different POS tags attributable to the same word, or multiple occurrences of the same POS tag but associated with different features, representing different meanings. However, the system is capable of choosing the most appropriate sense (based on a probabilistic study) of an ambiguous word based on its context (the sentence structure) thus adding an additional feature, `HMM-SELECTION`, to the set of features of the chosen tag node. This simplifies the problem of dealing with ambiguity, since during the parsing of the XML files, we only have to guarantee that our system's processor identifies the marking flag and captures solely the block corresponding to the chosen POS tag and its corresponding features.

The CETEMPúblico (Rocha and Santos, 2000) will be used to obtain the various contexts of those classified nouns. CETEMPúblico stands for "Corpus de Extractos de Textos Electrónicos MCT/Público" (corpus of extracts of MCT/Público electronic texts), and it is a large corpus (180 million words) of European Portuguese texts taken from the Público newspaper online edition, the corpus freely available and distributed by Linguateca.

The 5.000 Portuguese semantically classified nouns together with their surrounding contexts obtained from CETEMPúblico constitute the contextual features used as input for the classification task.

Another linguistic resources could be used, like Onto.PT (Oliveira and Gomes, 2010) or PAPEL (Oliveira et al., 2008). However, the choice of CETEMPúblico is related to fact that this corpus is already available and being used at INESC-ID.

3.3 STRING

The Statistical and Rule-Based Natural Language Processing Chain (STRING) (Mamede et al., 2012) developed at INESC-ID is a modular tool that performs all the basic text processing tasks for Portuguese, namely tokenization and text segmentation, POS tagging, morphosyntactic disambiguation, shallow parsing (chunking) and deep parsing (dependency extraction) in four steps:

1. Pre-processing;
2. Lexical analysis;
3. Statistical and rule-based POS Disambiguation;
4. Parsing.

The architecture of the STRING system is presented in *Figure 10*.

The STRING system works as follows:

In the pre-processing stage the input text is segmented into sentences and tokenized with all the POS tags. The tokenizer module is able to identify words, numbers, numerals, punctuation, etc.

LexMan (Vicente, 2013) is the lexical analyzer, responsible for the POS and morphosyntactic features (gender, number, tense, mood, case, degree, etc.) of each token. The system then proceeds with RuDriCo2 (Diniz et al., 2010), a rule-driven converter responsible for solving contractions of words, like ‘comigo’ = ‘com’/Prep + ‘eu’/Pron, executing some types of morphosyntactic disambiguation and merging compound words into a single token, validating the input data.

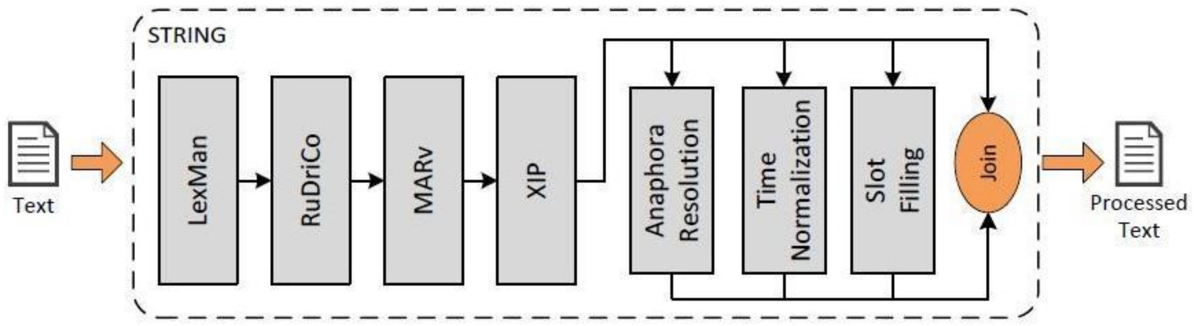


Figure 10: STRING architecture.

MARv (Ribeiro, 2003) on the third step, is the disambiguator responsible for choosing the most probable POS tag for each word using the Viterbi algorithm, which its language model is based on second-order (trigram) models, that codify the contextual information concerning entities, and unigrams, which codify lexical information, consisting this module in the statistical part of the STRING. The corpus used in the classification model is heterogeneous since it contains 250.000 words from a wide set of distinct sources.

Finally, XIP is the finite-state incremental parser already presented before. It chunks the text in elementary phrases identifying their heads and the relations between the heads of the various chunks, using as resources a Portuguese rule-based grammar and a set of lexical resources with linguistic (morphologic, syntactic and semantic) information. This allows the identification of relations such as subject, direct object, modifier, etc., and also to link verbal chains formed of strings of auxiliaries.

After these steps, other post-processing modules can be used to address specific NLP tasks, such as anaphora resolution (Marques et al., 2013), temporal expressions normalization (Maurício, 2011), or event ordering (Cabrita et al., 2014).

3.4 CO-TRAINING ALGORITHM IMPLEMENTATION

For the semantic classification aimed at this work, an application of the co-training algorithm is used, following an approach proposed by (Collins and Singer, 1999) and used in several works like (Mota, 2009).

The algorithm receives as input a set of nouns, constituted by elements of the classified nouns, named *seeds*. With these seeds, the first classifier of the algorithm searches for sentences containing these seeds, thus forming noun-context pairs. These noun-context pairs are fed into the second classifier of the algorithm, which is responsible for gathering other contexts that match the seeds’ ones, and then collecting the seed words that are in these new contexts, feeding the first classifier with the new nouns. This cycle repeats until no new information is acquired.

In each iteration of the algorithm, the process is the following:

1. The seeds are used on the first step classifier to *Learn* contextual rules;
2. These contextual rules of the seeds are used to *Label* the unlabeled contextual rules, during the phase two (second classifier);
3. The seeds contained in the resulting labelled examples are used to *Learn* new contextual rules, as on the first step classifier;
4. These are then fed to the second classifier, repeating the cycle of feedbacks.

This process explains the cyclic behavior of the solution, because every new obtained seed is reinserted in the seed set for further classifications of unlabeled data.

Figure 11 demonstrates this cyclic behavior of the co-training algorithm. *CLASSIFIER* 1 is responsible to extract contexts from the received seed words, while *CLASSIFIER 2* is responsible to pair contexts with other matching contexts, and from these last ones, extract new seeds.

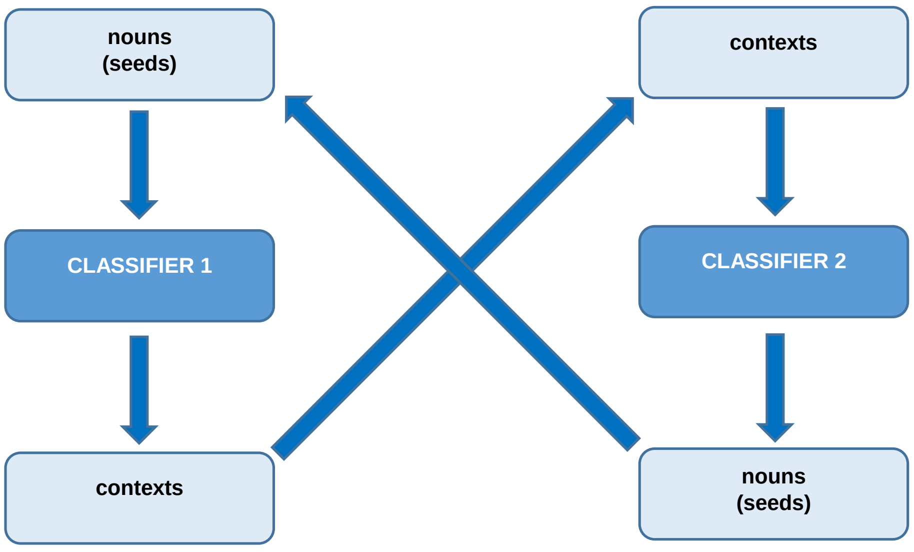


Figure 11: Cyclic behavior of the co-training algorithm.

3.5 Evaluation Methods

The evaluation of the results presented in this work is done by a user, through interaction with a simple interface to validate the results.

This interface includes simple options that allows for filtering of the results, regarding the success or failure of the classifications obtained from the system for the nouns presented. The human user must filter the final seed-words list to decide about the correctness of these classifications. In case of success, the new words are considered classified and are saved as a XIP file, together with the semantic tag matched, for further adding to the database of semantically classified Portuguese nouns. Thus, the user must be able to judge on the correctness of the results, by having the necessary knowledge to do so.

The system is not expected to achieve a higher percentage of successful classifications than wrong ones, due to the fact that the comparison method used in the machine learning method chosen is blind, meaning that it does not take into consideration anything else other than the surrounding context (dependencies) in the sentence. The context, namely the dependencies, is what allows for matching between noun-context pairs that share nothing else but the dependencies they establish among the sentence. However, these results are annotated and the parameters of the algorithm are subject to adjustments in order to refine and improve the percentage of correct results, as described in detail later in Chapter 5.

4 IMPLEMENTATION

The implementation of this project is divided in two phases: *input processing*, and *algorithm execution*. This last on, due to its extensive nature, is itself divided into smaller modules, where some can be considered to be merely secondary. The following list aims to summarize the phases and modules that constitute the implementation of this application, including the optimizations implemented during the development of the solution:

1. The first phase corresponds to the parsing of the corpus text files by the STRING system, in order to obtain the corresponding XML files constituting the syntactic and semantic analysis of the texts. The corpus of texts is organized in 20 Parts of about 200 files each;
2. The second phase regards the implementation of the main program, which implements the algorithm itself. This algorithm can be seen as constituted by multiple modules concerning different tasks:
    a. **Main/User Interface** – The *Main* module is responsible to call the *User Interface* module, in case the user triggers it, allowing the user to interact with a graphic interface for simpler configuration of the input parameters, such as semantic tag to search, initial seed list, which can be filtered, split factor to apply, and corpus main folder to use, and later for filtering of the final results. Alternatively, the Main program can make the configuration itself for default values, only allowing the user to decide the semantic tag to search and split factor to apply. These two modules are individually responsible to manage the user interaction either for configuration on startup, or for final results presentation along with information on how to filter them.
    b. **Algorithm Sequencer** – The second group of modules can be seen as part of a box called *Algorithm Sequencer*, where the modules are sequenced in order to obtain the desired execution process:
        i. For processing of the XML files and transformation of the initial data produced by the STRING system into a more manageable data structure, an auxiliary module was developed: the *Parser*. This tool is responsible for the corpora parsing and processing, in order to extract only the relevant information from the various XML files, saving it as textual files containing a filtered and more readable and usable description of the information;
        ii. *Processer* is another auxiliary module that transforms textual data contained in files into the respective data objects, either stored in memory structures (during the application runtime), which is done to the seed list file, or object streams stored in physical memory (hark disk), for resources limitations (space), as is done to the corpus text;
        iii. To sustain the execution of the program, it is necessary to know the resources available. With this in mind, one of the first steps towards the execution of the

process concerns a data analysis, responsibility of the *Data Analyzer* module, and corresponds to the acquisition of knowledge regarding the current state of the data set of classified nouns, to obtain a view on which semantic categories are available. This module is also important to support our decision on which semantic categories were lacking examples, and would produce less accurate results due to their poor representativeness;

iv. Another resources-directed module is the *Seed Producer,* very similar to the *Data Analyzer*, since it parses the same files and in the same way, however, while one is only responsible for counting occurrences of tags, the other is responsible for gathering the nouns that contain a certain tag attributed to it. This task is important to gather and supply to the user the list of initial seed words available for the semantic category he requested, either through the user interface, the console or the program arguments.

v. One auxiliary module had to be developed to deal with the biggest difficulty found during this work: *Resources Management*. This module is responsible to divide the text contained in each part of the corpus, to a smaller and more manageable quantity of data to be processed at a time, in order to make the algorithm provide results within a reasonable amount time and with a reasonable amount of storage consumption from the machine;

vi. **Co-Training** – The main task regards the proper execution of the learning algorithm, the kernel of this program. The algorithm is constituted by two classifiers that feed each other with their results, in a cycle that allows the seed set and contextual features to be reused and expanding dynamically during execution time.

    1. The first classifier, called *Seed Finder* is responsible for searching the corpus for sentences (contextual features) containing the seed words it received, and feed these to the *Context Finder*;
    2. The *Context Finder* is the second classifier on the Co-Training implementation, responsible for searching the corpus for sentences (contextual features) that match the sentences it received, thus acquiring new ones, and then extract new seed words from these sentences, that are then feed to the *Seed Finder* again, in a cyclic manner. The output of the execution of the algorithm takes the form of textual files containing a list of seed words together with the number of occurrences of each one, one per line, which is then merged into a single file and targeted to human validation, for results filtering.

vii. The last tasks regard the condensation of results, since each part is divided into subparts, by the *Resources Manager*, the results are divided as well. Thus, it is necessary to condensate this results in order to obtain a final single algorithm’s results file, to simplify the input for the presentation task. This is responsibility of the *Results Merger* module. At this final stage, the output of the algorithm is transformed into a XIP file, with the nouns, either filtered or not, tagged with the new semantic category label found

c. **Libraries** – The execution of the algorithm is supported by two auxiliary modules that are used multiple times throughout the whole execution of the algorithm:

   i. An important auxiliary module named *Directory Reader* was developed for filename gathering according to specific text patterns, in order to feed the different phases of the project with the necessary list of files to read. This module is capable of exploring a possible hierarchy of directories and subdirectories;

   ii. *Common Types* is an auxiliary module whose purpose is to supply an interface to deal with the sets of features of the elements of dependencies constituting sentences, but also to supply printing functions for the different structures involved in the application.

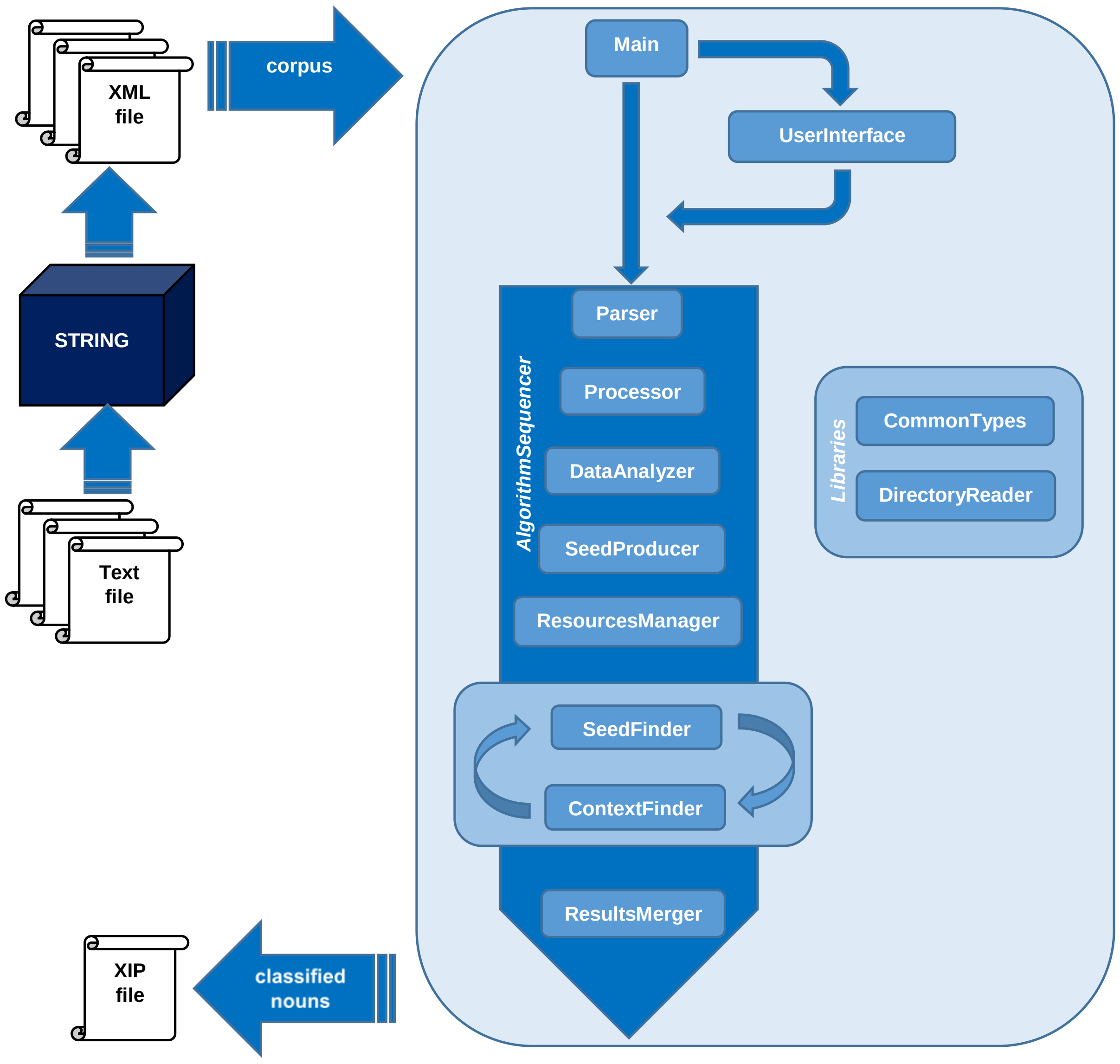


Figure 12: Architecture of the semantic classifier developed.

