LINGUISTIC INTERPRETATION OF SPEECH ERRORS

D. Gîfu M. Cioca

Daniela Gîfu
„Alexandru Ioan Cuza" University of Iaşi
16 General Berthelot St., Iași, 700483, Romania
daniela.gifu@info.uaic.ro

Marius Cioca
„Lucian Blaga" University of Sibiu
10, Victoriei Bd., Sibiu, 550024, Romania
marius.cioca@ulbsibiu.ro

Abstract

The paper is an attempt to illustrate the linguistic interpretation of speech, known that it remains insufficiently resolved, especially for Romanian. The cause is given by the multitude of criteria that can or should be considered important in speech processing. The aim of this study is to develope a computational tool in order to identify the possible errors related to the morphosintactic structure of speech. Furthermore, the key feature of the application is to assist users who can receive automatically different suggestions that can help them to improve the quality of their text. The analysis involves a careful reading of the text, implying the examination of the language used to understand how the speaker's communicative intention is reflected in language. This study intends to help direct beneficiaries (students, journalists, etc.), but, also, specialists and researchers in natural language processing field in order to improve the writing skills and comprehension of texts.

Keywords: *speech analysis, spellchecking, lexical and grammatical correction, writing skills, natural language processing.*

1. Introduction

The option for the topic of linguistics interpretation of speech errors is still a challenge in NLP (*Natural Language Processing*) field. This survey comes from the need to develop an application for Romanian language for morphosintactic analysis of the content in order to facilitate correct text spelling, named LAT (*Linguistic Analysis Tool*). It can be defined in a language-independent manner. Our model differes from other previous models, [Goldsmith], [Brent], [Creutz], etc., by finding a complex architecture that combines four modules - *spellchecking*, *POS-tagging*, *text categorization* and *creating diagrams*. Thus, we chose an interdisciplinary approach through speech analysis that brings together, especially, the fields of artificial intelligence, computer science, and linguistics, which combine different give a concise representation for any kind corpus.

The analysis involves a careful reading of the text, implying the examination of the language used to understand how the speaker's communicative intention is reflected in language. Therefore, the users would be able to visualize different texts and to analyse them at the lexical and grammatical level, in order to identify certain types of errors in the writing

related to morphosyntactic appearance. Then they can correct the texts by choosing the correct phrasing[1] based on grammatical features.

The speech is characterized by an important level of diversity and one of the most known typologies structures it in three classes, necessary in speech interpretation: *enunciative* [1], *communicational* and*situational* [7].

Moreover, speech recognition demands a certain level of spelling, which represents our goal for this paper.

The concept, "speech" has no plural; it designates a generic field, the relation between language and parameters of the nonlinguistcs reality. The literature provides a few classic definitions. For this study, we consider the functional part of it very important, the speech analysis being "the analysis of language in use" [2]. In other words, the language "is doing some job in some context" [5]. Only a part of what can be said is accessible, forming a system and defining an identity [6]. According to these opinions, we develop a method useful to fix different errors that might appear in a speech.

The paper is structured in five sections. After a brief introduction about the importance of this study, section 2 mentions some important computer applications focused on the examination of speech. Section 3 describes the LAT architecture, and section 4 presents shortly the work methodology. Last section highlights conclusions and mentions the future work.

2. Related Works

To develop this application, the folowing tools were explored as: Tropes[2], RO-Balie[3] [3], and other competing software as Gate[4], Oak[5] or Minor Third[6]. All these applications, except RO-Balie, could be trained, using machine learning techniques.

Tropes V1.0, developed in 1994 by Pierre Molette and Agnès Landré, is the first semantic analysis software based on propositional analysis principles [4]. Tropes is now available in two versions: French or English. It creates a specific thesaurus for a certain domain (literary, philosophic, politic and scientific), performing a morphosyntactic analysis on the text by identifying the morphologic category of all the words, including: nouns, verbs, adjectives, pronouns, connectors, etc. It is important to note that Tropes can yield information about a text. For instance, to detect and quantify different categories of the emotional lexicon in French (EMOTAIX-Tropes) [8].

The similarities between Tropes and LAT are that both perform a morphosyntactic analysis on the text, identifying the main part of speech.

RO-Balie is an extension of BALIE[7], known as a multilingual text processing tool designed to support information extraction. The tool supports five languages: French, German, Spanish, English, and Romanian. It is used for language identification, text segmentation in sentences, tokenization, and part-of-speech tagging. Balie text processing consists of a structured list of tokens, by applying machine learning techniques [7].

