UNIVERSITY OF ALGARVE
FACULTY OF SOCIAL AND HUMAN SCIENCE

THE UNIVERSITY OF WOLVERHAMPTON
SCHOOL OF LAW, SOCIAL SCIENCES AND COMMUNICATIONS

Munshi Asadullah

A Framework for Structural Ambiguity Resolution for Portuguese.

A Project submitted as part of a programme of study for the award of
MA Natural Language Processing & Human Language Technology

Supervisors

Prof. Doutor Nuno J. Mamede
Instituto Superior Técnico (IST) / INESC-ID Lisboa

Prof. Doutor Jorge Baptista
University of Algarve / INESC-ID Lisboa

Dr. Constantin Orasan
University of Wolverhampton

May 2012

UNIVERSITY OF WOLVERHAMPTON

SCHOOL OF LAW, SOCIAL SCIENCES AND COMMUNICATIONS
MA NATURAL LANGUAGE PROCESSING & HUMAN LANGUAGE TECHNOLOGY

Name: Munshi Asadullah

Date: 15-05-2012

Title: A Framework for Structural Ambiguity Resolution for Portuguese

Module Code: LN4036

Presented in partial fulfilment of the assessment requirements for the above award

Supervisor: Dr. Nuno J. Mamede, Dr, Jorge Baptista and Dr. Constantin Orasan

Declaration:

(i) EITHER:
*This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work, I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

OR:
*The work includes the following material which has been submitted for assessment/credit previously: *student to list material.* My Project Supervisor has expressly agreed that it is appropriate to include the material listed below within the work.

(ii) It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free licence to do all or any of those things referred to in section 16(i) of the Copyright Designs and Patents Act 1988 (viz: to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make adaptation of the work.

(iii) EITHER:
*This project did not involve contact with human subjects, and hence did not require approval from the LSSC Ethics Committee.

OR:
*This project involved contact with human subjects. Approval was given by the LSSC Ethics Committee, and I have abided by any conditions that were stipulated.

Signed:_____ Date:_____

* One of each pair of alternatives in (i) and (iii) should be deleted.

Abstract.

Structural ambiguity resolution in natural language text is an active field of research. Ambiguity is where a structural element in a text causes the interpretation of the text in more than one way. Prepositional Phrases (PP) introduce significant amount of structural ambiguity since how a PP modifies another phrase can often be traced beyond the syntactic scope of a sentence. It is rather intuitive that semantic knowledge is important to model the phenomenon properly. Proper semantic interpretation though can be difficult to achieve. Moreover lexical and syntactic information based approaches has been proven to be quite effective. One such approach was dependency based resolution of PP attachment ambiguity. We propose a heuristic based modeling of data from two different parsers namely Constraint Grammar (CG) based parser PALAVRAS and Phrase Structure Grammar (PSG) based Finite-State Parser (FSP) used as the parsing backbone of the STRING Natural Language Processing (NLP) chain for Portuguese. Different models using two parser output will be produced and put together in a linear combination for performance maximization. For the development of the research, a processing framework is also proposed and its development is presented. A dependency annotation tool is also developed within the scope of the research. The models performance was satisfactory if not extraordinary, although the primary objective was to present the modeling possibilities rather than the absolute performance.

Acknowledgement.

I am very thankful to my supervisors; Prof. Doutor Nuno J. Mamede and Prof. Doutor Jorge Baptista for helping me build the research. I would also like to express my gratitude to my supervisor at the University of Wolverhampton, Dr. Constantin Orasan for guiding me towards the completion of the work.

I am grateful to Wilker Aziz for his technical help and support along with his advice during the setup of the experiments. I would like to thank all the annotators who helped me in evaluating the annotation took and producing annotated data. My special gratitude to Justina Valotkaitė, for her help and support with the Portuguese language.

This research was partially supported by the European Commission, Education & Training, Eramus Mundus: EMMC 2008-0083, Erasmus Mundus Masters in NLP & HLT program.

Table of Contents

List of Figures:

List of Tables:

Chapter 1: Introduction.

This research will investigate a way to improve the **P**repositional **P**hrases (PP) attachment disambiguation for Portuguese using the output of two automatic parses. PP attachment is a structural ambiguity problem that makes it difficult for a parser to attach a PP with the phrase it modifies. We are proposing several heuristic based models for the task. Since very little work has been done in this direction, a framework has been proposed to design the models in a semi-automatic way. The parsers selected for the task are STRING (Mamede, 2011) **N**atural **L**anguage **P**rocessing (NLP) chain and PALAVRAS (Bick, 2000). This chapter will provide a general overview of the motivations, aims, challenges and the contributions of the research.

1.1. Motivation.

Ceccato et al. (2004, p. 1) argued that ambiguity, where something can be interpreted in more than one way, is a phenomenon present at all levels of linguistic analysis in natural languages. According to Roth (1998, p. 806) many important natural language problems can be regarded as problems of resolving ambiguity. The ambiguity may be semantic or syntactic or both based on the context. Thus, this research is motivated by the fact that mapping accurate dependencies in a sentence defines the relation of each syntactic element with respect to its surrounding elements and that will ease the task of structural ambiguity resolution and semantic interpretation.

Improving the accuracy of the dependency representation will make the current output of STRING more appropriate and information-rich for further analysis and processing. Moreover, the statistical models will be tried to enrich using the parser output of the Bosque tree-bank. Bosque was parsed using **D**ependency **G**rammar (DG) (Tesnière, 1959) based parser PALAVRAS. According to Covington (2001, p. 97) the parser in the human mind operates in the same way as in a DG. The author (idem: ibidem) also presented some advantages of DG over more traditional representation based on constituency namely that "dependency links are close to the semantic relationships needed for the next stage of interpretation", especially for semantic and pragmatic analysis, which also motivated this research. Best of our knowledge, a framework to incorporate DG based bi-lexical dependency information with **P**hrase **S**tructure

Grammar (PSG) based dependency information, such as the output produced by STRING, has never been attempted, thus, another important motivation for the research.

Furthermore, among different ambiguities, structural ambiguity, more specifically attachment ambiguity resolution in parsed texts, is one of the goals of this research. Structural ambiguity, occurs when a sentence has several alternatives dependency relationships between words or phrases. Nagao (1990, p. 280) stated that these kinds of ambiguities are difficult to resolve using syntactic knowledge alone. Semantic processing is often also necessary. He also argued that resolution of structural ambiguities is a problem of selecting the most acceptable dependency from several possibilities. It is often performed by using knowledge on dependencies between words (idem, ibidem).

Resolving attachment ambiguity, such as the **P**repositional **P**hrase (PP) attachment problem, can thus be performed more efficiently with the proper dependency annotation of a sentence. The correct attachment of PPs is a central disambiguation problem in parsing natural languages as Merlo & Ferrer (2006, p. 341) presented that "Incorrect attachment of prepositional phrases often constitutes the largest single source of errors in current parsing systems. Correct attachment of PPs is necessary to construct a parse tree which will support the proper interpretation of constituents in the sentence."

According to the authors (idem, ibidem), "recent approaches have formalized the problem of disambiguating PP attachments as a binary choice, distinguishing between attachment of a PP to a given verb or to another constituent". This is, though, a simplification of the problem, which does not take the type of the attachment into account. The authors (idem, ibidem) eventually proposed an extension of the problem of PP attachment as a "four-way disambiguation problem", arguing that what is needed in "interpreting prepositional phrases is knowledge about both the structural attachment, the traditional noun and verb attachment distinction, and the nature of the attachment, i.e. the distinction between arguments from adjuncts". The later distinction is a traditional linguistic problem for which there is no trivial solution (see Goldberg, 1995).

Foth & Menzel (2006) argued that the PP attachment can be well treated with the help of **M**achine **L**earning (ML) approach. ML techniques are used to resolve irregular and complex problems for several significant reasons that have been listed by Nilsson (1998, p. 2). Some tasks cannot be defined properly except for by example; i.e. one

might be able to specify input and output pairs but not a precise relationship map between an input and the desired output. Moreover, a system should be able to adjust their internal structure to produce correct outputs for a large number of sample inputs. Thus, it is possible that the performance of a system can be improved with large amount of better quality data. The other important aspect of ML based systems is that, it provides a way to extract the important relationships required to solve the problems from huge amount data.

This study will be dealing with the output of two rather different systems and model the PP attachment phenomenon using both the systems. The systems are rule-based system and thus relationship present in the data is limited by the features used to devise the rules. There is a possibility of the existence of more useful relations that might be discovered by using the distribution of certain features over the training data set. These relations may have been left out because of both the exhaustive development process of any rule-based system and the increasing complexity of predicting the hierarchy of rule triggering. Therefore, a ML system may be used to extract these relations and thus use them to model a phenomenon more accurately.

Furthermore, the Bosque corpus here used is quite large and the number of features available in the linguistic, information-rich, STRING output may be able to establish a map between the input and the output. Moreover, the size of the data and the number of features are overwhelming for human encoding. Besides, future editions of the corpus may be more accurate and STRING features are regularly updated. So, intuitively, a ML approach seems reasonable for this task.

In the realm of ML, the attachment disambiguation task can be categorized as a classification problem. It is considered to be a standard problem handled by **A**rtificial **I**ntelligence (AI) methods. According to Walt & Barnard (2006, p. 170), "the performance of classifiers depends on both the quality and the quantity of the training data, and the richness of the feature set selected for the classification". This research will be conducted with the pre-compiled corpus Bosque and thus the quality and quantity of the training data is constrained by the size and the quality of the corpus.

An information-rich feature set will be able to extract significant amount of linguistic knowledge embedded in the data to make the accurate modeling that is necessary to improve the dependency output. The data extraction though can be a difficult task

considering the irregularities which may be inherent from the underlying systems or noisy data. A robust framework is necessary to reduce the time required for developing application and research to study dependency relations. This is an important motivator for this research.

1.2. Objectives.

This dissertation will describe the methods that have been investigated in order to design a framework to analyze, annotate and develop dependency relation based research such as PP attachment disambiguation using the parsers under investigation. The framework is a part of the natural development of this research to build a statistical model to extract PP modifier dependency using the features that can be extracted from the data.

Lin (1997, p. 65) defined that traditionally, lexical dependencies (Hudson, 1984; Mel'čuk, 1987) are asymmetric, binary, primarily syntactic relations between a word called head (or governor, or parent), and another word called modifier (or dependent, or daughter). However the dependency generated by STRING defines the relationship between the heads of the chunks rather than the relationships between word units. The models will be generated using the output data of the systems, regardless of the rule set or underlying implementation of the parsing systems that produced the data. Thus the model will be adopted to extract relationships between the chunks. Thus, one of the primary objectives is to check for possible improvement in the current dependency output of the STRING.

The input data will be extracted from the parsed output of the parsers under investigation. The objective is to design a model that will select the candidate of a modifier and governor pair and select the most probable pair. This dissertation will also present the experiments conducted to evaluate the performance of the model in disambiguating the PP attachments. Although the experimental setup, tools and the data are developed for Portuguese, an argument in favor of the possible multi-lingual nature of the framework will also be presented.

The methods used for the statistical modeling, extracts pre-defined feature distributions from the parsed output of the STRING and the **C**onstraint **G**rammar (CG) (Karlsson, 1990) format output of PALAVRAS. The model thus is biased by the underlying rule

based system and the model's performance is expected be closer to the training dataset. The absence of large amount of gold standard training data will be attempted to be compensated by biasing the model's parameters using the Portuguese tree-bank Bosque[1]. The CG formalism with bi-lexical dependency is used to represent one of the many formats of the Bosque output. The separate model was developed using the Bosque and will be used in combination with the other models.

Bosque is the part of the larger tree-bank, Floresta Sintá(c)tica tree-bank project (Afonso et al., 2002). However Bosque has been checked by humans for the anomaly in the parsed output. It will be used to extract the biasing factors, such as, the existence of a traversal path between the modifiers and governor components and the path distance if such exists. Several models will be produced and the performance will be evaluated using human annotated test dataset. The human annotation will be produced using the annotation tool developed as a part of the proposed framework.

1.3. Methodology.

The methodology for the study is designed, concerning the primary goal of the research, which is extracting and using available linguistic knowledge in the data to resolve dependency mapping using ML methods. The linguistic knowledge is encoded in STRING and Bosque in the form of rules and features, and the data convey them. Thus pre-processing of the data is a vital part of this research. The extracted data can then be used to design the model and experimented with different features to incorporate in the model to improve its performance.

1.3.1. Data Pre-processing.

The model will be designed using the output of two different parses. Thus the training dataset needs to be parsed by both parsers. The amount of human modified data though, has been limited by the size of the dataset of Bosque. So, the raw text from the Bosque will be extracted and then parsed using the STRING NLP Chain. The parsed tokens are produces differently by each parser thus, token level alignment will be required for the proper data analysis and modeling using both parses. So, the data processing will start with processing the Bosque. We shall extract the raw text form the data and at the same time the parsed output will be transferred into a predefined data-structure.

[1] Distribution at http://www.linguateca.pt/floresta/

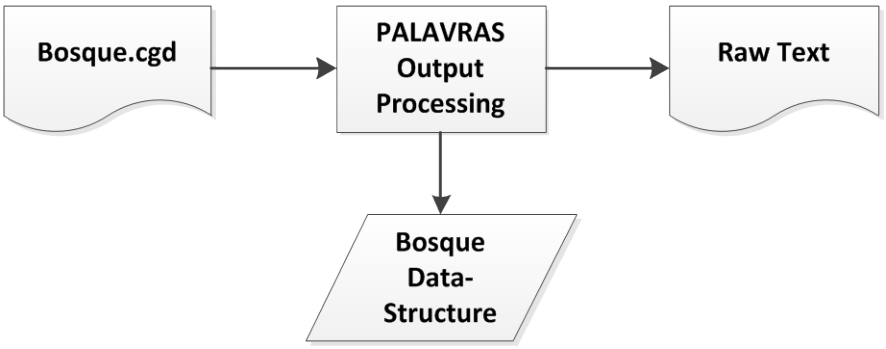

Figure 1 - 1: Primary data processing of the Bosque.

The raw text then will be parsed with the STRING NLP Chain and the output in XIP e**X**tensible **M**arkup **L**anguage (XML) format will be used for this study. The output data will then be processed into the detailed data-structure for future processing. From the data-structures of both Bosque and XIP sentence level alignment data will be produced.

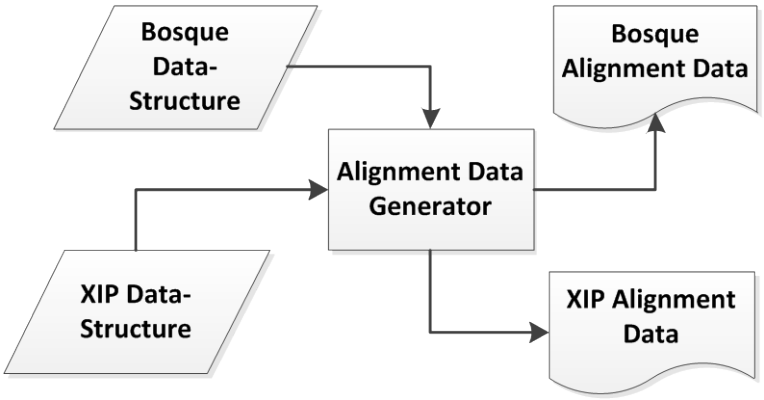

Figure 1 - 2: Alignment data generation.

Raw alignment ready text from the two systems will then be aligned using automatic aligner GIZA++ (Och, 2000). The automatic aligner align tokens by their relative index in the sentence, thus a transformation module will be needed to extract proper token level alignment. Once proper alignment is achieved the data is ready to be used to extract the modeling parameter distributions. The framework that we are developing within the scope of this research allows all such preprocessing in a structured manner and with any number of dataset.

1.3.2. Experimental setup.

The primary experiments conducted in producing the model is the parameter evaluation. First of all the dataset has been split into 90% training set and 10% test set. The

distribution of each of the feature selected will be extracted from the training dataset. The test set has been evaluated by human annotator for the correctness of the data. This test dataset will be the reference dataset, thus all evaluations will be performed using this data. Both training and test dataset will be made up of only the modifier elements that are PPs to model the attachment behavior. This dataset will be used to evaluate the proper feature set for the model.

1.3.3. Implementations.

As a part of this research, we implemented a simple annotation tool to annotate head dependency. This tool will be used to produce the reference dataset. The tool will have an innovative interface to allow the annotator perform the task without too much training and instructions. The design and implementation issues regarding the tool will be discussed in the ***Chapter 5***.

For the research the tool is needed to generate the reference test data, annotated by human annotator. The framework's core is the reversed engineered data-structure that provides a simple interface to produce experimental data needed to model and later use that model to perform specific tasks. The data-structure and the manipulation system built on top of it provide a fluid interaction between the objects defined to represent the data. In ***Chapter 5*** the framework will be used to model and then evaluate a statistical model for PP attachment disambiguation.

1.4. Structure of the dissertation.

The dissertation will be divided into six chapters. In the first chapter, a general presentation of the problem domain will be made. In this chapter we have also made the general introduction to the tools and resources to be used for the research. We have also presented the general outline of the research methodology to be used and the organization of the dissertation.

The second chapter will introduce the problem domain in finer details and report the findings of recent researches on the subject. We will be trying to model structural ambiguity, in particular attachment ambiguity, and resolution methods will thus be introduced along with the recent development in the field. We will be using ML methods and thus different methods and algorithms will be briefly introduced. The research trends and their performance will also be reported in this chapter.

The third chapter is dedicated to the data and its representation. The systems under study will be introduced and a reasonable analysis of their underlying theoretical background will be presented. The data pre-processing methods and the design decisions will also be discussed. One of the major issues that will be presented in this chapter is the data structures.

The forth chapter will discuss the data preprocessing steps, primarily the parsing the **X**erox **I**ncremental **P**arser - e**X**tensible **M**arkup **L**anguage (XIP XML) (XRCE, 2011) data into the data-structure. The other important issue is the token level alignment to allow the framework to use data from two different systems. The evaluation of the automatic alignment process will be presented and the results will be discussed. The alignment process will require some pre-processing and this chapter will present the tools required for the task.

The fifth chapter will discuss the experimental setup and the underlying algorithms. The design and development of the annotation tool **Dep**endency **An**notator (DpAn) will also be presented in this chapter along with the qualitative evaluation of the tool. The statistical model for PP attachment disambiguation will be introduced and performance with the reference data will be discussed. It will also present the automatic evaluation process and report the findings of the research.

The final chapter will present a general discussion on the achieved results. Along with the future direction of the research this chapter will also include some concluding remarks regarding the possibility of multi-lingual aspect of the work.

Chapter 2: Related work.

This chapter discusses the theoretical background of the problem domain and the studies conducted to explore the scope of structural ambiguity. It will also explain the relevant research regarding solving the problem, especially **P**repositional **P**hrase (PP) attachment disambiguation. Moreover, **M**achine **L**earning (ML) methods will be explored for some of the experiments, thus a general overview of ML research and its involvement in resolving PP attachment will also be provided. In the later part of this chapter, relevant work done for Portuguese language will be presented.

2.1. Structural ambiguity.

The presence of more than one interpretation or meaning of a single sentence is traditionally designated as ambiguity. Although the conflict of meaning interpretation puts ambiguity to be a problem in semantic or other higher level of analysis (pragmatic, discourse etc.), the source of the ambiguity can be defined in all the possible linguistic levels (morphological, lexical, syntactic, pragmatic, discourse etc.). Structural ambiguity is best defined in Hindle & Rooth (1993) as the ambiguity caused by the possibility of multiple syntactic representation of a single sentence i.e. having more than one parse-tree.

Let us meet the new European literature teacher. ***(example.2.1)***

Example.2.1 is syntactically ambiguous because of the two possible syntactic representation of the sentence, i.e. one attaching the adjective European to teacher and the other one to history. The sentence is though is ambiguous because of the two distinct meaning that can be extracted, i.e. new literature teacher is from Europe or the new teacher will teach European literature.

A sentence having multiple parse-trees does not automatically corresponds to ambiguity, rather each tree representing unique meaning for the sentence makes it structurally ambiguous. Hindle & Rooth (1993) argued that, Prepositional Phrase (PP) attachment is the most recognized instance of structural ambiguity. Therefore, this research will try to formulate a Machine Learning (ML) method to reduce PP attachment ambiguity in Portuguese using parsed output of the shallow parser and rule-

based automatic linguistic analysis tool, STRING **N**atural **L**anguage **P**rocessing (NLP) chain. The following sections will give a brief overview of the specific problem domain followed by different approaches attempted to resolve this ambiguity so far.

2.1.1. Prepositional Phrase (PP) Attachment.

Attachment ambiguity occurs when a particular syntactic constituent of a sentence, such as a prepositional phrase or a relative clause, can be correctly attached to two other syntactic constituent of that sentence. The most well-known pattern of attachment ambiguity is a **P**repositional **P**hrase (PP) that may modify either a **V**erb **P**hrase (VP) or a **N**oun **P**hrase (NP). The general form of the ambiguity can be explained by the following example (example.2.2).

The girl hit the boy with a book. ***(example.2.2)***

The PP with a book can be attached with either the VP ***hit*** or the direct object of the VP ***the boy***. The ambiguity is structural and (Figure 2.1) shows the possible parses.

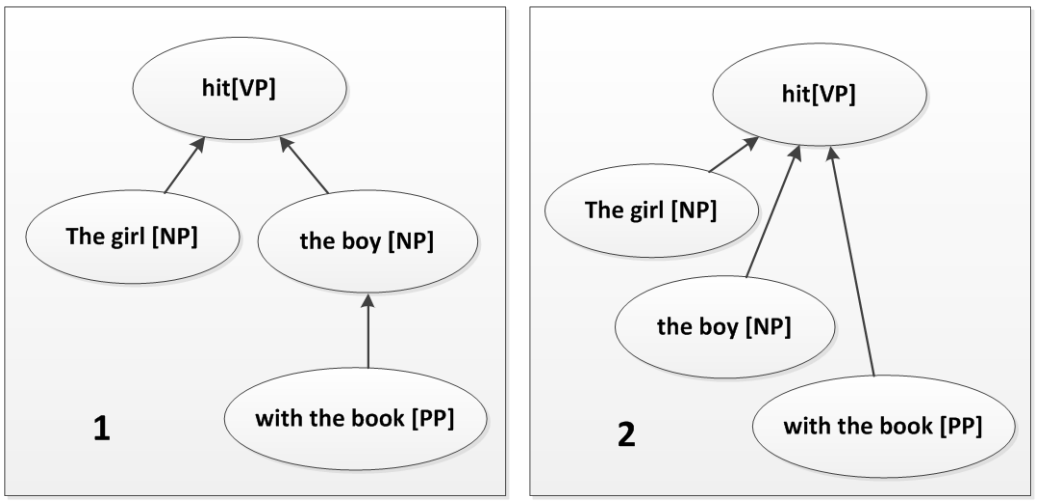

Figure 2 - 1: Possible Parse-Trees for Example.2.2.

