Formalising Natural Languages with NooJ 2013

Formalising Natural Languages with NooJ 2013: Selected papers from the NooJ 2013 International Conference

Edited by

Svetla Koeva, Slim Mesfar
and Max Silberztein

CAMBRIDGE
SCHOLARS
PUBLISHING

Formalising Natural Languages with NooJ 2013:
Selected papers from the NooJ 2013 International Conference,
Edited by Svetla Koeva, Slim Mesfar and Max Silberztein

This book first published 2014

Cambridge Scholars Publishing

12 Back Chapman Street, Newcastle upon Tyne, NE6 2XX, UK

British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

ISBN (10): 1-4438-5824-2, ISBN (13): 978-1-4438-5824-3

TABLE OF CONTENTS

EDITORS' PREFACE[1]

NooJ is a linguistic development environment that provides tools for linguists enabling them to construct linguistic resources for formalising a large gamut of linguistic phenomena: typography, orthography, lexicons for simple words, multiword units and discontinuous expressions, inflectional and derivational morphology, local, structural and transformational syntax, and semantics.

For each resource that linguists create, NooJ provides parsers that can apply it to any corpus of texts in order to extract examples or counter-examples, to annotate matching sequences, to perform statistical analyses, etc. NooJ also contains generators which can produce the texts described by these linguistic resources, as well as a rich toolbox allowing linguists to construct, maintain, test, debug, accumulate and reuse linguistic resources.

For each elementary linguistic phenomenon to be described, NooJ proposes a set of computational formalisms, the power of which ranges from very efficient finite-state automata to very powerful Turing machines. This distinguishes NooJ's approach from most of the other computational linguistic tools, which typically offer a single formalism to their users.

Since its release in 2002, NooJ has been enhanced with new features every year. Linguists, researchers in Social Sciences and more generally, professionals who analyse texts have contributed to its development and participated in the annual NooJ conference. In 2013 the European project META-NET CESAR brought a new version of NooJ, based on the Java technology and available to all as an open source AfferoGPL project. Moreover, several companies are now using NooJ to construct business applications in various domains, from Business Intelligence to Opinion Analysis. Silberztein's article "NooJ v4" presents the latest version of the NooJ software, designed to satisfy both the needs of the academic world (through the open source Java version) and the needs of private companies (through the enhanced engineering functionalities).

The present volume contains 18 articles selected among the 43 papers presented at the NooJ 2013 International Conference held on June 3-5, at

[1] We would like to thank Ivelina Stoyanova for her help in ensuring that the texts of all articles are in correct English and Dhekra Najar for her help with the formatting of the final document.

the Deutsches Forschungszentrum für Künstliche Intelligenz GmbH (DFKI, German Research Centre for Artificial Intelligence) in Saarbrücken. These articles are organised in three parts: "Vocabulary and Morphology" contains six articles; "Syntax and Semantics" contains six articles; and "NooJ Applications" contains five articles.

The articles in the first part are focused on the construction of dictionaries for simple words and multiword units, as well as the development of morphological grammars:

— Farida Aoughlis, Kamal Nait-Serrad, Annouz Hamid, Aït-Kaci Ferroudja and Habet Mohammed Said's article "A New Tamazight Module for NooJ" describes the formalisation of the conjugation of a class of Tamazight (Berber) Verbs.
— Kaja Dobrovolic's article "Introduction to Slovene Language Resources for NooJ" presents the first Slovene module for NooJ that contains a large-coverage dictionary of over 100,000 lemmas.
— Maximiliano Duran's article "Formalising Quechua Verb Inflections" presents the inflectional grammar that describes the morphology of verbs in Quechua.
— Sandrine Fuentes and Anubhav Gupta's article "Updated Spanish Module for NooJ" presents the latest version of the Spanish electronic dictionary that has been adapted to NooJ.
— Zoe Gavriilidou and Lena Papdopoulou's article "Derivation of Multiply Complex Negative Adjectives from Verbal Stems in Greek" describes a set of morphological grammars that handles the productive derivation of Greek verbs into adjectives.
— Mario Monteleone and Simonetta Vietri's article "The NooJ English Dictionary" shows how the authors have mined the WIKI English dictionary to construct an electronic dictionary that can be used by NooJ to parse English texts.

The articles in the second part discuss the construction of syntactic and semantic grammars:

— Pilar León-Araúz's article "Semantic Relations and Local Grammars for the Environment" presents the EcoLexicon terminological knowledge database that uses NooJ local grammars to automatically extract from texts hyponymic and meronymic semantic patterns, and can even disambiguate polysemic patterns in certain cases.
— Yury Hetsevich, Sviatlana Hetsevich, Alena Skopinava, Boris

Lobanov, Yauheniya Yakubovich and Yury Kim’s article “Describing Set and Free Word Combinations in Belarusian and Russian with NooJ” presents a set of local grammars designed to recognise set and free word combinations such as quantitative expressions in Belarusian and Russian.