---

[1] In linguistic analysis, a phrase is a group of words (or possibly a single word).

[2] Tropes Text analysis software, designed for TextMining, Qualitative Analysis, Semantic Categorization and Keywords extraction. In 1997, Acetic launches Tropes V3.0 automates the ACD (remarkable phrases), and chronological analysis of the text, starting from the political texts (http://www.semantic-knowledge.com/tropes.htm).

[3] RO-BALIE is available for download at http://www.site.uottawa.ca/~ofrunza/ RO-Balie/RO-Balie.html

[4] http://gate.ac.uk

[5] http://nlp.cs.nyu.edu/oak

[6] http://minorthird.sourceforge.net

[7] BALIE is a Java open-source software issued under the GNU General Public License. It is hosted by SourceForge and it is available at http://balie.sourceforge.net

The similarities between RO-Balie and LAT are that both allow extracting the tokens from the text and identifying the main part-of speech, and are using the Naïve Bayes classifier (here, to identify the speech type).

In addition, LAT performs grammar and spellchecker on the text.

3. LAT architecture

To implement the LAT application as a development environment we used *Spring Tool Suite (STS),* version 3.6.4 *RELEASE*, and as a programming language, the application was written in Java, within Java EE platform (*Enterprise Edition*).

The server part was performed using Spring MVC (*model-view-controller*) framework[8] and Apache Tiles 3 framework, and on the client side, for design, we used Bootstrap[9] [10], and for Ajax calls to server, the javascript was chosen (jquery 1.11 .1).

În urma studiilor efectuate, aplicația îi oferă utilizatorului posibilitatea de a printa sau copia datele din tabel, sau de a salva rezultatele, în format PDF, XLX or CSV. Această funcționalitate a fost realizată prin utilizarea plugin-ului "*TableTools*"[11] oferit de programul "*DataTables"*[12], a plug-in for the jQuery JavaScript library.

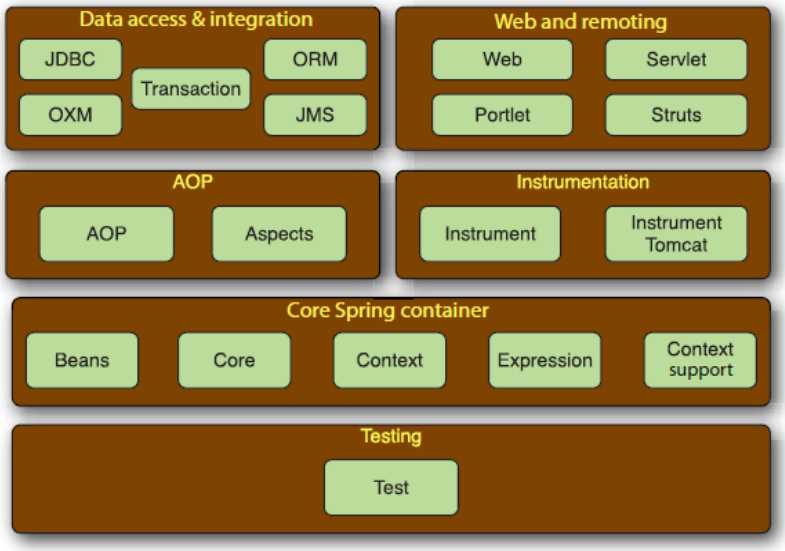
3.1 STS

Spring Tool Suite (STS)[13] este un mediu de dezvoltare bazat pe platforma Eclipse și creat pentru dezvoltarea de aplicații Spring, oferind suport atât pentru implementarea, debug-ul, rularea și lansarea aplicațiilor, cât și pentru refactoring, AOP (Aspect Oriented Programing)[14], validări, Spring beans, Spring Persistence (Hibernate + JPA), Spring MVC, suport pentru Java 8, integrarea cu Maven, navigare și căutare avansată etc. Procesul de dezvoltare al acestui mediu este mult mai ușor și mai rapid, atât datorită funcționalităților lui, cât și datorită posibilității de a combina toate acestea cu versiunea de Java dorită, (Java 7), cu partea de Web și Java EE din Eclipse. LAT a fost realizată cu Spring MVC, care va fi prezentată în cele ce urmează, urmată de descrierea framework-ului MVC utilizat pe partea de server.

