# 3
# Portuguese Chunk Definitions and an Extraction Heuristic

The previous chapter demonstrated how the text chunking problem has been approached for the English language, and how a chunk definition has been established based on a corpus annotated with full parsing information. One of our goals is to follow similar steps in order to investigate the impact of text chunking in the Portuguese language, and, to achieve that, we need to make use of a corpus in the same conditions.

In this chapter, we describe how we employ the *Bosque* corpus, from the *Floresta Sintá(c)tica* project, for that purpose. We also propose a heuristic that takes the sentences' phrases annotated in this corpus to generate a new chunk annotation for it. The possibility to choose between subsets of available types of phrases in the corpus allows us to investigate different chunk definitions, whose component chunk types correspond precisely to the selected phrase types.

Our approach, however, has only computational aspirations. Establishing a "correct" definition for chunks in a linguistic sense is out of the scope of this work. Therefore, the way we directly evaluate the usefulness of our extracted chunks is by applying them as input data to other NLP tasks. The two other tasks selected for this purpose are clause identification and dependency parsing. We feed Machine Learning-based systems targeted at those tasks with a chunk feature containing tags derived by our heuristic, and evaluate its impact on their performance.

## 3.1
## The Bosque corpus

The *Floresta Sinctá(c)tica* project is a publicly available treebank for Portuguese, created in 2000 as a collaboration between Linguateca[1] and the VISL Project[2] (28). It consists of texts written in both European and Brazilian Portuguese, annotated automatically by the parser PALAVRAS (7).

*Floresta Sinctá(c)tica* is composed by different sets of corpora, each having distinct sources. A few of these sets have received some degree of linguistic revision in order to correct errors from the parser. The *Bosque* set is

[1] `http://www.linguateca.pt`
[2] `http://visl.sdu.dk`

the only one that received thorough revision, and for that reason we use this corpus in our experiments.

*Bosque* is composed by two corpora, each of them being a subset of bigger corpora derived from newspaper articles. One of these bigger corpora is the *CETENFolha*, composed of Brazilian Portuguese text from the *Folha de São Paulo* newspaper. The other is *CETENPúblico*, in European Portuguese, extracted from the *PÚBLICO* newspaper. Table 3.1 demonstrates some statistics about the *Bosque* corpus.

| | |
|---|---|
| Number of tokens | 226,758 |
| Number of sentences | 9,368 |
| Average of tokens per sentence | 24.2 |
| Number of noun phrases (NP) | 61,537 |
| Number of verb phrases (VP) | 21,747 |
| Number of prepositional phrases (PP) | 32,949 |
| Number of adjectival phrases (ADJP) | 9,608 |
| Number of adverbial phrases (ADVP) | 6,505 |

Table 3.1: Statistics of the *Bosque* corpus

*Bosque* is available in many formats. One of them is the traditional Penn Treebank format, and another is a format called *Árvores Deitadas* (AD). We choose to work with the latter, extracting the data needed from its structure. Technical description for it is available on *Bíblia Florestal*, a comprehensive manual for the *Floresta Sintá(c)tica* project (2).

Figure 3.1 provides an example of a sentence in this format. The number of "=" signs preceding a constituent denotes its level in the tree, and the parent of a constituent is the closest preceding node at the level immediately above. Each AD node also contains information about clause and phrase structure and POS tags.

## 3.2 Encoding chunks in tags

We have described the chunking problem as approached by Ramshaw and Marcus in the previous chapter, mentioning their novel concept of tackling it as a classification problem. The idea is that each token receives a specific chunk tag, and a learning model tries to predict this tag based on the other known features of the present token and the context around it.