— Dehkra Najar and Slim Mesfar’s article “Political Monitoring and Opinion Mining for Standard Arabic Texts” presents a set of local grammars used to identify references to political actors and organisations in journalistic texts in the context of opinion expressions, and to produce annotations to mark the opinion holder, the opinion target and the polarity of the opinion (positive or negative).
— Héla Fehri, Kais Haddar and Abdelmajid Ben Hamadou’s article “Co-reference Resolution Using NooJ Recognition Process of Arabic Named Entities” presents a system based on local grammars capable of recognising referring expressions and linking them to the named entities (sportive place names) they refer to.
— Valérie Collec-Clerc’s article “Adapting Existing Japanese Linguistic Resources to Build a NooJ Dictionary to Recognise Honorific Forms” presents the set of dictionaries and local grammars that the author has created in order to extract from a corpus of Japanese texts sentences that contain honorific expressions.
— Hajer Cheikhrouhou’s article “Recognition of Communication Verbs with NooJ” presents the semantic class of communication verbs extracted from the LVF dictionary, and shows how to formalise it with NooJ, adding the Arabic translation for each lexical entry.

The articles in the third part describe applications that use NooJ:

— Sahbi Sidhom and Philippe Lambert’s article “Project Management in Economic Intelligence: NooJ as Diagnostic Tool for Nanometrology Cluster” shows how NooJ’s morpho-syntactic parser can be associated with a domain-specific knowledge organisation to construct an automatic cluster capable of managing large collections of texts and to extract interactions between actors in the domain.
— Edoardo Salza’s article “Using NooJ as a System for (Shallow) Ontology Population from Italian Texts” shows how the author used NooJ capabilities to construct an information extraction

application capable of building an ontology that contains both the concepts in the text and their relations.

— Ralph Müller's article "NooJ as a Concordancer in Computer-Assisted Textual Analysis. The case of the German module" shows how NooJ can be used in literature studies and how a better set of lexical resources would improve NooJ's usefulness dramatically for corpus analyses.
— Kristina Kocijan, Sara Librenjak and Zdravko Dovedan Han's article "Introducing Music to NooJ" presents a system built with NooJ capable of parsing sheet music. The authors have developed a set of "lexical" resources to recognise musical objects such as notes and pauses written using the LilyPond notation, and a set of syntactic grammars to recognise more complex musical objects such as chords and slurs.
— Rania Soussi, Slim Mesfar and Mathieu Faget's article "STORM Project: Towards a NooJ Module within Armadillo Database to Manage Museum Collection" presents a software application that allows users to retrieve information from a corpus of museum text collection, using queries in Arabic, English or French.

This volume should be of interest to all users of the NooJ software because it presents the latest development of the software as well as its latest linguistic resources, which are now free and distributed as open source thanks to the endorsement of the European META-SHARE CESAR project. As of now, NooJ is used as the main research and educational tool at over 30 research centres and universities across Europe and in the world; there are NooJ modules available for over 50 languages; more than 3,000 copies of NooJ are downloaded each year.

Linguists as well as Computational Linguists who work on Arabic, Belarusian, Bulgarian, English, French, German, Greek, Italian, Japanese, Quechua, Russian, Slovene and Spanish, will find in this volume state-of-the-art linguistic studies for these languages.

We think that the reader will appreciate the importance of this volume, both in view of the intrinsic value of each linguistic formalisation and the underlying methodology, as well as the potential for new applications of a linguistic-based corpus processor in the field of Social Sciences.

—The Editors

NOOJ V4

MAX SILBERZTEIN

Abstract

This paper presents the latest version of the NooJ software. We show the need for a major overhaul of the software: the CESAR META-SHARE project as well as the use of NooJ in an industrial environment. From a technical point of view, the fact that there are now three implementations of NooJ, including one in open source, has posed several problems of compatibility and protection of the resources. We present the various technical solutions that we have adopted.

Introduction

From its very beginning, NooJ was designed to become a "total" linguistic development environment, capable of formalising a large gamut of linguistic phenomena: Typography, Spelling, Inflectional and Derivational Morphology, Local and Structural Syntax as well as Transformational Syntax, and Semantics. For each of these levels of linguistic phenomena, NooJ provides users with one or more formalisation tools such as simple dictionaries, finite-state graphs and regular grammars, recursive graphs and context-free grammars, contextual and unrestricted grammars[1].

From the development point of view, NooJ provides linguists with a set of tools that allow them to construct, edit, test, maintain and share elementary pieces of linguistic description, which can be accumulated into consistent packages called NooJ modules. In order to apply the linguistic resources to texts, NooJ also includes a set of parsers that process these modules and access any property they contain — at any level of linguistic phenomena, from Typography to Semantics. NooJ is indeed used for applying linguistic modules to large texts for the purposes of automatic text annotation or information extraction.

[1] See (Silberztein 2005) and (Silberztein 2013).

Large linguistic modules for processing potentially large corpora of texts have been used in Social Sciences, e.g. in Psychological and in Literature studies[2], as well as in business-type applications[3]. Finally, the European Community got word of NooJ and the EU Competitiveness and Innovation Framework Programme Project CESAR, led by Prof. Tamas Varadi, decided to use NooJ to develop linguistic resources for several languages[4].