3.2 Spring, Spring Security și Spring MVC (11)

Spring[15] este cel mai popular mediu de dezvoltare, *open-source*, creat pentru a aborda complexitatea dezvoltării aplicațiilor Java *enterprise*, framework-ul fiind alcătuit din șase module bine-definite (Figure 1).

[8] http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm
[9] http://getbootstrap.com/
[10] http://bootswatch.com/cerulean/
[11] https://www.datatables.net/extensions/tabletools/
[12] http://datatables.net/extensions/tabletools/api#fnResizeButtons
[13] https://spring.io/tools
[14] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/aop.html
[15] http://www.tutorialspoint.com/spring/spring_overview.htm



Figură 1 Spring Framework Modules[16]

Printre beneficiile utilizării acestui framework se numără: dezvoltarea de aplicații *enterprise*, folosind obiecte Java called *POJO* (*Plain Old Java Objects*)[17], organizarea modulară a pachetelor, cuplarea slabă (En. *loose coupling*) prin *dependency injection* și prin utilizarea conceptelor procesului IoC (*Inversion of Control*[18]), programarea declarativă bazată pe aspecte și multe altele.

Dependency injection face ca, decuplarea să fie concretă, clasele, mai ușor de testat prin injectarea mock-urilor[19], iar codul, mai ușor de înțeles și de menținut, principle called *Clean Code.*

Spring Security 4.0.0 Release

Mecanismul de autentificare în cadrul aplicației a fost realizat folosind framework-ul de autentificare și de control al accesului *Spring Security*[20], care oferă o serie de servicii de securitate pentru aplicații de tip *enterprise*, putând fi integrat cu Spring MVC. Practic, se evită folosirea neautorizată a aplicației, sporind siguranța.

Spring MVC[21]

Spring MVC este un framework, *open-source*, bazat pe *design-pattern*-ul *model-view-controller*, realizat pentru a simplifica implementarea și testarea aplicațiilor Java web, fiind integrat complet cu Spring. Pattern-ul MVC asigură *loose coupling* prin separarea logicii aplicației în trei componente: input logic, business logic și UI logic. Modelul încapsulează datele aplicației prin utilizarea obiectelor *POJO*, view-ul este responsabil cu redarea datelor modelului. Acesta constă în output HTML ce poate fi interpretat de browser-ul clientului, iar controller-ul este responsabil cu procesarea cererilor utilizatorilor, construind modelul ce va fi trimis mai departe către view.

Bazat pe design pattern-ul Front Controller[22], Spring web MVC este construit în jurul unui servlet central, called Dispatcher Servlet[23], which treats all *HTTP requests&responses.*

[16] Source: http://docs.spring.io/spring-framework/docs/3.0.x/reference/overview.html
[17] More details: http://www.shaunabram.com/beans-vs-pojos/
[18] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/beans.html
[19] http://docs.spring.io/spring/docs/current/spring-framework-reference/html/testing.html
[20] More details: http://projects.spring.io/spring-security/
[21] http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm
[22] http://www.tutorialspoint.com/design_pattern/front_controller_pattern.htm
[23] http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

Practic, după primirea cererii HTTP, servletul consultă *Handler Mapping*-ul[24] pentru a decide ce Spring MVC Controller trebuie să trateze cererea. După aceasta, trimite o cerere controller-ului selectat și consultă un *View Resolver*[25] pentru a mapa cererea la implementarea unui view specific, view ce poate fi sau nu o pagină JSP (*Java Server Pages*).

JSP[26] este o tehnologie utilizată pe partea de server, comună implementării componentei view din modelul MVC. LAT este alcătuită dintr-o serie de pagini JSP, populate prin intermediul modelelor și mapate cu ajutorul controllere-lor. Pentru o mai bună structurare a paginilor s-a folosit *Apache Tiles 3.0*[27].

3.3 Apache Tiles

Bazat pe design pattern-ul *Composite*[28], *Apache Tiles*[29] este un framework, open-source, construit pentru a simplifica procesul de dezvoltare al interfeței utilizator. Acest framework, reprezintă cea mai simplă și elegantă soluție pentru dezvoltarea de aplicații web bazate pe orice tehnologie MVC.

Arhitectura generală a LAT se bazează pe *Apache Tiles*. A fost conturat șablonul principal, prin definirea footer-ului și a header-ului, componente ce vor fi conținute de fiecare pagină JSP a aplicației, la care s-a adăugat partea de conținut a paginii (*body*), ce va fi înlocuită, în funcție de context, cu informații particulare, specifice fiecărei acțiuni.