Figure 12 presents a modular overview explaining the architecture of the implementation of the Nouns Semantic Classifier application.

The cyclic behavior of the *SeedFinder* and *ContextFinder* modules corresponds to the cyclic behavior demonstrated in Figure 11 of Section 3.4, corresponding these modules respectively to *CLASSIFIER 1* and *CLASSIFIER 2*.

The following sections will describe the implementation of these modules in more detail.

4.1 DATA ANALYSIS

The **Data Analysis** task regards the acquisition of knowledge regarding the current state of the data set of classified nouns.

In this step, an application was developed to parse the files containing the data set of classified nouns (along with their semantic tags) and execute a count of the existing noun samples in each category.

This task allowed us to capture the distribution of nouns on semantic categories, thus helping on the decision of which categories should be expanded for data set richness purposes.

The script developed uses XIP files as input, and they must be located inside a folder named '*XIPFILES*' that must exist in the directory where the script is being run. The content of these files must obey to a strict syntax, in order to fulfill the script's expectations on what concerns the processing of the text and gathering of the relevant pieces of information.

An example of a valid line of one of these files is presented next:

```
castor: noun += [SEM-Adom=+,SEM-Azo=+].
```

The complete results of these task can be seen in the Section 5.1.

In face of the results, we first decided to expand the categories that had the least amount of seeds, aiming to acquire new knowledge to our data set, but we believe that not only the lack of representativeness of the words would negatively influence the quality of the results obtained, by the fact that seed examples are missing, but also the journalistic source of the corpus would influence the results as well, because some semantic categories would have low probability to be found in this kind of text.

Thus, and also with the objective of grounding our believes on what concerns the representativeness of seed examples and corpus source (in terms of kind of content) influencing the results obtained, we opted to explore opposite categories, in terms of their representativeness in the set of classified nouns, but also opposite on their probability to be found in a corpus of journalistic source, considering our believe that some topics are less likely to be found in this type of corpus. Thus, we considered two factors to choose the semantic categories to test:

- Probability of finding samples of the semantic category in a corpus of journalistic source;
- Quantity of nouns classified with the semantic category in the set of already classified nouns.

Our choices were:

- `SEM-ACT-CRIME` – corresponds to criminal acts, which we considered to be highly probable to be found in a journalistic corpus.
- `SEM-CC-STONE` – corresponds to stones or stone-sized round objects, like stone, ammonite, brick, diamond, etc., which we considered to be less probable to be found in journalistic text;
- `SPORTS` – corresponds to sporting events, which we considered to be reasonably probable to be found in journalistic text;
- `SEM-TOOL-MUS` – corresponds to musical tools (instruments), which we considered to have low probability to be found in the corpus.

4.2 SEED PRODUCER

This task concerns the gathering of seed nouns containing a determined semantic tag (requested by the human user through the user interface, console or program arguments as explained later in Section 4.9) attributed in its classification, by searching the XIP files contained in the '*XIPFILES*' folder. For gathering the XIP files this module takes advantage of the Directory Reader module (explained later in Section 4.3).

On what concerns implementation, this modules execution is very similar to the Data Analyzer module, since it parses the same files and in the same way, however, while the other one is only responsible for counting occurrences of tags, this one is responsible for gathering the nouns that contain a certain tag attributed to its classification.

As a result, this module produces a list of nouns that share the semantic tag requested by the user, and that after filtering, will serve as seeds to be used for the algorithm execution.

4.3 DIRECTORY READER

The *Directory Reader* is a tool whose goal is to analyze the current directory where the main application is being run, and extract the relevant files (by its filename), existent in a possible hierarchy of directories and subdirectories.

For this, the Directory Reader receives a pattern (for example: '*.xml*'), that is essential for matching purposes on the search for the corresponding files. Any pattern can be fed to the tool.

On this specific project, this tool is used to gather:

- The main directories containing files (pattern ‘*Parte*’);
- The XIP files for the *Data Analyzer* and *Seed Producer* modules (pattern ‘*.xip*’);
- The XML files existing in these directories (pattern ‘*.xml*’);
- The parsed XML files existing in these directories (pattern ‘*Parsing.txt*’);
- The list of seed files produced by each subpart (pattern ‘*listaSementesNova*”) existing in a directory;
- The list of result seed files produced for each part (pattern ‘*Results*’) existing in a directory.

These filenames are collected and provided to the subsequent module, responsible for parsing and processing the respective content of files, according to its needs.

These content is used in the following manner:

- The ‘*Parte*’ directories are essential for the program to know where to look for resources;
- The ‘*XML*’ files are needed for the *Parser* to know which files are supposed to be parsed;
- The ‘*XIP*’ files are needed for the *Data Analyzer* and the *Seed Producer* to know where to look for already classified nouns;
- The ‘*Parsing*’ files are fed to the *Resources Manager*, in order to evaluate the data partitioning necessary to make the program executable in appropriate time and with appropriate resources;
- The resulting seed files are used to feed the user interface responsible for allowing the user to filter the valid and invalid classifications obtained, but since each part is divided into subparts processed individually, the results are also divided accordingly, thus they need to be merged gradually into a final single results file in the final stage of the algorithm.

4.4 PARSER

The CETEMPúblico (Rocha and Santos, 2000), constituted by online newspapers text, was the corpora chosen to be used during this thesis’ project implementation, and its fundamental function is to provide contexts for the classification task target of this work.

A context is a collection of semantic and morphological features that are used to describe individual words as well as their relations with other words inside one sentence, named dependencies.

To obtain this contexts, the sentences that constitute the corpora were first parsed by the STRING system. The output of the STRING system is a XML file that fully describes the structure of each sentence individually, represented as a *LUNIT*, and its constituents using a hierarchy of *nodes*, *features*, *tokens*, *reading* tags and *dependency* tags. The meaning of these identities can be described as follows:

- **LUNIT** – represents a language unit;
- **NODE** – represents a syntactic function;

- **FEATURE -** represents a characteristic representative of the element it is part of.
- **TOKEN** - represents an individual word;
- **READING** - represents the lemma of the individual word;

Detailed information about the grammar of the XIP module, part of the STRING processing chain, can be found in the document (Mamede and Baptista, 2014).

To better understand this hierarchy of elements, an extract of a XML file is presented next:

```
<LUNIT language="Portuguese">
      <NODE num="38" tag="TOP" start="154224" end="154294">
            <FEATURE attribute="CAT" value="0" />
            <NODE num="52" tag="PP" start="154224" end="154230">
                  <FEATURE attribute="ENV3" value="+" />
                  <FEATURE attribute="QUANT" value="+" />
                  <FEATURE attribute="PP" value="+" />
                  <FEATURE attribute="START" value="+" />
                  <FEATURE attribute="FIRST" value="+" />
                  <NODE num="0" tag="PREP" start="154224" end="154226">
                        <FEATURE attribute="PREPLOCDEST" value="+" />
                        <FEATURE attribute="TOUTMAJ" value="+" />
                        <FEATURE attribute="MAJ" value="+" />
                        <FEATURE attribute="PREP" value="+" />
                        <FEATURE attribute="TOKENSTART" value="+" />
                        <FEATURE attribute="HMMSELECTION" value="+" />
                        <FEATURE attribute="START" value="+" />
                        <FEATURE attribute="FIRST" value="+" />
                        <TOKEN pos="PREP" start="154224" end="154226">
                        A
                              <READING lemma="a" pos="PREP">
                                    <FEATURE attribute="PREPLOCDEST" value="+"/>
                                    <FEATURE attribute="TOUTMAJ" value="+" />
                                    <FEATURE attribute="MAJ" value="+" />
                                    <FEATURE attribute="PREP" value="+" />
                                    <FEATURE attribute="TOKENSTART" value="+" />
                                    <FEATURE attribute="HMMSELECTION" value="+"/>
                                    <FEATURE attribute="START" value="+" />
                              </READING>
                        </TOKEN>
```

```
                    </NODE>
                    <NODE num="2" tag="ART" start="154224" end="154226">
                          <FEATURE attribute="DEF" value="+" />
                          <FEATURE attribute="MASC" value="+" />
                          <FEATURE attribute="PL" value="+" />
                          <FEATURE attribute="ART" value="+" />
                          <FEATURE attribute="TOKENEND" value="+" />
                          <FEATURE attribute="HMMSELECTION" value="+" />
                          <TOKEN pos="ART" start="154224" end="154226">
                          os
                                <READING lemma="o" pos="ART">
                                      <FEATURE attribute="DEF" value="+" />
                                      <FEATURE attribute="MASC" value="+" />
                                      <FEATURE attribute="PL" value="+" />
                                      <FEATURE attribute="ART" value="+" />
                                      <FEATURE attribute="TOKENEND" value="+" />
                                      <FEATURE attribute="HMMSELECTION" value="+"/>
                                </READING>
                          </TOKEN>
                    </NODE>
(...)
<DEPENDENCY name="DETD">
      <PARAMETER ind="0" num="22" word="barra" />
      <PARAMETER ind="1" num="20" word="o" />
</DEPENDENCY>
<DEPENDENCY name="MOD">
      <FEATURE attribute="POST" value="+" />
      <PARAMETER ind="0" num="22" word="barra" />
      <PARAMETER ind="1" num="28" word="baliza" />
</DEPENDENCY>
(...)
```

The *Parser* module concerns the parsing and processing of this XML data obtained from the STRING system into relevant information, by filtering unnecessary data from the XML files and merging the dependencies information with its respective constituent nodes' features, producing a succinct textual file with the resulting relevant data.

To filter the XML information, the tags are selected and only some of the presented elements in the XML are retained, since only those represent relevant information for the later task.

The POS tags used in this work are the following:

- PRONREL, ADJ, ADV, ART, CONJ, DET, INTERJ, NOUN, NUM, PASTPART, PREP, PRON, PUNCT, REL, SYMBOL, VERB

The morphological features used in this work are the following:

- PL, SG, 1P, 2P, 3P, FEM, MASC, PROPER, CARD, ORD, MULT, ROM, FRAC, DIG, DEG, MAJ, TOUTMAJ, SUB, REDUCEDMORPH, CLI, SPLIT, AUG, DIM, PFX, SFX, NOM, ACC, DAT, OBL, INCL, REF, REFL, PERS, POSS, REL, DEM

The morphological feature *PROPER*, representing a proper noun, takes a special role on the classification task during comparison, because it must be distinguished between common nouns and proper nouns: the aim of this work is to classify common nouns, thus the proper nouns must be treated differently.

The dependencies considered relevant are:

- MOD, SUBJ, CDIR, COORD, CINDIR, PREDSUBJ, ATTRIB, COMPL, APPOSIT, AGENT, PATIENT, FIXED

The semantic tags that classify nouns (example: `SEM-AMOUNT`, `SEM-CC-STONE`, `SEM-SIGN`, `SEM-AC`, `SEM-WATERMASS`, etc.) are all retained for the classification task. However, some of these tags underwent a special treatment, because despite being different, they are all associated with the human entity. This features related to the human being can be either a profession, act, or behavior. Thus, the semantic tags that are considered to be related to human entities and are replaced by a special tag `UMB-HUMAN`, are the following:

- `SEM-H, SEM-HH, SEM-Hattr, SEM-Hbio, SEM-Hfam, SEM-Hideo, SEM-Hmyth, SEM-Hnat, SEM-Hprof, SEM-Hsick, SEM-Htit, SEM-H-nomagent-crime, human, firstname, lastname, title, relative, affiliation, postpeople, SEM-Hindividual, SEM-Hpeople, SEM-Hprofession, SEM-Hcargo, SEM-Hcollective, SEM-Hadministration, SEM-Hinstitution, SEM-Hgroup, SEM-Huniversity, SEM-Hpartido, SEM-Hindgroup, SEM-Hjornal`

It is worth to remember that ambiguity is present in the set of already classified nouns, and must be dealt with. As explained before, the STRING processing chain deals with ambiguity in a very convenient manner: based on the surrounding context of the ambiguous word, it adds the feature `HMMSELECTION` to the set of features associated with the node corresponding to the chosen POS tag, marking it as the most appropriate sense. Thus, during the parsing of the XML files, we only have to guarantee that our system's processor identifies the marking flag and captures solely the block corresponding to the chosen *POS* tag and its corresponding features.

The result of this filtering of the XML file is a text file containing a description of each relevant dependency of a sentence per line, characterizing its elements on what concerns their features, according to the selections presented above, together with some more useful information, like the identifier of its node, or the lemma of the word.

An example of an output of these parsing and processing task is presented next:

```
MOD_ID=22,sem=SEM-CC-STONE,word=diamante,pos=NOUN#ID=24,lemma=lapidar,pos=PASTPART,
class=32C|MOD_ID=18,sem=SEM-AMOUNT,word=lote,pos=NOUN#ID=22,sem=SEM-CC-STONE,word=d
iamante,pos=NOUN|MOD_ID=6,lemma=relatar,pos=VERB,class=09#ID=18,sem=SEM-AMOUNT,word
=lote,pos=NOUN|MOD_ID=6,lemma=relatar,pos=VERB,class=09#ID=38,sem=SEM-HPROF,special
=UMB-HUMAN,word=proprietário,pos=NOUN|MOD_ID=6,lemma=relatar,pos=VERB,class=09#ID=4
4,morph=PROPER,word=Angola,pos=NOUN|MOD_ID=6,lemma=relatar,pos=VERB,class=09#ID=52,
sem=SEM-STATE-H,sem=SEM-CC,sem=SEM-BUILDING,word=dependência,pos=NOUN|SUBJ_ID=0,lem
ma=tratar,pos=VERB,class=02#ID=79,pos=PRON|SUBJ_ID=6,lemma=relatar,pos=VERB,class=0
9#ID=10,morph=PROPER,word=Lusa,pos=NOUN|CDIR_ID=0,lemma=tratar,pos=VERB,class=02#ID
=2,pos=PRON
```

4.5 PROCESSER

The output presented in the previous section is the input of the *Processer* module. This module is responsible for the transformation of textual data contained in files into the respective data objects, either stored in memory structures (during the application runtime), which is done to the seed list file and to the seed sentences found along the execution of the algorithm, or object streams stored in physical memory (hark disk), for resources limitations (space), as is done to the individual corpus text files.

The memory data structure used to keep the list of seeds during runtime is a *HashMap* that matches a string (the seed word) to the number of its occurrences during the whole processing of the respective subpart:

```
HashMap<String, Integer> _seedsList;
```

To store the seed sentences found during runtime, a complex data structure is needed, also based in the concept of *HashMap*:

```
ArrayList<HashMap<String, ArrayList<ArrayList<ArrayList<String>>>>> _seeds;
```

During runtime, the sentences from the corpus text files are stored into a similar structure, in order to be immediately written into a data objects file, using an *ObjectOutputStream* and an *ObjectInputStream* for further reading during the processing of the algorithm.

Since each object is written individually, they are stored into the following structure, for writing purposes:

```
HashMap<String, ArrayList<ArrayList<ArrayList<String>>>> _corpusSentence;
```

These objects stored in disk, are the main input of the co-training algorithm, as will be described later in Sections 4.7 and 4.8.

4.6 RESOURCES MANAGER

The *Resources Manager* module appeared as an optimization needed to reduce the execution time of the algorithm. This module is responsible for partitioning the data into smaller parts, in order to transform the large corpus into more manageable pieces, to reduce time and resources consumption during the processing task.

Before the development of this module, the algorithm considered the corpus to be used all at once, which revealed unacceptable execution times, for example, the test-case for the richest semantic category `SEM-ACT-CRIME` was unable to be finished with success due to the machine running out of memory to store the intermediate data used by the algorithm, and its execution was permanently crashing after two weeks of execution, without producing any results.

Many adjustments and experiments were needed, in order to find the better splitting factor. The files needed to be split in such a manner that did not harm the classification task (and for that, the larger the corpus, the better the results), but at the same time it could not be so large that the data to process would translate in such a large number of comparisons that it would affect the runtime and resources usage of the application.

As a first approach, this module divided the corpus text into *N* equal subparts. To do this, it counted and gathered the total quantity of files, and divided it into *N* parts, corresponding to *N* files with sequential names, containing each one its array of filenames from the bigger set, divided equally (except for the last part). However, as explained with the respective results further presented in Section 5.2, this splitting strategy would not resolve the unbalance existing among files of the same part, where some blocks could contain dozens of seeds, while others could have none, keeping the resources management in trouble.

To improve the time and reliability of the results obtained, the most delicate balance of this project, a seed-based-splitting strategy was chosen. This strategy consists of having the system read and calculate the seeds contained in each parsed file in a sequential manner, as in, it keeps reading and gathering content from these sequential files, until *N* seed sentences (sentences matched as containing any seed word) are found. Thus, each block (subpart) will be constituted by a group of files that have at most *N* seed sentences present in their content. This block of files' filenames are stored in a structure, accordingly, to further represent a subpart. Note that none of the subparts can have more than N seed sentences, however they can have less, as often occurs with the last block of files, or in situations where a single file contains more than one seed, which in case of passing the *N* limit would traduce in a file being stored in the next block (subpart). Thus, the corpus is divided in blocks of text containing *N* seed sentences each.