The first parse attaches the PP with the **N**oun **P**hrase (NP) thus, representing the meaning where, the girl hits a boy who had a possession relation with the book; whereas the second parse attaches the PP with the **V**erb **P**hrase (VP) and that expresses the meaning that, the book is an instrument of aggression (i.e. there is no possession relation). These structures though represent two different semantic representations for the sentence and thus can be considered ambiguous. However, this is an over simplified representation of the PP attachment ambiguity.

Nadh & Huyck (2009) argued that the presence of the frame [VP NP PP] introduce such ambiguity in English and they also presented the experiments by Ford et al. (1982) with human subjects using the example.2.3, where 35% of subjects attached the PP to the VP and 65% to the NP.

She discussed her daughter's difficulties with the teachers. ***(example.2.3)***

The problem thus appears to be more complex since even native speakers have disagreement over the attachment issue. Hence, a general overview of the problem domain is presented in the following sub-section.

2.1.2. Syntactic Environment of the PP Attachment Ambiguity.

Prepositions are often characterized as syntactic connecting words. However, they have both syntactic and semantic disclaimers that are distinctive to them. According to Merlo & Ferrer (2006, p. 341) PP attachment has been presented by many researchers as a two way i.e. binary syntactic disambiguation problem considering that the PP can attach to the verb or the direct object of the verb.

The idea was originally introduced by Hindle & Rooth (1993) as a means of disambiguate PP attachment using ML methods. Lexical association of the preposition in the PP with the verb or the verb's direct object was used to formulate the problem. A later study by Brill & Resnik (1994) extended the feature set to four elements, which contains the noun inside the PP as well.

The authors (Merlo & Ferrer, 2006, p. 341) however considered the problem as a four way disambiguation problem, considering the type of the PP in the disambiguation process, namely, PP argument and PP adjunct. For example,

The girl put the book on the table in the morning. ***(example.2.4)***

The sentence in (example.2.4) contains two PPs and both are attached to the verb put, but the type of relation is quite different. The first PP, on the table is a locative PP and it has an argument or necessary relation with the verb; whereas the second PP in the morning is an optional descriptor of time and has an adjunct relation with the verb. The authors (Merlo & Ferrer, 2006) also presented the difficulties in such distinction and introduced a method using corpus-based statistical correlates for the diagnostics used in linguistics to decide whether a PP is an argument or an adjunct. They found that the

most significant feature to be used for their proposed method was lexical classes (noun, verb etc.).

I saw the girl with the basketball ***(example.2.5)***

Niemann (1998) studied the PP attachment in relation to the semantic role that it plays in the sentence. The authors provided (example.2.5) as a general presentation of the study, explaining that, intuitively the PP with the basketball cannot play the semantic role of instrument for the verb saw. A human speaker would rather expect it to take the role of possession, thus attaching the PP with the NP the girl. He (Niemann, 1998) also refers to the seminal work of Taraban & McClelland (1988), which reported the regular anticipation of such roles by human speakers.

Fellbaum & Miller (1990) demonstrate that the semantic associations given as hypernym and troponym trees in WordNet (Miller et al., 1990) can be used to categorize lexical items when parsing PP attachment. They used partially parsed corpus and reported that clear preference patterns were presence and that assist the disambiguation with "relative success". They reported 80% accuracy for PP attachment disambiguation and also mentioned that, the presence of word sense ambiguity reduce the accuracy by 8%.

Another work that explores the scope of the PP attachment ambiguity was conducted by Mohanty et al., (2005). They performed a detail study of six English prepositions (for, from, in, on, to, and with) and the thematic role they play depending on the semantics of the preceding and the immediately following lexical heads. They used the British National Corpus (BNC) and reported that these six prepositions account for about 45% of the total 11 million PPs in the corpus. They studied these prepositions within the frame such as [V NP1 P NP2] among the eight frames defined in the study. They found up to eight sentence patterns for some of the prepositions. In conclusion, they argued that a deep linguistic analysis is needed to resolving the ambiguity and that, instead of analyzing millions of sentences, only a set of sentence types containing the relevant patterns, are needed to be tested.

2.1.3. PP Attachment Disambiguation.

Human speakers possess the knowledge about the environment where PPs occur, and thus have the natural ability to resolve the ambiguity, especially during oral

communication. Eysenck & Keane (2000, p. 339) made a general observation about the cognitive progression in a human speaker to resolve ambiguity using prosodic feature as follows,

> *"Spoken speech contains prosodic cues in the form of stress, intonation, and so on. This information can be used by the listener to work out the syntactic or grammatical structure of each sentence. For example, in the ambiguous sentence, 'The old men and women sat on the bench', the women may or may not be old. If the women are not old, then the spoken duration of the word 'men' will be relatively long and the stressed syllable in 'women' will have a steep rise in pitch contour. Neither of these prosodic features will be present if the sentence means that the women are old."*

Ratnaparkhi et al. (1994) obtained PP-attachment resolution performances of three tree-bank experts on a set of three hundred randomly selected test events from the Wall Street Journal (WSJ) corpus. They reported that human experts could reach an accuracy of 93.2%, resolving PP attachments, if cases were given as whole sentences out of context.

Altmann (1985) studied a number of syntactic resolution experiments to determine whether ambiguity resolution by humans is based on syntactic information alone or some other basis, e.g. the presence of contextual information. He found that syntactic ambiguity resolution by humans is largely based on existing knowledge. Even an isolated sentence with no context, that has a PP-attachment ambiguity, could be resolved based on prior knowledge.

The author concluded that computational models of syntactic ambiguity resolution which ignore contextual considerations are not true models of NLP, thus emphasizing the importance of semantic models. The general approach to the problem is to use a preference heuristics and the common preference heuristics used are presented in the next section.

2.1.4. Preference Heuristics for PP Attachment Disambiguation.

Linguistic information-based computational systems often use structure-based preference heuristics to resolve parsing ambiguities. One such approach is **R**ight **A**ssociation (RA), originally presented by Kimball (1973, p. 24), which states that constituents should be attached to the nearest (lowest i.e. right most) non-terminal node because of the tendency of natural language to be organized in a right-branching

structure. For PP attachment, this means that the PP should always be attached to the nearest constituent, i.e. the NP, in a [VP NP PP] frame.

Another approach to resolve the problem is the **M**inimal **A**ttachment (MA), originally proposed by Frazier (1983) as a part of the Garden-Path Theory. This approach essentially suggests that potentially unnecessary nodes will not be proposed and the new items will be attached to the recently processed phrase or clause. So, in a [VP NP PP] frame, the PP will always attach to the VP.

The other approach to the problem is to define **L**exical **P**references (LP) for the noun, the verb, or the preposition. The LP for verbs regarding PPs has been presented by Ford et al. (1982) while Rappaport (1983) studied the LP for nouns. Preposition themselves may have different likelihood to get attached to certain constructions. Preposition acting as functions, for example, in temporal PPs may be associated to pattern in attachment to events that have temporal properties, for details see Wilks et al. (1985). Huyck (2000) also presented this heuristic mentioning that, in the case of the preposition of, this approach is entirely effective; i.e., a PP with the preposition ***of*** always attaches to the NP.

Another heuristic searches for a similar PPs as modifiers with in the discourse and if a match is found the attachment take the appearance of the antecedent. Crain & Steedman (1985) termed the theory to be the principle of **R**eferential **S**uccess (RS). They stated that this principal can be generalized as a kind of presupposition satisfaction method i.e. the reading that satisfies the most presuppositions is the one to be preferred. They (Crain & Steedman, 1985, p. 170) explained the method as follows,

> *"A definite NP presupposes that the object or event it describes exists and that it is available in the knowledge base for unique reference. The attachment of a PP to an NP results in new presuppositions for the NP, but cancels its uniqueness. The attachment of a PP to a VP creates no new presuppositions but rather indicates new information".*

Volk (2001, p. 34) use the method to perform PP attachment disambiguation for German. He explained the method such that, if the attachment to a definite NP mapped to an undefined pattern, the verb attachment will be considered. On the other hand if the NP attachment is mapped to a definite reference, NP attachment will be accepted as proper. Finally, he concluded that in this process the definiteness is a feature to be used while deciding the attachment of the PP. However, he stated that such a detailed

knowledge representation is only possible for strict domain specific implementation with semantic analysis.

Hirst (1987) introduced a modification to the theory and conclude that, definite noun phrases require "the recipient of a discourse" to attempt a connection to the existing knowledge. So it is necessary to search the whole discourse space to locate the antecedent and thus the author concluded stating that indefinite, generic or simple plural noun phrases are preferable for searching over definite noun phrases. His method though relies more into the semantic representation than syntactic representation.

Whittemore et al. (1990) presented a comparative study on the preference heuristics presented above. These heuristics are the basis of any structural solution to the PP disambiguation strategy thus the result this study is quite significant. The author used a dataset of 910 sentences distributed equally over 13 different dialogs containing 745 sentences with potential attachment ambiguity. The author reported a 55% success using the RA heuristic whereas, the strict MA performs worse, with 36% accuracy. The author also reported that 81.86% instances of LP has been successfully extracted which is fairly high accuracy. In the case of RS, the author reported that, with all the NPs the performance was rather poor with 52% accuracy, but the accuracy increase to 90% when the definite NPs are excluded.

2.1.5. PP Disambiguation and Dependency Grammar (DG).

In this section some of the works that specifically used lexical dependency as a means to resolve PP attachment ambiguity. Nivre (2005), in his detailed work on **D**ependency **G**rammar (DG) and dependency parsing stated that modern DG is considered to achieve its formal structure with the inspiring work of Tesnière (1959). In another work on stochastic model for DG, Nivre (2002, p. 1) describes DG as follows,

> *"Dependency Grammar (DG) is a rather vague concept and can probably best understood as an umbrella term covering a large family of grammatical theories and formalisms that shares certain basic assumption about grammatical structure. The most significant assumption is that syntactic structures consist of lexical nodes linked by binary relations called dependencies. This representation thus lacks phrasal nodes unlike the traditional representation based on constituency".*

Blevins & Sag (2011, p. 1) argued that, "most of the history of linguistics is the history of DG". It is widely used as a method of syntactic representation by traditional

grammarians, especially in Europe, and predominantly in Classical and Slavic domains (Mel'čuk, 1988).

Nagao (1990) developed an experimental system called Dependency Analyzer. The system used, "instances of dependency structures" extracted from a terminology dictionary (IBM Dictionary of Computing) as a knowledge base. The process is explained by the author as follows,

> *"Structural (attachment) ambiguity is represented by showing that a word has several words as candidate modifies. The system resolves such ambiguity by searching the knowledge base for modification relationships (dependencies) between the word and each of its possible modifies, then assigns an order of preference to these relationships, and finally selects the most preferable dependency. It was aimed to overcome two serious problems in realizing practical semantic processing; semi-automatic construction of knowledge and efficient use of that knowledge. (Nagao (1990: 282)"*

He used a knowledge base which includes about 20,000 instances of dependency structure. The author evaluated the system by disambiguating the prepositional phrase attachment of about 2,000 sentences. He reported that out of 4,290 PPs, the system correctly disambiguated 3,569, which gives an 83.2% success for disambiguation.

A step forward is to choose the attachment based on the n-tuple formed by the preposition, verb, and noun. The authors (Hindle & Rooth, 1993) used 1,000 sentences with potential PP ambiguity and perform analysis based on some the structural heuristics mentioned on the previous section along with their proposed methodology. They reported 67% accuracy for RA heuristic and only 33% accuracy for MA heuristic. While human analyst performs with an average accuracy of 85% - 88%, the LP heuristic was found to have accuracy around 80%. They also reported that with their method which is a modified LP, the accuracy achieved was up to 89%.

Nuria & Lafourcade (2005) presented a system that used the output of the Xerox XIP parser using the grammar implementation for French. They extracted all possible attachments for a given sentence. Then they query the **W**orld **W**ide **W**eb (WWW) for the attachment distribution statistics and lexical signatures of the components of the pattern. Then all these information has been used to weight the dependency produced by the parser. Although they did not report the final results, they presented there estimate of 80.6% correct attachment.

Hsieh et al. (2007) presented an automatic method to produce erroneous yet unlimited amount of "word association" data to evaluate the best trees produced by a feature-extended PCFG grammar. The error in the data was mainly because of the word sense ambiguity and parsing error produced by the parser. They added lexical dependency or word-to-word dependency as a type of semantic information in the resolution process, extracted for the phrasal heads. They used a Gigaword Chinese corpus (from three different sources, namely, Taiwan's Central News Agency, Xinhua News Agency and Central News Agency) to extract word dependency pairs. Their system was evaluated by standard PARSEVAL metrics and they only used sentences longer than six words for testing. They reported F-Score between 83.99 and 88.83 on three separate test data set.

In another work to incorporate dependency parsing with PP attachment was attempted by Kübler et al. (2007). They investigated the performance of a DP's output in comparison to an independent PP attachment classifier. They also present a method to integrate the PP attachment information into the output of a parser without modifying the parser itself. The experiments used data extracted from the Tübingen tree-bank of Written German. TüBa-D/Z (Telljohann et al., 2005) is a syntactically annotated corpus consisting of newspaper articles comprised of approximately 27,000 sentences, or 470,000 words. For dependency parsing, MALTParser (Nivre et al., 2007) was used. MaltParser is an implementation of deterministic inductive dependency parsing, based on a memory-based or a **S**upport **V**ector **M**achines (SVM) classifier.

An extensive study by Nivre et al. (2007) tested the parser on 10 different languages. They presented that the approach is language-independent and reaches state-of-the-art results. The independent module for PP attachment was reported to reach an accuracy of 81.4% in contrast to the parser output accuracy of 71.8%. Incorporating PP attachment module to annotate parser output shows insignificant improvement in overall parsing output (0.3%). The output accuracy improvement for PP attachment was minor (3.1%) as well.

Cahill et al. (2009) presented a system to automatically extract large lists of tri-lexical dependencies from [PP NP VP] triples taken from a corpus of 230 million tokens of parsed newspaper text. They investigated how effective they are in PP attachment disambiguation by integrating them into a log-linear model (Agresti, 1990) for parse disambiguation. They used the **FSPar** parser (Schiehlen, 2003) to create dependency

structures for each sentence. They reported the F-Score of 80.05% using 10,000 dependency samples. They also reported that increasing the sample size to 250,000 reduces the F-Score to 79.85%. They conclude that higher number of samples may introduce noise to the system thus reducing the performance.

We are proposing a system that is using a form of RA heuristic. The basic idea is to model the PP attachments on the basis of the number of phrases between the modifier and the governor phrases. Using the training data this feature will be modeled and we have termed it **L**inier **P**hrase **D**istance (LPD). A similar traversal distance heuristic was used for the Bosque trees. It tries to obtain the traversal distance between the modifier and the governor token element and we have termed it **T**ree **T**ravers **D**istance (TTD) Heuristic. The heuristics were modeled manually but each heuristic value will be learned from the training data by the system.

2.2. Machine Learning (ML).

This section will provide a brief overview of basic concepts of **M**achine **L**earning (ML) and its components. Learning covers such a broad spectrum of processes that it is difficult to define it properly. ML or **D**ata-**D**riven **L**earning (DDL) though deals with the processes that allow computer systems to learn patterns from data and later identify those patterns in unseen data. Blum (2002, p. 2) presented a concise definition of ML:

> *"Machine Learning Theory, also known as Computational Learning Theory, aims to understand the fundamental principles of learning as a computational process. This field seeks to understand at a precise mathematical level what capabilities and information are fundamentally needed to learn different kinds of tasks successfully, and to understand the basic algorithmic principles involved in getting computers to learn from data and to improve performance with feedback."*

DDL can be crudely named as classification problem or regression problem depending on the type of data the system is dealing with. If the data contains discrete values, the term classifier is used and for real values regression analysis is often used. ML methods relevant to the research will be presented in the next subsection.

2.2.1. Machine Learning Methods.

The learning method categorization is primarily based on the representation of available experience and broadly categorized as Supervised Learning, Unsupervised Learning and Semi-supervised Learning. In supervised learning (Farley & Clark, 1954) the experience

representation or data is labeled with the class of pattern it is representing, often by human data resource provider. When the data is not labeled and the learning algorithm has to determine the label by recognizing a finite set of patterns present in the data, it is called unsupervised learning (Marr, 1970). A hybrid approach known as semi-supervised learning (Scudder, 1965) uses a small amount of labeled data to define a preliminary class definition and later use the acquired knowledge over a larger set of unlabeled data.

This research though will be dealing with discrete values once the training data is produced. The result we will be searching can have only two possible values, either an attachment between a modifier and a governor is correct or incorrect. Thus the problem can be defined as a binary classification problem. The training data production is a heuristic based method. Once the heuristics are defined, the process is an unsupervised method, even if it is not a clustering method. Thus the whole process is often termed as a semi-supervised or boot-strapping method. The heuristics were defined by studying examples but the modeling was performed automatically by extracting the parameters from the data.

The classification algorithms can be based on different learning philosophies regardless of the method. Furthermore, some algorithms classify data into finite discrete classes (Linear Classifiers, Support Vector Machine etc.). Learning philosophy for the algorithms include, Multi-Dimension Vector Geometry (Linear Classification, Support Vector Machine etc.); Statistical Inference (Maximum Entropy Model, Hidden Markov Model etc.); Logical Inference (Decision Tree) and Biological concept Based Models (Artificial Neural Network, Evolutionary Algorithm etc.). Most of these approaches have been used in some **N**atural **L**anguage **P**rocessing (NLP) tasks, more specifically for structural ambiguity resolution (See Duda et al., 2000). The model we are trying to devise can be classified as a linier combination of statistical inference i.e. results of multiple feature probability combined together.

2.2.2. Data and Feature.

Machine learning, like any learning process relies on acquiring knowledge. Increasing knowledge in a system to improve the solution for a target problem is the primary goal for ML methods. The source of the knowledge is the data and features are the knowledge units representing each datum. In NLP, data can be in different forms,

namely, speech, digitized text and images. Each data type poses unique set of issues regarding machine learning methods and algorithms. Data representation is another issue that deals with how the data may look like (numeric values, text, frequency values, etc.). Furthermore, the specific research question that a certain study is trying to answer often shapes the use of a certain data representation and the ML method that are applied to them.

The features are the backbone of any machine learning method. The distributions of the features are the key observation in a machine learning method to obtain a classifier. ML methods try to generalize the pattern present for one or a set of features in a dataset for each decision class. A real world problem may have hundreds of features. In most cases most of the features are noise or useless to the actual problem domain. Worse than that is, if a classifier manages to find patterns in those noises to classify. Feature selection thus is an important task in ML system.

The basic approaches in feature selection are Filtering, Wrapping and Feature Weighting. Filtering is the process of selecting features on the basis of prior knowledge such as, selecting only the feature that has strong correlation to the problem. A more practical example can be in the problem space of the study, while resolving structural ambiguity syntactic and semantic features may be more appropriate than lexical features or word length. In basic wrapping method will try to solve the problem or a small sub-set of the problem using all possible feature sets. Since the exponential nature of the new problem is impractical and greedy heuristics are often employed.

The more intuitive approach though is associating a weight to each feature and in the process of measuring performance the weights can be adjusted to an optimal level. A possible modified version of the proposed model thus can be a weighted linear combination i.e. each of the feature probability weighted with a co-efficient.

2.2.3. Resolving Structural Ambiguity Using Machine Learning.

Machine learning algorithms have been used to derive information that could be used to resolve the attachment decision. Ratnaparkhi et al. (1994) proposed a Maximum Entropy (ME) model that used lexical information within verb phrases obtained from the Penn Treebank WSJ corpus and no external semantic knowledge. They trained the model with both word features and word class features and a binary hierarchy of word

classes derived by mutual information clustering from the corpus obtaining a resolution accuracy of 81.6%.

A non-statistical supervised method by Brill & Resnik (1994) attempted a transformation-based approach (Brill, 1995) and incorporating word-class information. The system attained 81.8% accuracy in resolving PP attachment ambiguity. They also report that the top 20 transformations learned involved specific prepositions supporting the claim of Collins & Brook (1995) that the preposition is the most important lexical item for resolving the attachment ambiguity. The authors (Collins & Brooks, 1995) adopted a backed-off model to smooth for undetected cases. They manage to achieve 84.5% accuracy. They also discovered that preposition is the most informative lexical item for attachment disambiguation and keeping low frequency cases improve performance.

Stetina & Nagao (1997) presented a corpus-based supervised algorithm that employs a semantically tagged corpus based model using decision trees. They also used an unsupervised word-sense disambiguation algorithm with WordNet to sense-tag each word in a labeled corpus. They reported 88.1% attachment accuracy which is nearly as good as what humans can accomplish (88.2%) as it has been reported by Ratnaparkhi et al. (1994).

Sopena et al. (1998) used neural network for the PP attachment disambiguation. They scored higher than previous approaches on the Wall Street Journal corpus, namely 86.8%, and class information taken from WordNet not only for the NP1 and NP2 but for verbs as well. They explain their very good results by the fact that the previous approaches did not use classes over NP1, NP2 and VP, and if they did, they did not consider them simultaneously.

An unsupervised method attempted by Ratnaparkhi (1998) using an extraction heuristic, unambiguous prepositional phrase attachments of the form [V P N] and [N1 P N2] are extracted from a large corpus. Co-occurrence frequencies are then used to disambiguate examples with ambiguous attachments. 81.9% attachment accuracy was reported by the author.

