An interactive environment for creating and validating syntactic rules

Panagiotis Bouros,* Aggeliki Fotopoulou, Nicholas Glaros

Institute for Language and Speech Processing (ILSP),
Artemidos 6 & Epidavrou,
GR-151 25, Athens, Greece

{pbour, afotop, nglaros}@ilsp.gr

1 Introduction

Syntactic analysis is a key component in many Natural Language Processing applications. This is especially true when considering advanced spelling checkers, where the usage of contextual rules at the syntax level can significantly increase the spelling error detection and correction capability of such systems. The advantage of the contextual approach over the isolated-word approach becomes more clear in morphologically rich languages, in which it is very likely that a spelling error free word can, in fact, represent a misspelled word within a given context. In such cases, even a minimal set of syntactic rules can be proved very effective in obtaining high spelling performance levels. However, determining a consistent set of rules for spelling checking purposes is not always a straightforward task. In this paper, we design and implement an interactive linguistic environment for managing the grammatical and syntactic resources of an advanced spelling checker system for Greek.

2 Objectives - Specifications

The main purpose of the work presented in this paper is to provide a supportive environment for fast generating a consistent set of syntactic rules optimized for advanced spelling checking processes. Through a user-friendly interface, this tool allows language specialists to create, view, edit, real-time test monitor and validate syntactic rules, while leaving them out of the underlying computer programming technicalities.

As far as rule generation and editing is concerned, the environment provides a graphical rule representation mechanism. We consider that a tree graphical representation is suitable for presenting the word environments, the decision and generally the context of a syntactic rule. Moreover, in order for the tool to be speller technology-independent, we provide an XML-based mechanism for storing the rule tree representations. Furthermore, the tool automatically transcribes the user-defined rules into ready-to execute speller code (according to the speller being targeted), thus, providing a test-bed for the fast generation of a robust syntax analyzer.

By means of rich enough monitoring information, the system enables the user to evaluate the application of rules either individually or in combination with other user-specified rules. Emphasis is given on the production of a detailed report depicting the lexical analysis of the text, as well as details on the application of the user selected subset of rules, in order to identify or handle potential misusage, conflicts etc.

3 Architecture

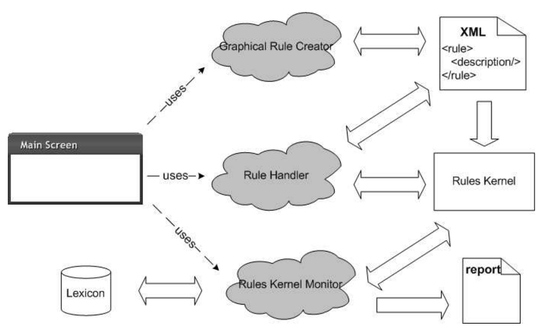

Figure 1: System architecture

Figure 1 illustrates the architecture of the implemented tool. Each syntactic rule created by the Graphical Rule Creator is stored in an XML document and integrated in the Rules Kernel. The Rules Kernel is an extension of the kernel used by Symfonia speller with extra features for supporting insertion, handling and monitoring of the rules' application. Graphical Rule Creator is also used for editing and updating a syntactic rule. Furthermore, in order to provide additional handling functionality on the Rules Kernel, we have introduced the Rule Handle component.

Finally, the Rules Kernel Monitor is responsible for testing and reporting on the usage of a subset of the rules, integrated into the kernel, across real unformatted text. The monitor procedure relies on the speller's built-in lexicon for the lexical analysis and on the Rules Kernel for syntactic analysis, in order to generate a detailed report.

4 Working Environment - Functionalities

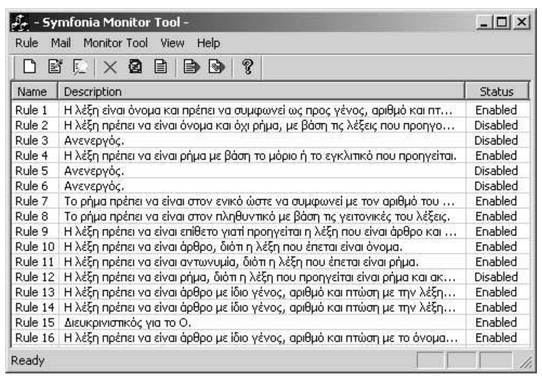

Figure 2: Main screen

Figure 2 displays the main screenshot of the implemented tool, being the first window interacting with the user. This window consists of the list of rules that are integrated into the kernel. For every rule, its name, description and status are indicated. The status of a rule is either *enabled* or *disabled*, meaning that can be either taken into account by the monitor procedure or not. Moreover, all functionalities of the developed system are available through the menu options and the toolbar icons of this window.

4.1 Rule Handling

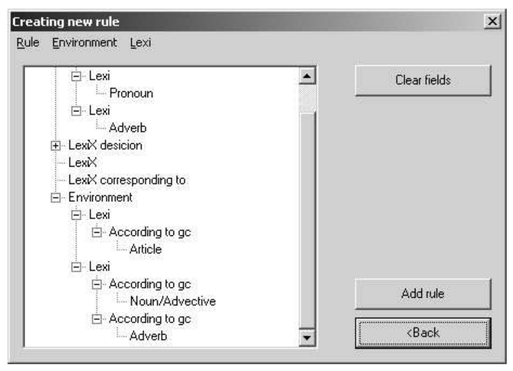

Figure 3: Rule tree

Rule handling mainly pertains to the management of the Rules Kernel component, i.e. addition of new rules, editing of the definition and of the status of an existing rule or simply its removal from the kernel.