4.7 Seed Finder

The **Seed Finder** module constitutes the first classifier of the Co-Training algorithm implementation used in this project.

This classifier is responsible for searching the corpus for sentences (contextual features) containing the seed words it received, and feed these to the *Context Finder*.

To achieve this, this classifier is provided with a list of seed words, which it uses to process the corpus looking for sentences containing these seed words, and a file containing the respective corpus (corresponding to the divided content provided by the *Resources Manager*.

The algorithm processes each sentence at a time, and for each sentence it looks for each seed words (thus, the corpus is only processed one time, while the seed list, for being smaller, was preferred to be read multiple times), marking the element of the dependency where it was found with a special tag 'SEED' and then gathering these matched sentences and storing them in a logical structure in memory, that will be used on the second module *Context Finder*. It admits the existence of seeds in any dependency of a sentence.

This process allows for data splitting:

- The corpus is read one single time, and each sentence is gathered for sentence at a time;
    - If the sentence contains a seed, it is stored in the logical structure in memory;
    - Else, the sentence is written again to an object data file, in disk, for further corpus processing.

To this process, the algorithm uses huge object data files stored physically in disk who are produced and read alternately by the two classifiers, containing the output that they must fed as input to the other classifier.

The learning algorithm starts with the *Seed Finder* using the first file (provided by the *Processer*) as input and produce a new file by removing the matched sentences from the original corpus to a structure in memory, and writing the non-matched ones to the second file, which is then provided to the *Context Finder* who uses it as input (because it contains the resulting corpus). The second file of the pair will contain the *Context Finder* filtered corpus (the filtering is described in Section 4.7) that is then provided to the *Seed Finder* as input. This cycle repeats as needed, eliminating the unnecessary files and alternating the newly created ones accordingly.

Thus, during runtime, each pair of objects' files will contain the corpus in use. From this gradual filtering, the corpus is shortened in each step of the algorithm.

The final task of this module is to count the number of seed sentences (marked as *trues*) found: this is what determines the continuity of the execution of the algorithm: If no other seed sentence is found, there is no need to run the *Context Finder* again because it will not find any new matched-context sentences and seeds. However, if it finds a match, it must continue the algorithm until no new knowledge is obtained.

The number of new seed sentences is printed as the number of *trues* found, while the number of remaining not-matched sentences in the corpus is printed as *falses*. The execution stops when the number of trues is zero: at this point, the collection of new seed words found and gathered during the whole algorithm as well as the counting of occurrences of each new seed on the corpus is written to a text file matching the name of the input file, in the output directory.

As a demonstrative example, consider the following group of sentences and its contextual information (the corpus), considering that *queijo* is a seed word (the only seed in the seed file):

(corpus sentence) `O Miguel comprou um queijo.`

Miguel bought a cheese.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-FOOD-C-H, sem=SEM-FOOD-H, word=queijo, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Miguel, pos=NOUN]]]
```

(corpus sentence) `O Rui comeu um bife.`

Rui ate a steak.

```
CDIR=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=8, sem=SEM-FOOD-C-H, word=bife, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Rui, pos=NOUN]]]
```

(corpus sentence) `A Joana comprou um frango.`

Joana bought a chicken.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-AORN,
sem=SEM-AZO, sem=SEM-ADOM, word=frango, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Joana,
pos=NOUN]]]
```

(corpus sentence) `O Pedro comeu um frango.`

Pedro ate a chicken.

```
CDIR=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=8, sem=SEM-AORN,
sem=SEM-AZO, sem=SEM-ADOM, word=frango, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Pedro,
pos=NOUN]]]
```

The *Seed Finder* processes each sentence by taking each dependency at a time, looking for the seed. Usually, each dependency has two constituents. For the first sentence, it takes the first constituent of the first dependency, which considering the example above is `[ID=4, lemma=comer, pos=VERB, class=32C]` and looks for the *POS* tag. It must be a noun to be compared with the seed, because this work aims to classify nouns. Since it is not a noun, it is a verb, it passes to the second constituent of the dependency:

```
[ID=8, sem=SEM-FOOD-C-H, sem=SEM-FOOD-H, word=queijo, pos=NOUN]
```

First condition: it is a noun. Next, it looks for the tag *word* and compares its content with each seed word in the seed file. Since the seed file only contains one word, only one comparison is needed. The content matched the seed *queijo*, thus it marks this constituent with the tag *SEED*, and the contextual information of the sentence becomes:

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-FOOD-C-H,
sem=SEM-FOOD-H, word=queijo, pos=NOUN, SEED]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Miguel,
pos=NOUN]]]
```

This sentence `O Miguel comprou um queijo` is classified as a seed sentence and stored in the logical structure in memory containing the seed sentences found. The comparison continues with the remaining three sentences: none of them contains the seed word, thus all of them are written to the corpus data objects file, and passed to the *Context Finder* as input.

The intermediate results of this step are printed on the screen, where '*true*' corresponds to the number of sentences found containing a seed word in its content, and '*false*' corresponds to the number of the remaining sentences in the corpus which do not contain any seed word, and thus constitute the corpus for the next iteration of the algorithm:

```
#TRUE: 1 | #FALSE: 3
```

The execution proceeds by feeding the structure with the contextual information of the matched sentence and the resulting corpus data objects' file to the *Context Finder*.

4.8 CONTEXT FINDER

The *Context Finder* is by far the most complex module of the whole project. This module is responsible for finding sentences that share the same context as the seed sentences received as input.

The concept of context used here refers to the dependencies constituting the sentence, thus not only its syntactic constitution, but also the elements that are part of each dependency, including its morphological and semantic tags. Thus, a sentence that matches the context of another sentence, is a sentence that has the same dependencies, and the dependencies relate the same kind of elements.

The objective of this module can be seen as matching seed sentences received from the *Seed Finder* with sentences with similar context from the corpus. To achieve this, sentences are compared individually against the seed sentences, matching their dependencies according to some imposed rules:

- The **POS** tag of the corpus element must be equal to the **POS** tag of the seed element:
    - If the **POS** tag is *VERB* or *PASTPART*:
        - Both must match in *lemma*, and if applicable, must also match in *class* – the class is a tag that represents verb construction in the sentence, in terms of positioning of the verb in the sentence, according to the ViPEr system (Baptista, 2012);
        - Otherwise, they do not match;
    - If the **POS** tag is *NOUN*:
        - If any of them contains the special tag *UMB-HUMAN*, the other must have it as well;

        - If any of them is marked with the morph tag *PROPER*, the other must have it as well;
        - If the seed element is marked as *SEED*, they match and the corpus element is marked as *SEED* as well;
        - If the seed element is not marked as *SEED*, they must match in the tag word;
        - Otherwise, they do not match;
    - If the **POS** tag is *ADJ*:
        - Both must match in the tag *word*;
        - Otherwise, they do not match;
    - For other tags, they match with no further requirements;
- Otherwise, they do not match;

When the sentences do not match during this comparison process, they are written in a data objects file that contains the corpus resultant from the current iteration of the algorithm. We can consider that the corpus is filtered in each iteration of the algorithm, where an iteration corresponds to a call to the *Seed Finder* or the *Context Finder* modules, because the corpus sentences are split gradually into one of two structures: resultant corpus, to use in the next iteration, or the seed sentences structure in memory. Thus, the resultant corpus dimension is gradually reduced.

If the sentences are matched, the module is then responsible to identify the word that occupy the same position in the sentence as the seed one in the seed sentence received and matched against it. These words are then fed to the *Seed Finder*, who searches the corpus for new sentences with the new seeds words, repeating the cycle.

To help understand the algorithm of this classifier, we present a brief demonstration that simulates the subsequent execution of the previous example by the *Context Finder*, reusing the same group of sentences and its contextual information, considering that *queijo* is a seed word and the first sentence is the seed sentence contained in the logical structure in memory and received as input from the *Seed Finder*:

(seed sentence) `O Miguel comprou um queijo.`

Miguel bought a cheese.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-FOOD-C-H,
sem=SEM-FOOD-H, word=queijo, pos=NOUN, SEED]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN,     sem=SEM-HINDIVIDUAL,     morph=PROPER,     word=Miguel,
pos=NOUN]]]
```

(corpus sentence) `O Rui comeu um bife.`

Rui ate a steak.

```
CDIR=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=8, sem=SEM-FOOD-C-H,
word=bife, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Rui, pos=NOUN]]]
```

(corpus sentence) `A Joana comprou um frango.`

Joana bought a chicken.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-AORN,
sem=SEM-AZO, sem=SEM-ADOM, word=frango, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Joana,
pos=NOUN]]]
```

(corpus sentence) `O Pedro comeu um frango.`

Pedro ate a chicken.

```
CDIR=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=8, sem=SEM-AORN,
sem=SEM-AZO, sem=SEM-ADOM, word=frango, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comer, pos=VERB, class=32C], [ID=2, sem=SEM-HPEOPLE,
special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Pedro,
pos=NOUN]]]
```

The seed word is *queijo*, thus it contains a special tag *SEED* marking this fact. The comparison process takes each corpus sentence at a time and compares it with each seed sentence, for optimization purposes, since by picking the corpus sentences and traversing the seed sentences, instead of the reverse, reduces the number of times the corpus has to be read. It takes each dependency at a time, and then compares its constituents individually. Thus, the comparison process works as follows:

The first corpus sentence, `O Rui comeu um bife`, contains the same dependencies, *CDIR* and *SUBJ*, as the first (and only) seed sentence, `O Miguel comprou um queijo`. Thus, they can be compared, taking each dependency at a time, and each constituent of each dependency at a time. The process starts with the following constituent of the *CDIR* of the first corpus sentence:

The element `[ID=4, lemma=comer, pos=VERB, class=32C]` of the *CDIR* of the corpus sentence is compared with the element `[ID=4, lemma=comprar, pos=VERB, class=36DT]` of the *CDIR* of the seed sentence. Their *POS* tag match, however their lemma and class do not match, which means that they do not correspond to the same verb. The comparison fails: this sentence does not match the context of the seed sentence. If a *SEED* tag was added during the matching of a noun element, it would be deleted as soon as a later matching of the other elements of the same sentence failed, which means the sentence is 'cleared' from SEED tags whenever a comparison fails. The sentence is written into the resulting objects data file, because it is still part of the corpus, the matching tracking structure is reset to the following comparison with another sentence, and the execution continues. The following sentence's processing will demonstrate the usage of the *SEED* tag and the matching tracking structure.

The following corpus sentence, `A Joana comprou um frango`, is now the target of comparison with the seed sentence `O Miguel comprou um queijo`. They share the same dependencies *SUBJ* and *CDIR*, thus the comparison process can be executed.

Considering the *CDIR*, the element`[ID=4, lemma=comprar, pos=VERB, class=36DT]` of the corpus sentence is compared with the element `[ID=4, lemma=comprar, pos=VERB, class=36DT]` of the seed sentence: it looks for the *POS* tag: both are verbs, and since their lemma and class also matches, this constituent is considered to match the seed sentence one, saving its tags *ID* in an appropriate structure for the matching tracking, storing that the *ID* 4 in the corpus sentence, has matched to the *ID* 4 in the seed sentence, proceeding to the following element. The purpose of this matching tracking structure is described later in this section.

The element `[ID=8, sem=SEM-AORN, sem=SEM-AZO, sem=SEM-ADOM, word=frango, pos=NOUN]` is compared with the corresponding element in the seed sentence `[ID=8, sem=SEM-FOOD-C-H, sem=SEM-FOOD-H, word=queijo, pos=NOUN, SEED]`: the *POS* tag match, and the element in the seed sentence is marked as *SEED* because it is present in the same contextual dependency as the one containing the seed word in the seed sentence, representing a new noun found. The tag *SEED* is added to this set of features of this element from the dependency *CDIR* of the corpus sentence. The *ID*'s of both the matched elements of the seed sentence and the corpus sentence are saved in the structure for the matching tracking, storing that the *ID* 8 in the corpus sentence, has matched to the *ID* 8 in the seed sentence, proceeding to the following element. The *CDIR* dependency is considered to match, and the comparison process is able to continue. Since the dependency has no more elements, it switches to the following dependency, *SUBJ*:

Considering the *SUBJ* dependency, the element `[ID=4, lemma=comprar, pos=VERB, class=36DT]` of the corpus sentence is compared with the element `[ID=4, lemma=comprar, pos=VERB, class=36DT]` of the seed sentence. Since the matching tracking structure already contains a matching between node 4 from the corpus sentences with node 4 from the seed sentence, the comparison is not needed, and the matching succeeds between these two elements without any more validation, which helps reducing execution's runtime. The following element of the SUBJ dependency is `[ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Joana, pos=NOUN]` in the corpus sentence, that will be compared with `[ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Miguel, pos=NOUN]` from the seed sentence. No matching exists for these nodes in the matching tracking structure, thus the comparison process has to be run. Both elements have the *POS* tag '*noun'*, and both contain the special tag *UMB-HUMAN* (review Section 4.4 for more information about this special tag) thus no more comparison is needed, and the element is considered to match. A correspondence between nodes 2→2 is added to the tracking structure. Since the dependency does not contain more elements, the *SUBJ* dependencies are considered to match.

Before declaring the sentences as matching, one more comparison is needed: The *SUBJ* element of the corpus sentence must contain at least one of the elements that were matched in the *CDIR* element, in order to impose a relationship between the pairings done, and they must correspond to the same pairings on the seed sentence.

The matching tracking structure of *ID*'s presents a correspondence between nodes 4→4 and 8→8 for *CDIR*, and for the next dependency, *SUBJ*, one of these *ID*'s is present, 4→4, along with a new one: 2→2 in both sentences. This establishes a relationship between the elements contained in the *CDIR* sentence and the *SUBJ* sentence, which were matched. This process is successful.

Since both sentences do not have more dependencies to be paired, the comparison between these two sentences finishes by storing the dependencies of the seed sentence in the new seed sentences' context structure.

The comparison proceeds to the following sentences, but none of them matches, so they are both written to the resulting corpus file, to be further fed into the *Seed Finder* module as input.

At the end of this task, the structure containing the matched sentences is traversed looking for the individual constituents whose features now contain the *SEED* tag, and it collects the corresponding content of the *word* tag: *frango*.

The intermediate results of this step are printed on the screen, where '*true*' corresponds to the number of corpus sentences matched against the seed sentence, and '*false*' corresponds to the number of the remaining sentences in the corpus which were not matched, and thus constitute the corpus for the next iteration of the algorithm:

```
#TRUE: 1 | #FALSE: 2
```

The execution proceeds by feeding the structure with the new seeds (*frango*) and the corpus file to the *Seed Finder.*

In the following steps of the algorithm, the *Seed Finder* will match the sentence `O Pedro comeu um frango`, producing: `#TRUE: 1 | #FALSE: 1`. Next, the *Context Finder* will match this sentence with `O Rui comeu um bife`, producing: `#TRUE: 1 | #FALSE: 0`. The *Seed Finder* will have no corpus content to process, thus producing: `#TRUE: 0 | #FALSE: 0`, terminating the execution of the algorithm by printing the content of the structure containing the resulting seed words and occurrences to a textual file:

```
frango 2

bife 1
```

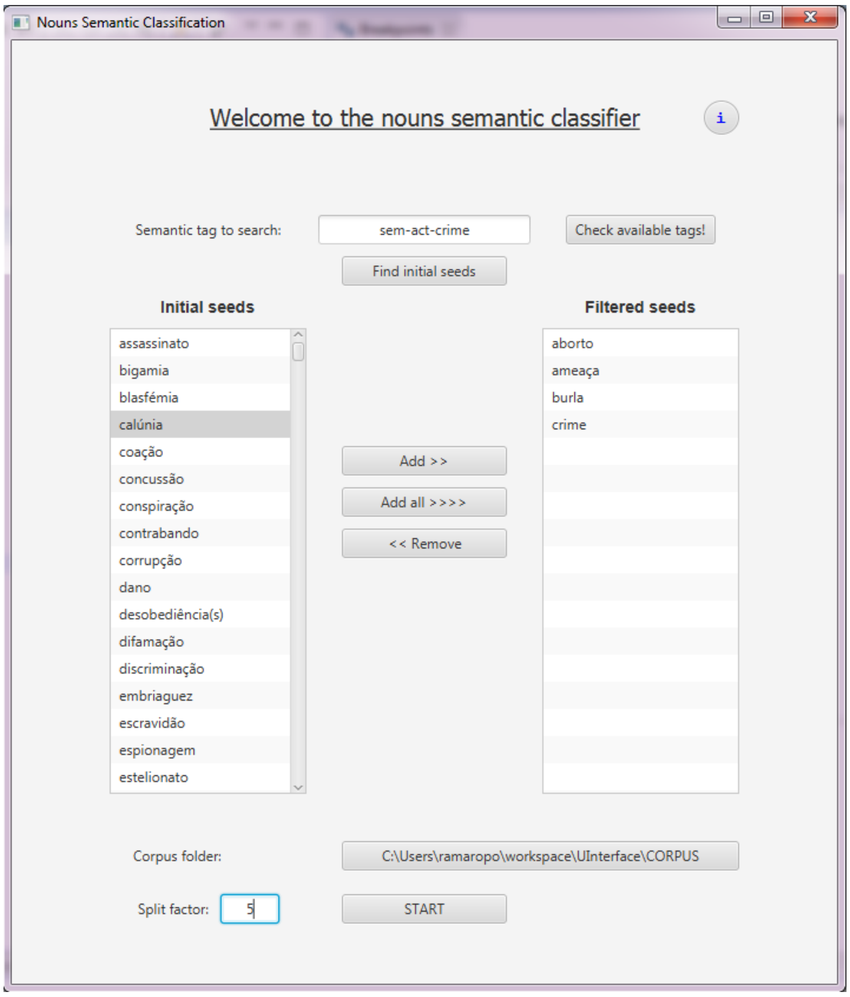
The development of a matching tracking structure, that stores matches between elements of the corpus sentence with elements of the seed sentence, by their *ID*'s, emerged as an optimization to reduce algorithm's execution time. This way, whenever an element is matched with another element, the stored relationship between *ID*'s allow the comparison process to avoid further comparisons of the same two elements, because before proceeding to the exhaustive comparison, the algorithm verifies if the *ID*'s are matched in the matching tracking structure, and if so, it considers them as matching and proceeds to the comparison process of the following elements of the dependency, saving one comparison process. The savings in comparisons help to reduce execution's runtime.

4.9 USER INTERFACE

This module provides a simple graphic interface to allow the user to easily interact with the application.

The application has two modes to be run:

- **Console Mode, with arguments**: If the program receives execution parameters, the console mode is triggered, which works over the corpus available at a preconfigured directory and with the preconfigured parameters, the user has only to provide the semantic tag to search for and the split factor to apply to the corpus, either through the first and second program's arguments respectively, or by typing in the console when requested. In this mode, nor the initial seeds list nor the final results are subject to any intermediate validation by the human user. The *Results* file will present all the words identified during the classification. The validation, in this mode, must be manually done to the content of this file, by deleting the erroneous classifications presented.



Semantic tag to search: sem-act-crime

Initial seeds	Filtered seeds
assassinato	aborto
bigamia	ameaça
blasfémia	burla
calúnia	crime
coação	
concussão	
conspiração	
contrabando	
corrupção	
dano	
desobediência(s)	
difamação	
discriminação	
embriaguez	
escravidão	
espionagem	
estelionato	

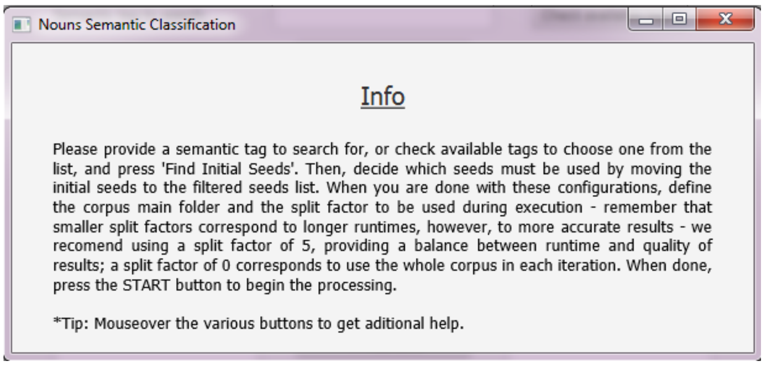
Corpus folder: C:\Users\ramaropo\workspace\UInterface\CORPUS

Split factor: 5

Figure 13: Graphic UI for user interaction with the application.

- **Console Mode, with no arguments**: If no arguments are passed, the user is asked to decide if he wants to run the graphic configuration mode or the manual mode. When using the **graphic configuration mode**, a new window opens with the graphic version of the application, presenting to the user a brief tutorial on how to fill the input texts, although when using the manual configuration mode, the user interacts with the application through the console. In any of the modes, the user has to provide information that configures the execution of the whole algorithm:

- The *semantic tag* s/he wants to search for, the one to which he wants to increase the number of nouns classified with that tag, which is used to provide the initial seed list of already classified nouns, by the *Seed Producer* module;



Nouns Semantic Classification

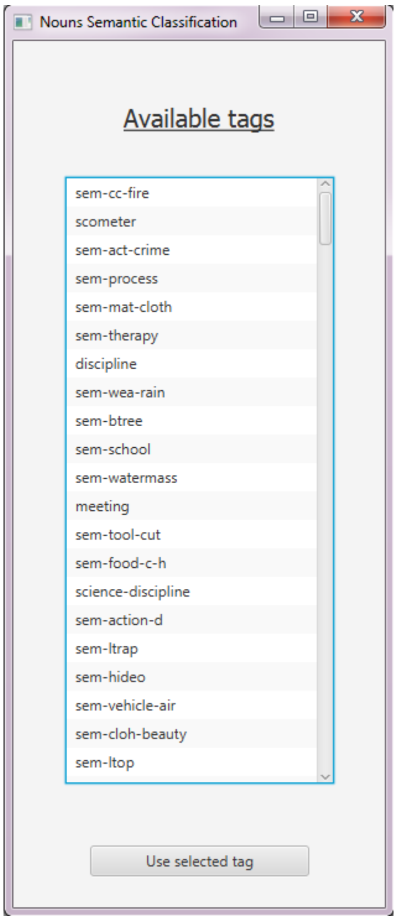
Info

Please provide a semantic tag to search for, or check available tags to choose one from the list, and press 'Find Initial Seeds'. Then, decide which seeds must be used by moving the initial seeds to the filtered seeds list. When you are done with these configurations, define the corpus main folder and the split factor to be used during execution - remember that smaller split factors correspond to longer runtimes, however, to more accurate results - we recomend using a split factor of 5, providing a balance between runtime and quality of results; a split factor of 0 corresponds to use the whole corpus in each iteration. When done, press the START button to begin the processing.

*Tip: Mouseover the various buttons to get aditional help.

Figure 14: Help screen, providing instructions on how to fill the UI fields of the configuration screen.

- In the graphic mode, if the user does not know the available semantic tags he can use, the button 'C*heck available tags'* provides a list of available tabs, as can be seen in *Figure 15*. This tag list is filled with information provided by the *Data Analyzer* module;



Nouns Semantic Classification

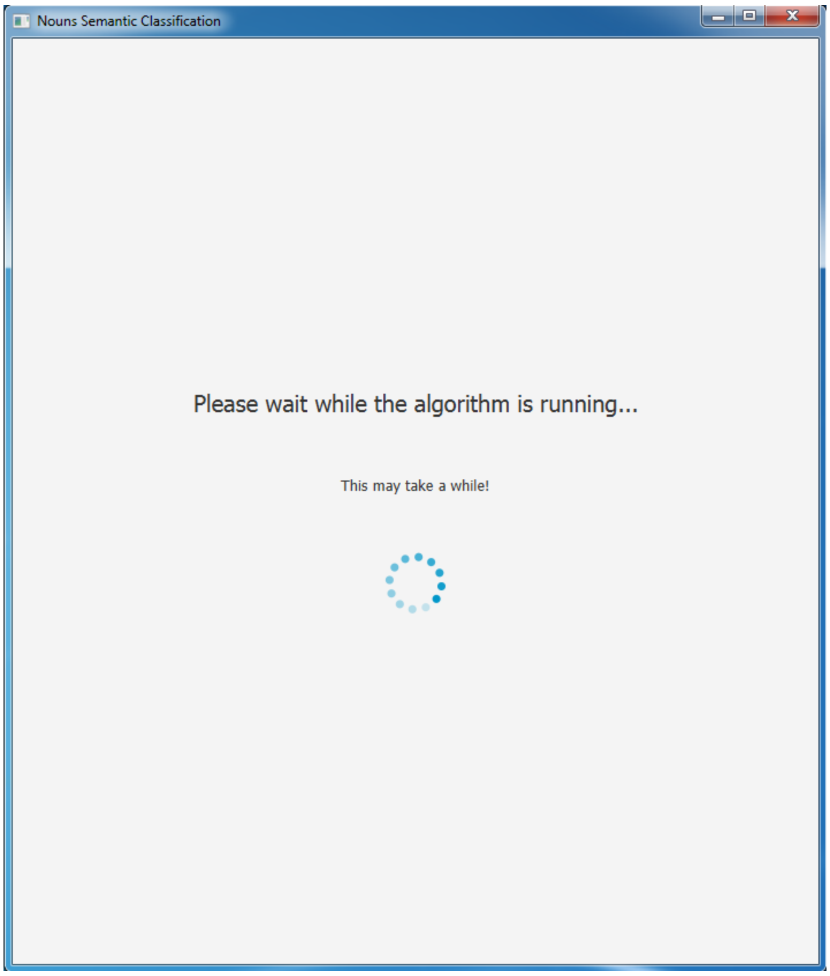
Available tags

sem-cc-fire
scometer
sem-act-crime
sem-process
sem-mat-cloth
sem-therapy
discipline
sem-wea-rain
sem-btree
sem-school
sem-watermass
meeting
sem-tool-cut
sem-food-c-h
science-discipline
sem-action-d
sem-ltrap
sem-hideo
sem-vehicle-air
sem-cloh-beauty
sem-ltop

Figure 15: Graphic UI that allows the user to check the list of available semantic tags and choose one.

- In the manual configuration mode the application informs the user that the available semantic tags can be consulted in a specific file;

- The list of *filtered seeds*, taken by selection of the items presented by the system as initial seeds. The initial seeds can be filtered by the user because there can be erroneous classifications;
  - The graphic interface provides means to filter the results, through the use of buttons;
  - In the manual configuration mode the user has to manually delete entries from a determined file communicated by the application;
- The path to the corpus main folder to use (the hierarchy is explored in depth starting in (and including) the main folder;
- The split factor (an integer number) to be used by the Resources Manager module, to divide the corpus text in smaller blocks of data.



Please wait while the algorithm is running...

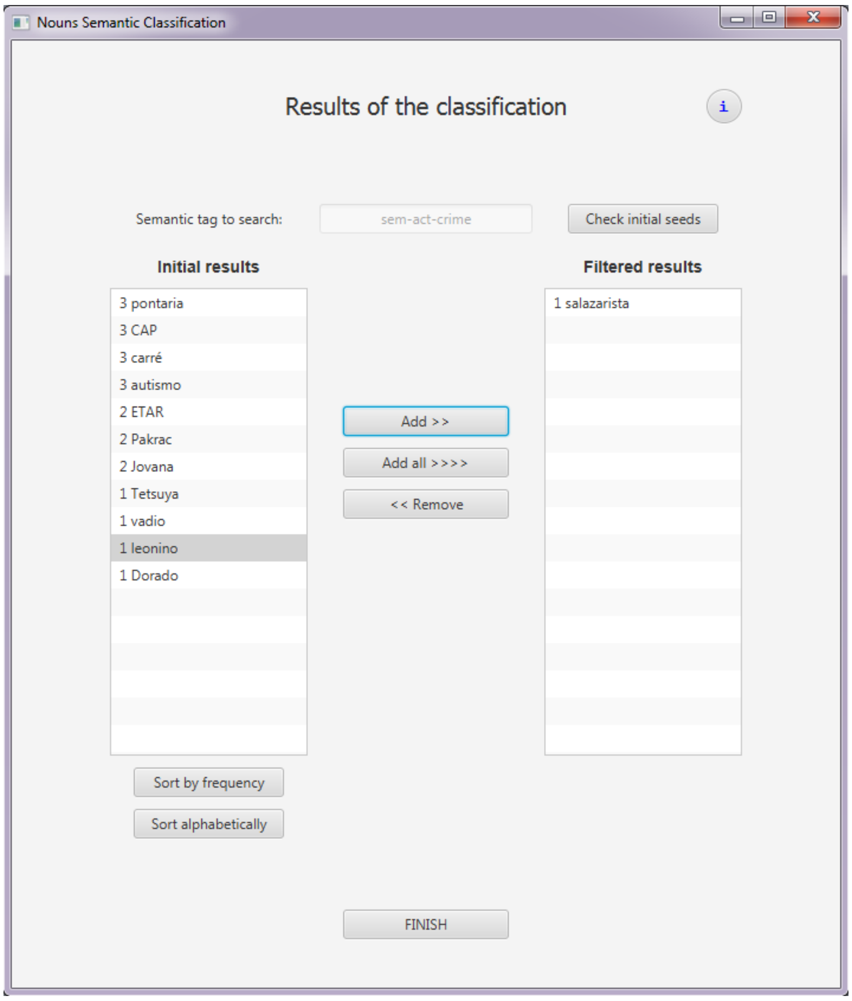
This may take a while!

Figure 16: Graphic UI to inform the user that the algorithm is running.

The user interface contains *help screens* accessible through the **i** (info) buttons available in both the configuration screen and the results screen, with the objective of helping the user to fill the various fields needed to interact with the algorithm.

When configuring the split factor, the following considerations must be taken into account:

- A split factor of 0 means the corpus must be used integrally on each iteration;
- Any other factor, *N*, means the corpus must be split in blocks of *N* seed sentences, as in, blocks of data that contain each N seed sentences, along with every non-seed sentence part of the corpus found in between (review Section 4.6 for more information about this splitting management). For example, a split factor of 5 (the recommended value) means the corpus must be split in blocks of text containing no more than 5 seed sentences, along with every other non-seed sentence found in between the search for these ones.


Nouns Semantic Classification

Results of the classification

Semantic tag to search: sem-act-crime

Initial results

3 pontaria
3 CAP
3 carré
3 autismo
2 ETAR
2 Pakrac
2 Jovana
1 Tetsuya
1 vadio
1 leonino
1 Dorado

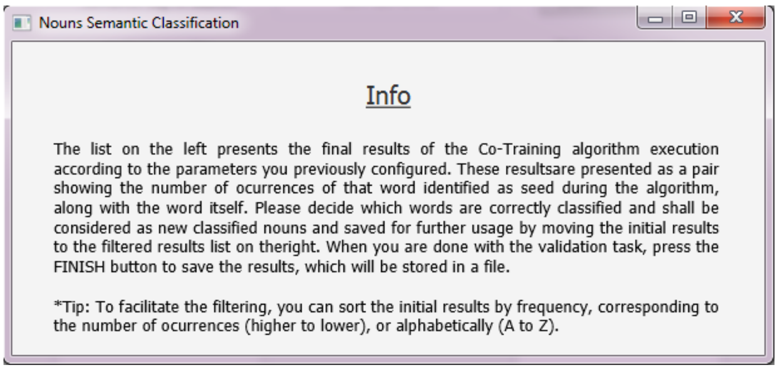
Filtered results

1 salazarista

Figure 17: Graphic UI with the final results and the filtering options.

Yet about the split factor, one must have in mind that the reliability of the results increase in inverse proportion to the split factor, as in, the most accurate results are achieved when the corpus is used integrally during each iteration of the algorithm, thus corresponding to a split factor of 0. However, this accuracy comes at the cost of much longer times of execution.

After providing these configurations, the user initiates the algorithm by pressing the *Start* button (or typing *CONTINUE*) that triggers the execution. He must then wait several minutes for the classification algorithm to terminate, so he is presented with the list of new nouns encountered along with the number of occurrences. Again, the user must filter this to separate good classifications from wrong classifications.



Info

The list on the left presents the final results of the Co-Training algorithm execution according to the parameters you previously configured. These resultsare presented as a pair showing the number of ocurrences of that word identified as seed during the algorithm, along with the word itself. Please decide which words are correctly classified and shall be considered as new classified nouns and saved for further usage by moving the initial results to the filtered results list on theright. When you are done with the validation task, press the FINISH button to save the results, which will be stored in a file.

*Tip: To facilitate the filtering, you can sort the initial results by frequency, corresponding to the number of ocurrences (higher to lower), or alphabetically (A to Z).

Figure 18: Help screen, providing instructions on how to interact with the UI fields of the results screen.

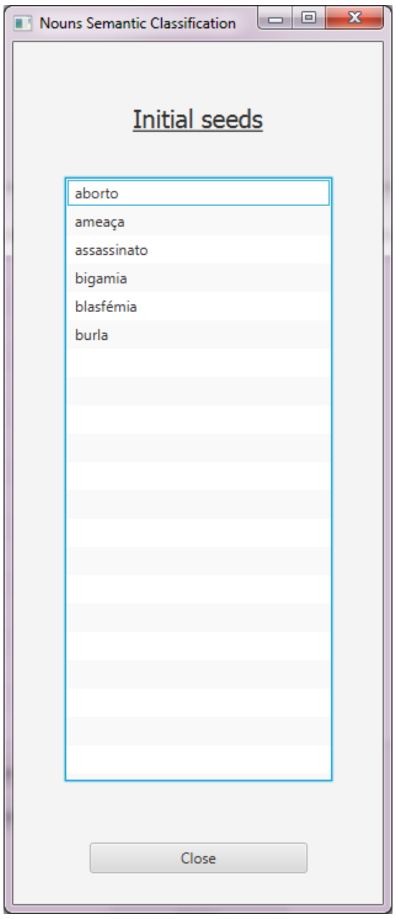


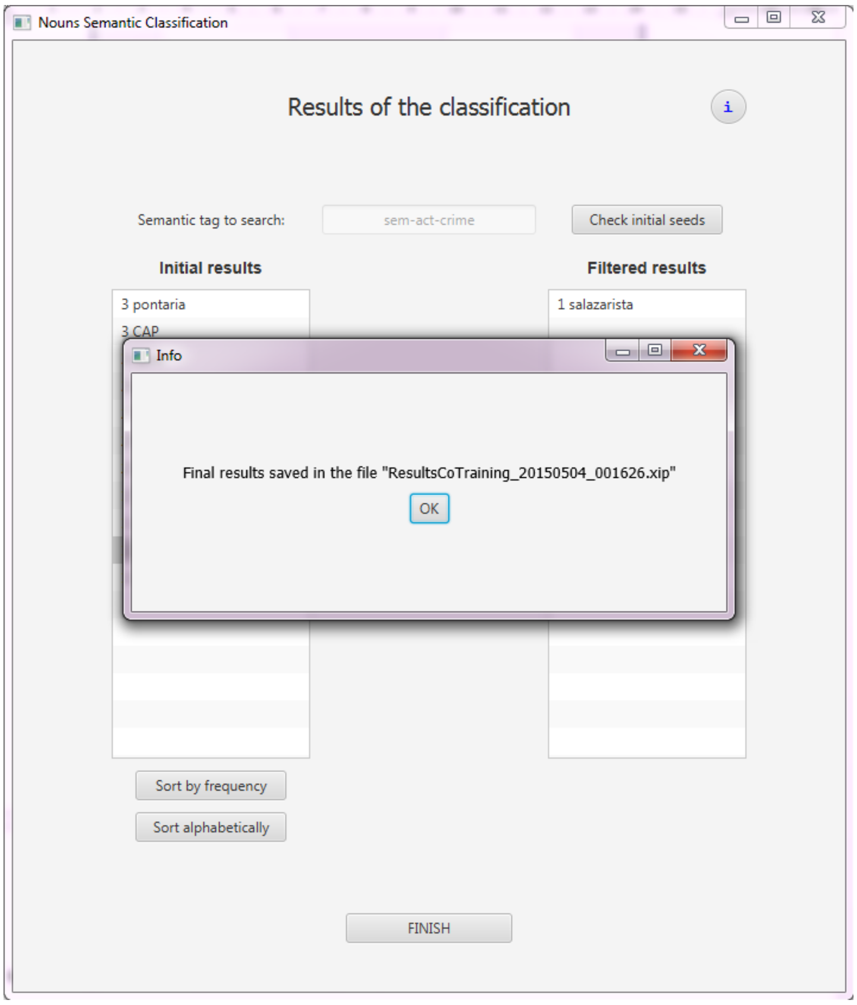
Figure 19: Graphic's UI facility to consult the initial seeds provided to the algorithm.

When the user presses *Finish* (or types *CONTINUE)*, the filtered list of new nouns he created is written to a textual XIP file.

The final results are presented in a file with the following template, which matches the template found during the Data Analyzer module development, in the XIP files (for exemplification purposes, the noun *salazarista* was considered as a correct classification):