Gamallo et al. (2003a) used an unsupervised method to model word classes (syntactic and semantic sub-categorization) from shallow parsed text corpora and only significant

work found during the research. The usage of sub-categorization information for parsing was originally introduced by Gamallo et al. (2003b). The authors attempted the learning strategy over a Portuguese **P**rocuradoria **G**eral de **R**epública (PGR) Corpus with 178,522 syntactic context and 1,543,659 binary dependencies. They evaluated three syntactic structures namely [NP PP PP], [VP PP PP] and [VP NP PP], found over the corpus. They reported an average precision of 91.20% over their test data of 633 manually disambiguated, randomly selected sentences from the test corpora.

Toutanova et al. (2004) applied a **R**andom **W**alk (RW) model to the task of PP-attachment attaining a resolution accuracy of 87.5%. They worked with the Penn Treebank Wall Street Journal data (Ratnaparkhi et al., 1994), a widely used dataset used by many researchers. It consists of four-tuples of head words and a specification of the type of attachment. There are 20,801 samples in the training set, 4,039 in the development set, and 3,097 samples in the test set. Their supervised method used a Markov chain model that used the training set to estimate empirical distributions and the development set to train the parameters of the random walk.

Zhao & Lin (2004) proposed a nearest-neighbor method that did not rely on any manually constructed knowledge bases, but instead worked by computing distributional word similarities. Their training set comprised of 4-tuples of ambiguous sentences [V N1 P N2] with attachment information, which were extracted from the Ratnaparkhi et al. (1994) dataset. Given a 4-tuple with no attachment information, the training set was searched for its top priority nearest neighbors and the PP attachments were determined based on the known classifications of the nearest neighbors. The experiment yielded a high resolution accuracy of 86.5% and they concluded that cosine of point wise mutual information was better than most commonly used word similarity measures.

Nakov & Hearst (2005) proposed a method that exploited the web as a very large training dataset, extracting its surface features and paraphrases based on the assumption that phrases found on the WWW are sometimes disambiguated and annotated by content creators. Using the Ratnaparkhi et al. (1994) dataset, they obtained an accuracy of 83.82% using N-gram models with statistics obtained by querying exact phrases including inflections and all possible variations of words derived from WordNet against WWW search engines.

We are trying to model the PP attachment using a different approach where the system learns the statistical parameters from the data but the specific feature distributions were chosen on the basis of the analysis of the data. It is a novel approach to the problem regardless of the fact that we are using a very limited feature set.

2.3. Summary.

To the best of our knowledge, very little research had been done for Portuguese. The other significant tree-bank based research other than the Floresta Sintá(c)tica is the CINTIL Treebank. Silva et al. (2010, p. 86) tried to "assess to what extent the available Portuguese tree-banks and available probabilistic parsers are suitable for out-of-the-box robust parsing of Portuguese". They first commented on the previous attempt of Wing & Baldridge (2006) who trained the Bikel Parser (Bickel, 2002) over the Bosque. The authors reported that their implementation of the Bikel Parser trained using Bosque (9,374 sentences) yielded PARSEVAL F-Score of 36.3%. They also reported that after enriching Bosque annotation the parser manage to achieve F-Score of 63.2%.

The authors then presented the CINTIL Treebank (1205 sentences) "was produced from the output of LXGram, a deep linguistic processing grammar (Branco & Costa, 2008) by manually selecting the correct parse for a sentence from among all the possible parses that are delivered by the grammar". According to the authors, Bickel's Parser trained with this corpus reported to achieve F-Score of 76.18%, by the authors.

However, these results are only indicative of the parser's performance on this new corpus, since its size and content is different from that used by Wing & Baldridge (2006). The authors then selected three freely available, open-source parsers for performance analysis, namely, the Bickel parser, the Stanford parser (Klein & Manning, 2003) and the Berkeley parser (Petrov et al., 2006). They reported that trained with CINTIL Treebank, these parsers attained F-Score between 85% and 90%. They also reported the Barkley parser as being the best and that it could be improved to attain 95.61% F-Score, after using dedicated POS Tagger, among other enhancements, namely, named entity recognition and lemmatization. The following chapter will give a brief overview of the parsing systems (i.e. PALAVRAS and STRING NLP chain and its parsing backbone XIP) and the data (i.e. Portuguese tree-bank Bosque).

Chapter 3: Data and Systems.

The research is designed to investigate the **P**repositional **P**hrase (PP) attachment ambiguity in Portuguese text and devise possible disambiguation solution. The source data for the experiments are machine outputs (automatically generated output). The training data is the publicly available tree-bank Bosque[2]. Bosque is available in different output format and the **C**onstraint **G**rammar (CG) (Karlsson, 1990) numbered format will be used for this research. The data will also be processed with the STRING **N**atural **L**anguage **P**rocessing (NLP) chain. STRING can produce outputs in many formats but the **X**erox **I**ncremental **P**arser (XIP) (XRCE, 2011) e**X**tensible **M**arkup **L**anguage (XML) format will be used in this research. This chapter will provide an overview of the data, the environment of the underlying systems and the pre-processing performed to make the data useable.

3.1. Data and systems overview

The source data for the research is the largest freely available tree-bank for Portuguese, Floresta Sintá(c)tica (Afonso et. al., 2002). Floresta Sintá(c)tica or Syntactic Forest was created as a collaboration project between the **V**isual **I**nteractive **S**yntax **L**earning (VISL)[3] project, and Linguateca[4] which is formerly known as the **C**omputational **P**rocessing of **P**ortuguese (CPP)[5] project. It is a set of trees automatically created from the CG output of the PALAVRAS (Bick, 2000) parser. The corpus corresponds to the first million tokens of the CETEMPúblico[6] and CETENFolha[7] corpora, with a size of roughly 1,640,000 words. The text came from the PÚBLICO[8] newspaper and the Folha de São Paulo[9] newspaper respectively.

The underlying system that produces the tree-bank i.e. the Portuguese parser PALAVRAS is designed in the framework of CG parsing. It was originally proposed by Karlsson (1990) and fully documented in Karlsson et al. (1995) and Tapanainen (1996).

[2] http://www.linguateca.pt/Floresta/ficheiros/Bosque_CP_7.4_cgd.txt
[3] http://visl.sdu.dk
[4] http://www.linguateca.pt
[5] http://www.linguateca.pt/proc_comp_port_en.html
[6] http://www.linguateca.pt/cetempublico/
[7] http://www.linguateca.pt/cetenfolha/
[8] http://www.publico.pt/
[9] http://www1.folha.uol.com.br/fsp/

It is a reductionist parsing framework based on the introduction and subsequent resolution of morphological and shallow syntactic ambiguities. The philosophy behind the formalism is to produce declarative constrains i.e. rules that will tell the system about patterns that are not possible rather than defining patterns that are possible. One of the main features of such a system is the high success rate. It is reported by Karlsson (1990) that the earliest system scored above 90% even for languages lack of considerable sized corpus to perform extensive study.

The target system is the STRING NLP chain. The system uses the XIP as its parsing backbone and also the inbuilt rule-based dependency extraction system to produce head dependency links. The dependency extraction rules were implemented at **L**aboratório de **S**istemas de **L**íngua **F**alada (L^2F) (Spoken Language Laboratory). XIP uses **I**ncremental **F**inite-**S**tate **P**arsing (IFSP) as its implementation specific computational framework to parse natural language text. The next subsections will provide details on the data environment and the underlying systems that produced the data.

3.2. CG parsing with PALAVRAS.

CG is an operational paradigm for natural language parsing. Instead of generative rules which is more common for grammar frameworks, it uses the notion of constrain which restricts certain forms or structures to be formed during parsing. The parsing philosophy of CG parsers in general has been clearly described in Karlsson (1990). This parsing formalism is designed to deal with running text (real world regular sentences) and not only the pre-processed, pre-defined perfect sentences.

As the name suggests, descriptive statements or constrains basically tries to discard as many alternatives as possible in a syntactically ambiguous sentence. Constraints are formulated by extensive corpus study. The rules can have structured formal rule-like pattern or probabilistic patterns. The rule-like constraint structure is considered to be preferable over probabilistic constraints. One of the central ideas is the use morphological information as extensively as possible. The parsing process as it has been described in (Karlsson, 1990), is very implementation oriented and explained as a process of resolving six sub-problems,

> *"Preprocessing, morphological analysis, local morphological disambiguation, morpho-syntactic mapping, context-dependent morphological disambiguation, determination of intra-sentential clause boundaries and disambiguation of surface syntactic functions."*

In Karlsson's (1990) implementation the first four modules were executed in sequence and the last two were executed in parallel, making it a five stage parsing process.

Several modifications over the basic CGP design have been made over the years during the development of PALAVRAS. Bick (1996) introduced attachment direction markers to all argument tags and double tags to the central linking word in sub-clauses to improve some weakness in the original CGP design, such as, lack of hierarchically motivated clause boundary, certain unsatisfactory valence feature and improve the overall dependency information over constituent information. He (Bick, 1996) also used a CGP for automatic grammatical analysis of spoken language data for Portuguese by introducing additional rules and by disambiguating pauses (in-utterance) and breaks (inter-utterance). He (Bick, 2000) later introduced the complete CG-based parsing system PALAVRAS for Portuguese.

PALAVRAS is a statistical robust Portuguese parser based on CG formalism and it was developed at the Institute of Language and Communication of the University of Southern Denmark. According to Bick (2000), it always returns at least one analysis even for incomplete or ungrammatical sentences, and has a high accuracy (96%). The parser also has a named entity recognizer (Bick, 2000) and provides some semantic information for nouns, verbs and adjectives (e.g. organization, date, place, etc.).

Bick (2003) introduced a **P**hrase **S**tructure **G**rammar (PSG) and CG hybrid representation with the suggestion that a complete bias towards either formalism (**D**ependency **G**rammar (DG) and PSG) is also practical and possible. Later he (Bick, 2005) presented a system to create a CG-Treebank with full dependency specification. In (Bick, 2006) and (Bick, 2007), he improved the PALAVRAS for semantic prototype annotation using CG framework. Later in (Bick, 2009), he introduce statistical information regarding the feature tags used by CG rules. Both the training and test data is coming from a corpus created by the CG parser PALAVRAS.

3.3. Bosque Tree-bank.

Bosque is a subset of the Floresta Virgem that has been fully revised and corrected in the scope of the Floresta Sintá(c)tica project. The version used for this research is marked as version 7.4. It is available in several significant formats such as Constraint Grammar Format (CGD), Phrase Structure Format (AD), Penn Tree-Bank and Tiger

XML. The significant statistics regarding the compilation of Bosque version 7.4 can be summarized as, it contains 9,368 trees, corresponding to 1,962 different extracts, featuring 162,484 tokens and it has approximately 140 thousand words. The CGD format output of the Bosque in a text editor looks as follows (Figure 3.1),

```
11  <s id="2" ref="CP1-2" source="CETEMPúblico n=1 sec=clt sem=92b" forest="1"
    text="O 7 e Meio é um ex-libris da noite algarvia.">
12  O    [o] <artd> ART M S @>N #1->2
13  7=e=Meio    [7_e_Meio]  PROP M S @SUBJ> #2->3
14  é    [ser] <mv> V PR 3S IND @FS-STA #3->0
15  um   [um] <arti> ART M S @>N #4->5
16  ex-libris    [ex-libris]  N M P @<SC #5->3
17  de   [de] <sam-> PRP  @N< #6->5
18  a    [o] <artd> ART  @>N #7->8
19  noite    [noite]  N F S @P< #8->6
20  algarvia    [algarvio]  ADJ F S @N< #9->8
21  $. #10->0
22  </s>
```

Figure 3 - 1 : CGD view of a sentence in Bosque.

The annotation schema for all the sentences is identical, although lots of inconsistency has been found in the annotation during the manual analysis of the corpus. A detail analysis of the data has been done to identify and later compensate during the parsing process. In a proper structure the first section of a sentence annotation is the sentence header. A header section can be depicted as in Figure 3.2.

```
11  <s id="2" ref="CP1-2" source="CETEMPúblico n=1 sec=clt sem=92b" forest="1"
    text="O 7 e Meio é um ex-libris da noite algarvia.">
```

Figure 3 - 2 : A typical header in the CGD view of a sentence in Bosque.

The XML type tag **[<s>]** marks the starting of a sentence. The tag also carries several attributes that conveys important information regarding the sentence such as source of the sentence, the raw text of the sentence etc. Primarily the **[text]** attribute that contains the raw text was used to prepare the raw text file to be parsed by the STRING NLP chain. The attempt was a failure since many sentences were found to have discrepancy between the text found in the tag and the actual parsed text. One such irregular sentence has been presented in the following figure (Figure 3.3). It was one of the early experiments and regardless of the outcome of those experiments provided valuable information regarding the corpus.

```
126293  <s id="4151" ref="CP808-3" source="CETEMPúblico n=808 sec=soc sem=94b"
        forest="1" text="Nas «prisões de deportação» alemãs, estão cerca de
        quatro mil «indesejados» à espera da expulsão.">
126294  Em  [em] <sam-> PRP  @ADVL> #1->10
126295  as  [o] <artd> ART F P @>N #2->4
126296  $« #3->0
126297  prisões [prisão]  N F P @P< #4->1
126298  de  [de]  PRP  @N< #5->4
126299  deportação  [deportação]  N F S @P< #6->5
126300  $» #7->0
126301  alemãs  [alemã]  ADJ F P @N< #8->4
126302  $, #9->0
126303  estão  [estar] <mv> V PR 3P IND @FS-STA #10->0
126304  cerca=de  [cerca_de]  ADV  @>A #11->12
126305  quatro  [quatro] <card> NUM M F P @>N #12->14
126306  $« #13->0
126307  indesejados [indesejado] <n> ADJ M P @<SC #14->10
```

Figure 3 - 3 : Inconsistency in raw text found in Bosque.

Figure 3.4 presents some of the parsed lines that contain all the sub elements of a parsed output are presented. There are seven (7) unique segments in each word representation in CGD format output of a sentence denoted by numbers 1 to 7 in Figure 3.4. The first element marked as 1 is always the token itself.

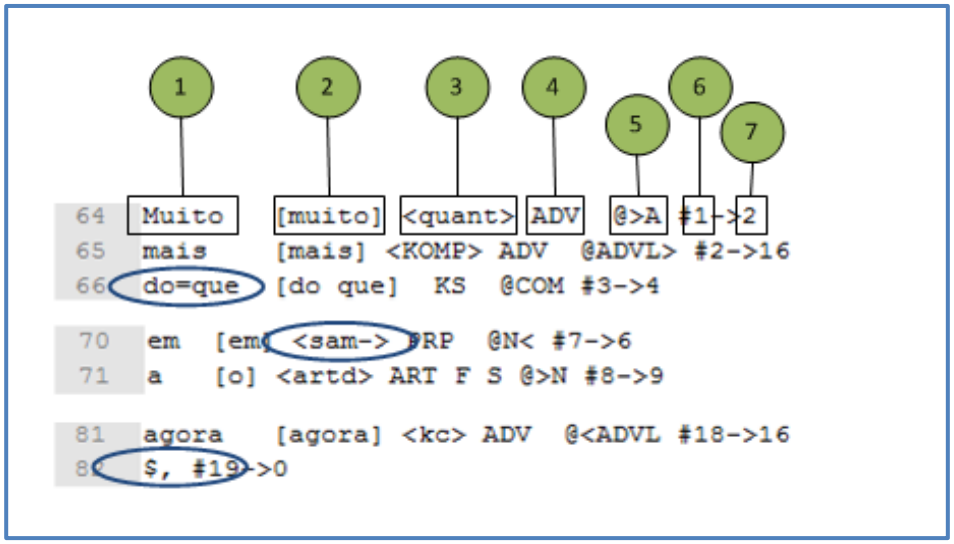

Figure 3 - 4 : Different segments of CGD output of a sentence in Bosque.

One significant issue is that tokens are not necessarily words, rather the unique lexical units processed by the parser as tokens. In Figure 3.4 the token **`[do=que]`** in line 66 and the token **`[$,]`** in in line 82 are the examples of diverse types of tokens produced by the

parser. The tokens in line 70 and 71 are originally a single word **[na]** which is a Portuguese contraction[10] **[em+a=na]** and decided by the parser to split it. The marking **[sam-]** is a special parsing marker to indicate contraction splits.

The second element marked as 2 always appears within square braces. This element is the lemma or the selected reading for the token. This particular element though is absent for punctuation tokens (see line 82 in Figure 3.4). The lemma is insignificant for tokens that are made up of multiple words (see line 66 in Figure 3.4). The third element marked as 3 in Figure 3.4 will always appear between angular braces and multiple such elements can be present. These elements are for semantic features and special parsing features generated by the parser.

The fourth group of elements (denoted by 4), are always in capital letters and multiple elements can appear. These are the morphological features associated to the specific token. The first element and in many cases the only element in this group is the word class tag. The word class tag generated by the PALAVRAS parser roughly represents the **P**arts **O**f **S**peech (POS) category of the token. It followed by (if any) the infliction tags such as gender, number, case, tense, mood, finiteness etc. This block ends with the start of the fifth block tagged as 5 in Figure 3.4.

The fifth block is the syntactic tag associated with a token. This tag always starts with a **[@]** symbol and often have a **[<]** or **[>]** symbol in the block, denoting the tokens association with either a previous element or an element posterior to the token. PALAVRAS is a CG based parser and thus bi-lexical (considering tokens to be lexical entities) relation is represented through this block. In CG representation each lexical entity is connected to only one other element, its parent. This block represents the type of the relation and the type of the expected parent block but not the exact identity of the parent (i.e. the token ID of the parent).

The next two blocks denoted as 6 and 7 in Figure 3.4 contains the unique identity of both the lexical element and its parent. The sixth element is the unique identifier for the lexical element and the seventh element is the parent. Some of the sentences have to be discarded because of faulty identity of the parent. Part of the flawed elements that have been listed by a regular expression search is presented in Figure 3.5. The manual

[10] Contraction (grammar) - From Wikipedia, the free encyclopedia.

analysis of the data yield that the flaw appears to be in the parent ID only and it is difficult to define a transformation function to correct the flawed mapping.

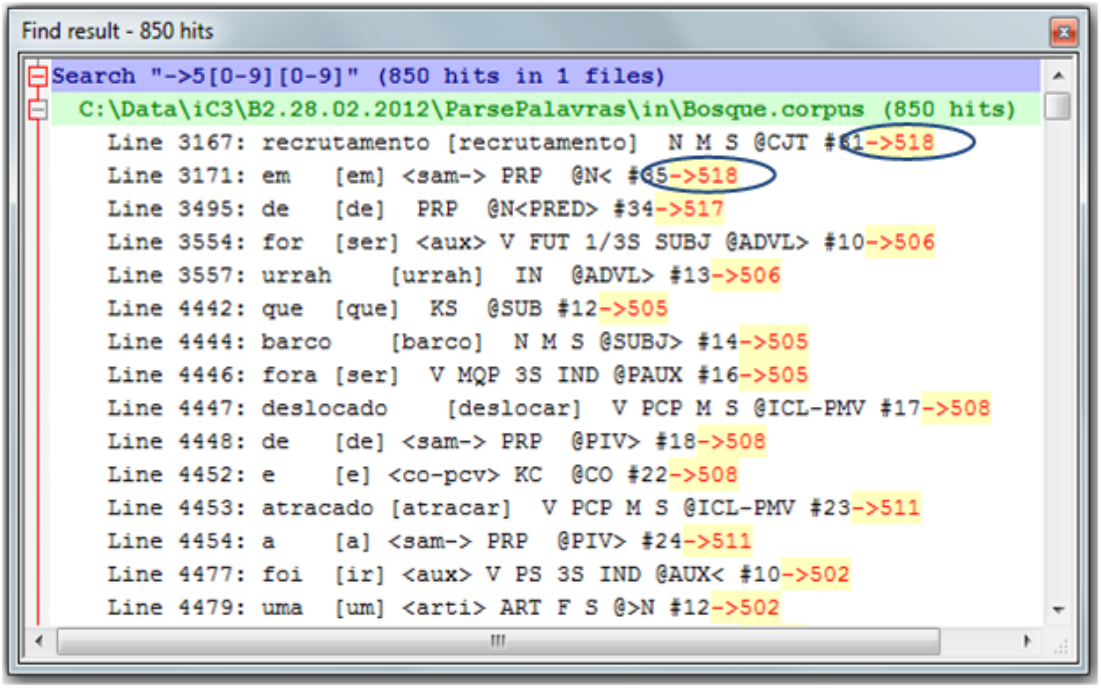

Figure 3 - 5 : ID error in CGD output of Bosque.

A total of 850 instances of such error have been identified and from the manual analysis of the data it is found that the errors are trend to group together i.e. if a sentence contains an error many can be found in the same sentence. Thus the sentences that contain such irregularity had to be ignored since a gap in the lexical mapping makes it unmanageable to be used in any experiment.

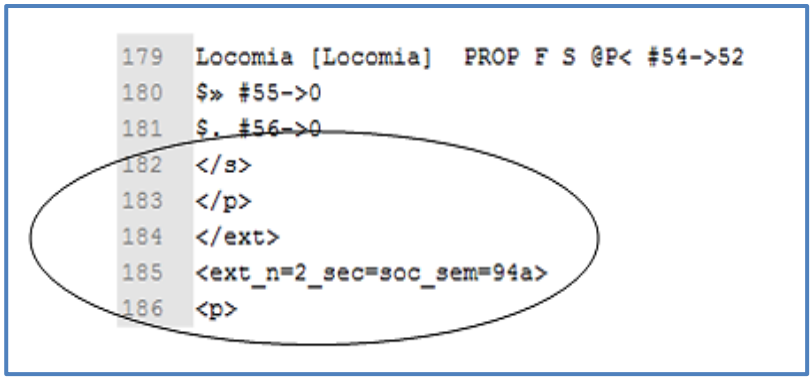

Figure 3 - 6 : Irrelevant random element error in CGD output of Bosque.

Any machine generated output is expected to be consistent, even when making errors (i.e. it is expected that the system will make errors consistently). Bosque however had erroneous parse output and in the analysis of the data they appear to be random. Figure

3.6 shows some random tags often found in the data that make any regular expression based data extraction impossible.

Moreover, the online version of the parser[11] has not been reported to make such random errors i.e. the errors appear to be exclusive to the corpus. The documentation for the corpus is rather poor and thus a manual error analysis of the corpus was the only alternative to properly evaluate the quality of the data. The difficulties faced at different stages of the pre-processing of the data will be reported in the later subsections as appropriate and necessary.

