Chapter X
CorTag:
A Language for a Contextual Tagging of the Words Within Their Sentence

Yves Kodratoff
University Paris-Sud (Paris XI), France

Jérôme Azé
University Paris-Sud (Paris XI), France

Lise Fontaine
Cardiff University, UK

ABSTRACT

This chapter argues that in order to extract significant knowledge from masses of technical texts, it is necessary to provide the field specialists with programming tools with which they themselves may use to program their text analysis tools. These programming tools, besides helping the programming effort of the field specialists, must also help them to gather the field knowledge necessary for defining and retrieving what they define as significant knowledge. This necessary field knowledge must be included in a well-structured and easy to use part of the programming tool. In this chapter, we present CorTag, a programming tool which is designed to correct existing tags in a text and to assist the field specialist to retrieve the knowledge and/or information he or she is looking for.

INTRODUCTION AND MOTIVATION

In this paper we present a new programming language, called CorTag, which is devoted to tagging words within the boundaries of the sentence in which they are contained. The context we are concerned with here is therefore limited to the sentence and the words within it. The tagging

process in CorTag includes syntactic, functional and semantic tags. Ultimately CorTag is designed to correct the existing tags in highly specialised or technical texts.

Our primary aim is to contribute to the creation of a system which is able to find interesting pieces of knowledge within specialised texts. There is no attempt being made towards the broader understanding of natural language. Our ambition is to be able to spot parts of the texts that may be of particular interest to the specialist of a given technical domain. As we shall see, the process does nevertheless require a kind of 'primitive' understanding of the text.

In creating this new language, we have been motivated by two facts which, despite being intuitively obvious, are challenging when used as a base for the building of a computer system.

The first of these is that the number of genre specific texts is increasing exponentially. It follows that humans can no longer handle these masses of texts and the whole process has to be automated. The scientific community is certainly aware of this need as it is exemplified by the large number of competitions and challenges, dealing with many topics expressed in many different languages. This has led to the development of software solutions devoted to solving at least one of the problems encountered for each step of the overall process. In order to make these steps explicit, let us propose a tentative list of the main steps involved. The text mining process starts by gathering the texts of interest, what we will refer to as 'text gathering'. The process ends when the desired information has been found in the text. This final step is identified here as 'information extraction'. There is a large set of intermediate steps which take place between these two steps, and the precise set of steps depends on the state of the retrieved texts and the nature of the information sought. The following sequence shows one possible ordering of the necessary intermediate steps:

text gathering → sorting → standardization → creation/improvement of lexicon → tagging and/or parsing → terminology → concept recognition → co-reference resolution → finding the relations among concepts → information extraction.

In the following, when speaking of any step in particular, we will always assume that all n-k steps have been executed before the current $step_n$. We shall not, however, assume that they have been correctly completed. One of the main difficulties is that these different levels of Natural Language (NL) processing are mutually dependent. In general, the context independent processes can be performed quite satisfactorily, while the context dependent ones are very challenging as we shall exemplify. Unfortunately, the users (and sometimes even the creators) of the '$step_n$ specialized software' are not aware that this software is absolutely unable to function properly if some of $step_{n-k}$ has not been properly completed. For example, 'sorting', a step which will be described later in the paper, illustrates well the dependencies amongst steps. Sorting is not really context-dependent, as we shall explain, and therefore it is a step which should be completed relatively easily. However, an improperly performed $step_n$ causes mistakes at $step_{n+k}$ which then spread throughout the process. It is the context dependent steps which are most greatly affected by this. Since many of the context dependent mistakes of $step_{n-k}$ cannot be detected before $step_n$, we need a language to backtrack and correct them. This defines the first primary constraint placed on CorTag's development.

The second motivating fact is that each specific genre tends to develop its own lexical and grammatical tendencies, often considered as jargon outside of the genre. As researchers, we cannot be put off because of the difficulties and challenges presented as the texts become more highly specialised or because they diverge from standard written English. Linguists, whether