In order for chunks to be encoded in tags, one per token, these tags need to provide more information than just the type of chunk their corresponding tokens are in. For example, if two adjacent tokens were also inside adjacent NP chunks and their tags only provided a value such as "NP", simply indicating

```
STA:fcl
=SUBJ:np
==H:pron-pers('ele' M 3S NOM) Ele
=ADVL:advp
==H:adv('só') só
=ADVL:advp
==H:adv('não') não
=P:vp
==MV:v-fin('jogar' IMPF 3S IND) jogava
=ADVL:fcl
==SUB:conj-s('porque') porque
==ADVL:advp
===H:adv('não') não
==P:vp
===MV:v-fin('estar' <fs-cause> <nosubj> IMPF 3S IND) estava
==SA:advp
===H:adv('bem' <quant>) bem
=.
```

Figure 3.1: A sentence in the *Árvores Deitadas* format

the type of their corresponding chunks, we would never be able to tell that they are not inside the same NP chunk. As proposed by Ramshaw and Marcus (51), the typical way of encoding chunks in tags is using a format called IOB.

The original IOB format, also called IOB1, has been established for the classification of noun phrases in particular. It specifies the following rules: tokens inside a chunk are tagged as `I`, tokens outside chunks are marked as `O`, and tokens starting a chunk immediately following another chunk are tagged as `B`, in order to make a distinction between these two chunks. This style can be extended for the identification of multiple types of chunk: each token receives an `X-YY` tag, where `X` is either `I`, `O` or `B` as defined above, and `YY` denotes the chunk type — NP, for instance. In this case, `B` is only used if a token is at the boundary of consecutive chunks of the same type.

IOB2 is a similar format proposed afterwards (52). Its only difference compared to IOB1 is in the `B` tag: all tokens that start a chunk are tagged as `B`. This format has been the one chosen for the CoNLL-2000 shared task. Other two formats have been used, though not as frequently: IOE1 and IOE2 (63). These are directly related to IOB1 and IOB2, respectively. However, instead of defining a `B` tag for tokens that start a chunk, they establish an `E` tag for tokens ending a chunk. Figure 3.2 shows a sentence in Portuguese split into chunks and with its contractions between prepositions and articles expanded, and table 3.2 illustrates those four tagging styles being applied to the same sentence.

$[_{NP}$ O futebol $]$ $[_{VP}$ precisa $]$ $[_{VP}$ seguir $]$
$[_{NP}$ o exemplo $]$ $[_{PP}$ de $]$ $[_{NP}$ a CPI $]$
$[_{PP}$ de $]$ $[_{NP}$ o orçamento $]$ .

Figure 3.2: A sentence in Portuguese divided into chunks

| | | **Chunks** | | | |
|---|---|---|---|---|---|
| **Word** | **POS** | **IOB1** | **IOB2** | **IOE1** | **IOE2** |
| O | art | I-NP | B-NP | I-NP | I-NP |
| futebol | n | I-NP | I-NP | I-NP | E-NP |
| precisa | v-fin | I-VP | B-VP | E-VP | E-VP |
| seguir | v-inf | B-VP | B-VP | I-VP | E-VP |
| o | art | I-NP | B-NP | I-NP | I-NP |
| exemplo | n | I-NP | I-NP | I-NP | E-NP |
| de | prp | I-PP | B-PP | I-PP | E-PP |
| a | art | I-NP | B-NP | I-NP | I-NP |
| CPI | n | I-NP | I-NP | I-NP | E-NP |
| de | prp | I-PP | B-PP | I-PP | E-PP |
| o | art | I-NP | B-NP | I-NP | I-NP |
| orçamento | n | I-NP | I-NP | I-NP | E-NP |
| . | . | O | O | O | O |

Table 3.2: Different tagging styles for chunks

Following the decision for the shared task and considering its frequent usage, the present work uses the IOB2 format in its experiments.

## 3.3 Chunk derivation heuristic

Our chunk derivation heuristic for Portuguese is similar to the general method used for the CoNLL-2000 shared task, albeit simplified. It is based on the phrase structure contained in the parse tree of sentences, hence the necessity of previous full parsing for the corpus used.

By our proposed method, a chunk is composed by all consecutive tokens within the same deepest-level phrase node. This means that, since all tokens are terminal nodes in the tree, a given token is within the chunk derived from its closest phrase ancestor. Naturally, a chunk inherits its type from its generating phrase.