3.4. Finite-State Parsing.

The STRING NLP chain uses XIP as its parsing backbone and XIP is an implementation of finite-state parsing. Finite-state parsing at sentence level falls into two categories, the constructive approach and the reductionist approach. The reductionist approach was influenced by the CG approach (Karlsson et al., 1990). The main idea is to reduce all possible readings of a sentence (represented by finite-state automata) to one correct reading by a set of elimination rules (Loftsson & Rögnvaldsson, 2007). On the other hand a common constructive approach is to string together a sequence of transducers to build incremental (or cascading) shallow parsers (Abney, 1997). Each transducer in this approach adds syntactic information into the text, such as brackets and names for grammatical functions.

Incremental **F**inite-**S**tate **P**arsing (IFSP) is an implementation specific computational framework that adopts a hybrid method that merges the constructive and the reductionist approaches. Syntactic information is added at the sentence level depending on the contextual information and often reported to achieve broad coverage and include richer information than typical chunking systems (Megyesi & Rydin, 1999).

One of the significant practical implementations and within the scope of the interest of the research is, XIP (Aït-Mokhtar et al., 2002). XIP is a natural language analysis tool designed for extracting dependency functions between pairs of words within the sentences. Sándor et al. (2006) explains that the concept-matching grammars are built on top of a general rule-based robust dependency grammar that has been developed in **X**erox **R**esearch **C**entre **E**urope (XRCE).

[11] http://beta.visl.sdu.dk/visl/pt/parsing/automatic/dependency.php

3.5. STRING NLP Chain.

Portuguese rule-base grammar for XIP was initially developed in collaboration with Xerox, since 2004 and it is the parsing backbone for the STRING NLP chain. STRING is a hybrid statistical and rule-base NLP chain for Portuguese, which has been developed by L2F, at INESC-ID Lisboa. The modular structure of STRING is its fundamental feature and it performs all the basic NLP tasks in four steps: Preprocessing, Lexical analysis, POS Disambiguation and Parsing. Figure 3.7 provides a detail process flow of the STRING NLP chain.

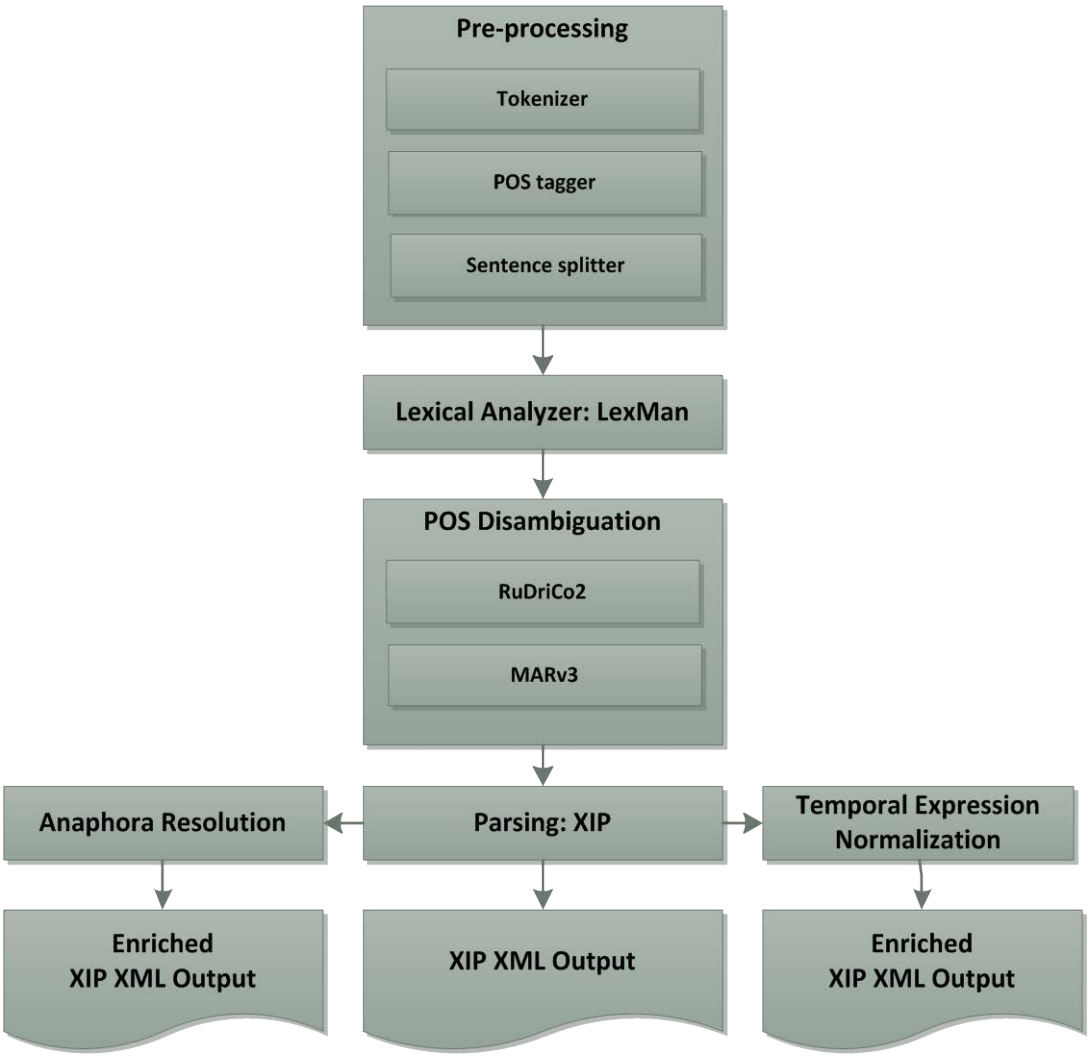

Figure 3 - 7 : The STRING NLP Chain.

The pre-processing stage is comprised of three modules. The first one, the tokenization module, is mainly responsible for dividing the input into individual segments or tokens. For example, consider the sentence [`O Pedro foi ao Brasil`] (*Pedro went to Brazil*) given as input to the tokenization module, the output would be,

```
word[0]: |O|
word[1]: |Pedro|
word[2]: |foi|
word[3]: |ao|
word[4]: |Brasil|
word[5]: |.|
```

The next module is the lexical analyzer LexMan (Diniz & Mamede, 2011). This module assigns each token with its part-of-speech and any other relevant morpho-syntactic features (gender, number, tense, mood, case, degree, etc.). The rich tag set has a high granularity featuring 12 POS categories and 11 feature fields. Some of these fields are category (CAT), subcategory (SCT), mood (MOD), tense (TEN), person (PER), number (NUM), gender (GEN), degree (DEG), case (CAS), formation (FOR) etc. (Mamede, 2011). At this stage, the output of the same sentence would be:

```
word[0]: |O| POS >[o]Td...sm... [o]Pp..3sm.as
word[1]: |Pedro| POS >[Pedro]Np...sm...
word[2]: |foi| POS >[ser]V.is3s=... [ir]V.is3s=...
word[3]: |ao| POS >[ao]S...sm..f
word[4]: |Brasil| POS >[Brasil]Np...sm...
word[5]: |.| POS >[.]O.........
```

At this point each token is assigned a POS tag and each tag has a corresponding code. For example, [`Pedro]` have the corresponding code [`Np...sm...]`, which basically means that they are proper nouns, singular and the gender is male. Whenever a token is ambiguous and might belong to several categories, all possible readings will be listed. The token [`foi]`, which might mean either the verb [`ser]` (to be), or it might be [`ir]` (to go). So, both the readings are kept at this stage. The final step of the pre-processing stage is the sentence splitter module. In order to build a sentence, the system matches sequences that end with one of the following characters `[.|!|?]`.

The next stage of the processing chain is the POS disambiguation modules. The first module in the process is a rule based disambiguation module called the **Ru**le **Dri**ven **Co**nverter 2 (RuDriCo2) (Diniz, 2010). RuDriCo2 provides adjustments on the results produced by a morphological analyzer to the specific needs of each parser. It makes segmentation changes such as `[ex-]` and `[aluno]`, into one segment: `[ex-aluno]` or `[nas]` into two segments: `[em]` and `[as]` depending on the parser's necessity. The new version RuDriCo2 is reported to be significantly (10 times) faster than the previous version. RuDriCo2 also validates the input data, displays error messages, and warns for potential problems (Mamede, 2011).

The next process is the statistical part-of-speech disambiguation module MARv (Ribeiro et al., 2003). Its function is to choose the most likely POS tag for each word, using the Viterbi algorithm. The language model used by MARv3 (newer version) is trained on a 250K word Portuguese corpus. The current implementation of MARv3 performs at over 97% precision and it is significantly (9 times) faster than the previous version (Mamede, 2011).

The final stage of the processing chain is the syntactic analysis performed by XIP. XIP makes use of a rich set of lexical resources, which add linguistic (syntactic and semantic) information to the output of the POS tagger. XIP parses the text by dividing it into chunks, that is, elementary phrases such as NP, PP, and identifies their heads. Then, it extracts the syntactic relations between the heads of those chunks. These dependencies correspond to the major deep parsing relations of Subject, Direct Object, Modifier, etc., but they also include auxiliary dependencies between different chunks and words, such as those necessary to link verbal chains formed of strings of auxiliaries (Baptista et al., 2010).

XIP is a language-independent parser that takes textual input and provides linguistic information about it. XIP can modify and enrich lexical entries, construct chunks and other types of groupings, and build dependency relationships. The fundamental data representation unit in XIP is the node. Being able to extract dependencies between nodes is very important because it can provide richer, deeper understanding of the texts. Dependency rules take the sequences of constituent nodes identified by the chunking rules and identify relationships between them. A special type of regular expression, **T**ree **R**egular **E**xpression (TRE), is used in XIP in order to represent the connections between distant nodes. Dependency rule files written in TRE are used by XIP to produce the dependency relations during the parsing.

This research is interested in the process of defining the dependency between a PP and its governor by the XIP system. While defining the dependency between chunks, the current version of the Portuguese grammar implemented in XIP deals with the issue of PP attachment using a three step disambiguation strategy. Firstly, a Modifier dependency (MOD) relation is set between the PP head and all other prior chunk heads within the same sentence. Some of the MOD dependencies are then eliminated to avoid the extraction of long-distance dependencies.

Secondly, some of the MOD dependencies are converted to complement dependency (COMPL); this is a binary dependency that links a predicate (verb, noun or adjective) to each of its essential complements. Finally, depending on the context, the remaining multiple MOD dependencies issuing from a single chunk are reduced to just one dependency by attaching it to the nearest prior **N**oun **P**hrase (NP) or PP or the MAIN head (this later is an unary dependency that extracts the main predicative element of a sentence; e.g. the main verb); this final stage may be considered to be a base-line for many parsing evaluation.

3.6. XIP XML Structure.

The STRING NLP chain can output a parsed sentence in several formats and this research will be interested in the XIP XML format output. The output is generated basically using the pre-defined XML **D**ocument **T**ype **D**efinition (DTD) of XIP (XRCE, 2011). A brief understanding of the DTD was required to process the data and to understand the significance of each segment produced by the XIP system. The common elements in a XIP XML output (an exact output of a sentence from Bosque) has been presented in the following figure (Figure 3.8).

```
  1  <XIPRESULT math="0">
  2  ...
  3      <LUNIT language="Portuguese">
  4          <NODE num="26" tag="TOP" start="0" end="49">
189          <DEPENDENCY name="MAIN">
192          <DEPENDENCY name="QUANTD">
196          <DEPENDENCY name="DETD">
200          <DEPENDENCY name="DETD">
204          <DEPENDENCY name="PREDSUBJ">
208          <DEPENDENCY name="VDOMAIN">
212          <DEPENDENCY name="MOD">
217          <DEPENDENCY name="SUBJ">
222          <DEPENDENCY name="ATTRIB">
226          <DEPENDENCY name="NE">
232          <DEPENDENCY name="NE">
237      </LUNIT>
238  ...
239  </XIPRESULT>
```

Figure 3 - 8 : First level of DTD's in XIP XML.

Each output produced by the XIP can be found within the XIPRESULT element. LUNIT (linguistic unit) element roughly represents one sentence and the XIPRESULT

contains a list of LUNITs. Each lexical element (tokens or chunks) are considered to be NODE elements. Each NODE element can have child NODE elements and having so the NODE element can be inferred to be chunk NODE. Token level NODE element will not have any children. Each LUNIT will have only one NODE child which is the top level node with attribute `[tag="TOP"]`. Each LUNIT element may also house several DEPENDENCY elements. The hierarchy that exists in a NODE element has been presented in the following figure (Figure 3.9).

Figure 3 - 9 : Hierarchy of a NODE element in XIP XML.

As it has been mentioned, in Figure 3.9 we can see just one top level NODE element with attribute `[num="26"]`. The node hierarchy can be clearly seen in the figure and it

roughly corresponds to the parse tree (or chunk tree considering the nature of XIP being a shallow parser) of the sentence. Each NODE element always contains four (4) attributes and they are presented in the following table,

Attribute	Description
num	A unique identifier for each NODE element.
tag	The name of the NODE element.
start	The character count of the first character of the first word in a NODE.
end	The character count of the last character of the last word in a NODE.

Table 3 - 1: NODE element attributes.

Each NODE element can also have embedded FEATURE elements. Each FEATURE element has two (2) attributes, **[attribute]** and **[value]**. Using the FEATURE elements all possible features associated to a particular NODE can be encoded as binary features. It can be easily seen from Figure 3.9 that the presence of each feature **[attribute]** indicates its presence in the NODE element. The NODE elements that represent tokens rather than chunks also have another child element, TOKEN. Within the TOKEN tag the text or the PCDATA element represents the actual token found in the input. Each TOKEN element contains the result of tokenization, morphological analysis and lexical disambiguation and it can have FEATURE elements as child. It also has another child element READING, which provides the disambiguated lexical unit. A TOKEN element can have multiple READING elements and each READING element have a lemma of the token for the specific reading.

```
  3    <LUNIT language="Portuguese">
  4        <NODE num="26" tag="TOP" start="0" end="49">
189        <DEPENDENCY name="MAIN">
192        <DEPENDENCY name="QUANTD">
196        <DEPENDENCY name="MOD">
197            <FEATURE attribute="POST" value="+"/>
198            <PARAMETER ind="0" num="12" word="ex-libril"/>
199            <PARAMETER ind="1" num="18" word="noite"/>
200        </DEPENDENCY>
201        <DEPENDENCY name="DETD">
```

Figure 3 - 10 : DEPENDENCY elements in XIP XML

The other significant child element of the LUNIT element is the DEPENDENCY elements (Figure 3.10). Each DEPENDENCY element corresponds to one dependency

relation between two NODE elements. Each DEPENDENCY element has an attribute [name] and the value of this attribute is the type of a DEPENDENCY element (MOD, MAIN, CDIR etc.).

DEPENDENCY elements are results from linguistic analysis performed on the NODE elements. Each DEPENDENCY element contains two child elements, PARAMETER and FEATURE. FEATURE elements in a DEPENDENCY element represent in a similar manner as in NODE elements. PARAMETER elements contain the reference to the NODE elements among those the dependency relation is between. Each PARAMETER element contains three (3) attributes, an index ([ind]) a NODE element's number ([ind] attribute), a number ([num]) that corresponds to the parameter's count (starts from zero (0)) and the word ([word]) attribute that contains the token if the node is a TOKEN element.

3.7. Summary.

The understanding of the data and the system is vital for the proper execution of the research. Both systems have rule-based and statistical components yet they are not quite similar. Looking at the data representations from the systems provided an overview of the source data and possible interaction between the systems. It is important to understand the choices made in selecting the data, i.e. Bosque, to conduct the research on PP attachment disambiguation using only machine produces data.

The target system i.e. XIP XML output of the STRING NLP chain, using the heuristics from both systems provides target system specific features to identify potential candidates and human influence in one of the source system's features (Bosque have human input embedded in the parsed output) to improve accuracy of each selection. Regardless of the automated nature of the outputs, the complexity of a rule-based system and the hierarchy in the rule triggers embed significant amount of discrepancies in the output. Thus, a lot of the pre-processing was significant in the research. During these pre-processing stages one of the major objectives came to light, designing a framework to conduct research on PP attachment disambiguation using two parsers. The next chapter will provide details on these pre-processing experiments.

Chapter 4: Data Pre-Processing.

Data pre-processing is a significant part of the research because all the data is text outputs. Raw text output is considered to be useless for any formal processing, either generic data analysis or using **M**achine **L**earning (ML) to model a phenomenon. The first processing tasks were to produce suitable data-structures for both systems. The task led to several experiments and to improve the original code several times. The primary assumption was that machine produced data follows strict patterns and thus producing data-structure was considered to be defining all patterns, but during the experiments the assumption was proven to be wrong. The details of these processes and the problems that had to be dealt with will be reported in the following subsections.

Once the data-structure was produced, the next step was to establish token level alignment between the outputs of the systems. During the parsing of Bosque to produce the data-structure, raw text was extracted. The raw text was then parsed with the STRING NLP chain. The XIP XML output did not have sentence level alignment because the sentence splitter of the system always splits according to pre-defined sentence marker. So, alignment was achieved in two levels, at sentence level and later at token level. After achieving alignment, the experimental data was produced. The later sections will provide the details of the process and their evaluation.

4.1. Data-structure generation.

Data-structure generation process can be split into three sub-tasks, (1) parsing the Bosque corpus; (2) parsing the same text with the STRING NLP chain and (3) XIP data model generation. Each of these sub-tasks may perform more than one task that contributes to other tasks. Each of these tasks and the challenges they present will be discussed in the following sub-sections.

4.1.1. Parsing Bosque.

The data presented in Bosque is in text format and each parsed token contains seven distinct features, token, lemma, semantic information, morphological information, syntactic information, token ID and the parent ID. The primary parsing modules output has been presented in the following figure (Figure 4.1).

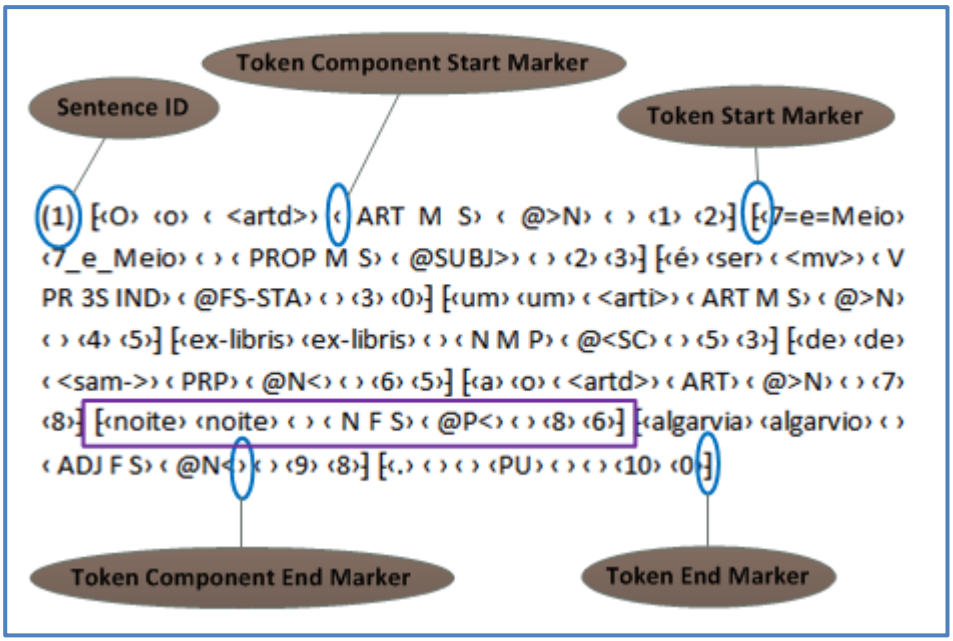

Figure 4 - 1: Bosque data-structure of one sentence.

Each sentence will be parsed into a separate line in the data-structure file and the format is Unicode (UTF-8 encoded) text. Each line will be in the same format and always start with a sentence ID. Sentence IDs are given to each sentence in a sequential order and regardless of the sentence ID present in the header of each sentence in Bosque. Sentence ID's in the parsed output are presented within parenthesis and they are always integers.

Each token element is presented between the token start marker and token end marker shown in the figure. One token component is shown in the figure within the rectangular selection. Each component within a token element is enclosed within component start marker component end marker. One significant point is the number of components in each token is eight (8) rather than seven (7). While testing the parsing with the output of the web interface of PALAVRAS, it was found that the newer version contains an additional feature. Although, Bosque does not have the specific feature, the data-structure has been designed with the provision for both newer version of the corpus or in case of being used to parse the automatically produced output.

All the components of each token are kept as text including the node ID and the parent ID. Some components such as the morphological information component is actually space separated multiple features. These features are kept together at this stage since they are not expected to be used for this research. Semantic and special features are also kept in their original format within the angular brackets.

```
38935  <s id="1297" ref="CP243-5" source="CETEMPúblico n=243
       sec=nd sem=95a" forest="1" text="Tinha calçado um par de
       pantufas inchadas de pêlo velho.">
38936  Tinha   [ter] <aux> V IMPF 3S IND @FS-STA #1->0
38937  calçado [calçar] <mv> V PCP  @ICL-AUX< #2->1
38938  um  [um] <arti> ART M S @>N #3->4
38939  par [par]  N M S @<ACC #4->2
38940  de  [de]  PRP  @N< #5->4
38941  pantufas    [pantufa]  N F P @P< #6->5
38942  inchadas    [inchar] <mv> V PCP F P @ICL-N< #7->6
38943  de  [de]  PRP  @<ADVL #8->7
38944  pêlo    [pêlo]  N M S @P< #9->8
38945  velho   [velho]  ADJ M S @N< #10->9
38946  $. #11->0
38947  Tinha   [ter] <aux> V IMPF 3S IND @FS-STA #1->0
38948  calçado [calçar] <mv> V PCP  @ICL-AUX< #2->1
38949  um  [um] <arti> ART M S @>N #3->4
38950  par [par]  N M S @<ACC #4->2
38951  de  [de]  PRP  @N< #5->4
38952  pantufas    [pantufa]  N F P @P< #6->5
38953  inchadas    [inchar] <mv> V PCP F P @ICL-N< #7->6
38954  de  [de]  PRP  @N< #8->6
38955  pêlo    [pêlo]  N M S @P< #9->8
38956  velho   [velho]  ADJ M S @N< #10->9
38957  $. #11->0
38958  Tinha   [ter] <aux> V IMPF 3S IND @FS-STA #1->0
38959  calçado [calçar] <mv> V PCP  @ICL-<OC #2->1
38960  um  [um] <arti> ART M S @>N #3->4
38961  par [par]  N M S @<ACC #4->1
```

Figure 4 - 2 : Redundant parsed output in Bosque.