Mai jos este redat șablonul-ul principal al aplicației construit folosind *Tiles*, template ce a fost extins pentru a adăuga conținut nou paginii JSP definite în template-ul principal.

```
<tiles-definitions>
    <definition            name="defaultTemplate"            template="/WEB-
INF/views/template/default/template.jsp">
    <put-attribute name="title" expression="No title"/>
        <put-attribute              name="header"              value="/WEB-
INF/views/template/default/header.jsp" />
        <put-attribute              name="footer"              value="/WEB-
INF/views/template/default/footer.jsp" />
    </definition>
</tiles-definitions>
```

Iată un exemplu de extindere a șablonului principal:

```
<definition extends="defaultTemplate" name="upload/viewUploadedContent">
    <put-attribute                  name="body"                value="/WEB-
INF/views/upload/viewUploadedContent.jsp"/>
    <put-attribute name="title" expression="View uploaded content"/>
  </definition>
```

3.4 Apache Commons FileUpload

Încărcarea fișierelor care sunt analizate se face folosind Apache Commons FileUpload, API ce permite procesarea unei cereri HTTP de tip POST a cărui conținut este de tip “multipart / form-data”, conform cu standardul *RFC 1867*. Este o aplicație care ușurează procesul de upload de fișiere, răspunsul returnat fiind ușor de utilizat în cadrul aplicației web.

---

[24] http://www.codejava.net/frameworks/spring/understanding-spring-mvc?start=4
[25] http://www.studytrails.com/frameworks/spring/spring-mvc-view-resolver.jsp
[26] http://www.oracle.com/technetwork/java/javaee/jsp/index.html
[27] https://tiles.apache.org/framework/whats-new.html
[28] http://www.tutorialspoint.com/design_pattern/composite_pattern.htm
[29] https://tiles.apache.org/

3.5 Hibernate

Pentru salvarea datelor s-a folosit *PostgresSQL*, version 9.3, iar pentru maparea obiectelor Java cu tabelele din baza de date și maparea datelor din Java cu date de tip SQL, *Hibernate*.

Hibernate is an *open-source* framework ce oferă posibilitatea interogării datelor prin generarea de apeluri SQL pentru a prelua sau căuta informații stocate în baza de date. Astfel, prin intermediul facilităților oferite de *Hibernate,* maparea obiectelor se face în mod automat, fără a mai fi nevoie de conversia manual.

Toate aceste tehnologii și framework-uri au ușurat procesul de dezvoltare, structurarea și realizarea aplicației fiind mai rapidă.

4. Work methodology

LAT becomes a very useful application for the Romanian linguistic analysis, being able to determine the defining features of the type of speech that it represents.

The application (Figure 2) was focused on identifying the context of the analyzed text. It allows analysis of TXT, DOC, DOCX, ODT files. On entry the texts can have or not diacritics, with LAT we can make the correct diacritical marks, especially when the word can be confused with another Romanian word (e.g. *laturi* vs. *lațuri* vs. *lături*). These situations are not too rare.