```
salazarista:    noun += [sem-act-crime=+].
```

As mentioned before, when running the console version of the algorithm, the final *Results* file will contain all the words found, without any filtering nor validation. The validation, in this mode, must be done manually on the content of this file, by deleting the erroneous classifications presented.



Final results saved in the file "ResultsCoTraining_20150504_001626.xip"

Figure 20: Graphic UI with final results' file information.

5 ANALYSIS OF RESULTS

During the development of this work, many adjustments and experiments were needed in order to achieve success.

This chapter presents these experiments results, the description of the various attempts on achieving results by the algorithm, the adjustments made, and the final sampling results obtained.

5.1 DATA ANALYSIS

The main result and conclusion achieved from this step is that **the source (in terms of content) of the corpus determines the final results**.

Since our application aims to expand the set of Portuguese semantically classified nouns available at the lexicon of the L^{2}F's STRING processing chain, the first step of the implementation of this project was developing a **Data Analysis** module, whose purpose and implementation is fully described in Section 4.1.

In resume, this module concerns the acquisition of knowledge regarding the current state of the data set of classified nouns, in order to obtain a view of the distribution of nouns on semantic categories to sustain our decision of which categories should be expanded for data set richness purposes.

The data acquired was manually processed to cluster the various semantic tags in their higher parent-tags (named *umbrellas*), because the purpose is to obtain an overview by the most generic category (e.g. 'Animal') rather than its different sub-categories (e.g. 'Aquatic Animal', 'Animal Anatomy', etc.). For this purpose, the file '*features.xip*', part of the XIP grammar, was used as guide, because it describes the hierarchy of the tags that are part of the XIP system.

Table 1 presents the results of the Data Analysis module after processing the XIP files containing the set of Portuguese classified nouns, available at the lexicon of the STRING system. The results in this table are organized according to the umbrella – a parent category – to whom they belong. Considering these results, we observe that the set of Portuguese classified nouns is rich in nouns belonging to the semantic categories related to human beings (HUMAN), animals (ANIMAL) or actions (ACTION), for example, while it is clearly poor in nouns regarding the semantic categories related to weather (WEATHER), food (FOOD) or clothing (CLOTHING).

After analyzing the results, we first decided to expand the categories that had the least amount of seeds, aiming to acquire new knowledge to our data set. However, we believe that the source of the corpora influences the final results. In this case, since we used a corpus of journalistic sources, we expected some semantic categories to be poorly represented in the corpus, so we knew this would affect the results negatively:

- It reduces the accuracy of the results, since a lack of seed sentences leads to a lack of first-iteration gathered sentences, which are the most accurate ones, because they are found by comparison with the initial seed sentences, without contexts being matched by second-degree (or later) comparisons;

While journalistic text can be rich in crime-related issues (`SEM-ACT-CRIME`) or actions developed by humans (`UMB-HUMAN`), one easily confirms that it is very poor on what concerns food (e.g. `SEM-FOOD`) or musical instruments (`SEM-TOOL-MUS`), for example. It is completely understandable, since we do not usually find this kind of information on journals dedicated to news reports.

This has to be taken into consideration when using the application, since one cannot expect to find good quality results on semantic categories poorly represented in the corpus.

9044	**HUMAN**	**386**	**THING**	**408**	**GEO**
7623	SEM-HGROUP	125	SEM-CC	142	SEM-BUILDING
374	SEM-HPROF	110	SEM-CC-R	70	SEM-BUILDING-DENP
335	HUMAN	36	SEM-CC-MACH	67	SEM-LTOP
217	SEM-HH	32	SEM-CC-FURN	29	SEM-LPATH
166	SEM-H	19	SEM-CC-BAR	27	SEM-URB
91	SEM-HNAT	14	SEM-CC-LIGHT	21	SEM-SCHOOL
85	SEM-HATTR	11	SEM-CC-STONE	12	SEM-WATERMASS
39	SEM-HTIT	10	SEM-CC-HANDLE	11	SEM-LSTAR
32	SEM-HIDEO	9	SEM-CC-STICK	11	SEM-WATERCOURSE
30	SEM-HFAM	7	SEM-CC-RAG	11	SEM-LPATH2
21	SEM-HMYTH	6	SEM-CC-FIRE	5	SEM-ISLAND
19	SEM-HBIO	5	SEM-CC-BOARD	2	SEM-DIVISION
11	SEM-H-NOMAGENT-CRIME	1	SEM-CC-PARTICLE		
1	SEM-HSICK	1	SEM-CC-BEAUTY		
505	**CONCEPT**	**7755**	**ANIMAL**	**507**	**SEMANTIC PRODUCT**
141	SEM-OCC	3062	SEM-AORN	170	SEM-SEM-R
119	SEM-INST	1648	SEM-ADOM	121	SEM-SEM-C
104	SEM-MON	1592	SEM-AZO	61	SEM-SEM-S
54	SEM-ISM	1150	SEM-AICH	52	SEM-SEM-W
42	SEM-LING	217	SEM-AENT	40	SEM-SEM-L
25	SEM-GEOM	27	SEM-A	39	SEM-SICK
11	SEM-GAME	24	SEM-AA	18	SEM-SICK-C
7	SEM-META	13	SEM-AADOM	5	SEM-SEM
1	SEM-GEOM-LINE	12	SEM-ACELL	1	SEM-SEM-NONS
1	SEM-GENRE	10	SEM-AMYTH		

Table 1: Results (1/3) of the data analysis made to the set of already classified nouns (tags HUMAN, THING, GEO, CONCEPT, ANIMAL, and SEMANTIC PRODUCT).

1533	ACTION	178	THING/UNCOUNTABLE	326	EVENT
646	SEM-ACTION	72	SEM-CM	181	SEM-EVENT
386	SEM-ACT-CRIME	32	SEM-MAT	50	ORGANIZED
182	SEM-ACTION-D	23	SEM-CM-LIQ	25	SPORTS
166	SEM-ACTIVITY	19	SEM-CM-CHEM	25	EVENT
106	SEM-ACTION-S	14	SEM-MAT-CLOTH	20	NATURAL
19	SEM-ACADEMIC-DEGREE	8	SEM-CM-H	10	ARTISTIC
12	SEM-ACTION-FIGHT	7	SEM-CM-REM	7	SCIENTIFIC
9	SEM-ACTION-TRICK	3	SEM-CM-GAS	8	POLITICAL
7	SEM-ACTION-BEAT				
267	**STATE-OF-AFFAIR**	**195**	**LOCATION**	**175**	**TOOL**
88	SEM-SIT	114	SEM-LABS	119	SEM-TOOL
64	SEM-STATE-H	33	SEM-LOPENING	22	SEM-TOOL-MUS
41	SEM-TEMP	21	SEM-L	13	SEM-TUBE
30	SEM-STATE	14	SEM-LCOVER	9	SEM-TOOL-GUN
24	SEM-TALK	7	SEM-LTIP	9	SEM-TOOL-CUT
18	SEM-SPORT	3	SEM-LTRAP	3	SEM-TOOL-SAIL
2	SEM-THERAPY	3	SEM-LSURF		
253	**PERCEPTION**	**245**	**FEATURE**	**86**	**FOOD**
75	SEM-PROCESS	90	SEM-F	20	SEM-FOOD
64	SEM-PERCEP-F	70	SEM-F-PSYCH	19	SEM-FOOD-H
37	SEM-PICT	26	SEM-F-Q	15	SEM-DRINK
35	SEM-PERCEP-L	19	SEM-F-H	15	SEM-FOOD-C-H
21	SEM-POS-SOC	18	SEM-F-RIGHT	13	SEM-FRUIT
14	SEM-PERCEP-W	12	SEM-F-AN	3	SEM-FOOD-C
2	SEM-POS-AN	10	SEM-F-C	1	SEM-SPICE
5	SEM-PERCEP-O				
164	**DIVERSOS**	**63**	**PLANT**	**106**	**COLLECTIVE**
96	SEM-DOMAIN	28	SEM-BB	71	SEM-AMOUNT
25	SEM-DIR	15	SEM-BTREE	18	SEM-COLL-CC
21	SEM-CORD	11	SEM-B	10	SEM-COLL-SEM
9	SEM-COL	6	SEM-BVEG	6	SEM-COLL
7	SEM-CONV	3	SEM-BFLO	1	SEM-COLL-B
6	SEM-DANCE				
60	**CLOTHING**	**72**	**VEHICLE**	**26**	**EVENT FEATURE**
42	SEM-CLOH	37	SEM-VEHICLE-LAND	9	ART-SESSION
7	SEM-CLOH-BEAUTY	21	SEM-VEHICLE-WATER	5	PARTY
6	SEM-CLOH-HAT	10	SEM-VEHICLE-AIR	5	MEETING
5	SEM-CLOH-SHOE	4	SEM-VEHICLE-COLLECTIVE	4	CULT
				3	PARADE

Table 2: Results (2/3) of the data analysis made to the set of already classified nouns (tags ACTION, THING/UNCOUNTABLE, EVENT, STATE-OF-AFAIR, LOCATION, TOOL, PERCEPTION, FEATURE, FOOD, DIVERSOS, PLANT, COLLECTIVE, CLOTHING, VEHICLE and EVENT FEATURE).

29	**GROUP**	**26**	**ANATOMICAL/ANIMAL**	**152**	**ANATOMICAL**
23	SEM-RELIEF	17	SEM-ANBO	53	SEM-ANMOV
3	SEM-HISTORY	4	SEM-ANZO	40	SEM-ANORG
2	SEM-ASTRO	3	SEM-ANENT	40	SEM-AN
1	SEM-CIV	2	SEM-ANORN	19	SEM-ANOST
244	**DISCIPLINE**	**1007**	**ABSTRACT**	**23**	**WEATHER**
122	DISCIPLINE	513	SEM-AM	7	SEM-WEA
84	SPORTS-DISCIPLINE	359	SEM-AC	7	SEM-WEA-WIND
33	SCIENCE-DISCIPLINE	90	SEM-CAT	5	SEM-WEA-C
5	ART-DISCIPLINE	45	SEM-SIGN	4	SEM-WEA-RAIN
28	**TAGS NP**	**98**	**UNITYPE**	**97**	**PART**
25	SEM-ORG	65	SEM-CON	80	SEM-PART-BUILD
1	SEM-MEDIA	17	SEM-CURRENCY	12	SEM-PART
1	SEM-PARTY	16	SEM-MEASOTHER	5	SEM-PIECE
1	SEM-ADMIN				
104	**TIME**	**10**	**GEO-TOPONYM**	**30**	**LANGUAGE**
91	SEM-PERIOD	5	LEGAL	29	SEM-LUS
12	SEM-DUR	5	VIRTUAL	1	SEM-BR
1	SEM-MONTH				
8	**PARSING FEATURE CIRCUNSTANCE**	**39**	**AUXILIAR-FEAT. NER**	**89**	**OTHER**
4	MANNER	39	INTROD-HUMAN-CONST	89	ACRON
4	NOP				
55	**USAGE**	**14**	**PREDICATIVE NFEAT**	**6**	**HABITATION**
55	SEM-RARE	14	SCOMETER	6	LIVE-IN

Table 3: Results (3/3) of the data analysis made to the set of already classified nouns (tags GROUP, ANATOMICAL/ANIMAL, ANATOMICAL, DISCIPLINE, ABSTRACT, WEATHER, TAGS NP, UNITYPE, PART, TIME, GEO-TOPONYM, LANGUAGE, PARSING FEATURE CIRCUNSTANCE, AUXILIAR-FEAT NER, OTHER, USAGE, PREDICATIVE NFEAT and HABITATION).

5.2 RESOURCES MANAGER

The main result and conclusion achieved from this step is that **no perfect balance can be achieved between resources management and quality of the results.**

The management of resources revealed to be one of the most complex tasks, and it needed a lot of experiments with the corpus' size in order to achieve results in a reasonable time, without dramatically damaging and influencing the quality of the results

The first experiments considered the whole corpus being processed at once, what lead to no results in reasonable time – either because the application ran out of memory and physical resources to store the data created on runtime, being terminated by the system, or because the application was unable to provide any kind of result in reasonable time, since the way it was designed would only provide results when the algorithm ended all the iterations, what was impossible due to the resources extinction.

Secondly, the corpus was considered to be divided into its 20 partial folders and processed according to these blocks. However, the application was still having resources management issues, and no results were obtained in the time lap of 2 weeks, when testing the semantic category `SEM-ACT-CRIME`, which we considered to be highly probable to be found in the corpus, thus a longer runtime was expected for this test case.

Lastly, we started equating the processing of the parts of the corpus in smaller parts.

The splitting of each part in equal blocks of text led to unbalanced resources usage and, even if the results were being provided after the processing of each block of text, the final results were still an issue: splitting the text in equal blocks proved to be fallible on what concerns balancing the load on the system, because, considering the worst-case scenario, one cannot guarantee that the whole seed set of sentences would not be present in a single one of these text blocks, clearly creating an huge gap of load balance between poor-sentence-blocks and rich-sentence-blocks.

Another problem verified in this first implementation was the fact that we were considering the results of each block of the corpus algorithm's execution to be propagated to the following blocks of corpus, by propagating the seed sentence's structure in memory and passing to the subsequent blocks' execution. Thus, although the various blocks were processed individually, the final results of their execution were propagated to the following blocks' execution. Despite this providing a more accurate comparison process, because the whole corpus would had a chance to match with a good (or bad – containing a wrongly classified word) context, and achieving more accurate results, it also provided a lot more garbage words, with the disadvantage of maintaining an exponential increase in runtimes of the test-cases, because for each seed sentence found, the whole corpus had to be matched against it, which is a very heavy task.

A different approach was chosen to solve this issue: the corpus should be split dynamically, in such a way that the seed sentences presence would be split among the whole set of blocks of file, instead of having the risk of getting all of them in a single block. To achieve this, the *Resources Manager* module split the blocks of text in parts containing no more than *N* seed sentences among the remaining corpus. This blocks should be processed individually, without sharing their results with further blocks being executed, to avoid heavy comparison processes.

After this result, the focus became that of discovering the better value of *N*, which should be the smallest possible to reduce the risk of quality loss induced in the final results by having the corpus being processed in too many slices.

Experiments were made with N values starting at 20, and decreasing. None of the values was able to provide results in less than 3-4 days. The *N* value we found to be reasonable was 5, which divides the corpus in blocks of text that contain at most 5 seed sentences among the remaining sentences, and which can produce results in 1 to 2 days maximum, depending on the richness of the semantic category being processed.

However, since this split factor is a matter of discussion and disagreement, we considered that the human user should be the one to judge about what would be the reasonable time, to him, to get results from the application algorithm. Thus, this parameter is configurable from the user interface or the console mode, allowing the user to decide if he wants to privilege quality of results or runtime.

5.3 PARSER

The dependencies being filtered and collected from the XML files with output from the STRING system are another of the elements that influences the results obtained from the classification task.

While considering the whole universe of dependencies available in the system would lead to more accurate comparisons, it would, at the same time, reduce the matching rate of contextual features between seed sentences and corpus sentences, because it would enforce for more rules to match in the process.

From the relevant dependencies mentioned in Section 4.4, some underwent a filtering process in the pairing process of dependencies and respective nodes, during a later stage of the parsing process.

Sentences containing the *FIXED* dependency are discarded during the parsing, because this dependency is associated with idiomatic expressions, a peculiar language case that is still under research at the L^{2}F. Using this sentences would influence the results negatively, because the meaning of the sentence is not solely dependent on the meaning of the words in its content, but also from the meaning it acquires in the popular expression of an idiom.

Similarly, sentences containing the dependencies *AGENT* and/or *PATIENT*, which are related to sentences whose verb had been subject to the passive transformation, are also discarded due to the complexity their syntactic structure would impose in the algorithm processing.

The *MOD* dependency was first considered to be discarded from the comparison process as well, since, at first sight, it provides no relevant information about an action, since it only classifies the way it was done. For example, consider the following sentence:

`O Pedro comeu a sopa rapidamente.` MOD (comeu, rapidamente)

(Pedro ate the soup quickly).

The modifier (*MOD*) 'fast' provides no relevant information about the main action of the sentence, which is the fact that Pedro ate the soup. It only informs about the way Pedro did it.

Forcing the comparison of contexts to match the *MOD*'s of the sentences, would reduce the success and range of the classification, because it would enforce the existence of modifiers in both sentences: in case of the absence of the *MOD* in one of them, even if all the remaining elements match, the comparison would fail because of the inexistence of the *MOD*. Consider the following example:

```
O Miguel comeu o bife.
```

(Miguel ate the steak).

An human would clearly consider this sentence as matching the first one, however, since the *MOD* dependency does not exist in this last sentence, the system would fail the matching comparison, and we would be losing a relevant word that was semantically related to the seed sentence matched against,

Worst results would arise if not only we forced the existence of the *MOD* dependency in both sentences being compared, but also that the word that determines the way the action was done should be equal in both sentences. For exemple, consider the following sentence:

```
O João comeu a bolacha lentamente.
```

(João ate the cookie slowly).

MOD (comeu, lentamente)

While the contexts of the sentences clearly match, and the nouns are related to the food semantic category, the comparison of the *MOD* dependency would fail, since one describes the action as fast while the other describes it as slowly, since the words do not match, the comparison would fail, and we would be lacking a relevant sentence.

However, in the XIP grammar (Mamede and Baptista, 2014), the *MOD* dependency do not represent uniquely adverbial modifiers of verbs, as the examples presented in the previous paragraphs.

A *MOD* dependency is also used to represent:

- Associations of adjectives with nouns:

  ```
  Um dia solarengo.
  ```

  (One sunny day.)

MOD (dia, solarengo)

- Certain determinative focus adverbs:

  `Li precisamente esse livro.`

  (I read exactly that book.)

  MOD_FOCUS (livro, precisamente)

- Locative prepositions:

  `O livro estava em cima da mesa.`

  (The book was on the table.)

  MOD (em cima de, livro)

- Prepositional phrase adjuncts:

  `O medo de ficar preso nos elevadores paralisa-me.`

  (The fear of getting stuck in the elevator paralyzes me.)

  MOD (preso, elevadores)

- Sequential (circumstantial) adjunct sub-clauses:

  `O Pedro comia rebuçados enquanto lia o jornal.`

  (Pedro ate candies while reading the newspaper.)

  MOD (comia, lia)

- Relative clauses:

  `O Pedro leu o livro que a Ana lhe deu.`

  (Pedro read the book that Ana gave him.)

  MOD_POST_RELAT (livro, deu)

Thus, the *MOD* dependency revealed to be essential in various syntactic constructions of sentences, and whose exclusion would compromise the correct pairing process while matching contexts between corpus and seed sentences. For this reason, the *MOD* dependency was not excluded from the list of relevant dependencies.

5.4 Co-Training

In this section, results of the test-cases applied to the algorithm are presented. The algorithm was configured to run under the following conditions:

- Run in the *console mode*, by passing the semantic tag to search for as a first argument of the program, and the split factor as a second argument of the program;
- The algorithm uses the CETEMPúblico corpus available in the FFS system at the INESC-ID machines;
- The initial seed lists are not subject to filtering, thus they are considered to be reliable;
- The final seeds list are not subject to filtering as well, thus we present a long list of all the seeds found for the semantic category provided, in order to study the percentage of correct results among all the discarded ones.

Since we are running the console version of the algorithm, the final *Results* file will contain all the words found. The validation, in this mode, must be manually done to the content of this file, by deleting the erroneous classifications presented. We will be doing this validation in the test-cases present, to evaluate the success rate of the classification.

We opted to test the execution of the algorithm with two different split factors in order to evaluate the difference in the results obtained by both, and detect how it influences the quantity and quality of the results. However, we tested values that were not significantly distant, in order to evaluate the difference that a small augment or diminution on the split value causes to the results and success rates, because we believe it effectively affects them, but we want to measure how much it does with a small difference in the value, rather than testing extreme values and conclude that it affects, but now knowing exactly how much for a small portion.

The split factor is, by default, 5 – the value that offers the better balance between time consumption to run the algorithm, and corpus division in blocks, in order to not dramatically harm the results. With the purpose of testing the influence of the split factor in the results obtained, the search task is run with the default split factor of 5, and with a split factor of 10, two different values that are able to provide results in a reasonable time.

Results with a split factor of 0, meaning no corpus partition, would produce the most accurate results. However, these results take approximately 20 days to be generated (approximately 1 day per Part of the corpus in use), thus for time limitations it was impossible to present this kind of results.

As mentioned before, the chosen categories for testing and demonstration purposes are the ones that we considered to be capable of reflecting and sustain our belief that the corpus source influences the resources obtained, due to the poor representativeness of seed examples of some semantic categories in a corpus of journalistic source, because some topics are, in our point of view, not usually mentioned in texts of newspapers, as ‘clothing’, for example. This fact is what leads to poor representativeness of some semantic categories.

To study this influence, we opted to explore opposite categories, in terms of their representativeness in the set of classified nouns, but also opposite on their probability to be found in a corpus of journalistic source, considering our believe that some topics are less likely to be found in this type of corpus. Thus, we considered two factors to choose the semantic categories to test:

- Probability of finding samples of the semantic category in a corpus of journalistic source;
- Quantity of nouns classified with the semantic category in the set of already classified nouns.

Considering this, we choose some categories which we considered highly expected to be found in the corpus, while others are less probable to be found.

Our choices were:

- `SEM-ACT-CRIME` – corresponds to criminal acts, which we considered to be highly probable to be found in a journalistic corpus.
- `SEM-CC-STONE` – corresponds to stones or stone-sized round objects, like stone, ammonite, brick, diamond, etc., which we considered to be less probable to be found in journalistic text;
- `SPORTS` – corresponds to sporting events, which we considered to be reasonably probable to be found in journalistic text;
- `SEM-TOOL-MUS` – corresponds to musical tools (instruments), which we considered to have low probability to be found in the corpus.

Since the collection of non-filtering results' files present a huge quantity of words (~50.000 words) we cannot present them in this text, thus we opted to present a brief description of the relevant results obtained along with our analyze.

Table 4 and *Table 5* present the results of the execution of the co-training algorithm for the semantic categories chosen for testing, with the corresponding split factors of 5 and 10. The columns' labels explanation is the following:

- **Initial seeds** – presents the number of nouns classified with the target semantic tag which were contained in the set of classified nouns that is part of the lexicon of the STRING system, and are thus fed to the system as initial seeds;
- **Nouns already classified** – presents the number of classified nouns returned by the algorithm that were already part of the initial lexicon used as input, they do not provide new information, thus they are not provided to the user for filtering;
- **Nouns classified and not matching** – presents the number of classified nouns returned by the algorithm that were already part of the lexicon of the STRING system, however they are classified with a different semantic tag, thus they represent *false positives*, thus they are not provided to the user for filtering;
- **Nouns not classified** – presents the number of nouns returned by the algorithm that are not part of the lexicon of the STRING system, thus they represent new nouns whose classification must be validated by the user. These constitute the *resulting set presented to the user*, containing potential new classifications that might enrich the lexicon, after validation;

- **Validated nouns** – presents the number of nouns that, after manual validation by us, revealed to be correct new classifications acquired for the target semantic tag requested;
- **Success rate** – percentage that relates the number of *validated nouns* with the number of *nouns not classified*;
- **Runtime** – presents the execution runtime of the test-case.

split factor	**5**						
category	**initial seeds**	**nouns already classified**	**nouns classified and not matching (FP)**	**nouns not classified**	**validated nouns (correct classifications) (TP)**	**success rate (correct new classifications) %**	**runtime**
SEM-ACT-CRIME	381	5	3400	3236	27	0,834	9h30m
SEM-CC-STONE	11	1	2777	1770	6	0,339	8h30m
SEM-TOOL-MUS	22	4	3115	2446	6	0,245	11h30m
SPORTS	25	0	3875	5016	9	0,179	14h40m

Table 4: Summary of the results of the co-training algorithm for the test cases with split factor of 5.

split factor	**10**						
category	**initial seeds**	**nouns already classified**	**nouns classified and not matching (FP)**	**nouns not classified**	**validated nouns (correct classifications) (TP)**	**success rate (correct new classifications) %**	**runtime**
SEM-ACT-CRIME	381	5	3400	3236	27	0,834	22h20m
SEM-CC-STONE	11	2	4129	7133	11	0,154	13h15m
SEM-TOOL-MUS	22	4	3127	2483	6	0,242	11h10m
SPORTS	25	0	4528	13364	14	0,105	21h40m

Table 5: Summary of the results of the co-training algorithm for the test cases with split factor of 10.

The validated nouns for each semantic category are discriminated in the tables *Table 6* to *Table 9* presented next. Each table presents the results obtained for a specific semantic category, with a column for each split factor. The nouns are presented with the number of occurrences (number of times the algorithm identified the noun as belonging to the semantic category) in the column on its right. We considered this value could be helpful during the validation task, since it can be interpreted as the confidence of the classification by the algorithm.

It is worth mention that despite the poor results, many words related to the topic covered by the semantic tags were detected by the system – this analysis is presented together with the tables of results of each semantic category. Words marked with (*) correspond to these topic-related cases.

semantic tag	SEM-CC-STONE				
split factor	5			10	
	gravilha	1		brilhante	6
	magma	2		calcário	1
	oiro	2		estalagmite	1
	pedrouço	1		gravilha	1
	rochedo	1		menir	1
	xisto	1		mineral	1
				oiro	3
				rubi	1
				sedimento	1
				taipa	2
				talismã (*)	2
				xisto	1

Table 6: Nouns identified as belonging to the SEM-CC-STONE category.

Considering the semantic category SEM-CC-STONE, the algorithm curiously identified the noun '*talismã*'. Although this noun does not correspond to a correct classification, it is related to the topic, if we consider that a '*talismã*' can be considered a jewel to the person using it. This reveals this semantic category as an ambiguous one, on what concerns judgment of its elements, because what one individual considers to be a SEM-CC-STONE (stone, rock, mineral, gemstone), another judge can discard, it depends on each one's interpretation of the category and also on the 'emotion' or emotional attachment a word transmits to him, in this particular case, a '*talismã*' can be considered a gemstone depending on the emotional value we attribute to it.

semantic tag	SEM-TOOL-MUS				
split factor	5			10	
	bombo	2		bombo	2
	carrilhão	1		carrilhão	1
	guizo	1		guizo	1
	harmónica	2		harmónica	2
	maraca	1		maraca	1
	tamborzinho	1		tamborzinho	1

Table 7: Nouns identified as belonging to the SEM-TOOL-MUS category.

The semantic category SEM-TOOL-MUS does not have the particularity of suffering from ambiguous considerations about the validation of its elements, because musical instruments are objects very well defined and not dependent on each one's interpretation, nor personal emotional considerations of the object.

semantic tag	SEM-ACT-CRIME				
split factor	5			10	
	apartheid (*)	1		apartheid (*)	1
	cabala	1		cabala	1
	carnificina	6		carnificina	6
	cilada (*)	1		cilada (*)	1
	crucificação (*)	1		crucificação (*)	1
	dominação (*)	1		dominação (*)	1
	dopagem	1		dopagem	1
	enforcamento	2		enforcamento	2
	esfaqueamento	1		esfaqueamento	1
	esticão (*)	1		esticão (*)	1
	flagelação	1		flagelação	1
	hooliganismo	1		hooliganismo	1
	ilicitude (*)	1		ilicitude (*)	1
	incesto	1		incesto	1
	infâmia (*)	2		infâmia (*)	2
	intimação (*)	1		intimação (*)	1
	masoquismo (*)	1		masoquismo (*)	1
	mutilação	1		mutilação	1
	obscurantismo (*)	1		obscurantismo (*)	1
	perversidade (*)	1		perversidade (*)	1
	pide (*)	1		pide (*)	1
	poligamia	1		poligamia	1
	selvajaria (*)	1		selvajaria (*)	1
	separatismo (*)	1		separatismo (*)	1
	tirania (*)	3		tirania (*)	3
	ultraje (*)	1		ultraje (*)	1
	viciação	1		viciação	1

Table 8: Nouns identified as belonging to the SEM-ACT-CRIME category.

The identification of nouns belonging to the semantic category `SEM-ACT-CRIME` is perhaps one of the most ambiguous task. This arises from the fact that the definition of a crime is not well defined, unless we study laws to become experts on the topic and become able to clear identify what is a crime, and what is not, according to law. Considering this, most of the words validated by us are ambiguous enough so that a judge can decide to exclude them, while other can decide to consider them as crimes, again, using his own emotion or emotional connection to the word being validated, which is determined by each one's living experience.

Although these words constitute an high number of results, many other words, that we were sure enough could not be classified as crimes, were returned by the algorithm and it was clearly noticeable that the algorithm was able to establish a connection between the topic englobing the semantic category (crimes, malicious actions, words with negative connotation in the society) and the contexts matched during the execution of the algorithm, since a pattern was somehow established. The words that we encountered and consider to be related to the topic covered by this semantic category, thus we consider them to be malicious actions or have a negative connotation in the society, are the following:

- coscuvilhice (1)
- cábula (1)
- drogado (1)
- compulsão (1)
- borlista (6)
- demérito (4)
- cobarde (2)
- cobardia (3)
- rebate (1)
- gueto (1)
- dominação (1)
- dominador (1)
- dominante (2)
- mafioso (2)
- prevaricador (1)
- racista (11)
- desavença (1)
- descaramento (2)
- bruxaria (1)
- trafulhice (1)
- gatuno (2)
- gangue (1)
- detrator (1)
- machadada (2)
- alfinetada (2)
- infâmia (2)
- sanha (3)
- maldade (2)
- aldrabão (1)
- vagabundagem (1)
- devedor (3)
- manipulador (1)
- álibi (1)
- codícia (1)
- desertor (1)
- maledicência (2)
- taliban (2)
- desacato (1)
- estupefação (1)
- tarado (2)
- processo-crime (1)
- vaia (1)
- barbaridade (4)
- vadiagem (1)
- tabagismo(1)
- tabefe (1)
- asfixia (1)
- machismo (1)

As visible, many words considered related to negative acts or events, or having a negative connotation in the society, were returned by the algorithm, thus it is conclusive that the algorithm is able to establish a pattern that relates contexts and a determined topic that englobes not only the nouns belonging to the target semantic category but also related words.

semantic tag	**SPORTS**				
split factor	**5**			**10**	
	contra-relógio	1		contra-relógio	26
	despique (*)	1		despique (*)	3
	EuroLiga	1		EuroLiga	2
	F1	4		F1	4
	jogo-treino	1		jogo-treino	3
	Roland-Garro	1		meia-maratona	1
	sub-16	1		regata-treino	2
	sub-20	1		Roland-Garro	1
	sub-21	1		sub-16	2
				sub-18	1
				sub-20	1
				sub-21	6
				sub-22	1
				treino-conjunto	1

Table 9: Nouns identified as belonging to the SPORTS category.

On what concerns the semantic category SPORTS, many considerations arise. Not only problems of ambiguity of identification of events arise, but also the fact that some nouns correspond to the shortest form of event-nouns. For example, the noun '*Roland-Garro*' represents the shortest form of referring to the '*torneio de ténis de Roland-Garro*' ('*Roland-Garro tenis' tournament*') or simply '*torneio de Roland-Garro*' ('*Roland-Garro's tournament*').

The particular case of the marked word, '*despique*', concerns the fact that despite it not being clear if it can be considered an event, it surely is related to the competitive spirit existing during the practice of sports.

Also in this test-case, some words related to the topic sports, but not specifically sports events, were found, such as:

- penalti (36)
- fora-de-jogo (5)
- presidente-árbitro (1)
- treinador-jogador (6)
- futebolístico (1)
- crosse (17)
- rali (2)
- cricket (1)
- equitação (2)
- dérbi (2)

During the analysis of the lists of nouns to validate, we noticed that some words could be automatically excluded from the lists, because they would never be categorized in any of the semantic tags available at the lexicon of the STRING system. This is the case of single letters, like '*a*', or words constituted by numbers, like '*12-34-NM*' which corresponds to a car registration, for example. Thus, considering the possible increase in the quality of the results, we decided to apply some filters to the existing lists of nouns produced as results of the test-cases, and also include this improvement in the results validation procedure of the algorithm, applied at its latest stage, right before results presentation to the user.

The filtering of the lists of nouns to be validated allows to reduce the number of erroneous classifications (which correspond to garbage), and thus save time and resources during the user's validation task, but also results in an improvement of the success rates of the algorithm.

The filters we found to be appropriate are:

- Remove words containing numbers;
- Remove single letters;
- Remove words starting with capital letter;
- Remove words containing foreign letters (k, y, w).

These filters are applied through the use of regular expressions, which are used to test each noun for matching with the corresponding regular expression. Nouns that match the regular expression of a filter are excluded.

Table 10 presents the regular expressions corresponding to each filter applied to the resulting lists of nouns, and implemented in the results' validation procedure of the application, the latest stage of the algorithm's execution, right before presenting the results to the user.

FILTER'S DESCRIPTION	REGEX APPLIED
(1) Remove words containing numbers	^\d+ [^\n]+\d[^\n]+\n
(2) Remove words constituted by single letters	^\d+ .\n
(3) Remove words starting by capital letter	^\d+ [A-ZÇÁÀÉÍÓÚÃÕ][^\n]+\n
(4) Remove words containing foreign letters (k, y, w)	^\d+ [^\n]*[kyw][^\n]*\n

Table 10: Regular expressions corresponding to each filter applied to the resulting list of nouns.

Table 11 present the results of applying these filters to the lists of nouns to be validated, demonstrating the reduction of existent garbage in these lists, which will improve the success rates of the algorithm but most of all, save time to user who is validating the results. Each pair of intermediate rows corresponds to the application of a filter identified by an identifier (e.g. '*(1)*') that matches the filter's description presented in *Table 10*. The first and last rows of the table are highlighted to easily identify the consequence of applying the filters, allowing for easy comparison between the initial number of nouns and the resulting number of nouns, for each semantic category.

FILTER'S DESCRIPTION	SEM-ACT-CRIME		SEM-CC-STONE		SEM-TOOL-MUS		SPORTS	
	5	10	5	10	5	10	5	10
initial number of nouns	**3236**	**3236**	**1770**	**7133**	**2446**	**2483**	**5016**	**13364**
number of words containing numbers	9	9	5	11	9	10	19	24
number of nouns after applying filter (1)	3227	3227	1765	7122	2437	2473	4997	13340
number of words containing singular letters	11	11	11	16	9	9	13	18
number of nouns after applying filter (2)	3216	3216	1754	7106	2428	2464	4984	13322
number of nouns starting with capital letter	953	953	473	2440	695	710	1595	5293
number of nouns after applying filter (3)	2263	2263	1281	4666	1733	1754	3389	8029
number of nouns containing foreign letters (k, y, w)	32	32	10	85	21	21	61	154
number of nouns after applying filter (4)	2231	2231	1271	4581	1712	1733	3328	7875
resulting number of nouns	**2231**	**2231**	**1271**	**4581**	**1712**	**1733**	**3328**	**7875**

Table 11: Results of the filtering applied to the resulting lists of nouns.

Although these filters help improving the quality of the resulting list of nouns, by reducing the amount of garbage in the results' lists, they can, however, negatively influence the number of nouns correctly matched and validated by us, reducing the correct identifications as well, by removing words that were actually correctly classified for a determined semantic tag.

Considering the test-cases chosen for demonstration, and analyzing the lists of validated nouns prior to application of filters, only the `SPORTS` semantic category identifications could be negatively affected by the application of the enumerated filters, because it is the only list containing words containing numbers, but also some nouns starting with capital letter, corresponding to names of events that could eventually be removed by this filters.

All the other categories' test-cases validated identified nouns do not match any of the filters applied, thus the amount of validated nouns remain the same for them, and the filtering only produces a positive impact, by reducing the amount of nouns to be validated, and consequently the existent garbage and time to validate these lists.

The list of correctly classified nouns found in the filtered list of results for the semantic tag `SPORTS`, resulting from the application of the filters described above, is presented in *Table 12*.

semantic tag	**SPORTS**				
split factor	**5**			**10**	
	contra-relógio	1		contra-relógio	26
	despique (*)	1		despique (*)	3
	jogo-treino	1		jogo-treino	3
				meia-maratona	1
				regata-treino	2
				treino-conjunto	1

Table 12: Nouns identified as belonging to the SPORTS category, after filtering.

Analyzing this new lists of validated nouns, for the semantic category `SPORTS`, we notice the loss of 6 nouns for the test-case with split factor 5, and the loss of 8 words for the test-case with split factor 10, as a result of the filtering applied. This loss in quantity of validated nouns affects the success rate percentage of the algorithm, however, one must not forget the fact that the usage of this filters also led to the decrease in the number of total nouns available for validation, by reducing the garbage-words existent in the set. The way this two conditions affected the percentage rate will be evaluated with the help of the following results' tables.

Table 13 and *Table 14* present the updated results, after applying the filters described above, of the execution of the co-training algorithm for the semantic categories chosen for testing, with the corresponding split factors of 5 and 10.

split factor	5						
category	initial seeds	nouns already classified	nouns classified and not matching (FP)	nouns not classified (after filtering)	validated nouns (correct classifications) (TP)	success rate (correct new classifications) %	runtime
SEM-ACT-CRIME	381	5	3400	2231	27	1,210	9h30m
SEM-CC-STONE	11	1	2777	1271	6	0,472	8h30m
SEM-TOOL-MUS	22	4	3115	1712	6	0,350	11h30m
SPORTS	25	0	3875	3328	3	0,090	14h40m

Table 13: Summary of the results of the co-training algorithm for the test cases with split factor of 5, after filtering.

split factor	10						
category	initial seeds	nouns already classified	nouns classified and not matching (FP)	nouns not classified (after filtering)	validated nouns (correct classifications) (TP)	success rate (correct new classifications) %	runtime
SEM-ACT-CRIME	381	5	3400	2231	27	1,210	22h20m
SEM-CC-STONE	11	2	4129	4581	11	0,240	13h15m
SEM-TOOL-MUS	22	4	3127	1733	6	0,346	11h10m
SPORTS	25	0	4528	7875	6	0,076	21h40m

Table 14: Summary of the results of the co-training algorithm for the test cases with split factor of 10, after filtering.

Analyzing the data documenting the success rates after filtering the list of results, it becomes clear that this filtering is essential, because all the success rates benefitted by a significant amount from the cleaning of the lists of validated nouns; for some test-cases, the success rates almost tripled.

The exception correspond to both test-cases of the `SPORTS` semantic category, justified by the fact that despite it being the category that achieved the highest decrease in the size of the lists of nouns to validate, as can be confirmed in *Table 11*, this filtering also resulted in a significant loss of correctly classified nouns, especially if we have in consideration that the total number of validated nouns was, since the beginning, very low; any removal from this small set could only result in a significant loss on the success rate, because the amount of garbage removed did not compensate for the loss in correctly classified nouns.

This is an important consideration to retain from the study of these filtering's impact: it can have a good or a bad consequence, or even a balance between the two. It is closely related to the matching of nouns achieved by the algorithm for the semantic tag being searched, considering that for some specific tags, like the `SPORTS`' one, it would be predictable that some events' nouns could start by a capital letter, or include numbers in its noun, thus suffering negative impact by the application of the filters, because those particular nouns end up being removed.

However, in the generality of the cases, the benefit from the filtering is very positive, thus the decision of applying them and implementing them in the latest stage of the algorithm's execution, before presenting the final results to the user, saving him time on the validation task.

The lists of nouns to validate were sorted alphabetically. The criteria of sorting by number of occurrences did not reveal benefic results, since at the top of the sorted results' lists can appear words that despite having high frequency, for being matched many times during the corpus' processing by the algorithm, are not related at all to the semantic category being searched. This is determined by the corpus characteristics, which can be rich in a specific context containing a specific word, that although it matches one relevant seed-context that includes a correct seed word, it is not semantically related to it, thus it is constitutes garbage, and pollutes the list of results. This makes impossible to use a sorting criteria based on frequency of occurrence of nouns.

For similar reasons, the usage of a filtering criteria based on frequency of occurrence of nouns cannot be applied as well. One could be tempted to decide to filter the resulting lists of nouns by having in consideration the number of occurrences, with the objective of excluding words that were found punctually, and thus contain 1 (or another small number) single occurrence in the algorithm's results. However, evaluating the tables of validated nouns presented, one easily verifies that a filtering criteria that excluded words with a single occurrence would dramatically harm the results, because many important nouns occur only a single time, but they are correctly identified. As mentioned before, this is related to the corpus characteristics: the fact that the corpus was obtained from a journalistic source, determines its content, which greatly influences the representativeness of some semantic categories.

Another factor that is greatly influencing the results obtained is the split factor. The division of the corpus in independent blocks where the algorithm runs individually is greatly affecting the contexts matched, because one good context can be found at block number one, but since the results are not propagated among blocks, if a matching context with a correct word exists in block number 2, it will not be found. Each block is processed individually, thus it is solely dependent on the contexts of the seeds it contains on its own. The propagation of results among different partitions of the corpus, although it would provide more accurate results, would also augment the execution times exponentially, leading to the biggest problem found during the implementation of this work: resources management, which is the objective of implementation of the split factor methodology. Thus, it was necessary to keep each block algorithm's execution independent.

The algorithm is based solely on comparisons among dependencies' constituents. Although this comparison has to obey to the rules described in Section 4.8, this revealed to be a very poor criteria, because many sentences are being matched, considering their composition in terms of dependencies, however they are not related at all to the semantic category being searched, neither can be considered to be somehow related to the topic that englobes that semantic category. This is visible by the amount of nouns that were matched and do not correspond to valuable identifications, constituting garbage that pollutes the resulting lists of nouns, decreases the success rates achieved and penalizes the validation task to be developed by the user.

To demonstrate why this matching procedure is fallible, the following example uses the seed sentence presented before in the example of Section 4.8 (resulting from the prior execution of the example described in Section 4.7) together with a new sentence that would be wrongly matched during comparison.

(seed sentence) `O Miguel comprou um queijo.`

Miguel bought a cheese.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, sem=SEM-FOOD-C-H, sem=SEM-FOOD-H, word=queijo, pos=NOUN, SEED]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Miguel, pos=NOUN]]]
```

