SuSAna: Shallow Parsing with Preferences

Fernando M. Batista and Nuno J. Mamede

1 L²F - INESC-ID Lisboa/ISCTE
2 L²F - INESC-ID Lisboa/IST
Spoken Language Systems Lab
Rua Alves Redol 9, 1000-029 Lisboa, Portugal
(Fernando.Batista,Nuno.Mamede)@inesc-id.pt

Abstract. This paper presents a shallow parsing module – SuSAna – that performs efficient analysis over unrestricted text. The module recognizes, not only the boundaries, but also the internal structure and syntactic category of syntactic constituents. In addition to the definition of syntactic structures, the grammar supports a hierarchy of symbols and a set of restrictions known as preferences. During analysis, a directed graph is used for representing all the operations, preventing redundant computation. The algorithm has \(O(n^2)\) complexity, where \(n\) is the number of lexical units in the segment.

1 Introduction

The syntactic analysis of a corpus returns information otherwise hidden, allowing the development of more powerful and complex applications. The syntactic processing of corpora may be applied to areas such as information retrieval, information extraction, speech synthesis and recognition [1] and automatic translation. Syntactic analysis is also frequently the starting point for semantic processing systems [2].

The shallow parsing module, SuSAna (Surface Syntactic Analyzer) performs efficient surface analysis over unrestricted text. The development of the module is based on the work of Hagège [3], and recognizes the internal structure and the syntactic category of syntactic constituents, in a given text segment. The grammar supports a hierarchy of symbols, in addition to the definition of syntactic structures, and a set of restrictions known as preferences [4]. During the analysis, a directed graph is used for representing all the operations, preventing redundant computation. The algorithm has \(O(n^2)\) complexity, where \(n\) is the number of lexical units in the segment [5]. SuSAna can be used as a standalone application, fully integrated in a larger system for natural language processing, or in a client/server platform.

The next section describes the linguistic information structure required by SuSAna, and sect. 3 presents the main algorithm and internal organization of the module. Sect. 4 describes the parameterization of the module. Evaluation results are presented in sect. 5. Finally, the paper presents concluding remarks and topics concerning future work.
2 The Knowledge Base

The structures SuSAna identifies, known as models, are defined from a set of properties. In the scope of the analysis, morphosyntactic categories are also viewed as models, thus the concepts terminal model and non-terminal model are used to distinguish categories from models.

The grammar structure defined for SuSAna has been adapted and improved from the grammar used by the shallow parsing prototype AF [3]. This grammar uses three different structures for representing lexical information: block – structures for defining the behavior of models inside other models; preferences – for choosing between different interpretations, according to confidence levels; and a symbol hierarchy, for definition of classes and subclasses of models, leading to a clear and reduced number of rules. Figure 1 shows, the grammar’s DTD.

```
<TXT
<!ELEMENT LangSpec (topmodel, (superclass|block|preference)*)>
<!ELEMENT topmodel EMPTY>
<!ATTLIST topmodel name CDATA #REQUIRED>
<!ELEMENT superclass (subclass)>
<!ATTLIST superclass name CDATA #REQUIRED>
<!ELEMENT subclass EMPTY>
<!ATTLIST subclass name CDATA #REQUIRED>
<!ELEMENT block (textmod)> 
<!ATTLIST block same CDATA #REQUIRED
sup CDATA #REQUIRED
start (0|1|2) #REQUIRED
end (0|1|2) #REQUIRED
>
<!ELEMENT textmod EMPTY>
<!ATTLIST textmod name CDATA #REQUIRED>
<!ELEMENT preference EMPTY>
<!ATTLIST preference
premod CDATA #REQUIRED
discmod CDATA #REQUIRED
supmod CDATA #IMPLIED
premod CDATA #IMPLIED
wrd CDATA #IMPLIED
cat CDATA #IMPLIED
confidence CDATA #IMPLIED
>
```

Fig. 1. Document Type Definition of the grammar.