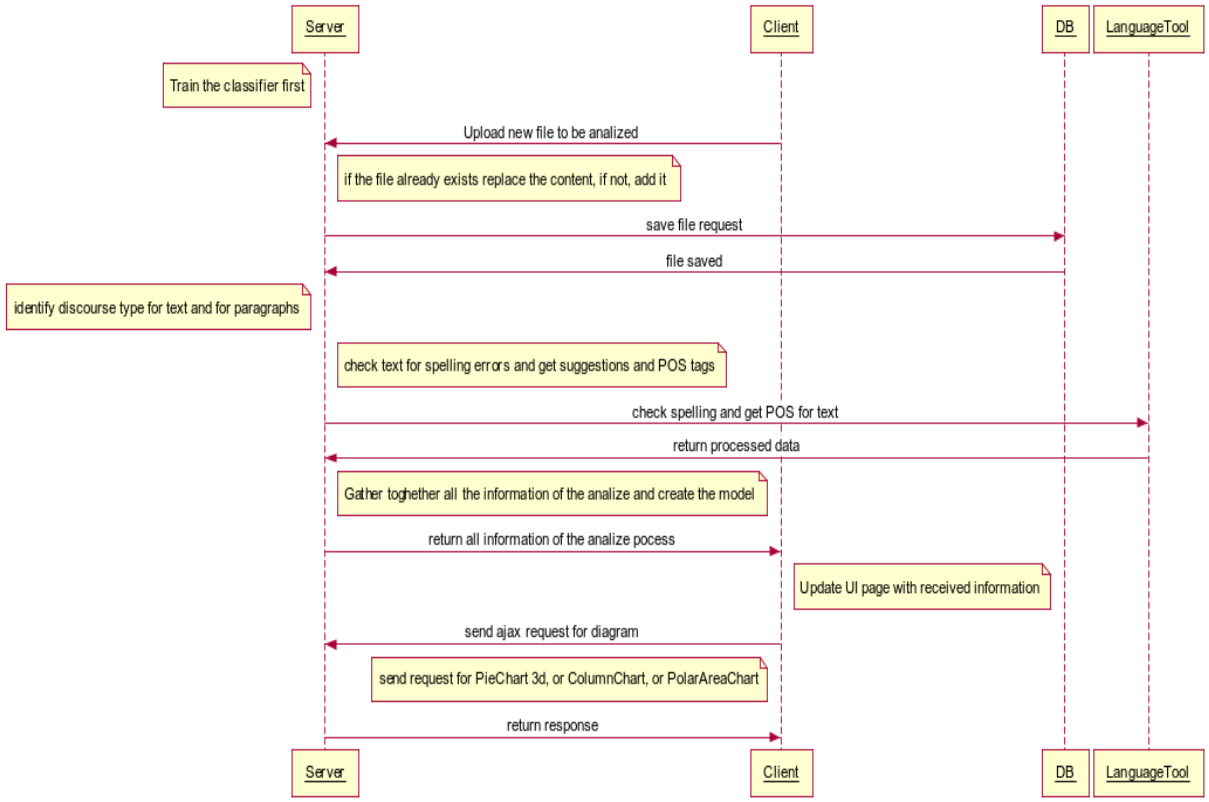


Figure 2. Work session with LAT

LAT is structured in four modules:

(1) *spellchecking* – to identify errors in spelling or grammar, repeated words, and also identify if there is no punctuation or it used incorrectly, by integrating the open-source program LanguageTool[30] (LT) (Figure 3). Also, it can detect the text language.

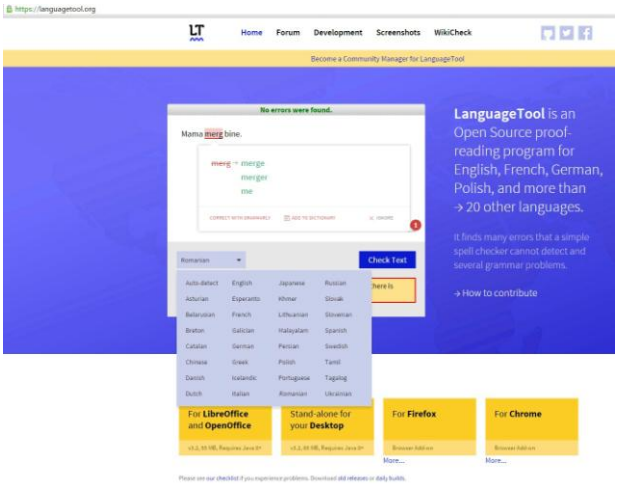


Figure 3. Work session with LT

Fpr instance

(2) *POS-tagging* – to identify parts of speech categories and morpho-syntactic information of tokens, by using the POS-tagger tool [9].

For example: *oameni sfinți*

```
  <W    Case="direct"    Definiteness="no"    Gender="masculine"
LEMMA="om"      MSD="Ncmprn"      Number="plural"      POS="NOUN"
Type="common" id="55.39" offset="205">oameni</W>
  <W Case="direct" Definiteness="no" Gender="masculine"
LEMMA="sfânt" MSD="Afpmprn" Number="plural" POS="ADJECTIVE"
id="55.44" offset="229">sfinţi</W>
```

(3) *identifying the speech type*, using Naïve Bayes classifier. To increase the speed of processing a classification request and, also, to improve performance, the @PostConstruct[31] method annotation was used.
(4) *creating diagrams* – after dividing text into paragraphs, a list of ChartDTO's objects (diagrams) is created. There are 3 types of diagrams supported by the app: *PieChart 3D*, *ColumnChart* by using *Google Charts* API[32] and *PolarAreaChart* by using *Chart.js* API[33].