We provide a depiction of the output of the heuristic for a given sentence in figure 3.3. The nodes shown in the figure roughly correspond to an expected phrase structure in the *Bosque* corpus. In this figure, for example, the higher-level prepositional phrase "*de a escola*" spawns two distinct chunks: the PP chunk "*de*", since the deepest-level phrase node that includes this token is prepositional, and the NP chunk "*a escola*", since these tokens are included

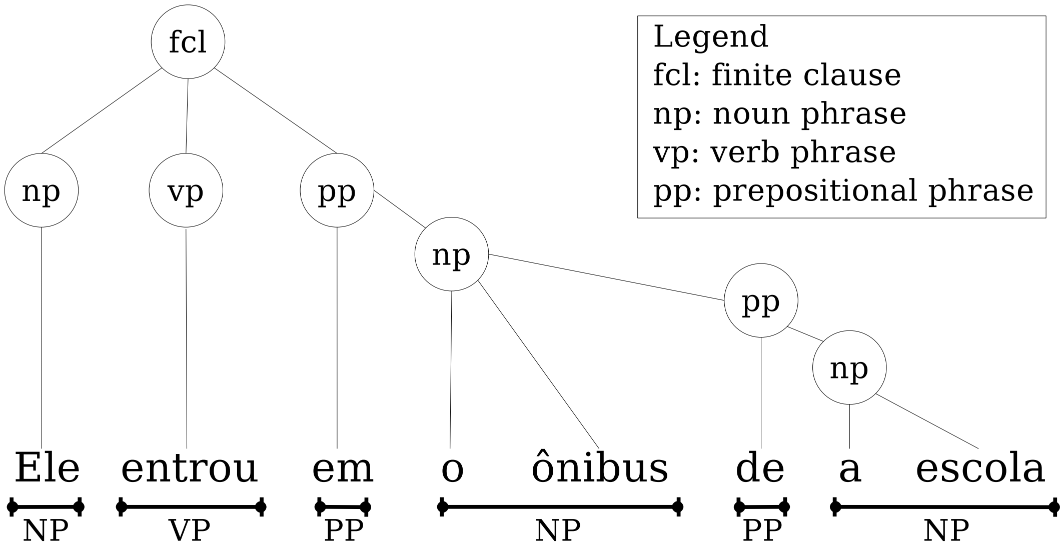

Figure 3.3: Output of the chunk derivation heuristic for a given sentence

by a deepest-level noun phrase.

## 3.4 Chunk definitions

Figure 3.3 shows examples of noun, verb and prepositional phrases, and their chunk counterparts. It also depicts a situation where all kinds of phrases present in the sentence are considered, i.e., no phrase type is discarded when generating chunks.

Since our goal with chunk extraction is to improve the effectiveness of other systems focused on more elaborate problems, it makes sense to investigate how the syntactic information associated with each phrase type affects these other systems in terms of performance. To evaluate that, we establish the notion of *chunk definitions*.

A chunk definition is a subset of chunk types resulting from a group of specifically selected phrase kinds. The phrase kinds not selected for a given chunk definition are ignored by the heuristic when it is determining the chunks corresponding to this definition for a sentence.

Table 3.3 elucidates this concept. It displays two sequences of IOB2 chunk tags derived from the sentence shown in figure 3.3, resulting from two definitions: $(NP, VP)$, which only considers noun phrases and verb phrases, and $(NP, VP, PP)$, which also takes prepositional phrases into account. Notice that the former definition ignores the prepositional phrases displayed in figure 3.3, and therefore no PP chunk is generated. Also, ignoring the second prepositional phrase causes the preposition "*de*" to be included in an NP chunk, since its corresponding noun phrase becomes the closest phrase ancestor of the preposition.

| | | **Chunk definitions** | |
|---|---|---|---|
| **Word** | **POS** | **(NP, VP)** | **(NP, VP, PP)** |
| Ele | pron-pers | B-NP | B-NP |
| entrou | v-fin | B-VP | B-VP |
| em | prp | O | B-PP |
| o | art | B-NP | B-NP |
| ônibus | n | I-NP | I-NP |
| de | prp | I-NP | B-PP |
| a | art | B-NP | B-NP |
| escola | n | I-NP | I-NP |
| . | . | O | O |