The process of describing the properties of a language, suitable for SuSAna, becomes simple, given the reduced number of grammar elements, which are described in the following subsections.

2.1 The Starting Model

The starting model, also known as top model, corresponds to the linguistic structure that the user wants to analyze, and is defined by the element topmodel. This element is used only once in the whole grammar and has only an attribute which
corresponds to the name of the model. If a given grammar was written for analyzing sentences, the value sentence could be used as the starting model; address for analyzing addresses; paragraph for analyzing paragraphs. This element may be changed during algorithm execution in order to analyze any other linguistic structure supported by the grammar.

Example 1. Defining the model ph as the starting model.

```xml
<topmodel name="ph"/>
```

2.2 Model Behavior

Except for the starting model, every model must occur inside another model. Therefore, the linguistic description of a language is described in terms of the behavior of models inside other models. The element block is used for this purpose, describing the behavior of a model name inside of a model sup. The attributes start and end expresses restrictions on the occurrence, and may assume one of the values: 0 (never), 1 (always) or 2 (sometimes). For example, start=1 means that the model name must occur at the beginning of sup.

Example 2. This example describes the occurrence of the terminal model arti_s (article def. sing.) inside the model mpp_n (nuclear propositional model).

```xml
<block name="arti_s" sup="mpp_n" start="0" end="0">
  <nextmod name="nc"/>
  <nextmod name="nadj"/>
  <nextmod name="inconnu"/>
</block>
```

The previous example shows that arti_s may not start nor end mpp_n, and also dictates all possible following models. Any model written in up-case chars is considered as a variable, that can take the value of any model. The example 3 defines the occurrence of category coord inside any model, using variables.

Example 3. The following XML block shows that coord neither can start nor end any other model, and can be followed by any other model inside that model.

```xml
<block name="coord" sup="X" start="0" end="0">
  <nextmod name="Y"/>
</block>
```

2.3 Symbol Hierarchy

Hierarchical relations can be defined between models, through the superclass element. The resulting relations can be expressed using a directed graph as shown in figure 2, which establishes hierarchical relations between noun models.
Example 4. Describing classes and subclasses. The example shows that nc (common noun), npr (proper noun) e adj (noun/adjunctive) are subclasses of the generic model n (noun); and nc is a superclass for nc_s (common noun singular), nc_p (common noun plural).

```xml
<superclass name="n">
    <subclass name="nc"/>
    <subclass name="npr"/>
    <subclass name="adj"/>
</superclass>
<superclass name="nc">
    <subclass name="nc_s"/>
    <subclass name="nc_p"/>
</superclass>
```

This mechanism allows for writing specific rules and generic rules, which can be used by default. For example, it is possible to write a block structure for the generic model nc, which will be then used for nc_s, whenever no block exists for such model. An alternative way of using sets of models without variables, consists on using the symbol hierarchy, as shown in the example 5, for that purpose.

Example 5. Supposing that the model all was defined as a generic model for all the other models, the following example could be used to describe the behavior of coord, as a replacement for the declaration of example 3.

```xml
<block name="coord" sup="all" start="0" end="0">
    <nextmod name="all"/>
</block>
```
2.4 Preferences

This mechanism is a way of selecting the most promising results based on previous corpora observations. Each preference establishes a confidence value for selecting between two different models, based on several restrictions, and may be applied whenever two solutions exist having the same conditions. The grammar's DTD defines two required attributes for this element: preferred model; and disc model corresponds to the discardable model. Attributes such as supmod, premod, word, cat are restrictions to the context where the rule applies.

The confidence level associated with each preference, allows the introduction of probabilistic elements in the grammar, and is given by the attribute confidence. This value must be in the set [-1,1], such that 1 shows an absolute confidence level, 0 is equivalent to discard the rule, and -1 is equivalent to 1 switching premod by disc mod. The confidence level for a preference of a given model A from B in a given context, can be defined on corpora occurrences, by the formula:

$$confidence(A, B) = \left( \frac{\text{occurrences}(A)}{\text{occurrences}(A) + \text{occurrences}(B)} - 0.5 \right) \times 2,$$

where occurrences(x) is the number of occurrences of x in such a context.