5. Conclusions and future work

LAT tools can be used successfully for speech analysis, including at lexical level by grammar checking, and spellchecking. Moreover, it becomes a very useful and interactive application, being able to recognize the text type, and determine the defining features of the type of speech that it represents.
Now, we are working to compare the results obtained by the program himself and the results obtained using a rule-based languages similarities method (especially, Latin languages). This method is based on pattern matching. We need a gold corpus of hundreds sentences, extracted

[30] http://mvnrepository.com/artifact/org.languagetool/languagetool-core
[31] https://docs.oracle.com/javaee/5/api/javax/annotation/PostConstruct.html
[32] https://developers.google.com/chart/interactive/docs/dev/dsl_get_started
[33] http://www.chartjs.org/docs/

from documents within different genres (a work in progress). We are also interested to identify the semantic level. Thus, the application should be extended by adding a new component responsible for to relate meanings to syntactic structure.

Acknowledgments: The LAT software has been developed by Iuliana Mariana Bejan from the Faculty of Computer Science "Alexandru Ioan Cuza" of Iaşi.

References

[1] Benveniste, E. (1966): Problèmes de linguistique générale. Paris, Gallimard, pp. 118-131 (republished in 1974).
[2] Brent, M.R. (1999: An efficient, probabilistically sound algorithm for segmentation and word discovery. *Machine Learning*, 34, pp. 71–105.
[3] Brown, G., Yule, G. (1983): Discourse analysis, Cambridge University, Press, Cambridge, p. 1.
[4] Creutz, M. and Lagus, K. (2002): Unsupervised discovery of morphemes. In Proc. Workshop on Morphological and Phonological Learning of ACL'02, Philadelphia, pp. 21–3.
[5] Frunză O., Inkpen, D., and Nadeau, D. (2005): A text Processing Tool for Romanian Language. In *Proceedings of the EuroLAN 2005 Workshop on Cross-Language Knowledge Induction*.
[6] Goldsmith, J. (2001): Unsupervised learning of the morphology of a natural language. *Computational Linguistics*, 27(2), pp. 153–198.
[7] Grivel, L., and Bousquet, O. (2011): A discourse analysis methodology based on semantic principles – an application to brands, journalists and consumers discourses. In *Journal of Intelligence Studies in Business* 1, pp. 76-86.
[8] Halliday, M. A. K. (1985). 1994. An introduction to functional grammar, pp. 1-32.
[9] Maingueneau, D. (1984): Genèses du discours, Mardaga, Liège, 5.
[10] Petitjean, A. (1989). Les Typologies textuelles. In *Pratiques*, no. 62, pp. 86-125.
[11] Piolat, A. and Bannour, R. (2009): An example of text analysis software (EMOTAIX-Tropes) use: the influence of anxiety on expressive writing, Current Psychology Letters [Online].
[12] Simionescu, R. (2011): *UAIC Romanian Part of Speech Tagger*, resource on nlptools.info.uaic.ro, "Alexandru Ioan Cuza" University of Iași.

Daniela Gîfu (b. January 10, 1973), licensed in Physics at the "Alexandru Ioan Cuza" University of Iași (UAIC), master degree in Communication and PR (2004, SNSPA București), PhD in Philosophy (2010, UAIC), and now she prepares her doctoral thesis in Computer Science. She finished a Postdoctoral fellowship at the UAIC (a co-tutelle between Faculty of Computer Science and Faculty of Psychology and Educational Sciences) in March 2013. Now she is scientific researcher at the Faculty of Computer Science, and she has got a temporary position of Associate Professor at the UAIC, since 2011. Her current research interests include Natural Language Processing tasks, most of them in correlation with discourse analysis. She has (co-)authored 12 books and dozens of articles, and more than 50 conferences participation. She was/is member of the Organizing Committees of the 13th International Conferences and member of the Scientific Programme Committees of the 10th International Conferences.

Marius CIOCA (b. November 22, 1969) received his M. Sc. in Computer Science (1995) and PhD in Automatic Control (2004) from Politehnica University of Bucharest. Now he is full Professor of Computer Science at Engineering Industrial and Management Department,

"Lucian Blaga" University of Sibiu, Romania. His current research interests include different aspects of Web engineering, References Arhitectures, Informatic Systems, Decision Support Systems and Data mining. He has (co-)authored more than 9 books, 70 papers, has received 7 grants and was member of 20 research projects.