(corpus sentence) `O Rui comprou um bombo.`

Rui bought a bass drum.

```
CDIR=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=8, word=bombo, pos=NOUN]]]

SUBJ=[[[ID=4, lemma=comprar, pos=VERB, class=36DT], [ID=2, sem=SEM-HPEOPLE, special=UMB-HUMAN, sem=SEM-HINDIVIDUAL, morph=PROPER, word=Rui, pos=NOUN]]]
```

Considering the explanation of the *Context Finder*'s module execution, and applying the same logic to the comparison process between these two sentences, one easily verifies that the *CDIR* dependency of both sentences is matched, because they are constituted by the same verb (it contains the same lemma and class) connected to a noun. Thus, the noun '*bombo*' is paired with the seed noun '*queijo*' and assumes the semantic category of the later one, being '*bombo*' declared as a *SEM-FOOD-C-H*.

This is why the comparison process needs to be stricter in order to avoid this kind of pairings that, despite being correct for the matching criteria imposed, reveal to be very poor on what concerns accuracy of the results obtained later in the execution.

An improvement to the quality of the results could be achieved if the classification task of this work focused not so specifically on some semantic-tags, but on the umbrella-tags that englobe all the possible semantic tags of a larger category (for example, instead of searching specifically for sports' events, one could search for the generic umbrella category EVENT), or one could search for a group of categories that englobed everything related to sports.

Similar classification works, like (Bel et al., 2012), used a cue-based approach to achieve automatic noun semantic classification in English and Spanish. This cues were obtained by exploring particular aspects of linguistic contexts, and resulted in the conclusion that nouns belonging to a semantic class tend to show up in a number of particular contexts, for example, they concluded that "*members of the class EVENT tend to co-occur with prepositions that refer to duration, i.e., during, and also with verbs whose meaning refers to events, i.e., to last.".* This approach presented very satisfactory results, with accuracy rates higher than 65%, and thus it was proven to be helpful in reducing the human work that is necessary in a classification tasks. This work identified the inadequacy of the lexicon as a cause of poor performance of many classification applications, and in their particular case, the well-known problem of data sparseness, concerning the low frequency of words and the particular cues needed to classify them, also affected their results negatively.

These conclusions correspond to the same conclusions our implementation of a classification algorithm demonstrated to us, through the analysis of the results obtained.

The study of the work of (Bel et al., 2012) reveals the faults of our implementation of the classification algorithm: although the co-training algorithm is a learning cyclic algorithm, where the two constituting classifiers gradually teach each other the rules to apply to achieve new classifications, this approach is very poor on what concerns accuracy of results, because it is based solely on comparison and matching of elements constituting dependencies. Although we imposed some matching constraints, as described in Section 4.8, these reveal to be insufficient to grant correct and relevant matches that actually result in correct classifications. This is revealed in the large amount of not classified nouns provided as results of our algorithm, which after validation presented very few relevant identifications and even lowest correct classifications.

Although our co-training approach provides some correct results, we consider the success rates to be unsatisfactory. A better solution would be to conjugate a co-training approach with some deeper study of the contexts of the sentences that are related to sample nouns of the semantic categories, in order to identify particular cases, like verb' lemmas, adjective words, etc., which are related with a particular semantic category, similar to what (Bel et al., 2012) did, and then including this data in the comparison process of the co-training, in order to make the matching process more strict and accurate, reducing the number of false positives acquired in the task.

However, despite not being accurate on the results provided, the algorithm revealed to be able to find nouns related to the semantic tag being searched, thus we consider it to be helpful in the task of classification of nouns, reducing the amount of human work, because instead of having to search in a whole corpus for relevant words, the algorithm is able to provide a shorter list of relevant words extracted from the corpus and only that list needs validation by the user.

To reduce the amount of erroneous classifications, the idea of filtering the matching of nouns in each iteration of the algorithm was discussed. This would avoid the continuing of the execution of the algorithm using erroneous seed sentences, because a user would be filtering the nouns as soon as they were returned by the algorithm, thus not propagating the error further. However, it would imply the dynamic participation of the user in this task, which would reveal to be very extensive on time, and imply a lot of intervention and availability by the user. Thus, we decided to make the algorithm self-sufficient and let the filtering of the nouns occur one single time at the end of the algorithm execution, with the consequence of having a larger amount of nouns to filter at that time, due to the error propagation during the non-supervised execution.

Another improvement that would control the garbage produced by the matching of erroneous words, during the algorithm's comparison stage, and the subsequent propagation of the garbage into further matches with other sentences, would result from the inclusion of the filters presented in *Table 10* in the matching procedure, right before a noun is declared validated and added to the seeds' structure, by comparing the identified noun with the rules of exclusion, and act accordingly: after concluding that a determined corpus sentence matches a seed sentence, and identifying the new noun obtained from this matching, the algorithm could enter the validation stage where the new noun is matched against the regular expressions of the filters described in *Table 10*. If the word matched any of the filters, it would not be added to the seed lists, being discarded, since it was considered to constitute garbage. Otherwise, the new noun would be added to the seeds' list structure, and used as seed on further comparisons. This would eliminate wrongly classified words right at their identification point, avoiding propagation of the error and reducing the amount of garbage present in the results' files.

6 Conclusions and Future Work

6.1 Conclusions

This work led to an interesting, however disappointing conclusion about natural language processing tasks: **the corpus determines the final results**.

This influence of the corpus is visible in the fact that the representativeness of sample nouns belonging to a semantic category determines the number of initial seeds and respective contexts available to be used in the comparison process, for matching purposes: the more seeds available, the more number of sentences containing theses nouns gathered, thus, higher probability of matching these sentences with new sentences that contain new nouns.

The representativeness of seed words is also intrinsically related to the source of the corpus, which revealed to be another characteristic that impacts on the quality of the results. The fact that we used a corpus of journalistic source, determined the quantity of seed samples for some semantic categories and influenced the quality of the final results, especially for poorly represented categories.