Table 3.3: Chunk sequences derived from two distinct definitions

*Bosque*'s annotation for full parsing presents five main types of phrases: noun phrases (NP), verb phrases (VP), prepositional phrases (PP), adjectival phrases (ADJP) and adverbial phrases (ADVP). Following previous studies for the English language mentioned in chapter 2, we derive the following three chunk definitions to apply to other Machine Learning systems:

1. $(NP, VP)$
2. $(NP, VP, PP)$
3. $(NP, VP, PP, ADJP, ADVP)$

In table 3.4, we show statistics concerning how many chunks of each type are derived from *Bosque* for each chunk definition.

| **Chunk definition** | **NP** | **VP** | **PP** | **ADJP** | **ADVP** |
|---|---|---|---|---|---|
| (NP, VP) | 74,104 | 21,236 | — | — | — |
| (NP, VP, PP) | 68,166 | 21,232 | 34,321 | — | — |
| (NP, VP, PP, ADJP, ADVP) | 68,536 | 21,235 | 34,129 | 9,586 | 6,440 |

Table 3.4: Quantity of chunks derived using different chunk definitions

## 3.5
## Tested problems: clause identification and dependency parsing

As previously mentioned, the two other systems to which we provide our derived chunks as a feature solve the clause identification and dependency parsing problems. Both models are based on the ETL algorithm, which we describe in chapter 4.

Clause identification is a task consisting in splitting a sentence into clauses (62). A clause is defined as a word sequence that contains a subject and

( Ninguém percebe ( que ele quer ( impor sua presença ) ) . )

Figure 3.4: The component clauses of a sentence in Portuguese

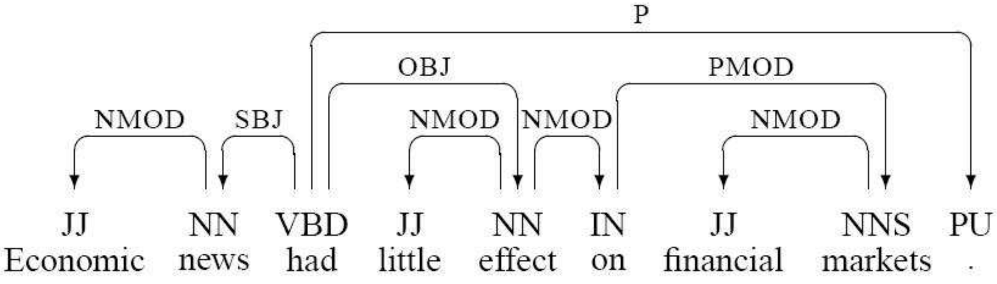

Figure 3.5: Example of a dependency graph

a predicate. This problem is regarded as more elaborate than text chunking because clauses allow recursiveness, and thus they may contain other embedded clauses. The output of this problem is also useful as input data for problems like full parsing and semantic role labeling.

We use the system for clause identification described by Fernandes et al. (23). A sentence divided into its composing clauses is shown in figure 3.4.

The second problem, dependency parsing, consists in the identification of a syntactic head word for each word in an input sequence, making its output a rooted tree where the nodes are the words in the sentence. Problems like question answering, machine translation and information extraction benefit from its results.

For this problem, we apply the model described by Crestana (17). Figure 3.5 displays a dependency graph corresponding to what the outcome for this problem should look like.

Our method for the empirical evaluation of chunk definitions consists in using a fixed configuration for the systems tackling each of the mentioned problems. Each system uses a specific set of features during their training and extraction phases. The only different characteristic between the systems' runs for each definition is the chunk feature used, which contains the tags derived by our heuristic corresponding to that particular chunk definition. We also do one run for each system without using a chunk feature in order to verify the absolute contribution of the proposed definitions. Then, after the extraction phase for each system is executed using a test corpus, we apply the evaluation metric used for the corresponding problem.

Clause identification and dependency parsing have different evaluation metrics. For the former, $F_{\beta=1}$ is traditionally used. The latter has three common metrics, as reported by Crestana (17): labeled attachment score (LAS), unlabeled attachment score (UAS) and label accuracy (LA). In this work, following Crestana's method, we choose UAS as our evaluation metric for dependency parsing. We expand on these metrics in chapter 5, where we also provide the results for the chunk definition experiments.