Example 6. Describing that inside of a m_nn, the model nc1_p should be preferred to the model adj3_p with a confidence level of 0.9.

```xml
<preference premod="nc1_p" discmod="adj3_p" supmod="m_nn"
    confidence="0.9"/>
```

3 Algorithm and Internal Organization

This section presents the adopted architecture and internal structure of the module, which allows analysis results to be stored, and information about previous calculated information to be provided, allowing efficient analysis.

3.1 Architecture

The overall analysis process is performed in two stages. The first stage consists of generating the information concerning the input data and storing it into a repository. The repository will then provide, in a second stage, all the information required for producing the desired output. As shown in figure 3, the analysis and extraction tasks are performed independently and can be independently parameterized. Besides providing all required data to the extraction module, the repository saves information about previous calculations, thus preventing redundant computation.

3.2 The Algorithm

In order to cover unusual linguistic constructions, the algorithm finds all possible sequences for the analysis during the first phase, then selects the most promising
Fig. 3. SuSAna internal architecture.

ones, either according to preferences or by applying psycholinguistic principles, such as longest model principle [6,3].

The analysis of a given sentence is represented using an in-memory DAG (Directed Acyclic Graph). Each vertex of the graph is associated with a lexical unit of the sentence and contains information about the occurrence of a model inside other model, in that position of the sentence. The DAG makes use of two types of edges, one for specifying sub-vertices (child vertices) and the other for specifying following vertices (sibling vertices). Each edge has an associated cost, given by the preferences specified in the grammar. The analysis consists on, being at a given vertex, finding all possible child vertices and, when done, finding all sibling vertices. Whenever possible, the algorithm reuses previously calculated analysis fragments, achieving results faster. Figure 4 shows the overall analysis process.

Fig. 4. Analysis process diagram in detail.
Selection of the most promising paths consists of ranking paths from the starting point of the graph, based on the cost associated with each edge and on the longest model principle.

The algorithm has a $O(n^2)$ complexity [3], where $n$ is the number of lexical units in the segment, since it deals with non-recursive syntactic domains, similar to chunks [6] and minimal syntagms [7], known as nuclear syntagms. The algorithm is also robust, in the sense that it can skip unexpected, or out of context, lexical units and reduce as much as possible the number of hypotheses for each analysis, thus providing suitable output for further processing.

3.3 Restriction mechanisms

The syntactic analysis of a text segment may lead to multiple solutions. In order to cover a broader set of constructions in a language, a grammar may contain generic rules, which can lead to a vast number of solutions. Mechanisms for refining solutions, reducing the most uncommon solutions, without limiting coverage, becomes necessary.

Besides preferences, SuSAna makes use of psycholinguistic principles [8,9], for choosing between different interpretations that the parser might be able to find. Currently, the module uses the longest model principle [6,3], which states that, all other things being equal, new constituents tend to be interpreted as being part of the constituent under construction (rather than part of some constituent higher in the parse tree) [9]. In the future, other psycholinguistic principles, such as minimal attachment and right association, may be applied. The use of restriction mechanisms by SuSAna is optional for each analysis.

4 Parameterization

The previously presented architecture allows a flexible way of setting analysis and extraction options. In what concerns analysis options, one of the most important is the possibility of defining the starting model, overriding the default one, during execution. Another important option is the possibility of skipping un-treatable lexical units at the beginning and at the end of the analysis, making it possible to find the best solution without considering those words. This option can be used to find large linguistic structures in the segment when boundaries are not feasible. By default, each segment corresponds to a linguistic structure. However, it is possible to search for multiple linguistic structures in a segment, allowing for example to identify sentences in a paragraph. This option can be used simultaneously with the option for skipping models, in order to extract all the linguistic structures of some type in a given segment.