The most significant parsing issue during this stage was the redundancy in the parsed output. Figure 4.2 is taken from the actual parsed output of Bosque and the circled lines indicate the start of the same parsed output three times. Although the total number of occurrences had not been counted, significant numbers of cases were encountered during the primary parse attempt. Later consistent sentence boundary had been established by considering the start of each sentence. Throughout the corpus all the sentences always starts with the sentence tag (**<s id="999" … >**). So, once a sentence start-point is found an identifier is initialized and it was only reinitialized if another start point is found.

Moreover each sentence starts with token ID one (1), thus once a sentence boundary is initialized only one token with ID one (1) is expected. The algorithm starts to ignore all the lines from the line where a second token with ID one (1) is found. The parsing of

lines only restarts once the sentence start identifier is reinitialized. This algorithm also helps to ignore irregular tags in between the parsed output. The output parsing algorithm has been tested with all irregular outputs found during the first attempt and found to parse properly.

4.1.2. Parsing with STRING NLP chain.

The parsing process with the STRING NLP chain is rather straight forward and dealt with a shell script to access the processing module in the server provided the input file. The input file is produced by extracting the raw text from the corpus. The extraction was initially attempted from the XML attribute of the sentence header. Due to the inconsistency in the attribute text and actual parsed token made the attempt a failure since token level alignment was not possible from such data. So, later text tokens were extracted directly from the parsed data.

The corpus had its own tokenization protocol and thus the tokens do not correspond to running text. A systematic pattern based token concatenation protocol was established to produce the transformation method to generate reasonable text to be parsed. One of the primary concerns was the representation of the reflexives in the Bosque corpus. Since, the XIP tokenize on the basis of patterns, the input text representation is very important for proper tokenization. The reflexives representation in Bosque is presented in the following figure (Figure 4.3).

```
27156   juntar- [juntar] <mv> V FUT 3P IND @STA #8->0
27157   se-ão   [se] <refl> PERS M 3P ACC @<ACC #9->8

39203   levá-   [levar] <mv> V INF  @ICL-AUX< #17->16
39204   las [ela]  PERS F 3P ACC @<ACC #18->17

39144   apresenta-  [apresentar] <mv> V PR 3S IND @FS-STA #31->0
39145   se- [se]  PERS F 3S ACC @<ACC #32->31
39146   lhe [ela]  PERS F 3S DAT @<A<T #33->31
```

Figure 4 - 3 : Representation of the reflexives in Bosque.

The general transformation method designed to extract text from the parsed output was to insert a space character between each token. The reflexives were one of the exceptions, since inserting a space character between these tokens presented in Figure 4.3 introduced tokenization complexity for XIP. The third example in Figure 4.3 is found to be unique and just one such entity was found in in the corpus where the token

is split into three tokens. The algorithm used to extract raw text compensate for this entity. The other lexical level choice made during the parsing was to discard any sentence that has less than three (3) words or sentences without a verb.

One of the primary experiments was an attempt to align the outputs of the parsers manually and it was found that some of the punctuation characters in the sentences were causing inaccurate sentence splitting while parsed by the STRING NLP chain. Moreover, these punctuation characters do not influence the research and thus can be discarded. The raw text extraction algorithm implements a transformation map to replace specific punctuations at specific locations. A list of the map is presented in the following table (Table 4.1).

Punctuation Character	Location	Mapped Character	Mapped Character Description
Missing termination	-	.	ONE DOT LEADER
…	End of line	…	HORIZONTAL ELLIPSIS
..	End of line	‥	TWO DOT LEADER
:	In the line	☺	WHITE SMILING FACE
;	In the line	⁉	INTERROBANG
!	In the line	‼	DOUBLE EXCLAMATION MARK
?	In the line	⁇	DOUBLE QUESTION MARK
--	In the line	—	EM DASH
.	In the line	.	ONE DOT LEADER
...	In the line	…	HORIZONTAL ELLIPSIS
..	In the line	‥	TWO DOT LEADER

Table 4 - 1 : Punctuation map for raw text extraction.

The other punctuations the algorithm had to handle directly, was the termination characters. The primary punctuation characters that XIP deals with are full stop (.), exclamation mark (!) and question mark (?). If a sentence ends with any other punctuation characters or missing punctuation character, a predefined punctuation symbol is put in its place. It is important to end a sentence with a proper punctuation because if XIP does not find a proper termination punctuation it marge the sentences, yet create another inconsistency. The Bosque also make compound words, mostly noun phrases and it is constructed by putting an equal (=) character between individual words elements. The equal (=) characters were replaced with space character during the raw text generation.

Regardless of all the efforts, the system still creates some sentence splitting because of the presence of some ambiguous elements such as date that contains the dot (.) character

(e.g. 13.12.2000). After all the selection 4663 sentences were selected to be parsed with STRING NLP chain and 4671 parses were produced.

4.1.3. XIP data model generation.

The raw text is parsed with the STRING NLP chain and the output in XIP XML is extracted. The data then is parsed using an XML parser specifically designed for the **D**ocument **T**ype **D**efinition (DTD) of XIP XML. The XML parser extracts the data directly into the data-structure designed within the scope of the research. The data-structure is designed to access all the information directly as required for the experiments. The class interaction diagram for the data-structure is presented in the following figure (Figure 4.4).

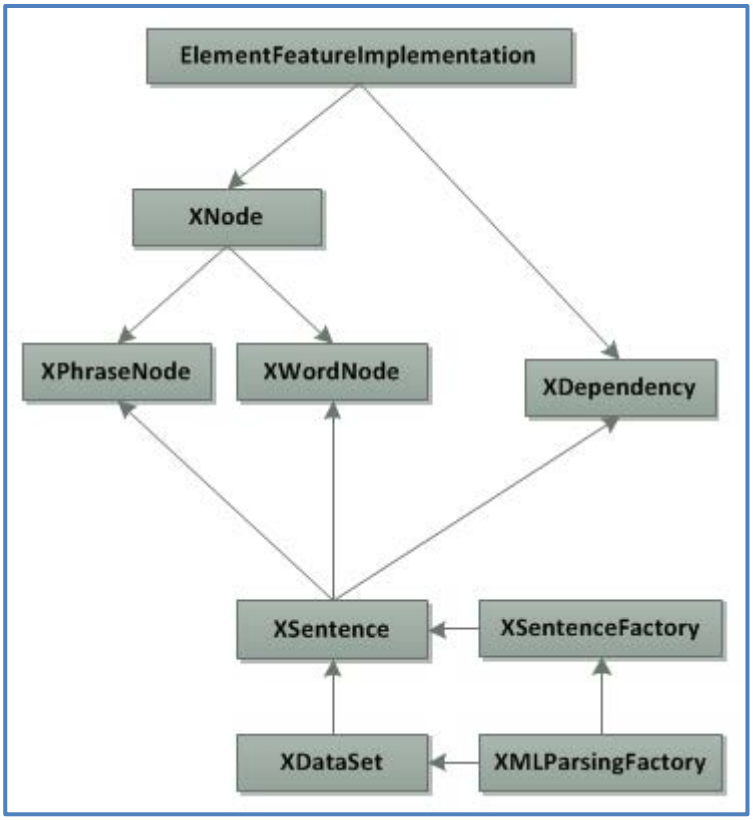

Figure 4 - 4 : Class interaction in XIP data model.

The process can be defined as reverse engineering the parsed data to generate the parser environment that allows the interaction between different elements. In the figure `XPhraseNode`, `XWordNode`, `XSentence` and `XDependency` are the basic data-structure elements readily corresponds to chunk nodes, token nodes, sentence and the dataset components of the XIP output. The `XNode` class encapsulates the generic node attributes common to both types of node. All node elements and the dependency structure have

feature elements and it is encapsulated by the abstract class `ElementFeatureImplementation`. The proper definition of the dependency structure requires identifying the word nodes and the chunk nodes explicitly. Each phrase node contains a list of child nodes, both chunk and word nodes. The sentence element is primarily a structure to contain all the nodes and thus the phrase structure map of the sentence.

The dataset component contains a map of all the sentences in the dataset according to the sentence ID. The `XMLFactory` class contains the XML parser and the means to generate the dataset. It also uses the `XSentenceFactory` to create and populate a sentence object for each sentence in the dataset. The dataset object creates an `XMLFactory` object and calls the method responsible for parsing the input XML file and populate the data-structure with the data. The `XDataSet` object can be extracted for the future processing once the parsing is completed. It also acts as the entry point to call different methods for further processing. The next stage of the processing will be described in the next subsections.

4.2. Token alignment.

The second major data processing task was achieving token level alignment between the tokens of XIP XML and Bosque. These two parsers tokenize text in different ways thus, alignment was absolutely necessary if the tokens are to be used in the experiments. The alignment process was quite complicated once the experimentation started. For the actual alignment, automatic statistical aligner Giza++ (Och, 2000) was used. The whole process was split into three sub-tasks alignment data structure alignment data generation, automatic alignment with Giza++ and head alignment generation.

Before even start the token level alignment process sentence level alignment was required to be achieved. The first step was to generate a compound form of data structure for the Bosque since all the components of the data-structure is not necessary for the alignment process. Moreover the automatic aligner, Giza++, tokenize each sentence considering that the input is a stream of tokens separated by space. Thus the output acquired from the Giza++ is the alignment data between the token ID's specified by the aligner's tokenizer. Thus, a transformation method was needed to map the alignment token ID to the parsing output's token ID. The following sub-sections will provide the details of each of the sub-tasks.

4.2.1. Data-structure for Alignment.

The first sub task was to create a reduced data structure just for the alignment data generation. The task of achieving alignment between two sets of tokens is a task performed at the lexical level. Thus, having the tokens is enough to perform the pre-processing but the symbols, introduced to produce proper text for the input of STRING NLP chain and the punctuation symbols are not needed for this task, yet can only be identified using the POS information extracted from the XIP XML data file. Moreover, the token ID for the XIP nodes were extracted from the XIP XML a data file as given by the system during parsing. Punctuations for example were rather simple to handle since they represent a small and unambiguous set of characters. Symbols on the other hand were sometimes presented ambiguity since some predefined character sequences such as currency symbols (USD, GBP), measuring units (KM, KG) etc. was considered to be symbols. So for the XIP Data structure only three (3) elements were selected the token itself, the POS tag and the token ID.

The Bosque data-structure was kept as it was primarily to reduce one level of unnecessary processing. The reduced XIP XML data-structure was generated using the same data element split markers used in the generation of Bosque data-structure. Thus a single parser can be used to parse both data-structures once saved in text format for future use. The data-structure parser was designed by reverse engineering the actual data parser for Bosque. Although token types such as punctuation and symbols was decided to be excluded in the alignment input data, all the tokens were kept in the data-structure.

4.2.2. Alignment data generation.

The first step of data generation is the sentence level alignment processing. The sentences from the dataset are found to split sometimes but multiple sentences do not marge together since the termination of each sentence is handled at pre-processing level. So, a crude yet effective method was employed that compares the last non-punctuation and non-symbol type element from the XIP output dataset with the non-punctuation type element in the Bosque data-structure. If they are not similar, the sentence is supposed to be split and a matching end element is searched in the next XIP outputs' data-structure. The sentences that are found to get split were discarded and all the other sentences are recorded in a text file by putting the sentence ID from each dataset

together with a **[:]** separated each of them. This alignment information is crucial for the next stage of the alignment process.

Alignment data is generated from the primary data-structures produced by the alignment data-structure generation module. The data-structure contains the tokens in the original format, so the tokens are in the form as there were found in the parsed output. The primary inconsistencies in this data are the phrase representation where multiple words are found to be a single token (e.g. the phrase ***um pouco*** (a little) is represented as ***um=pouco***). In Bosque output these phrases are words connected by the equal sign [**=**], whereas in the XIP XML output these tokens are words separated by space. The data generation module thus produces an intermediate data-structure and saved it for later use to reconstruct the token elements and create the token ID groups representing these phrases. It is important to mention that only the sentences that have been aligned are converted to the data-structure.

The intermediate data-structure is a transformational representation where the phrase elements are represented in a similar structure yet identifiable. Each of these compound tokens are split into word lists and a similar data-structure is created for each of the words. The first word element will contain all the feature elements of the compound token whereas the rest of the words do not contain any feature element.

All the elements other than the first one contains the token ID ***–1***, which is an indicator that it is a part of the last element that has a positive token ID. These features can be used during the reintegration stage. Once the data-structure is produced the alignment text generation is rather simple. The token text of each data element is put in a text file with a space separating each element. The same process is performed for both datasets and it outputs the data-structure and a raw text alignment input file.

4.2.3. Automatic alignment with Giza++.

Automatic alignment is a very important part of this research, since the human bias that will be introduced can only be achieved by once token level alignment is achieved. Giza++ was the tool of choice for the task and it managed to achieve high accuracy during the experiments. Giza++ (Och & Ney, 2000, 2002) is an extension of the Giza program (Al-Onaizan et al., 1999) and it is a part of the Statistical Machine Translation

(SMT) toolkit EGYPT[12]). Giza++ is a free statistical word alignment system that implements **I**nternational **B**usiness **M**achine (IBM) Models 1-5, **H**idden **M**arkova **M**odel (HMM) alignment, and parameter smoothing (Och & Ney, 2003). The Giza++ implementation used for the experiments had a default alignment model that performs the estimation using five iterations of each of Model 1, Model 2, Model 3 and Model 4. For the experiments though, two different parameter settings were tested.

```
1  0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-8 9-9 10-10
2  0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-8 9-9 10-10 11-11 12-12
   13-13 14-14 15-15 16-16 17-17 18-18 19-19 20-20 21-21 22-22
   23-23 24-24
3  0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-8 9-9 10-10 11-11 12-12
   13-13 14-14 15-15 16-16 17-17 18-18 19-19 20-20 21-21 22-22
   23-23 24-24 25-25 26-26 27-27 28-28 29-29 30-30 31-31 32-32
   33-33 34-34 35-35 36-36 37-37 38-38 39-39 40-40 41-41 42-42
   43-43 44-44 45-45 46-46 47-47 48-48 49-49 50-50 51-51 52-52
   53-53 54-54 55-55 56-56
4  0-0 1-1 2-2 3-3 4-4 5-5 6-6 7-7 8-8 9-9 10-10 11-11 12-12
   13-13 14-14 15-15 16-16 17-17 18-18 19-19 20-20 21-21 22-22
   23-23 24-24 25-25 26-26 27-27 28-28 29-29 30-30 31-31 32-32
   33-33 34-34 35-35 36-36 37-37 38-38 39-39 40-40 41-41 42-42
   43-43 44-44 45-45 46-46 47-47
```

Figure 4 - 5 : Primary alignment output.

Giza++ also finds the most probable alignment for each sentence pair based on the estimated parameter values. This alignment is called the Viterbi alignment. Although, the implementation used for the experiments, outputs the alignments in a space separated token ID pair in a text file. The output is then subjected to several post-processing to achieve the required alignment format. An example of the primary alignment format is presented in the following figure (Figure 4.5).

4.2.3.1. Statistical alignment.

In the statistical approach to token alignment, a statistical alignment model is estimated directly from parallel texts with sentence level alignment. An alignment model models the conditional probability of a source token given a target token. These probabilities are estimated from corpora using an alignment model that connects tokens in a source sentence with tokens in a target sentence. Several models are used to create the alignment between the token lists of two systems.

[12] http://old-site.clsp.jhu.edu/ws99/projects/mt/toolkit/

For this research the aligner is used strictly as a monolingual aligner regardless of its original purpose to be used as bilingual aligner. Moreover instead of estimating a translation models from the parallel corpus to train the alignment probabilities, a similarity model will be used, although the pure statistical nature of the process there is not much difference between these models.

Statistical alignment models introduce a temporary alignment $\boldsymbol{a}$ that is defined as the set of all possible connections between each token position $\boldsymbol{i}$ in the source string to exactly one token position $\boldsymbol{j}$ in the target string. The similarity probability $\boldsymbol{Pr(s|t)}$ can be calculated as the sum of $\boldsymbol{Pr(s,a|t)}$ over all possible alignments, where $\boldsymbol{Pr(s,a|t)}$ is the joint probability of the source string $\boldsymbol{s}$ and an alignment $\boldsymbol{a}$ given the target string $\boldsymbol{t}$ (Eq^{n} 4.1).

$$\boldsymbol{Pr(s|t) = \sum_{a} Pr(s,a|t) \quad \ldots \; Eq^{n}\; 4.1}$$

The joint probability $\boldsymbol{Pr(s,a|t)}$ is not estimated directly from a parallel corpus. Instead, the process of mapping the source string $\boldsymbol{s}$ from the target string $\boldsymbol{t}$ is broken down into smaller steps and the probability of each step is estimated from the corpus. The IBM models 1-5 (Brown et al., 1993) decompose the alignment model into a set of parameters that describe this generative process. Model 1 and 2 is built on simple process of mapping how source token is generated from the target token. Whereas, model 3, 4 and 5 are more complex model dealing with relevant factors that affect alignment probability.

4.2.3.2. The models.

Since this research will use only models 1 through 4, Model 5 will not be discussed. The simplest of the IBM-models is the Model 1 and in this model the probability $\boldsymbol{Pr(s,a|t)}$ only depends on one parameter, the mapping probability $\mathbf{Pr}\left(\boldsymbol{s_j|t_{a_j}}\right)$. This is the probability that the target token $\boldsymbol{t}$ aligned to the source token at position $\boldsymbol{j}$ can be mapped to the token $\boldsymbol{s_j}$. Model 2 includes an additional parameter for alignment positions $\boldsymbol{a(i|j,I,J)}$ where the position of the target token $\boldsymbol{i}$ depends on the position of the source token $\boldsymbol{j}$, the length of the target sentence $\boldsymbol{I}$ and the length of the source sentence $\boldsymbol{J}$. In this model, the alignment depends on the source and target tokens as well as the absolute position of the source token (Holmqvist. 2008).

The IBM Model 3 adds several new parameters to the alignment model. In this model, each target token can give rise to several source tokens as in ***juntar-se*** (to join with something) in the XIP XML output can be mapped to the tokens ***juntar*** and ***-se***. The fertility (the probability of the source token set) parameter $\boldsymbol{f(n|t)}$ models the probability that a target token generates $\boldsymbol{n}$ source tokens. Model 3 also assumes that source tokens can be generated from an empty token token at each position in the target sentence. The probability of generating such an empty token is also used as a parameter in this model. Finally, a reversed position model $\boldsymbol{a(j|i,J,I)}$ is used that models the probability of the source token position $\boldsymbol{j}$ based on the target token position $\boldsymbol{i}$.

Model 4 adds two additional parameters, a relative token order model and a first order dependence on token classes. The word order model acknowledges the fact that tokens (since these tokens are representative of a natural language) tend to appear in groups. This is modeled by having two reversed alignment models, one for the first token of a group $\boldsymbol{d_1\left(\Delta j|A(t_{i-1}),B(s_j)\right)}$ and the second model for the relative positions of the following tokens $\boldsymbol{d_{>1}\left(\Delta j|B(s_j)\right)}$. $\boldsymbol{\Delta j}$ is the relative position of the source token being placed and $\boldsymbol{A(t)}$ and $\boldsymbol{B(s)}$ are the token classes of a target token and a source token respectively. Consequently, in Model 4, the placement of the first token of a token group depends on the token class of the previous aligned target token and the token class of the source token being placed. The placement of the other tokens in a group depends only on the token class of the source token. Token classes are automatically induced from data (Och & Ney, 2003).

4.2.3.3. Parameter estimation.

The **E**xpectation-**M**aximization (EM) algorithm (Dempster et al., 1977) is used to iteratively estimate alignment model probabilities according to the likelihood of the model on a parallel corpus. In the Expectation step, alignment probabilities are computed from the model parameters and in the Maximization step, parameter values are re-estimated based on the alignment probabilities and the corpus. The iterative process is started by initializing parameter values with uniform probabilities for IBM Model 1. The EM algorithm is only guaranteed to find a local maximum which makes the result depend on the starting point of the estimation process. Therefore, the result of simpler models is used as initial guesses to bootstrap more complex models.

4.2.4. Alignment data generation.

Alignment data was generated in multiple steps and the first step is to use the automatic aligner. The data generative previously has been used at this step. A predefined implementation of the automatic aligner Giza++ at the University of Wolverhampton was used. This particular implementation uses four IBM-models and five iterations for each by default. A script has been developed to pass the parameters in a systematic way along with the iteration count for each of the models. The data for the aligner have to be process before actually being used for the alignment task.

The input text is general in nature and thus contains all the formatting common to running text. It is important to understand the working principals of the statistical aligner to grasp the requirement of this pre-processing step. Each token is a stream of characters to the aligner and an upper-case letter in a token is makes it a different token then the same token having only lower-case letters. Thus, all the tokens are converted into lower-case strings to have more accurate statistical data. A script written in Perl at the University of Wolverhampton application server is used to perform this task.

Both the data from the XIP XML output and Bosque were case converted and saved as a separate file. The script to run the training data needs to be passed several parameters, the source file, the target file, the name of the output file and the pass count for each of the models (total 4). Once the alignment data is generated it was evaluated using an automated system. The system was designed following the evaluation method proposed by Lopez (2007). The comparison for a correct link was rather crude and it basically tries to find if the tokens are the same or the token with smaller length is a part of the larger one. Either way it is considered to be correct and on the basis of the findings the evaluation is performed.