This corpus influence' was also reflected in the success percentages obtained: in the test-cases where more initial seeds were available, the comparison process had more examples to be matched against, and provided results with better quality, finding more new classified words to enrich our set of Portuguese classified nouns. It is worth mention that in this cases, where more initial seeds were available, on what concerns the non-matching words found as results, despite not matching the specific semantic category we were searching, some of them were somehow related to the topic in question, what demonstrates that the algorithm was able to establish a pattern that relates contexts and a determined topic's content, that englobes not only the nouns belonging to the target semantic category but also related words. However, the validated nouns are limited by the scope of the semantic category chosen that determines the criteria to apply during human validation, since one semantic category does not cover all the words related to that topic, but only a subgroup of them.

A curious aspect that arises from the analysis of the resulting lists of nouns is that the judgment about the correctness of classified nouns, according to the semantic category being expanded, revealed to be subject of ambiguity of evaluation, because the interpretation a judge makes about a noun is dependent on its living experience and 'emotion' or emotional attachment a word transmits to him, because each individual attributes its own emotional value to words that are associated with events of his life, thus some nouns can represent good or bad memories to us, depending on our living experience, being this personal interpretation an influence on our judgment. This is particularly interesting on what concerns the current topic of analysis in NLP: the analysis of sentiments contained and transmitted in a text. Thus, what one judge can consider to be correctly classified considering a determined semantic category, other judge can decide to consider as an erroneous classification, and exclude the noun.

The co-training revealed to be a really heavy algorithm to apply on corpus of such dimensions, because it implies very heavy comparisons on each iteration, forcing the corpus to be read and filtered many times, what degrades the runtime and load balance of the system. This leads to the most difficult task in any heavy-input parsing application: achieving a reasonable balance between resources management and quality of the results, because one cannot deliberately sacrifice accuracy of results in order to achieve shortest runtimes: it would deprecate the quality of the work. A delicate balance is needed.

We confirmed that our partition of the corpus, determined by the split factor applied, influences the results. Although it did not reveal any differences for some of the semantic tags, it seems to greatly influence the test-cases where less initial seeds exist, probably because the lack of initial trustable seeds and a larger block of corpus to analyze leads to more erroneous matches executed inside each corpus' block, resulting in much more erroneous classifications obtained at the end of the algorithm. It is worth notice that this rare situation, where the different partitioning of the corpus did not affect the results obtained (not forgetting we used relatively close values) occurred for the test cases that had the more probability to be found in the corpus, and it is reflected on its largest set of initial seeds. Thus, this is related to the fact that by having a good representativeness of the semantic tag in the corpus, the algorithm benefits with more examples for the comparison process, thus the probability of getting matches is also higher than in the low representativeness case, especially for the first iterations. The low representativeness cases are more affected by the corpus content because in this case the algorithm will depend on the matches of contexts, during the later iterations, starting from a small set of seeds whose initial contexts are also rare. The data available during algorithm's execution increases with the more number of iterations executed. However, this comes at the cost of more garbage being collected. This will result in final results with less quality.

We used a small difference between both split factors and it revealed a significant influence in this poorly represented semantic categories, thus if we were using more distant values, the consequences and differences in the results could be dramatic.

The way the corpus partition is implemented also influences the algorithm's performance, because each block of the corpus is being processed individually, the results of one block are not being propagated to the following blocks, which limits the matching of contexts, both correctly and wrongly classified. The choice of running the blocks of corpus individually is related to the major problem found during the development of this work: resources management. The propagation of results between different blocks would imply that each sentence would be compared with the whole corpus in order to find new contexts, and so on. Since this comparison would involve the whole corpus, and would be executed so much times, it would result in an exponential augment of the runtimes of the algorithm.

Performance has also revealed to be influenced by the split factor chosen, because tests have revealed a difference in the runtimes with the same configuration parameters, only differing in the split factor. This is particularly visible in the `SEM-ACT-CRIME` test-case, where the results obtained were exactly the same, however for the largest split factor (implying larger blocks of corpus to process at each time) the runtime has more than doubled its execution. This results from the fact that by having larger blocks of corpus to process at each time, the comparison processes take longer, however since the semantic category had good representativeness in the corpus, the algorithm achieved the same results, only loosing on performance, due to the longer process of comparisons.

The co-training algorithm has proven to suit this kind of task, because we had a smaller set of classified nouns for different semantic tags, and a much wider set of non-classified nouns, contained in a very large corpus and, despite being influenced by the source of the corpus, it lead to a small percentage of successful results. However, this implementation's usefulness is not in doubt, since the human's required amount of work to search for nouns belonging to a specific semantic category in a whole corpus is much more extensive than using this application to produce lists of nouns, much smaller than the corpus, and validate solely the nouns contained in those lists. The human's cost of this task is clearly reduced.

However, this implementation of the co-training classification algorithm presents faults on what concerns accuracy of results, because it is based solely on comparison and matching of elements constituting dependencies. Although we imposed some matching constraints, they revealed to be insufficient to grant correct and relevant matches that actually result in correct classifications. This is visible in the large amount of not classified nouns provided as results of our algorithm, which after validation presented very few relevant identifications of correct classifications.

The analysis of the resulting lists of nouns to validate led to the conclusion that some filters could be applied to improve the quality of these results. These filters revealed significant improvements, thus the decision to implement them as a permanent component of the latest validation applied to the lists of resulting nouns, right before presentation to the user. However, in some specific situations, where it is expected that nouns matching some of the filtering conditions might arise as correctly classified samples, this filters can lead to inferior success rates, because although they reduce the amount of nouns present in the list of nouns to validate, facilitating the human's validation task, they will also exclude nouns that otherwise would count as correctly classified nouns. Despite this disadvantage, that only occurs on some specific and rare situations, the advantages arising from the usage of filters surpass this low-probability of occurrence disadvantage.

6.2 FUTURE WORK

As any work developed, this implementation has many room for improvements.

As future work, it would be interesting to test this application using different available corpus at the L^{2}F at INESC-ID, from different nature/source, and collecting evidences that formally confirm that the kind of journalistic provenience really affects the results obtained, since journalistic text proved to be very strict on topics addressed, influencing the semantic categories of the nouns found on the corpus.

Another topic for further investigation would be finding the better split factor to apply to the corpus, by actually quantifying the results obtained and its accuracy, studying the fluctuation of the results according to the increasing or decreasing the split factor, and documenting the degree of proportional or inversely proportional relation between these factors.

For the sake of accuracy improvement, it would also be very useful to find a conjugation of another semantic analysis task with this co-training implementation, enriching the features being used to decide on the matching of seed sentences with corpus sentences. We suggest an approach that makes a deeper analysis of the lexical elements and its characteristics, extracted from an analysis of contexts, identifying patterns that relate contexts and semantic categories, for example, the conjugation of co-training with a technique that makes a deeper analysis over the relation between contexts themselves, but also how nouns belonging to a specific semantic-category are particularly connected to some contexts, by identifying, for example, verb' lemmas or adjective words, which are related with a particular semantic category, similar to what (Bel et al., 2012) did, and then including this data in the comparison process of the co-training, in order to make the matching process more strict and accurate, reducing the number of false positives acquired in the task.

Considering the positive impact of the filters applied at the later stage of the algorithm, as future work we suggest implementing these filters on the comparison process. This would control the garbage produced by the matching of erroneous words, during the algorithm's comparison stages, and the subsequent propagation of this errors into further matches with other sentences, decreasing the probability of obtaining wrongly classified contexts, and consequently wrongly classified nouns, which otherwise would continue to be matched with other sentences and propagate their error, generating more garbage. This would improve both performance of the algorithm, by decreasing the amount of comparisons executed (because some contexts would be discarded), but also the quality of the results' files provided to the user for manual validation, significantly reducing the time it takes to judge about the correctness of the classifications.

It would also be interesting to have the same classification task implemented with different learning techniques, and compare the resources consumption of those implementations versus the long runtimes of this Co-Training algorithm implementation. Also, the quality, in terms of percentage of success, of the results obtained with different learning algorithms would be relevant to figure out what is the best algorithm to use for a semantic classification task, concerning a corpus with similar dimension and content-type to the CETEMPúblico used in this project.

7 REFERENCES

Ait-Mokhtar, Salah, Chanod, Jean-Pierre and Roux, Claude. Robustness beyond shallowness: incremental deep parsing. *Natural Language Engineering*, 8(2-3):121-144, 2002.

Bick, Eckhard. Noun sense tagging: Semantic prototype annotation of a Portuguese treebank. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories*, pages 127-138, Prague, Czech Republic, 2006.

Bird, Steven, Klein, Ewan and Loper, Edward. *Natural Language Processing with Python: Analyzing Text with the Natural Language Toolkit*. O'Reilly, Beijing, 2009.

Blum, Avrim and Mitchell, Tom. Combining labeled and unlabeled data with cotraining. In *Proceedings of the Eleventh Annual Conference on Computational Learning Theory*, COLT' 98, pages 92-100, Madison, Wisconsin, USA, 1998. ACM.

Collins, Michael and Singer, Yoram. Unsupervised models for named entity classification. In *Proceedings of the 1999 Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora*, pages 100-110, 1999.

Diniz, Cláudio, Mamede, Nuno and Pereira, João. Rudrico2 - A faster disambiguator and segmentation modifier. *II Simpósio de Informática (INForum)*, pages 573-584, September 2010.

Duda, R. and Hart P.. *Pattern Classification and Scene Analysis*. John Willey & Sons, New York, 1973.

Emms, Martin and Luz, Saturnino. Machine learning for natural language processing. In *European Summer School of Logic, Language and Information (ESSLLI 2007)*, Dublin, Ireland, January 2011. course reader.

Giménez, Jesús and Márquez, Lluís. SVMTool: a general POS tagger generator based on support vector machines. In *Proceedings of the 4th International Conference on Language Resources and Evaluation*, pages 43-46, 2004.

Oliveira, Hugo, Santos, Diana, Gomes Paulo and Seco, Nuno. PAPEL: a dictionary-based lexical ontology for Portuguese. In António Teixeira, Vera Lúcia Strube de Lima, Luís Caldas de Oliveira, and Paulo Quaresma, editors, *Proceedings of the 8th International Conference on Computational Processing of Portuguese (PROPOR 2008)*, volume 5190, pages 31-40, Aveiro, Portugal, September 2008. Springer Verlag.

Jantan, Hamidah, Hamdan, Abdul and Othman, Zulaiha. Classification techniques for talent forecasting in human resource management. In *Proceedings of the 5th International Conference on Advanced Data Mining and Applications*, ADMA '09, pages 496-503, Beijing, China, 2009. Springer-Verlag.

Joachims, T.. *Learning to Classify Text Using Support Vector Machines*. The Springer International Series in Engineering and Computer Science. Springer US, 2002.

Kotsiantis, S.. Supervised Machine Learning: a review of classification techniques. Technical report, University of Peloponnese, Greece, 2007.

Kulkami, Anagha, Heilman, Michael, Eskenazi, Maxine and Callan Jamie. Word sense disambiguation for vocabulary learning. In Beverly Park Woolf, Esma Aïmeur, Roger Nkambou, and Susanne P. Lajoie, editors, *Proceedings of the 9th International Conference on Intelligent Tutoring Systems*, volume 5091 *of Lecture Notes in Computer Science*, pages 500-509, Montreal, Canada, 2008. Springer.

Lee, Yoong and Ng, Hwee. An empirical evaluation of knowledge sources and learning algorithms for word sense disambiguation. In *Proceedings of the ACL- 02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 41-48, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics.

Mamede, Nuno, Baptista, Jorge, Diniz, Cláudio and Cabarrão, Vera. STRING: An Hybrid Statistical and Rule-based Natural Language Processing Chain for Portuguese. *International Conference on Computational Processing of Portuguese (PROPOR 2012)*, Demo Session, April 2012.

Maurício, Andreia. Identificação, classificação e normalização de expressões temporais. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, November 2011.

McCallum, Andrew. MALLET: a machine learning for language toolkit. http://www.cs.umass.edu/ mccallum/mallet, 2002.

Mota, Cristina. *How to keep up with language dynamics: A case study on Named Entity Recognition*. PhD thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, May 2009.

Márquez, Lluís, Padró, Lluís and Rodríguez, Horacio. A machine learning approach to POS tagging. *Machine Learning*, 39(1):59-91, 2000.

Nigam, Kamal. Using maximum entropy for text classification. In *In IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61-67, 1999.

Nobre, Nuno. Resolução de expressões anafóricas. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, June 2011.

Nulty, Paul. Semantic Classification of Noun Phrases Using Web Counts and Learning Algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 79-84, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

Oliveira, Hugo and Gomes, Paulo. Onto.PT: Automatic construction of a lexical ontology for Portuguese, *FCTUC Eng.Informática - Artigos em Livros de Actas*, August 2010.

Orphanos, G., Kalles, D., Papagelis, A. and Christodoulakis, D.. Decision trees and NLP: a case study in POS tagging. In *Proceedings of ACAI 1999*, page n/d, 26500 Rion, Patras, Greece, 1999. Source: http://www.u.arizona.edu/˜echan3/539/decison_tree99.pdf (Date: 06-01-2014).

Platt, John. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical report, Advances in Kernel Methods - Support Vector Learning, 1998.

Pustejovsky, J.*. The Generative Lexicon*. Bradford Books. MIT Press, 1998.

Quinlan, J.. Induction of decision trees. *Machine Learning*, pages 81-106, 1986.

Quinlan, J.. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers, 1993.

Ratnaparkhi, Adwait. A maximum entropy model for part-of-speech tagging. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 133-142, April 16 1996.

Ribeiro, Ricardo. Anotação morfossintáctica desambiguada do português. Master's thesis, Instituto Superior Técnico, March 2003.

Rocha, Paulo and Santos, Diana. Cetempúblico: Um corpus de grandes dimensões de linguagem jornalística portuguesa. In Maria das Graças Volpe Nunes, editor, *Actas do V Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR 2000)*, pages 131-140, São Paulo, 2000.

Sebastiani, Fabrizio. A Tutorial on Automated Text Categorization. In *Proceedings of ASAI-99, 1st Argentinian Symposium on Artificial Intelligence*, pages 7-35, Buenos Aires, AR, 1999.

Socher, Richard, Perelygin, Alex, Wu, Jean, Chuang, Jason, Manning, Christopher, Ng, Andrew and Potts, Christopher. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1631-1642, Seattle, WA, October 2013. Association for Computational Linguistics.

Specia, Lucia. *Uma Abordagem Híbrida Relacional para a Desambiguação Lexical de Sentido na Tradução Automática*. PhD thesis, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, 2007. Tese de Doutorado em Ciências de Computação e Matemática Computacional.

Stella, Nwigbo and Chuks, Agbo. Expert system: a catalyst in educational development in Nigeria. In *Proceedings of the 1st International Technology, Education and Environment Conference (TEEC 2011)*, pages 566-571, Omoku, Rivers State, Nigeria, 2011. African Society for Scientific Research (ASSR).

Toker, G. and Kirmemis, O.. Text categorization using k-nearest neighbor classification. Survey paper, Middle East Technical University, n/d. Computer Engineering Department.

Vicente, Alexandre. Lexman: um segmentador e analisador morfológico com transdutores. Master's thesis, Instituto Superior Técnico, Universidade Técnica de Lisboa, June 2013.

Witten, Ian, Frank, Eibe and Hall, Mark. *Data Mining: Practical Machine Learning Tools and Techniques*. Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann, 3rd edition, 2011.

Yarowsky, David. One sense per collocation. In *Proceedings of the Workshop on Human Language Technology*, HLT '93, pages 266-271, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics.

Zampieri, Marcos. A supervised machine learning method for word sense disambiguation of Portuguese nouns*. Bulletin de Linguistique Appliquée et Générale*, 34:187-203, June 2010.

Zhang, G.. Neural networks for classification: a survey. *IEEE Transactions on Systems, Man, and Cybernetics*, Part C, 30(4):451-462, 2000.

Zhang, Harry. The optimality of naive bayes. In Valerie Barr and Zdravko Markov, editors, *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference (FLAIRS 2004)*, page 562. AAAI Press, 2004.

Strickland, Jeffrey. *Predictive Modeling and Analytics*, pages 54-56. LULU Press, 2014.

Mamede, Nuno and Baptista, Jorge. - Nomenclature of Chunks and Dependencies in Portuguese XIP Grammar, 2014.

Marques, João, Mamede, Nuno and Baptista, Jorge. Anaphora Resolution in Portuguese: an hybrid approach, MSc Thesis on Instituto Superior Técnico, Universidade de Lisboa, Nov. 2013.

Cabrita, Viviana, Mamede, Nuno and Baptista, Jorge. Identificar, Ordenar e Relacionar Eventos, MSc Thesis on Instituto Superior Técnico, Universidade de Lisboa, Oct. 2014.

Baptista, Jorge. ViPEr: A Lexicon-Grammar of European Portuguese Verbs, presented at Proceedings of the 31st International Conference on Lexis and Grammar, Sep. 2012.

Bel, Núria, Romeo, Lauren and Padró, Muntsa. Automatic lexical semantic classification of nouns. In *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istambul, Turkey; pages 1448-1455, 2012.