1. **Create a new rule**. In order to create a new syntactic rule the user takes advantage of the rule graphic tree representation presented in Figure 3. Each rule is focused on a single lexi[1] called LexiX. The user defines the description, the explanation and the context of a syntactic rule. i.e. the valid combinations of grammatical characterizations (lexis) for LexiX, the lexi which the new rule should conclude to, which words will the application of the rule be restricted to, the adjacent words whose grammatical characteristics will be inherited to LexiX and finally the alternative environments of the new syntactic rule.
2. **Edit an existing rule**. The procedure of editing an existing rule is alike to the one of creating a new rule as it reproduces the tree representation of the rule (Figure 3).
3. **Remove an existing rule**. Removal of an existing rule can be done through the respective menu option or toolbar icon located in the main screen (Figure 2).
4. **Disable/enable an existing rule**. By default, the status of a new rule is set to *enabled*. The status can be altered from the main screen in Figure 2 either to *disabled* or *enabled*.
5. **Export of existing rules**. Apart from XML format, a single or the entire set of the syntactic rules can be exported and e-mailed to the programmers group of the targeted syntactic speller, in a high level programming language code.

4.2 Monitor

Efficient syntactic rules-based spell checking leads to the problem of generating and choosing syntactic rules that on the one hand optimize the performance of the spelling checker engine and on the other constitute a consistent set of rules. For this purpose, the system provides a monitor functionality for the evaluation of Rules Kernel while being on text documents. The system also takes advantage of the lexicon of Symfonia in order to perform the additional grammatical and lexical analysis required.

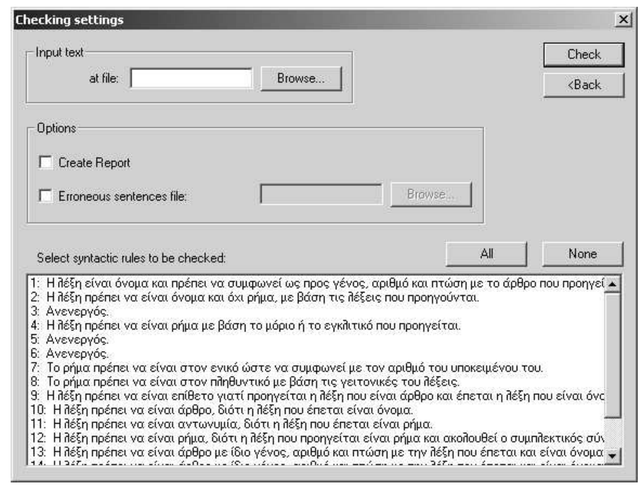

Figure 4: Checking procedure settings

Rules checking can be done either interactively or automatically. In the first case, the user has to select one of the automatically generated system suggestions that attempt to correct the syntax error encountered. In the second case, the system by default adopts the first suggestion. Figure 4 presents the settings dialog of the checking procedure. In this dialog the user specifies the input text containing the sentences that should be checked, the set of syntactic rules that will be used and also if the system will produce a report of the check and a document containing the erroneous sentences. Figure 5 denotes the structure of a report document.

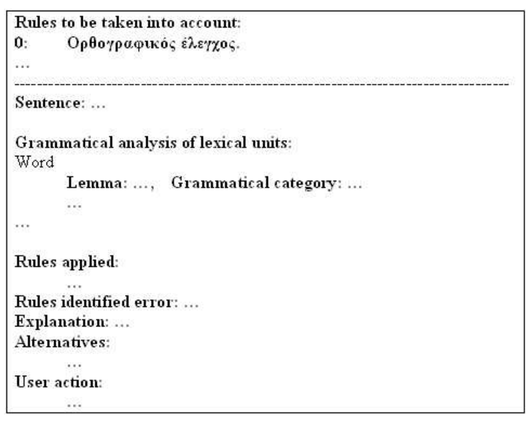

Figure 5: Report structure

5 Real-World scenario

Let us assume that we wish to solve the ambiguity between the greek words for "more" and "which": "πιο" and "ποιο". Although these two words have the same phonetic transcription /pjo/, the first one is an adverb and the second is a pronoun. We create a syntactic rule with the following environment:

Lexi1 LexiX Lexi2

If LexiX is characterized by the ambiguity "πιο" - "ποιο" and Lexi1 is an article and Lexi2 is either an adjective or a noun or an adverb, then LexiX is an adverb, i.e. "πιο".

The previous rule resolves the ambiguity by rendering LexiX as an adverb. We can also define another rule for specifying that LexiX should be a pronoun, i.e. "ποιο". The environment of the required rule would be:

LexiX Lexi1 Lexi2 Lexi3 Lexi4 Lexi5

LexiX is "ποιο" if Lexi1 is an article, Lexi2 an adjective, Lexi3 a noun, Lexi4 a particle and Lexi5 a verb. In addition, some or all of Lexi1, Lexi2, Lexi3 and Lexi4 can be missing.

* Current affiliation is National and Kapodistrian University of Athens (NKUA), Department of Informatics and Telecommunications.

[1] The term *lexi* (lexis in plural) is used in this text to denote the set of grammatical characteristics of a word - On the other hand *words* are simply the tokens of a sentence.