There were two models generated for the alignment of the datasets. The first model used five passes for all the models and the evaluation of the output shows 99.27% of the links produced were accurate. Moreover, 99.69% of the Bosque tokens and 99.81% of the XIP XML tokens were found to be aligned. The other experiment was performed with ten passes for Model 1, five passes for both Model 2 and Model 3 and 3 passes for Model 4. The evaluation showed 99.11% of the links were accurate. It also showed that 99.73% of the XIP XML tokens and 99.77% of the Bosque tokens were aligned. The very high accuracy of both the models made testing any other patterns to be

unnecessary. For the rest of the experiments the basic model (5 passes for all models) was used. Once the alignment was achieved, the next step was to generate the experimental data and perform the experiments to model the Prepositional Phrase attachment from the dataset and evaluate the models performance.

The output obtained at this stage is similar to what had been presented in Figure 4.5. The specific token level representation is not useful for the research under investigation. Thus the first level of transformation was to convert the token ID produced by the automatic aligner into corresponding token ID from the data-structure of the corresponding data file. The converted data-structure and the alignment systems tokens have a one to one correspondence at this level. Our objective though, is to achieve the alignment at the token level identified by the systems. During the data structure generation, dummy nodes with token ID -1 were created to represent the compound tokens. From the aligners representation these tokens are unique entities and at this stage, those tokens are grouped as a single element.

```
4  4:3-1:0-2:2-3:4-4:6-6:10-7:12-8:14-9:16-10:18-12:38-13:24-15:
   28.30-16:32-22:40-23:42-24:44-25:46-26:48-27:50-28:60-29:54-3
   0:56-31:58-34:62-35:64-36:66-37:68-38:70-40:74-41:76-42:78-44
   :82-45:84-46:86-47:88-48:90-49:92-50:94-51:96-52:98-53:100-54
   :102
5  5:4-1:0-2:2-4:6-5:10-8:12-9:14-10:16.18.20.22.24.26.28.30.32-
   11:34-12:36-13:38-14:40-15:42-16:44-17:46-18:48-19:50-20:52-2
   1:54-22:56-24:60-25:62-26:64-27:66-28:68-29:70-30:72-31:74-32
   :76-33:78-35:108-36:84-37:86-39:90-40:92-41:94-42:96-43:98-44
   :100-46:104-47:106-50:110-51:112-53:116-54:118-55:120-56:122
6  6:5-1:0-2:2-3:4-5:8-6:10-7:12-8:14-9:16-10:18-13:24-14:26-15:
   28-17:32-18:34-19:36-20:38-21:40-22:42-23:44-24:46-25:48-27:5
   2-28:54-29:56-30:58
```

Figure 4 - 6: Token level alignment map.

The representation in the original file was a one to one map between two tokens from the systems, whereas, the final token level map is a one to many map between Bosque nodes ID to XIP XML nodes ID. The same map can eventually be used to have a many to many map at the processing level. To reach this level of alignment a complex set generation algorithm was developed that generates the alignment map set using a clustering method.

4.3. Summary.

The pre-processing of the data is an important part of the experiment. Since the data from two different systems will be used to model the phenomenon, token level alignment is the most important part of the pre-processing. To reach the stage of alignment, proper data-structure is a primary requirement and later the manipulation of the data-structure made the task possible. Once the alignment is achieved, the map can be used for further processing. The next step is thus is the experimental data generation and performing the experiments. Experimental data generation requires some experiments and data analysis themselves to isolate significant data elements for the experiment. The next chapter will illustrate the experimental setup and the results acquired.

Chapter 5: Experments & Evaluations.

The objectives of this research and thus the experiments can be split into two categories; experiments for designing a framework for dependency based research, especially **P**repositional **P**hrase (PP) attachment research and a PP disambiguation method using machine generated data and its evaluation. The framework generation is a complicated subtask and comprised of several components such as, experimental data generation and dependency annotation tool etc. On the other hand, once the framework is designed, experiments were designed to devise a model for the PP attachment from the data. The following subsections will provide details on the experimental data analysis, designing of the annotation tool and the PP attachment model generation and evaluation.

5.1. Experimental data analysis.

This research was trying to create a framework that can be used for PP attachment disambiguation and evaluation. Once the pre-processing was completed, the data was ready to be analyzed for the specific research task. The framework is designed to work with any types of dependency relations present. For the research though, only PP attachments were chosen and they are presented as modifier (MOD) dependencies in the parsed output. Each dependency has two components, the modifier and the governor. The following figure (Figure 5.1) illustrates a modifier dependency represented in XIP XML output.

```
236     <DEPENDENCY name="MOD">
237         <FEATURE attribute="POST" value="+"/>
238         <PARAMETER ind="0" num="18" word="aparelho"/>
239         <PARAMETER ind="1" num="20" word="grave"/>
240     </DEPENDENCY>
```

Figure 5 - 1: Dependency representation in XIP XML

In Figure 5.1 the word ***grave*** (serious) is modifying the word ***aparelho*** (equipment) and the type of modification is POST. There are two major types of MOD dependencies POST and PRE. They represent the relative position of the modifier with respect to the governor. For the PP attachment disambiguation system designed using the framework will be using POST type dependencies only. The experimental dataset is the properly aligned Bosque dataset and it is split into two segments, 10% of the trees (442 sentences) were separated and used as the test data and the rest were used as the training

data. The test sentences were annotated manually by a human annotator, thus for this research we used these fixed sentences. The POST modifiers were distributed among several chunk types. Some of these chunk types are **PRON** (pronoun), **PASTPART** (past participle), **VTEMP** (tense feature verb phrase), **VPP** (past participle verb phrase), **VCOP** (copulative verb phrase), **VCPART** (past participle verb phrase), **NP** (noun phrase), **VGER** (gerundive verbal phrase), **VINF** (infinitive verb phrase), **VF** (finite verb phrase), **ADVP** (adverbial phrase), **NOUN** (common or proper noun), **AP** (adjectival phrase), **PP** (prepositional phrase) etc.

The distribution of the modifier (head) chunk types among 21377 dependencies found in the training data has been presented in the following figure (Figure 5.2).

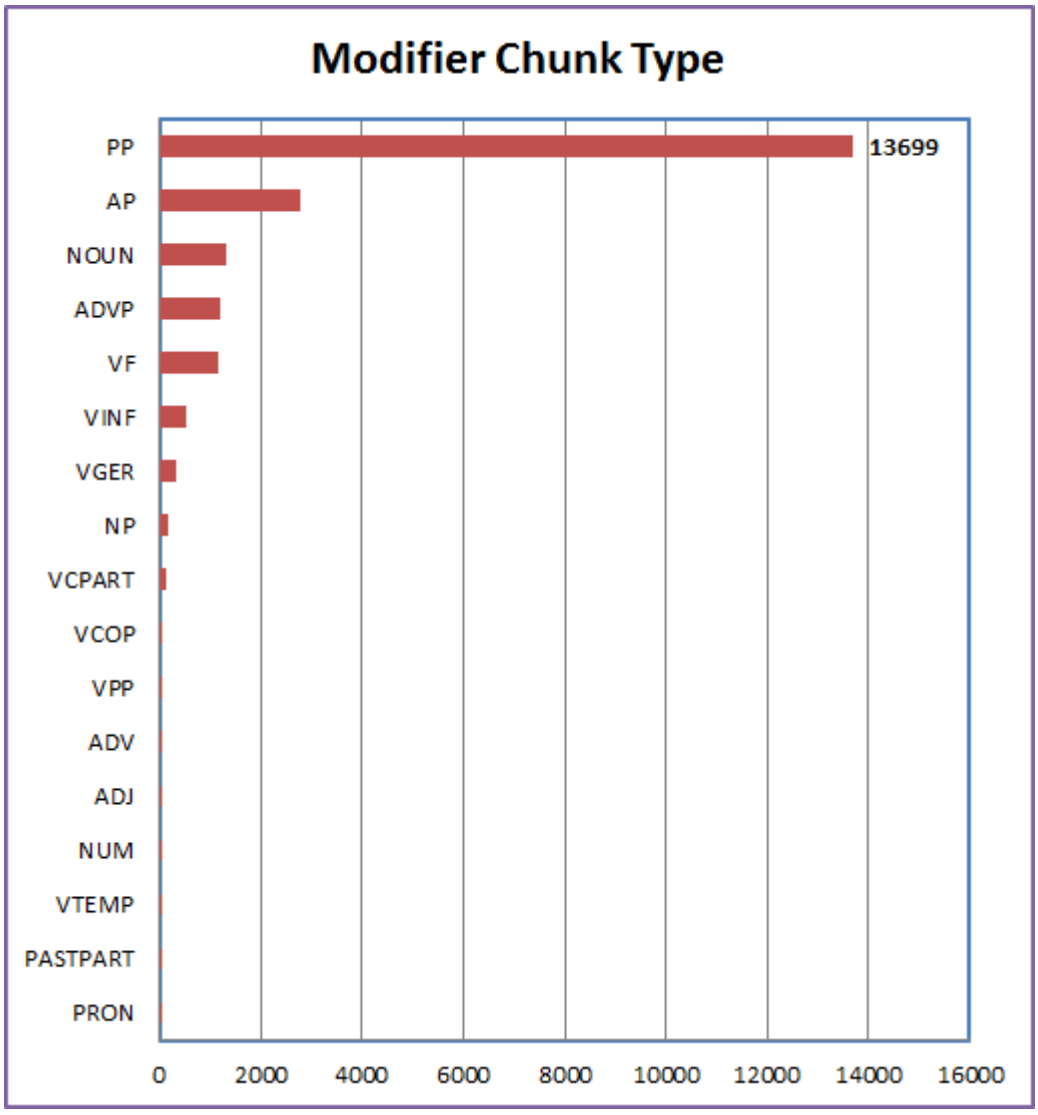

Figure 5 - 2: Modifier (head) distribution in the training data.

It is found that the modifiers (head) in the MOD relationship can be of different chunk type. Our interest was the PPs for this research and it is found that 64.08% of the total modifiers were PP's. Rests of the distributions were calculated for the PP modifiers only. The governor distribution among the PP modifiers is presented in the following figure (Figure 5.3).

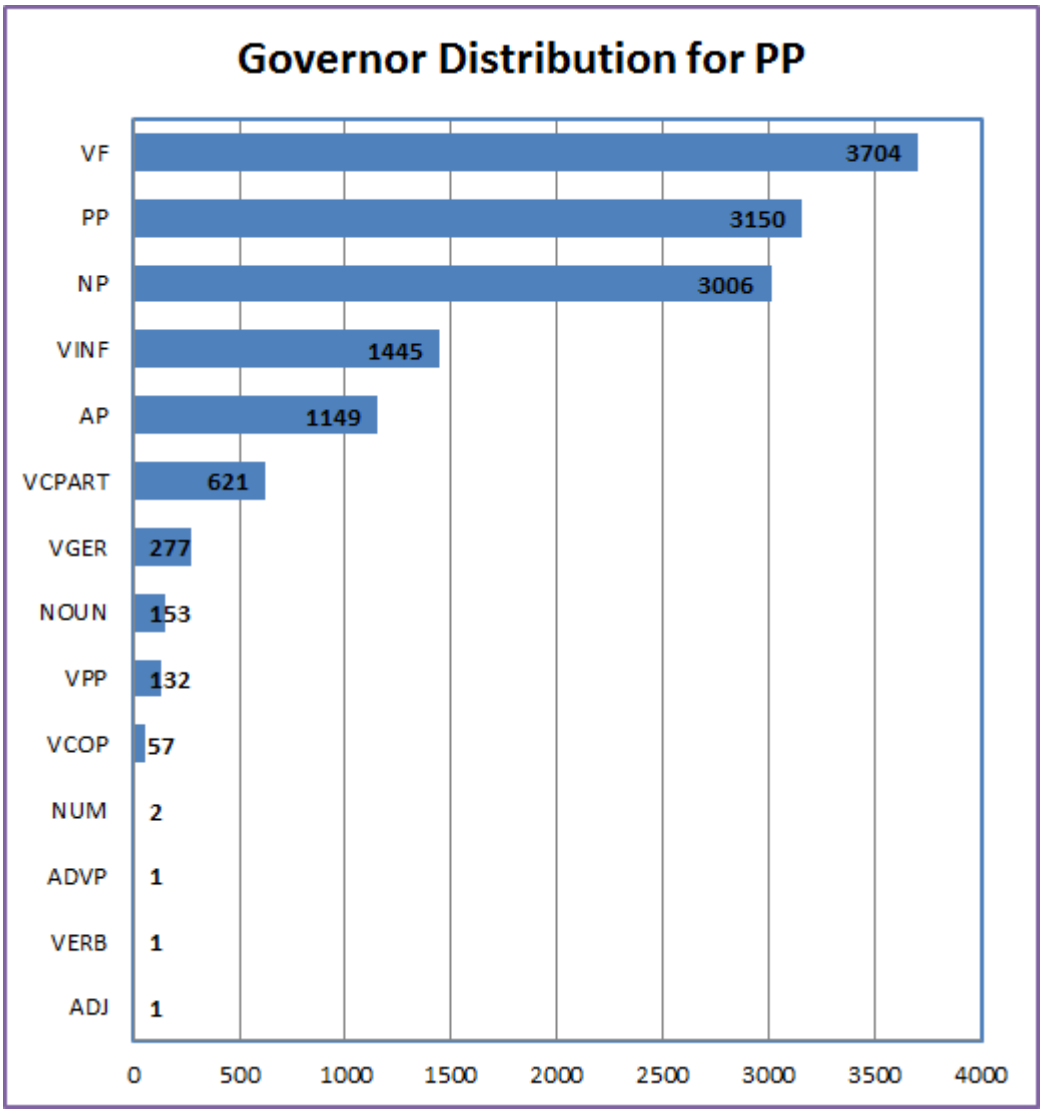

Figure 5 - 3 : Governor distribution of PP in the training data.

The numbers in Figure 5.3 presents that the governors are distributed among many chunk elements. Although the major numbers of governors are actually fall into four (4) types of chunks (token groups above the token level are more precise), **N**oun **P**hrases (NP), PP, **A**djective **P**hrases (AP) and seven (7) different types of **V**erb **P**hrases (VP). The other significant information discovered that each modifier can modify up to seven (7) governors. A distribution of the modified governor count is presented in Figure 5.4.

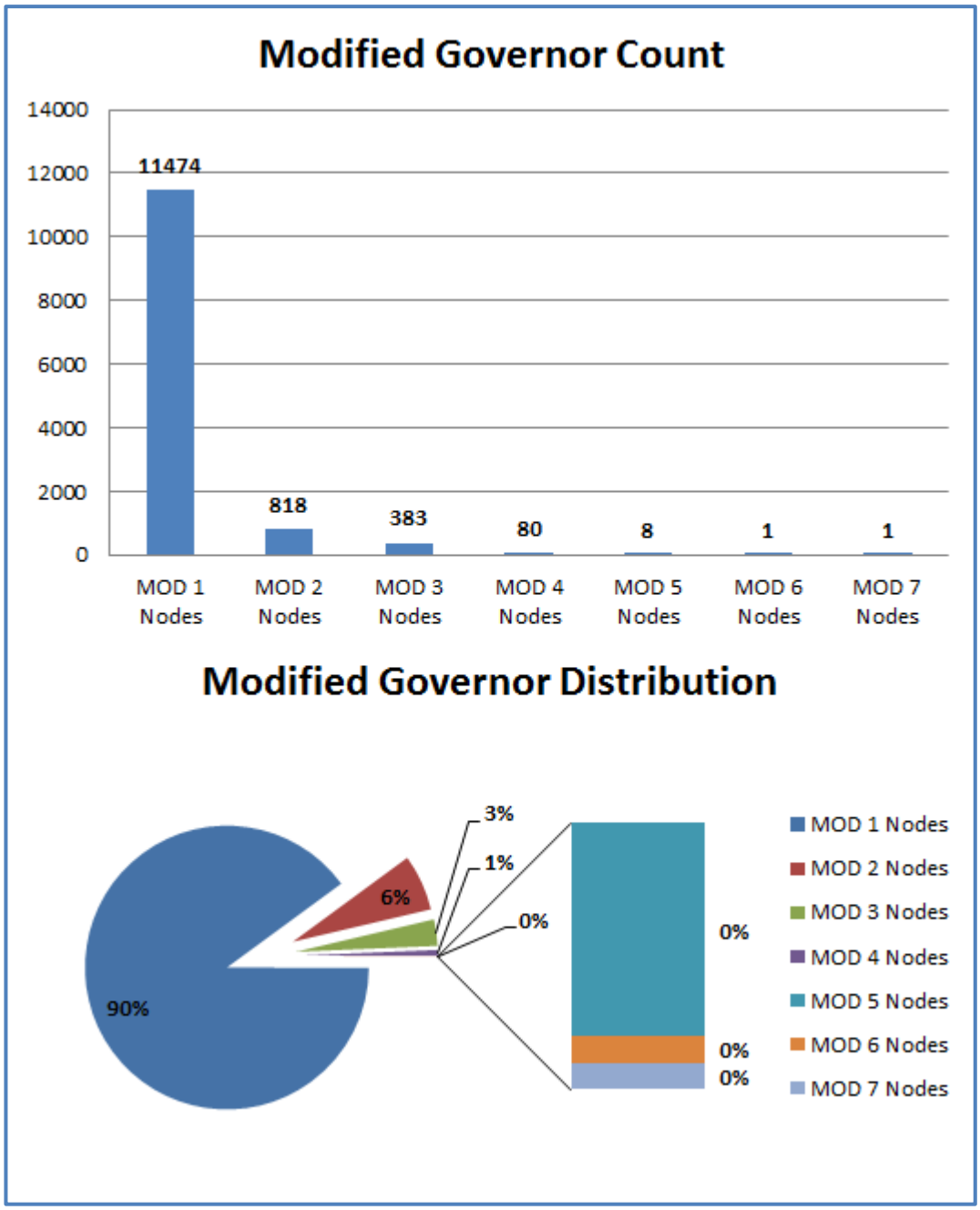

Figure 5 - 4: Modified governor count of PP in the training data.

In Figure 5.3 the distribution is presented in the form of **MOD n Nodes**, and **n** represents the number of nodes modified by a modifier. From the graph it can be seen that 90% of the modifiers modifies only one governor. Moreover, the experiments designed to model the PP attachment will be far too complicated if one PP modifier is allowed to modify more than one element. So, it is considered to be a trade-off between complexity and accuracy. It is also taken under consideration that the amount of data available for multiple modifications is not significant enough to be used to model the phenomenon. Once the data is purged of the modifiers that modify multiple governors,

the distribution changed drastically. The following figure (Figure 5.5) is an illustration of the modified governor distribution.

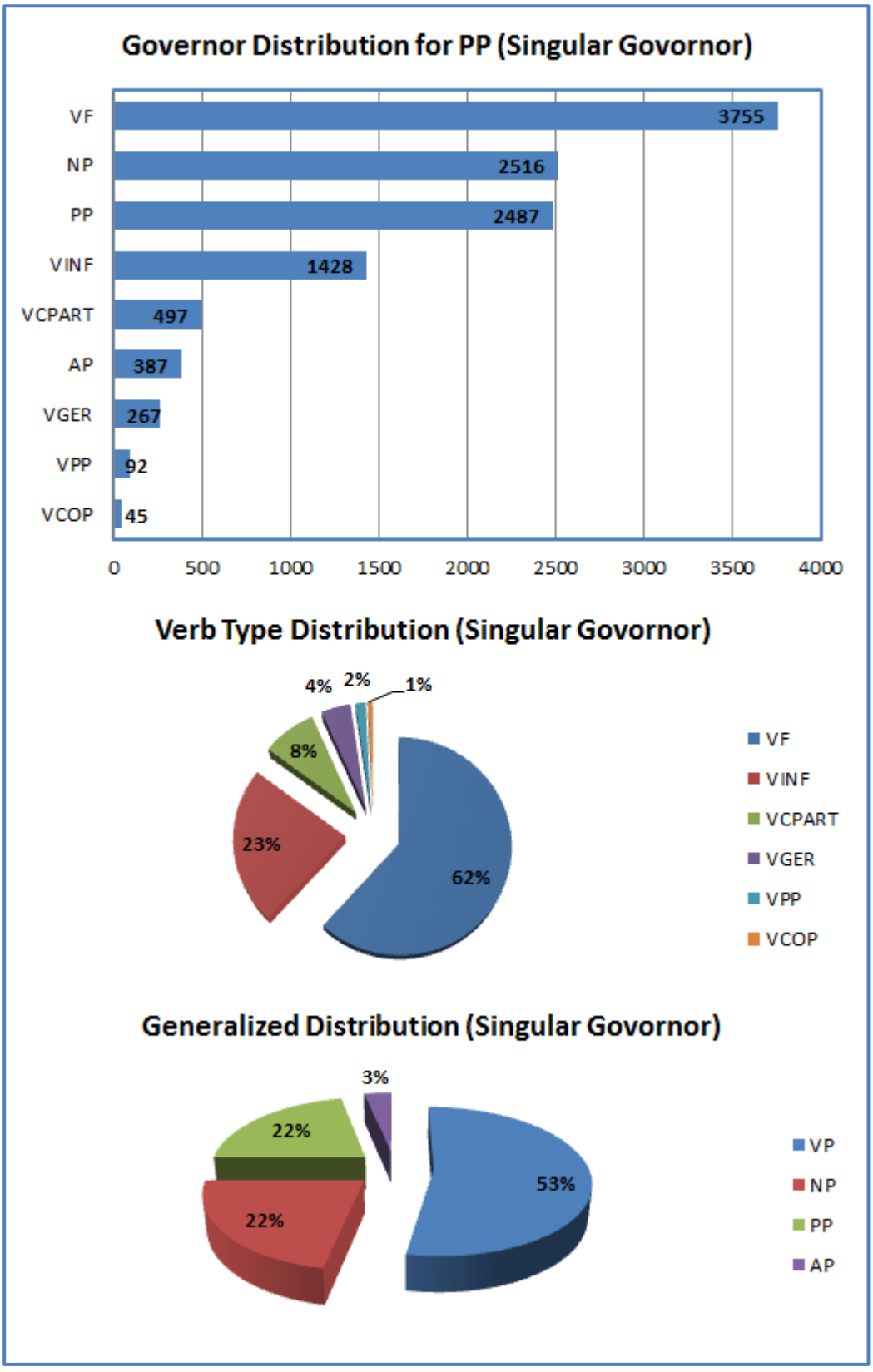

Figure 5 - 5: Modified governor distribution of PP in the training data.