Another important option for SuSAna is the ability of processing incomplete structures. This is useful in the case when there are no solutions and the user wants to known the largest analysis found. This can also be applied to guess, for an incomplete sentence, the categories that may follow the last lexical unit, so that the sentence remains correct according to the grammar.
SuSAHa can also be parameterized for producing the results in several formats. The XML format is the default, and allows to extract all the information about the syntactic analysis. The generation of the analysis graph structure, suitable for graph production using tools, such as GraphViz [10,11] is also possible. This last option can be used, either for analyzing complex structures, or for debugging the analysis process.

5 Evaluation

In what concerns linguistic correctness, at the moment, only small tests have been performed, but they show promising results. The grammar currently in use was written by Hagège [3] for extracting noun phrases (NPs). Linguistic phenomena, such as verb phrases, are superficially treated, preventing a feasible full linguistic evaluation of the system.

The performance analysis of the system was performed with several parameterizations, in order to study the module in different situations. The input information was previously tagged at the morphosyntactic level. The segmentation was also previously performed, using very basic prediction heuristics.

5.1 Evaluation Parameters

The quality of the obtained results depends on the linguistic information used for the algorithm. Comparisons between SuSAHa and AF showed that SuSAHa produces more feasible results, using the same linguistic information, due to incoherent information produced by AF. Nevertheless, results concerning the linguistic correctness can be found in [3].

Processing time is a relevant factor, in what concerns the integration of the module in broader processing chains, or Natural Language Processing systems. For this reason, it is important to evaluate SuSAHa's time performance, and to identify factors that contribute to that performance.

The combinatorial explosion of the solutions for a given analysis may impose limitation on the way results are treated by posterior stages. Therefore, efficiency measures of restriction mechanisms, such as preferences and long model principle, is another important aspect to focus on.

5.2 Corpus Properties

Tests were conducted over a corpus of about 4.6 million words, consisting of 51 editions of the newspaper Público. Table 1 presents a resume of corpus characteristics.

In what concerns the number of words per segment, Figure 5 presents the distribution of segments by number of lexical units. The biggest segment has 752 lexical units, the average length is 27 lexical units, and most of segments have from 19 to 39 lexical units. In what concerns the ambiguity, the average is about 1.31 tags per word.
<table>
<thead>
<tr>
<th>Element</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>Editions</td>
<td>51</td>
</tr>
<tr>
<td>Segments</td>
<td>170,126</td>
</tr>
<tr>
<td>Lexical Units</td>
<td>4,399,088</td>
</tr>
<tr>
<td>Tags</td>
<td>6,034,521</td>
</tr>
</tbody>
</table>

Table 1. Summary of the characteristics of the corpus used for evaluation.

Fig. 5. Distribution of text segments by number of lexical units.

5.3 Performance Analysis Strategy

Four performance analysis performed with different sets of parameters are now described.

The first test was performed using default options, considering each text segment as a sentence, and every word in that segment as valid for that sentence.

The second test was performed using the skips option, allowing the module to discard words in the segment for those the analysis process could not be successful. This test was conducted to perceive the number of segments that may contain a sentence, and no syntactic structure was assigned due to incompatible words. An interesting result from this test, is to obtain the number of skipped words for each segment.

The third test consisted of analyzing the corpus with the options skips and multiple. The purpose of the test was to identify the number of segments that contained more than one sentence. Once the segmentation was previously performed at morphosyntactic level, with basic heuristic rules, we expected an augmented number of sentences.

The last test, consisted of analyzing the whole corpus discarding previous segmentation, using skips and multiple options. In this process, each edition was considered as a single segment, and was analyzed as a whole. In this test, each segment had thousands of words, and the segmentation process was made by SuSAna, during analysis.
5.4 Results

All tests were conducted using an Intel Pentium III processor at 800 Mhz, Linux operating system, kernel version 2.4.20-8, with 500Mb of RAM.

