HOSTED BY

ELSEVIER

Available online at www.sciencedirect.com

ScienceDirect

journal homepage: www.elsevier.com/locate/jides

A typed applicative system for a language and text processing engineering

*Ismaïl Biskri**, *Marie Anastacio*, *Adam Joly*, *Boucif Amar Bensaber*

LAMIA – DMI, Université du Québec à Trois-Rivières, CP 500, Trois-Rivières, Québec, Canada

ARTICLE INFO

Article history:
Received 23 October 2014
Received in revised form
19 January 2015
Accepted 15 February 2015
Published online 4 March 2015

Keywords:
Processing chains
Language
Text analysis
Categorial and applicative systems
Combinatory logic

ABSTRACT

In this paper, we present a flexible, modular, consistent, and coherent approach for language and text processing engineering. Each processing chain dedicated to text processing is regarded as a serial or parallel assembly of modules, underlying particular tasks a user wants to apply to a text. Users, according to their needs and perspectives might want to build and validate their own processing chain by assembling a set of modules according to a certain configuration. In this paper, we suggest a theoretical formal system based on the model of the typed applicative grammars and the combinatory logic. This approach allows providing a general framework in which users would be able to build multiple language and text analysis processes according to their own objectives. It will also systematize the verification of the logical consistency of the sequence of modules in the assembly that characterizes a given processing chain.

1. Introduction

Nowadays, what is known as 'Language and Text processing' is all the fields related to information retrieval, categorization, classification, indexation, syntactic analysis, semantic analysis, knowledge extraction, knowledge management, etc. These fields are fundamental; they can impact economical, scientific, political, cultural and social sectors. This is particularly amplified by the fact that corpora, textual databases, and mainly the web (especially social networks), represent an endless source of information.

Recently, some voices have been raised among the scientific community to denounce the actual limits of these fields. The critics object based on the following hypothesis; the domain expert – or alternatively the computer scientist expert – would be the designer of an implemented system that would only require periodical updates.

This hypothesis has been proven to be unproductive, given the fact that it does not take into account the subjectivity, or the point of view of the user, may he be an expert or not on the topic.

Furthermore, the hypothesis stated above does not allow the "collaboration of multiple points of view", that often originates from different disciplines such as computer science, artificial intelligence, linguistic, psychology, semiology, logic, philosophy, terminology, ontology, etc.; reading and analysis

Peer review under responsibility of Qassim University.
* *Corresponding author*.
E-mail addresses: Ismail.Biskri@uqtr.ca (I. Biskri), Marie.Anastacio@uqtr.ca (M. Anastacio), Adam.Joly@uqtr.ca (A. Joly), Boucif.Amar.Bensaber@uqtr.ca (B. Amar Bensaber).

http://dx.doi.org/10.1016/j.jides.2015.02.003

of texts are considered to be at the intersection of these disciplines.

For example, users who wish to retrieve information with a search engine, like Google, may decide to use one or more downstream filters to refine their results. They may also decide that the combination of the search engine and the filter is the ideal processing chain for their needs, and thus want to keep it for reuse. Therefore, users can create, thereby, a processing chain that meets more specifically their needs. In order to illustrate this concept, consider the case of a "librarian" who wants to retrieve papers of one given author, and access to definitions and to citations contained in these papers. Giving the name of the author as a keyword is not sufficient. Adding the word "definition" and "citation" as keywords will not adjust the result of the query, since those key words are not significant terms in the papers. A linguistic filter as the one presented in [3,4] used downstream of the search engine will allow the extraction of definitions or citations. Librarians can create a processing chain for their specific needs. They can save or reuse this same processing chain, to access to definitions and citations contained in papers from another author.

In fact, it seems that the solution to this type of problem does not simply reside in supplying one or many software tools. Although developed technologies were successful since they were made more available to users, dissatisfaction has been observed among them due to the following significant limitations; (i) they offer a limited set of closed functionalities; (ii) they are often designed within an architecture that has limited communication capabilities with external softwares that would provide additional functionalities; (iii) it is difficult, if not impossible, to integrate new functionalities to the tool without having to rebuild a significant part or all of it; (iv) the sought collaboration between experts and their intermediaries, all while taking into account the copyrights of each of the creators, is also very complicated.

A methodological reflection on the topic is essential, which brings us to what we could consider a new postulate for text processing. In fact, text reading and analysis – the foundations of any function underlying a task in text processing – is a "dynamic" process that allows multiple "point of views" that can lead to different "understandings" and thus, must be undertaken, while taking into consideration "multiple objectives".

We find in the literature, in data-mining and text-mining, projects on the creation of complex processing chains that offer assembling of many functions and operations and the creation of software platforms for language engineering which integrate statistical analysis, such as RapidMiner [19], WEKA [25], D2K/T2K [14] and Knime [23], or linguistic analysis, such as Context [9] and Gate [10]. Although some of these platforms have enabled the collaboration of researchers in projects like NORA and TAPoR, limitations persist, especially for the assembly of modules, which requires knowledge about the platform and in some cases on the programming code. These new platforms highlight the importance of methodological development surrounding the creation of processing chains [24,18].

In our paper, we will present a flexible, modular, consistent, and coherent architecture for text processing, in which each task will be addressed by an autonomous function that is independent from the other functions of the architecture. A formal theoretical framework based on the model of the typed applicative systems and the combinatory logic will be suggested.

Before presenting the formal model itself, we will introduce in the two following sections combinatory logic and the *Applicative and Combinatory Categorial Grammar*.

2. Combinatory logic

The origins of combinatory logic bring us back to the works of Schönfinkel who defined the concept of combinators in 1924 and sometime later, those of Curry and Feys [11]. This notion was introduced with the purpose to bring a logical solution to some paradoxes, such as Russell's Paradox, but also to eliminate the need for variables in mathematics in order to avoid variables telescoping.

Combinators are abstract operators that use others of the same kind to build more complex ones. They act as functions over arguments, within an operator–operands structure. Each specific action is represented by a unique rule called β-reduction rule, which defines the equivalence between a logical expression with a combinatory, versus one with no combinator.

Although many more combinators exist, we demonstrate in this paper that the combinators we used in our works and their corresponding β-reduction rule [11,15] (for other combinators, the reader may refer to [11,13,15]).

Combinator	Role	β-Reduction rule
B	Composition	**B** x y z → x (y z)
C	Permutation	**C** x z y → x y z
S	Distributive composition	**S** x y u → x u (y u)
C*	Type raising	**C***x y → y x
W	Duplication	**W**x y → x y y
$\mathbf{B}^2$	Composition–Power 2	$\mathbf{B}^2$ x y z u → x (y z u)
$\mathbf{C}_2$	Permutation–Distance 2	$\mathbf{C}_2$ x y z u v → x y v z u

B, **C**, **S**, **W** are elementary combinators. The composition combinator **B** combines two typed operators x and y together in order to form the complex typed operator **B** x y that acts on a typed operand z according to the β-reduction rule. The permutation combinator **C** uses a typed operator x in order to build the complex typed operator **C** x such as if x acts on the typed operands y and z, **C** x will act on those typed operands in the reverse order, that is to say z and y. Given the two typed operators x and y, and the typed operand u, the general composition combinator **S** distributes the typed operand u with the two precedent typed operators x and y. (y u) becomes the typed operand of the complex typed operator (x u). The combinator **C*** is applied on a typed operand x (x functions as the operand of y). It allows to build the complex typed operator (**C*** x) in order to apply it to y. Finally, given the binary