The distribution presented in Figure 5.5 is the training data distribution to be used for the actual modeling process. From the distribution we found that there can be roughly four governor types, VPs, NPs, PPs and APs. The AP's though are very small percentage of the total governors and do not agree to the general PP attachment theories. If it is considered to be wrong annotations and removed from the distribution, the distribution agrees with the previous studies of the PP attachment presented in chapter 2 of this dissertation. Thus, the **M**achine **L**earning (ML) method has to learn how and when a PP is modifying PP's, NP's or VP's. The next section will explain the framework's dependency annotation tool **"DpAn"**.

5.2. The dependency annotation tool DpAn.

An important part of the framework was to design a simple tool to produce human annotated dependency data. The designed system is capable of annotating only one type of dependency using one input file at one time. The input data initialization can be supported by specific dependency model, depending on the application and the specific implementation. For this research the distribution model presented in the previous section was used to initialize the PP attachment annotation input file. The tool was used to produce the human annotated PP attachment test data that was used for the evaluation of the model.

The tool was designed using The Eclipse **G**raphical **E**ditor **F**ramework (GEF). GEF is consists of three frameworks, Draw2D, Zest and GEF. DpAn is designed using Zest, a layer on top of Draw2D for adapting a data model to a graphical interface. Zest provides an easier way to present model information in diagram but it has limits to the presentation format and the ability of the user to edit that information. The graphics is generated by creating a viewer then specify a content provider for providing dynamic data to the viewer and a finally a label provider to display the data in the viewer. Since it is a diagram rather than a list, tree, or table, it is also possible to specify the layout algorithm to tell the viewer how to display the data. Regardless the fact that Zest is a simpler framework with limited functionality, it is a very fast development environment. The tool only provides the basic functionality such as next and previous tree traversal, adding a new dependency link and deleting an existing dependency link. The objective of the tool is to be simple yet effective for the annotators with basic or no annotation experience. Figure 5.6 is an illustration of the basic window of the tool.

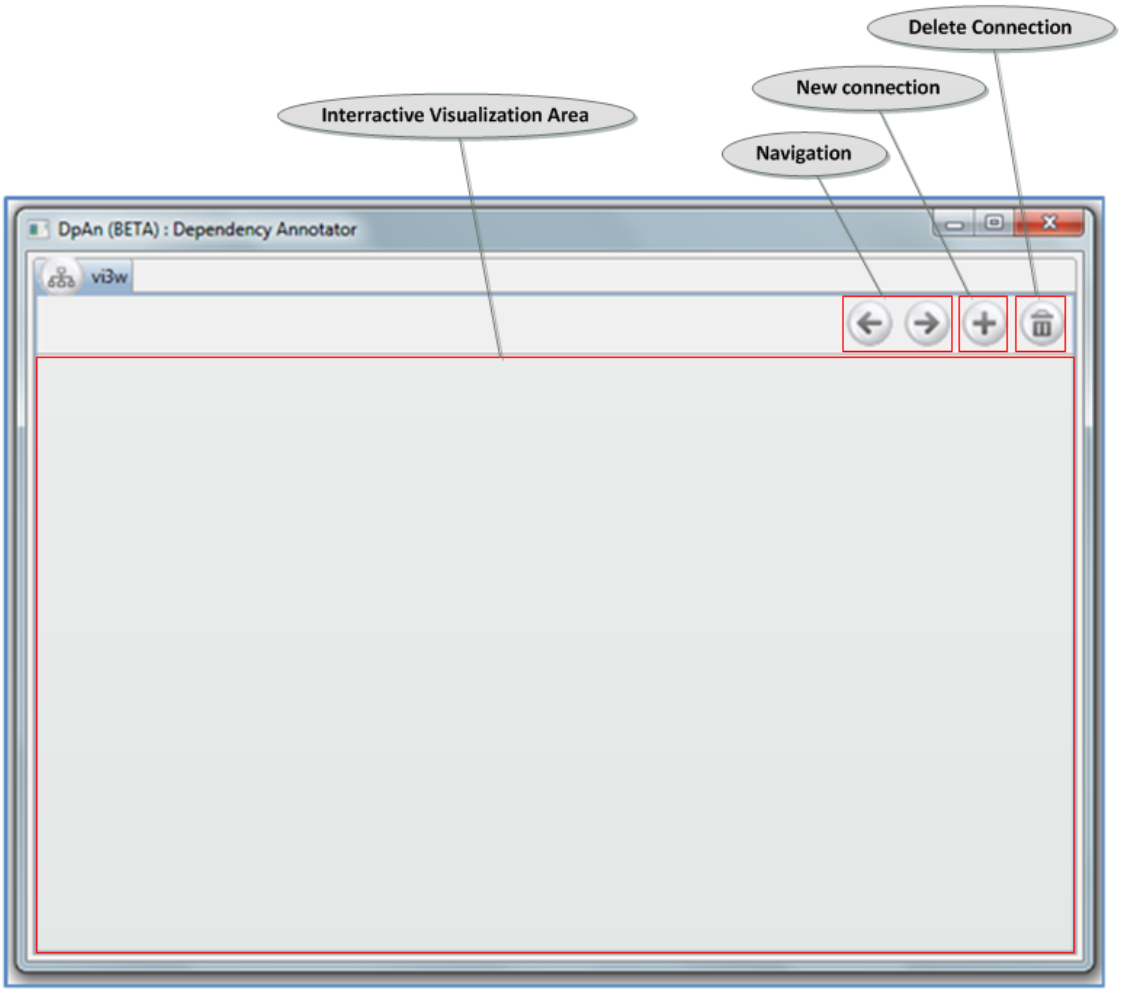

Figure 5 - 6: DpAn Basic Window.

The basic window is the main operational window for the tool. It contains few controllers that can be used by the annotators to perform their task. The interactive visualization area is the designated area, where the **P**hrase **S**tructure **T**rees (PST) are displayed. The navigation buttons allow the user to move among the dataset, one sentence at a time. The **"+"** button is a toggle type button. Once it is put to active state, the annotators only have to select the nodes that have a relation in proper order. The connections are directed thus proper order is very important, although it entirely depends on the objectives of a specific research. The **"delete"** button is used to remove a pre-existing dependency link. The instructions how to annotate are provided at the point of loading of each sentence. The updated data is only saved at the end of each session, i.e. once the program is closing it will ask the user if he/she wishes to save the changes. The following figure (Figure 5.7) presents the screenshots of the instructions and save prompt window.

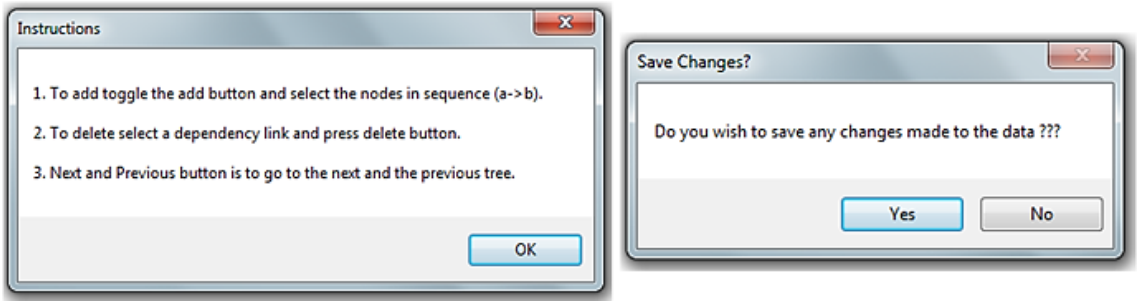

Figure 5 - 7: DpAn Prompts.

Each tree in this tool is presented as a modified tree since it is easier to present a tree that is both easy to read as a sentence and to represent the dependency links. The following (Figure 5.8) figure presents different components of a tree representation.

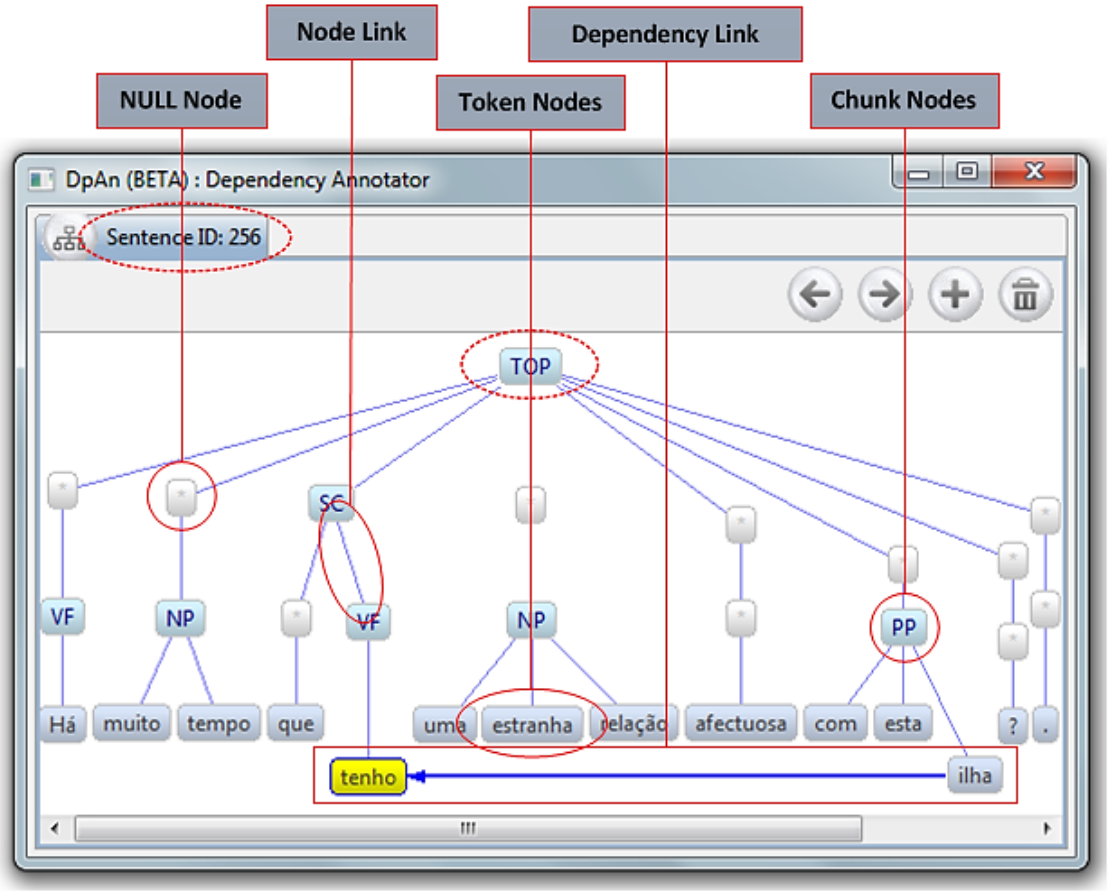

Figure 5 - 8 : DpAn Parse Tree Representation.

The development framework Zest provides the necessary structures to build the graphical interface. The tree representation is adopted for the specific task and it represents the XIP XML phrase tree in the figure. The tree structure can be described by the following elements,

1. On the top-left corner of the window the sentence ID associated with a tree is displayed as can be found in the input file.
2. The trees always initialized with a virtual **"TOP"** node. All the nodes are children to this node.
3. All the edged that connects a parent node to its child are tree nodes (Figure 5.7) and they are different from the dependency links and cannot be selected for deletion.
4. To keep the token nodes in the same level null nodes are inserted using an algorithm developed within the scope of the research. Chunk nodes or null nodes cannot be selected for dependency mapping.
5. Dependency links can connect two token nodes and can be deleted if required.

The tool has been used to produce the test data for the PP attachment disambiguation model evaluation. The tool has been tested by annotator to annotate PP attachment with a fifty sentence test dataset. The five aspects of the tool have been asked to three annotators to evaluate on a scale of zero to five (0-5). The results have been compiled below in Figure 5.8.

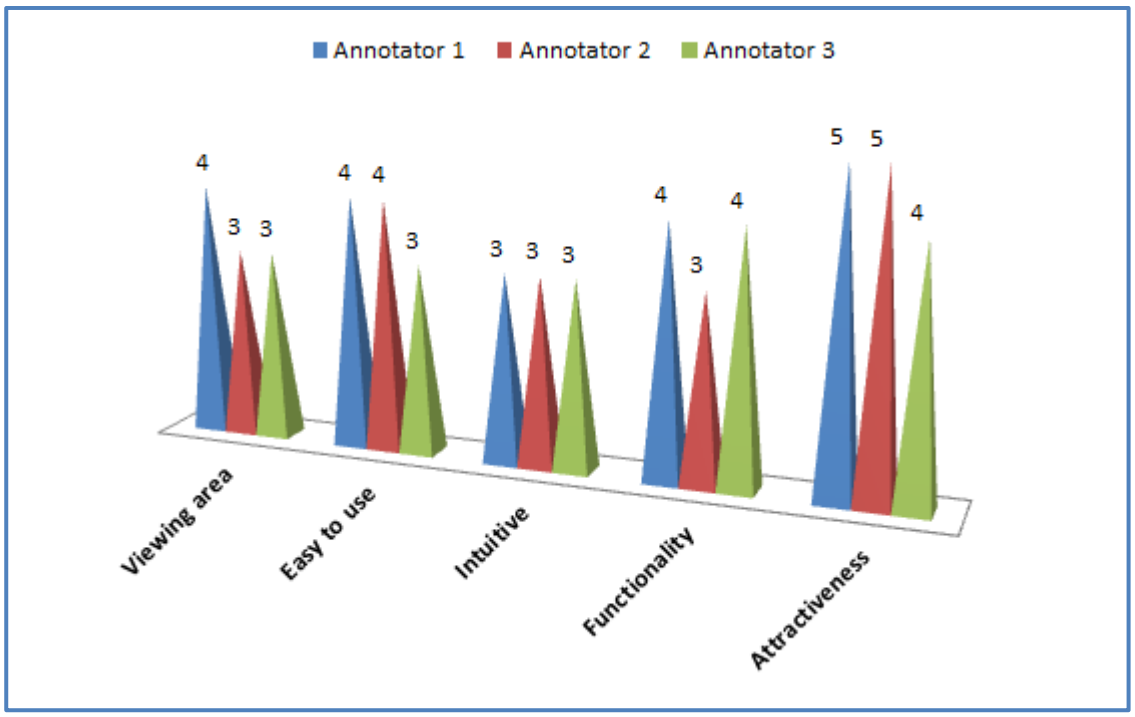

Figure 5 - 9: DpAn response from the annotators.

As it can be seen that the evaluation is not quantitative, rather qualitative from a small number of evaluators yet the significant features to evaluate the system was adopted

from the features presented by Dipper et al. (2004). It can be clearly seen that although the tool is easy to use and functional, the way to use it does not seem to be intuitive to the evaluators. Attractiveness or how a user feels about the look of the tool receives an above average score.

The evaluator's general complaint was about the viewing area. All of them reported that it is easy to follow a sentence, even a very long sentence. Nevertheless the complicated each sentence tree becomes, the more difficult it is to follow the tree. All of them reported word overlapping in some sentences. The layout algorithm is called `GXTreeLayoutAlgorithm` (courtesy of Michelle Tadmor[13] at Eclipse™ open-source community) and the most effective algorithm found so far. Regardless of the difficulty of resolving overlapping nodes the algorithm works fairly well. Since it is an open-source algorithm, in future versions of the tool the algorithm can be modified to perform better with the overlapping issue.

As mentioned before, Zest is a very simple framework for developing applications like **DpAn**. The framework thus does not allow customized edge generation and it makes the dependency links somewhat unmanageable. All the dependency links are rendered under the node row that represent the sentence and makes it difficult for the annotators to understand the start and the end of a link. Using the more advanced framework (i.e. GEF) this problem can be resolved. All the evaluators reported that they could not understand the use of the tool intuitively i.e. even with the guideline, it took some time to get used to the tool.

The input data can be generated using another tool that uses the same data-structure and developed within the scope of the framework. The data generation tool is needed to provide a set of sentence ID's to use to produce a dataset. The sentence IDs themselves can be compiled using the dataset generator on the basis of predefined criterions such as **n%** of the total input or sentences having certain chunk element (PP, NP etc.) present etc. The input for data for the tool is split into two files. One file contains the node configuration of each sentence (i.e. Tree Descriptor) and the other file contains dependency links for the sentence (i.e. Dependency Descriptor). It is a way to split the model from the data and the following figure (Figure 5.9) shows the input files and its representation.

[13] http://www.eclipsezone.com/forums/profile.jspa?userID=264977

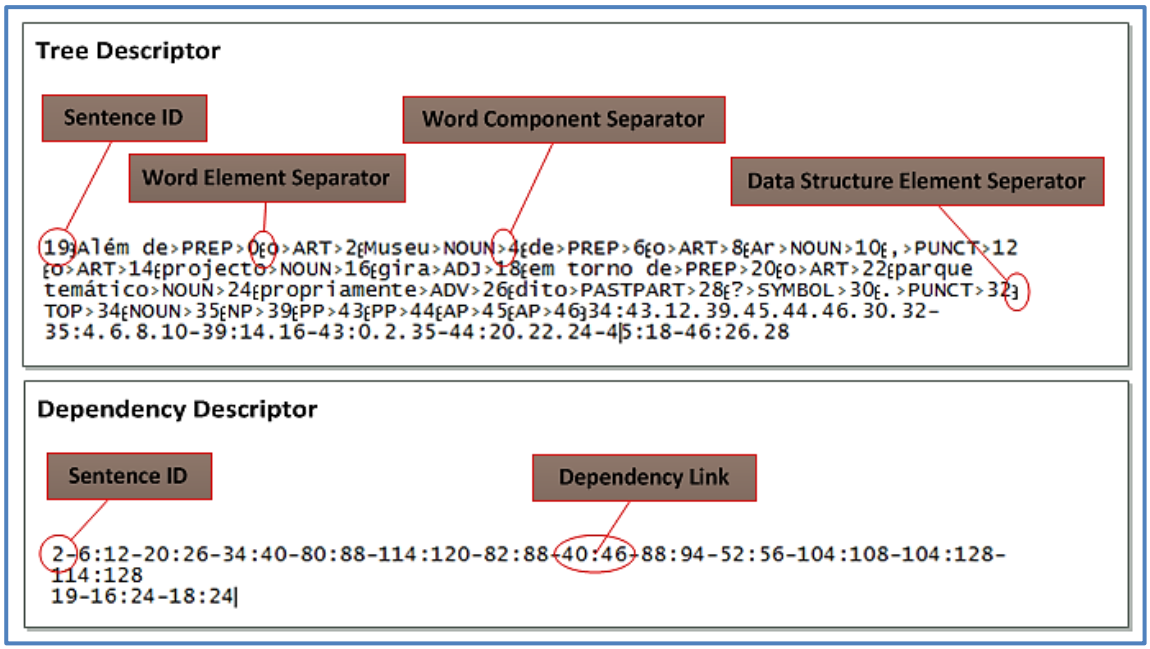

Figure 5 - 10 : DpAn Input File Format.

Each sentence representations have the same format and as it can be seen in Figure 5.9 starts with the sentence ID. Each data-structure elements are split by data-structure element separator (ĝ). The second set of components is the token node descriptors. Each token node is consist of the token, its **P**arts **o**f **S**peech (POS) and the node ID separated by word element separator (›). The third set of components is the chunk descriptors, the name of a chunk and its ID again separated by word element separator (›). The last set is the tree map, list of children node IDs for each chunk node since only chunk nodes can have children. The dependency descriptors represent only one type of dependency and the representation is a list of pairs followed by the sentence ID.

The data representation is basic test format and difficult to standardize. A future modification can be generating the data in XML format. If the size of each dataset can be regulated, a basic **D**ocument **O**bject **M**odule (DOM) parser implementation can be used for data representation and quicker navigation. Moreover a XML based data representation is considered to be a standard. The module to generate this data is implemented within the data-structure implementation. It can be used to generate specific dataset which is a subset of the extracted dataset. For the XML representation this data generation module has to be modified as well. The tool is an important part of the framework and it provides a means to produce gold standard data for both training and testing of dependency based applications.

5.3. PP attachment disambiguation model.

The framework allows the developer to model a phenomenon of interest from the output data of the STRING's XIP XML output. Moreover, the alignment module allows incorporating information from the parsed output of the PALAVRAS parser. To demonstrate the use of the framework this subsection will present the development of a PP attachment disambiguation model. The evaluation of the system will also be presented, although the performance of the model is not the primary objective of the experiments.

Designing a statistical model to extract PP attachment relations was a two-step process. The data analysis presented in section 5.1 is the basis of the primary data selection process. Only PP's modifying one element were considered for the training process. Moreover the definition of POST was enforced using only modifiers with head node ID appear later in the token sequence than governor head node ID. Then that model train itself using the data for the predefined heuristics. For this research two heuristic based model were defined from two different systems i.e. PALAVRAS CG output and the XIP XML output.

5.3.1. Linier Phrase Distance (LPD) Heuristic.

This heuristic was developed on the basis that PP's were found to modify phrases closer to them according to the input data. The distribution observed in the data is presented in the following figure (Figure 5.11).

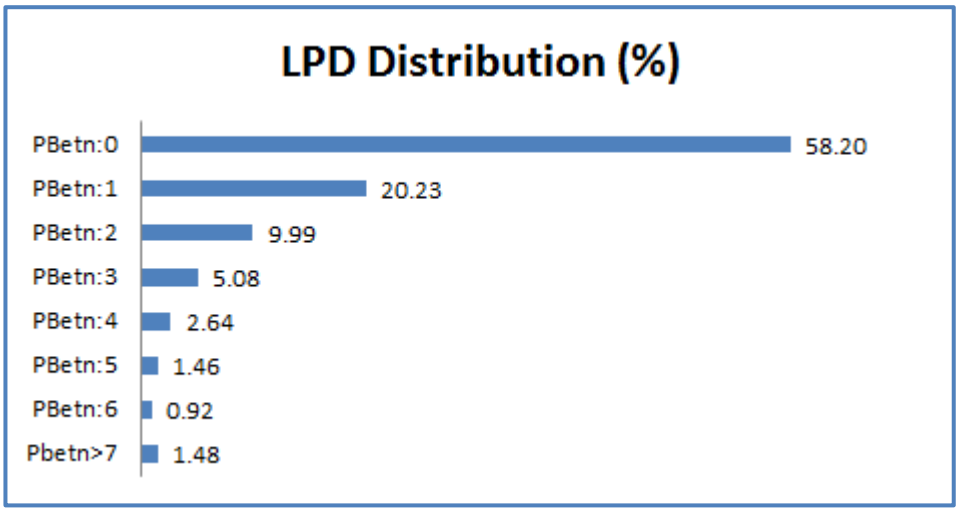

Figure 5 - 11 : LPD distribution observed in the data.