<table>
<thead>
<tr>
<th>Test</th>
<th>Segments</th>
<th>Skips</th>
<th>Multiplé</th>
<th>time</th>
<th>Coverage segments</th>
<th>Coverage words</th>
<th>Coverage sentences</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>170,226</td>
<td>No</td>
<td>No</td>
<td>2h 32m</td>
<td>61.6%</td>
<td></td>
<td>105,157</td>
</tr>
<tr>
<td>2</td>
<td>170,226</td>
<td>Yes</td>
<td>No</td>
<td>4h 18m</td>
<td>98.7%</td>
<td>42.2%</td>
<td>168,011</td>
</tr>
<tr>
<td>3</td>
<td>170,226</td>
<td>Yes</td>
<td>Yes</td>
<td>4h 44m</td>
<td>98.7%</td>
<td>95.3%</td>
<td>245,288</td>
</tr>
<tr>
<td>4</td>
<td>51</td>
<td>Yes</td>
<td>Yes</td>
<td>4h 45m</td>
<td>97.7%</td>
<td></td>
<td>243,724</td>
</tr>
</tbody>
</table>

Table 2. Results from the analysis. Test 4 - unsegmented test.

Table 2 presents the results, using the previously described strategy. SuSAna performed all the analyses at an average of 232-303 words/second, depending on the performed test. The percentage of segments for which a syntactic structure was found is 61.6%-98.7%, depending on the performed test. In what concerns word coverage, the percentage of segments for which a syntactic structure was found varies from 42.2% to 97.7%. The value 61.6% corresponds to the percentage of segments for which a syntactic structure was found, considering that each word was correctly placed in the segment. Using SuSAna to segment the corpus, 97.7% of the lexical units were covered.

![Fig. 6. Distribution of the number of solutions for each sentence.](image)

Figure 6, shows the number of correctly analyzed segments, and their distribution according to the number of solutions for each result. The figure shows that most of the results for an analysis have one or two possible solutions. For example, 37% of the analyses performed with skips option, have only one solution.
The average time per lexical unit, taking into consideration the segment length, is presented in figure 7.

6 Remarks and Future Work

SuSAna uses an efficient algorithm and is a flexible module in what concerns the combination of options. The integration within other systems was an important aspect taken into consideration during its development. In order to increase its portability, a client and a server module were developed in order to provide a client/server platform. This facility permits to use SuSAna in remote computers, using the RPC (Remote Procedure Call) protocol [12].

SuSAna has been, so far, integrated in several systems. examples are:

- Poeta: a system for helping to write poems [13];
- ATA: a system for automatic term extraction [14];
- JaVaLI: a questioning interpreting system [2];

Some points of continuity for this work are now presented.

The current lexicon information is reduced, oriented for NP extraction, thus not covering several linguistic phenomena for Portuguese. In order to correctly process real text, the information concerning sub-categorization for grammatical classes, such as verbs, adjectives, and nouns, must be provided.

Restriction mechanisms are now applied during the analysis, just after building partial constituents, reducing the analysis complexity. Other restriction mechanisms should be investigated, and integrated in the system, in order to achieve less complexity for the analysis.

The current confidence level values, used in preferences, are not based in corpora observations. An automatic process can be conceived of to produce such information, resulting a better accuracy for the results.
An interactive syntactic disambiguation tool could provide an easy way of semi-automatically adjust the linguistic information.

One important step to follow is to obtain performance indicators and linguistic accuracy comparisons with other parsers. To achieve this purpose is necessary to import linguistic information from other parsers and to execute each system under the same conditions.

Writing rules in SuSAna is now limited to the use of simple tags. The introduction of mechanisms such as concordance and feature values, would provide a more sophisticated way for dealing with linguistic phenomena, preventing the enormous growth of the linguistic information in the future.

References