On the basis of the observed distribution the heuristic was designed. This heuristic calculates the joint probability of the number of phrases between the modifier and the governor given the chunk type of the governor. The distribution was computed over the correct modifier data and it can be formulated as follows,

$$\boldsymbol{Pr(C_C, G_{CT}) = Pr(C_C|G_{CT})\,\mathrm{Pr}(G_{CT})} \quad \ldots \textbf{Eqn. 5. 1}$$

In the equation $\boldsymbol{C_C}$ is the number of chunks between modifier and governor node and $\boldsymbol{G_{CT}}$ is the governor's chunk type. The governor's chunk type is computed separately over the data. For each governor type the distance distribution was extracted.

5.3.2. Tree Travers Distance (TTD) Heuristics.

The existence of a path in the CG parse tree from the modifier head to the governor head. If such path exists, the inverse distance factor (i.e. $\boldsymbol{1/d}$ given that $\boldsymbol{d}$ is the distance between the heads) was used. The CG formalism dictates that modifier and governor are more probable to be in close proximity. The XIP data only provides head dependency which actually reflects the relationship between the phrases. So, all the nodes of the phrase were taken under consideration. The aligned node list of all the phrase nodes was taken under consideration while searching for the inverse path distance. After all the computation the maximum inverse path distance was used.

If $\boldsymbol{n}$ modifier nodes and $\boldsymbol{m}$ governor nodes were found in the aligned Bosque nodes the computation time complexity is linear and it is $\boldsymbol{m \times n}$. In our experiments it was found that some of the alignment nodes were missing within a phrase alignment and thus makes the alignment incomplete. This phenomenon actually reduces the time complexity since some of the traversals will be aborted prematurely.

5.3.3. Model evaluation.

We experimented with only two basic models. We attempted the LPD heuristic and the TTD heuristics as individual models. Several linear combinations were then tested, starting with the most basic combination of adding the values. We also tried weighted linear combinations by assigning weights to each of the heuristics.

In a given sentence all the PPs and all possible governor heads that can be modified by each of the PP's are listed. The best governor is then selected using the heuristics. The output data is then used for the evaluation of the model. The test set was originally

produced by the STRING NLP chain. The annotation tool **DpAn** was then used to revise the annotation by a human annotator. All the evaluation even the original machine produced output was performed against the reference human annotation. The evaluation output is listed in Table 5.2.

Model	Recall	Precision	F-Measure
Dependency in STRING NLP Chain (Analyzed)	76.86%	75.23%	76.03%
Dependency in STRING NLP Chain	76.86%	72.27%	74.49%
Linier Phrase Distance (LPD) Heuristic	60.97%	63.73%	62.32%
CG Tree Travers Distance (TTD) Heuristics	14.73%	47.05%	22.44%
LPD+TTD Heuristics	55.29%	57.72%	56.48%

Table 5 - 1: Evaluation Results.

As it can be seen the original data is closer the human annotation. The analyzed data only contains the relations that have been considered consistent. It is notable that the recall for both analyzed and the whole dataset was the same. The LPD heuristic is fundamentally modeled the system itself thus shows somewhat similar performance. The original assumption of incorporating information from CG output namely, CG TTD heuristic proven to be less useful.

The simple model was designed to evaluate the hypothesis of using two systems output to improve PP attachment output of the STRING NLP chain. The simple TTD model was ineffective primarily because of the head based dependency interface in STRING. It was often difficult to have the heads to be traversable. The head defined in STRING is basically the right most token thus, does not reflect the function of a phrasal head. Moreover, the alignment was not totally accurate thus produce a lot of missing tokens in a chain of tokens. Lot of the times the chains were necessary to find proper traversal distance. The lack of proper definition of the phrase structure within the CG frame leads to such poor result. The next chapter will provide an insight on the possible future prospects of the research.

Chapter 6: Future Prospects and Conclusion.

The objectives of this research were mostly achieved, although the models performance was not better than the original output. The data was generated almost entirely in automatic manner. Moreover the annotation tool developed provides a means to incorporate gold standard data in the research. The framework will provide any further research in the direction of dependency based language processing.

The improvement for the annotation tool is a likely future work that can be useful even beyond the scope of this research. Improved models can be developed with more heuristics to disambiguate PP attachment. Better alignment and human annotated gold standard dataset can improve the performance of the models. The analysis tools can be used to analyze larger gold standard data to create improved system to model linguistic phenomenon within the scope of the framework. Moreover, both parses are available in multiple languages thus the study can be conducted for the common languages.

References.

Abney, S. (1996). Partial Parsing via Finite-State Cascades. In Workshop on Robust Parsing. 8th European Summer School in Logic, Language and Information, Prague, Czech Republic. pp. 8-15.

Afonso, S., Eckhard, B., Renato, H. and Diana, S. (2002) Floresta sintá(c)tica: a treebank for Portuguese. In the Proceedings of LREC 2002, the Third International Conference on Language Resources and Evaluation Las Palmas de Gran Canaria, Spain. pp. 1698-1703.

Agresti, A. (1990) *Categorical Data Analysis.* John Wiley & Sons.

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

Al-Onaizan, Y., Curin, J., Jahr, M., Knight, K., Lafferty, J., Melamed, D., Och, F.J., Purdy, D., Smith, N.A. and Yarowsky, D. (1999) Statistical machine translation. Final Report, JHU Summer Workshop

Altmann, G. (1985) The resolution of local syntactic ambiguity by the Human Sentence Processing Mechanism. In Proceedings of the Second Conference on European Chapter of the Association for Computational Linguistics, Geneva, Switzerland. pp. 123-127.

Baptista, J., Mamede, N., Gomes, F. (2010) Auxiliary verbs and verbal chains in european portuguese. In: Computational Processing of the Portuguese Language. Number LNCS/LNAI 6001, Berlin, PROPOR 2010, Springer

Bick, E. (1996), Automatic Parsing of Portuguese. In García, Laura Sánchez (ed.), Anais / II Encontro para o Processamento Computacional de Português Escrito e Falado. Curitiba: CEFET-PR.

Bick, E. (2000) The Parsing System Palavras - Automatic Grammatical Analysis of Portuguese in a Constraint Grammar Framework. Aarhus University Press. Aarhus, Denmark.

Bick, E. (2003) A CG & PSG Hybrid Approach to Automatic Corpus Annotation, in Kiril Simow & Petya Osenova: Proceedings of SProLaC2003 (at Corpus Linguistics 2003, Lancaster), pp. 1-12.

Bick, Eckhard (2005) Turning Constraint Grammar Data into Running Dependency Treebanks. In: Civit, M. & Kübler, S. & Martí, M. A. (red.), Proceedings of TLT 2005, Barcelona, Dec 9th - 10th, 2005), pp.19-27

Bick, E. (2006) Noun sense tagging: Semantic prototype annotation of a portuguese treebank. Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT 2006).

Bick, E. (2007) Automatic semantic role annotation for portuguese. Em TIL, V Workshop em Tecnologia da Informa¸c˜ao e da Linguagem Humana, pp. 1715–1719.

Bick, E. (2009) Introducing Probabilistic Information in Constraint Grammar Parsing. Proceedings of Corpus Linguistics 2009, Liverpool, UK.

Bikel, D. (2002) Design of a multi-lingual, parallel processing statistical parsing engine. In Proceedings of the Human Language Technology Workshop.

Blevins J. P.and Sag I. A. (2011) Phrase Structure Grammar. To be appeared in M. den Dikken (ed.), Cambridge Handbook of Generative Syntax - Chapter 7. Cambridge University Press. 2012.

Blum, A. (2002) Machine Learning Theory. Lecture Notes, Carnegie Mellon University.

Branco, A. and Costa, F. (2008) A computational grammar for deep linguistic processing of Portuguese: LXGram, version A.4.1. Technical Report DI-FCUL-TR-08-17, University of Lisbon. Lisbon, portugal.

Brill, E. (1995) Transformation-based error-driven learning and natural language processing: a case study in part of speech tagging. Computational Linguistics, 21, 543–565.

Brill, E. and Resnik, P. (1994) A Rule-Based Approach To Prepositional Phrase Attachment Disambiguation. In Proceedings of COLING'94. Kyoto. Available at: http://www.cs.jhu.edu/~brill/pp-attachment.ps.

Cahill, A., Heid, U., Rohrer, C. and Weller, M. (2009) Using tri-lexical dependencies in LFG parse disambiguation. In: The 14th International LFG Conference, Trinity College, Cambridge, United Kingdom.

Ceccato, M., Kiyavitskaya, N., Zeni, N., Mich, L., and Berry, M. (2004) Ambiguity identification and measurement in natural language texts. Technical Report, University of Trento. Trento, Italy

Collins, M. and Brooks J. (1995) Prepositional phrase attachment through a backed-off model. In David Yarowsky and Kenneth Church, editors, Proceedings of the Third Workshop on Very Large Corpora, pp. 27-38, Cambridge, MA, June.

Covington, A. (2001) A fundamental algorithm for dependency parsing. Proceedings of the 39th Annual ACM Southeast Conference, pp. 95–102

Crain, S. and Steedman, M. (1985) On not being led up the garden path: the use of context by the psychological syntax processor. In Natural language parsing: Psychological, Computational, and Theoretical Perspectives. Cambridge: Cambridge University Press. pp. 320-358.

Dempster, A.P., Laird, N.M and Rubin, D.B. (1977) Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Statist. Soc. B(39) pp. 1–38.

Diniz, C. (2010) Rudrico2 - um conversor baseado em regras de transformaçao declarativas. Master's thesis, Instituto Superior Téecnico, Universidade Téecnica de Lisboa.

Diniz, C., Mamede, N.(2011) Lexman - lexical morphological analyser. Technical report, L2F / INESC ID Lisboa, Lisboa, Portugal.

Duda, R., Hart, P. and Stork, D. (2000) Pattern Classification (2nd Edition). Wiley-Interscience.

Eysenck, M. and Keane, M. (2000) Cognitive psychology a student's handbook 4th ed., Hove: Psychology Press.

Farley, B. and Clark, W. (1954) Simulation of self-organizing systems by digital computer. IEEE Transactions on Information Theory, 4(4), pp. 76-84.

Fellbaum, C. and Miller, G. (1990) Folk psychology or semantic entailment? A reply to Rips and Conrad. Psychological Review 97. pp. 565–570.

Ford, M., Bresnan, J., and Kaplan, M. (1982) A competence-based theory of syntactic closure, in Joan Bresnan, ed., The Mental Representation of Grammatical Relations , MIT Press, Cambridge, MA , pp. 727-796.

Foth, K. and Menzel, W. (2006) The benefit of stochastic PP attachment to a rule-based parser. In Proceedings of the COLING/ACL on Main Conference Poster Sessions (COLING-ACL '06). Association for Computational Linguistics, Stroudsburg, PA, USA. pp.223-230

Frazier, L. (1978) On comprehending sentence: syntactic parsing strategies. PhD Thesis. ETD Collection for University of Connecticut. USA.

Gamallo, P., Agustini, A. and Lopes, G. (2003a) Acquiring Semantic Classes to Elaborate Attachment Heuristics. In Moura Pires & S. Abreu (eds.), Progress in Artificial Intelligence 11th Portuguese Conference on Artificial Intelligence, EPIA 2003, Beja, Portugal Springer-Verlag. Lecture Notes in Artificial Intelligence, pp. 478-487.

Gamallo, P., Agustini, A. and Lopes, G. (2003b) Learning subcategorisation information to model a grammar with co-restrictions. Traitement Automatic de la Langue, 44(1). pp. 93–117.

Goldberg, A. (1995) Constructions; A construction grammar approach to argument structure. University of Chicago Press. Chicago, USA.

Hindle, D. and Rooth, M. (1993) Structural ambiguity and lexical relations. Computational Linguistics, 19(1):103–120.

Hirst, G. (1987) Semantic Interpretation and the Resolution of Ambiguity. Cambridge University Press, New York, NY, USA.

Hsieh, Y., Yang, D. and Chen, K. (2007) Improve Parsing Performance by Self-Learning. Computational Linguistics and Chinese Language Processing. 12(2) pp. 195-216.

Hudson, R. (1984) Word grammar. Oxford, England: B. Blackwell.

Huyck, C. (2000) A practical system for human-like parsing. In the Proceedings of the 14th European Conference on Artificial Intelligence, ECAI 2000, Berlin, pp. 436-440.

Karlsson, F. (1990) Constraint grammar as a framework for parsing running text. In Proceedings of the 13th Conference on Computational Linguistics. Helsinki, Finland. 168-173.

Karlsson, F., Voutilainen, A., Heikkilä, J. and Anttila,A. (eds.) (1995) Constraint Grammar - A Language-Independent System for Parsing Unrestricted Text. Natural Language Processing, No 4. Berlin & New York: Mouton de Gruyter.

Kimball, J. (1973) Seven principles of surface structure parsing in natural language. Cognition, 2(1) pp. 15-47.

Klein, D. and Manning, C. (2003) Fast exact inference with a factored model for NLP. Advances in Neural Language Processing Systems 15. pp. 3–10.

Kübler, S., McDonald, R. and Nivre, J. (2009). Dependency Parsing. Synthesis Lectures on Human Language Technologies. Morgan and Claypool Publishers.

Lin, D. (1997) Using syntactic dependency as local context to resolve word sense ambiguity. In the Proceedings of the Eighth Conference on European Chapter of the Association for Computational Linguistic (ACL). Madrid, Spain. pp. 64-71.

Loftsson, H. and Rögnvaldsson, E. (2007). IceParser: An Incremental Finite-State Parser forIcelandic. In Proceedings of the 16th Nordic Conference of Computational Linguistics (NoDaLiDa, 2007), Tartu, Estonia.

Lopez, A. (2007) Statistical machine translation. ACM Computing Surveys. Earlier version: A Survey of Statistical Machine Translation. U. of Maryland, UMIACS tech. Report pp. 2006-47.

Mamede, N. (2011) STRING - A Cadeia de Processamento de Língua Natural do L2F. Slides from seminar at NILC/ICMC/USP, São Carlos, Brasil.

Marr, D. (1970) A theory for cerebral neocortex. In the Proceedings of the Royal Society of London, Series B, Biological Sciences, 176(1043). pp. 161-234. The Royal Society.

Megyesi, B. and Rydin, S. (1999). Towards a Finite-State Parser for Swedish. In Proceedings of NoDaLiDa, Throndheim, Norway.

Mel'čuk, I. (1988) Dependency syntax : theory and practice. State University of New York Press. Albany, N.Y. USA.

Merlo, P. and Ferrer, E. (2006) The notion of argument in PP attachment. Computational Linguistics, 32(2) pp. 341–378

Miller, G. A. et al. (1990) WordNet: An on-line lexical database. International Journal of Lexicography, 3, pp.235–244.

Mohanty, R., Ashish F. and Pushpak B. (2005) Prepositional Phrase Attachment and Interlingua. In J. Cardenosa, A. Gelbukh & E. Tovar, (eds.) Universal Networking Language: Advances in Theory and Application. Special Issue of Research on Computing Science, pp. 242-253, Instituto Politécnico Nacional, Mexico.

Nadh, K. and Huyck, C. (2009) Prepositional phrase attachment ambiguity resolution using semantic hierarchies. In Ninth IASTED International Conference on Artificial Intelligence and Applications, 18th February 2009, Innsbruck, Austria.

Nagao, K. (1990) Dependency Analyzer: A Knowledge-Based Approach to Structural Disambiguation. In the Proceedings of 13th International Conference on Computational Linguistics, Coling-90, 282-287.

Nakov, P. and Hearst, M. (2005) Using the web as an implicit training set: Application to structural ambiguity resolution. In Proceedings of HLT-NAACL.

Niemann, M. (1998) Determining PP Attachment through Semantic Associations and Preferences. In D. Estival (ed.) Abstracts for the ANLP Post Graduate Workshop, pp. 25-32.

Nilsson, N. J. (1998) Introduction to Machine Learning: An Early Draft of a Proposed Textbook. [Accessed August 11, 2011] Available at: http://robotics.stanford.edu/people/nilsson/mlbook.html.

Nivre, J. (2002) Two Models of Stochastic Dependency Grammar. MSI Report 02118. Växjö University: School of Mathematics and Systems Engineering.

Nivre, J. (2005) Dependency grammar and dependency parsing. Technical Report MSI 05133, Växjö University: School of Mathematics and Systems Engineering.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S. and Marsi, E. (2007) MaltParser: A language-independent system for data-driven dependency parsing. Natural Language Engineering, 13(2), pp. 95-135.

Nuria, G. & Lafourcade, M. (2005) Combining corpus-based pattern distributions with lexical signatures for PP attachment ambiguity resolution. In Proceedings of the 6th Symposium on Natural Language Processing (SNLP-05), Chiang Rai. Chaina.

Och, F. J. (2000) Giza++: Training of statistical translation models. Available at http://www-i6.informatik.rwthaachen.de/~och/software/GIZA+.html.

Och, F.J. and Ney, H. (2000) A comparison of alignment models for statistical machine translation. In COLING'00: The 18th International Conference on Computational Linguistics, pp. 1086–1090, Saarbrücken, Germany.

Och, F.J. and Ney, H. (2002) Discriminative training and maximum entropy models for statistical machine translation. In Proc. of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia, PA, July.

Och, F.J. and Ney, H. (2003) A Systematic Comparison of Various Statistical Alignment Models. Computational Linguistics 29(1) pp. 19-51.

Petrov, S., Barrett, L., Thibaux, R. and Klein, D. (2006) Learning accurate, compact, and interpretable tree annotation. In: Proceedings of the 44th ACL. pp. 433–440.

Rappaport, M. (1983) On the nature of derived nominals. In Papers in lexical-functional grammar. Bloomington, Ind.: Indiana University Linguistics Club, pp. 113-142.

Ratnaparkhi, A., Reynar, J. and Roukos, S. (1994) A maximum entropy model for prepositional phrase attachment. In Proceedings of the workshop on Human Language Technology - HLT '94. Plainsboro, NJ, pp. 250-255.

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

Roth, D. (1998) Learning to resolve natural language ambiguities: a unified approach. In the Proceedings of AAAI-98, 15th Conference of the American Association for Artificial Intelligence, 806-813, July 1998, Madison, Wisconsin, United States.

Sándor, A., Kaplan, A. and Rondeau, G. (2006) Discourse and citation analysis with conceptmatching. International Symposium: Discourse and document (ISDD), Caen, France.

Schiehlen, M. (2003) A Cascaded Finite-State Parser for German. In roceedings of the Research Note Sessions of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL'03), Budapest.

Scudder, H. (1965) Probability of error of some adaptive pattern-recognition machines. IEEE Transactions on Information Theory, 11(3). pp. 363-371.

Silva, J., Branco, A., Castro, S. and Reis, R. (2010) Out-of-the-Box Robust Parsing of Portuguese. In Lecture Notes in Artificial Intelligence, 6001, pp. 86-89, Berlin: Springer.

Sopena, J. M., LLoberas, A. and Moliner, J. L. (1998) A connectionist approach to prepositional phrase attachment for real world texts. In Proceding 17th International Conference on Computational Linguistics, pp. 1233–1237,Montreal.

Stetina, J. and Nagao, M. (1997) Corpus based pp attachment ambiguity resolution with a semantic dictionary. In In Proceedings of WVLC, pp. 66–80.

Tapanainen, P. (1996) The Constraint Grammar Parser CG-2. Number 27 in Publications of the Department of General Linguistics, University of Helsinki.

Taraban, R. and McClelland, J. (1988) Constituent attachment and thematic role assignment in sentence processing: Influences of content-based expectations. Journal of Memory and Language, 27(6) pp. 597-632. Available at: http://www-psych.stanford.edu/~jlm/papers/PublicationFiles/80-89_ Add_ To_ONLINE_Pubs/T arabanMcClelland88ConstituentAttachment.pdf.

Telljohann, H., Hinrichs, E. W., Kübler, S. and Zinsmeiste, H. (2005) Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z).

Tesnière, L. (1959) Éléments de syntaxe structurale, Klincksieck, Paris.

Toutanova, k., Andrew Ng Y. and Manning, C. (2004) Learning random walk models for inducing word dependency distributions. In In Proceedings of ICML.

Volk M. (2001) The Automatic Resolution of Prepositional Phrase – Attachment Ambiguities in German. Habilitation thesis submitted to the University of Zurich, Faculty of Arts. Zurich

Walt, C. and Barnard, E. (2006) Data characteristics that determine classifier performance. In the Proceedings of the Sixteenth Annual Symposium of the Pattern Recognition Association of South Africa, Parys, South Africa. 166–171. Available at: http://www.meraka.org.za/pubs/CvdWalt.pdf.

Whittemore, G., Ferrara, K. and Brunner, H. (1990) Empirical study of predictive powers of simple attachment schemes for post-modifier prepositional phrases. In Proceedings of the 28th Annual Meeting of the Association for Computational Linguistics, Pittsburgh, Pennsylvania. pp. 23-30.

Wilks, Y., Huang, X. and Fass, D. (1985) Syntax, preference and right attachment. Syntax, preference, and right attachment. In Proceedings of the 9th international joint conference on Artificial intelligence - Volume 2 (IJCAI'85), Aravind Joshi

(Ed.), Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 779-784.

Wing, B. and Baldridge, J. (2006) Adaptation of data and models for probabilistic parsing of Portuguese. In: Proceedings of the 7th Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR). pp. 140–149.

XRCE (2011) Xerox Incremental Parser - Reference Guide [online]. France: Xerox research Center Europe (XRCE). [Accessed 14 May 2012].Available at: http://open.xerox.com/Repo/service/XIPParser/public/XIPReferenceGuide.pdf.

Zhao, S. and Lin, D. (2004) A Nearest-Neighbor Method for Resolving PP-attachment Ambiguity. In Proceedings of IJCNLP-04.