In the framework of the CESAR project, a team of computer scientists at the Mihajlo Pupin Institute (Belgrade) implemented Mono and Java versions of NooJ[5]. In particular, the Java version is now available as an open source package and is freely distributed under a GPL license by the CESAR project (http://www.meta-net.eu/projects/cesar).

Three Implementations for NooJ

NooJ was initially developed in the C# programming language on the .NET virtual machine, developed by Microsoft. The .NET virtual machine runs on all versions of Windows (from Windows XP to Windows 8), therefore NooJ can run on any Windows PC.

Mono-NooJ

Because .NET is proprietary technology, a group of developers decided to create an open source version of it: this initiative became the Mono project[6]. Mono is an open source implementation of Microsoft's .NET Framework based on the standard specifications of the C# programming language and the Common Language Runtime standard (used by .NET's virtual machine). Mono is largely compatible with .NET and can run most software developed with/for .NET. The Mono virtual machine is available for Mac OSX, Linux[7] and Solaris UNIX; therefore, Mono-NooJ can be used on these operating systems as well.

[2] See for instance (Ehmann 2012) and (Pignot, Lardy 2012).
[3] See (Sidhom, Lambert 2014). Also: in the framework of the STORM French national research project, Armadillo is constructing a search engine capable of performing queries in Arabic, English and French in a large corpus of Arabic texts that describe Archeological items and Architectural monuments, see (Soussi et alii 2014).
[4] Cf. http://www.meta-net.eu/projects/cesar.
[5] Cf. (Spasic et alii. 2013).
[6] See www.mono-project.com.
[7] Versions 3.X of Mono support the openSUSE desktop interface for Linux.

Note, however, that the Mono and .NET versions of NooJ are not perfectly identical: for instance, the .NET version of NooJ contains a few functionalities that are specific to the Windows operating system (such as DLLs that originated from the Windows or Microsoft Office platforms). Because these functionalities are not available on Linux, UNIX or Mac OSX, the Mono version of NooJ does not contain them: for instance, Mono-NooJ running on Linux would not be able to access neither a Microsoft Office Outlook database, nor a FrameMaker document[8].

The biggest limitation of Mono-NooJ is that it cannot process languages with non-European alphabets such as Arabic (which is written from right to left) and Khmer (in which vowels can be placed above or below consonants). This limitation was impossible to overcome, as the version of Mono available until April 2013 did not process RTF files in these languages correctly. However, the focus of the CESAR project was on European languages, so it was not considered a serious limitation. Despite these limitations, Mono-NooJ has been used by a fairly large number of Linux users.

Java-NooJ

In parallel to the effort to port NooJ on Mono, a second team at the Pupin Institute decided to translate the complete NooJ software source from C# to Java. Although the two languages have a very similar syntax, the amount of work involved proved to be challenging[9].

The two technical problems to be solved were the consequences of the differences between the C# and Java programming languages, and the differences between the .NET and Java graphical user interfaces (GUI). In particular, C# allows various methods to obtain references of parameters (via the “ref” or “out” keywords), whereas Java's methods only accept copies of parameters. To rewrite NooJ methods in Java, the team at the Pupin Institute had to encapsulate all objects that needed to be modified by a C# method, inside new, temporary objects that were then passed to the corresponding Java method. Before each method call, the temporary objects had to be created; after each method call, their content had to be copied back to the initial objects. When the method in question is called recursively, potentially millions of times (for instance, when NooJ’s syntactic parser processes a large corpus of texts), this overhead becomes significant.

[8] These capabilities are added to any Windows system as soon as Microsoft Office is installed. The Mono version of NooJ does have the capability to read .DOC files though. It would be possible to add support to Open Office file formats.
[9] Cf. (Spasic et alii 2013).

Another solution would have been to redesign NooJ's data architecture, but that was not possible given this limited resource and time frame.

The second problem was to reconstruct NooJ's GUI, because the API support to construct a GUI in Java is very different than the API support in .NET. The team at the Pupin Institute could not reuse any of the .NET GUI resources. Therefore, they had to redesign a whole new GUI from the ground up. Because of the limited resources available for this project, the Java version of NooJ has a much simpler GUI than its .NET version. However, it is perfectly suitable for educational purposes and Java-NooJ has been used successfully by a large number of students in NooJ tutorials.

In conclusion: we now have three versions of NooJ: the original .NET version that runs on Windows, the Mono version that also runs on openSUSE Linux, Mac OSX and Solaris UNIX (provided Mono is installed), and the Java version that runs on any PC (provided Java is installed).

Compatibility Issues

All "open" NooJ linguistic resources can be read by any of the three versions of NooJ. These resources are comprised of:

— Dictionaries (.dic format);
— Dictionary property definition files (.def format);
— Character Equivalence Tables (charvariants.txt files);
— Textual grammars (.nof, .nom and .nog formats);
— Any text file;

Moreover, the Mono and .NET versions of NooJ can share the following binary files:

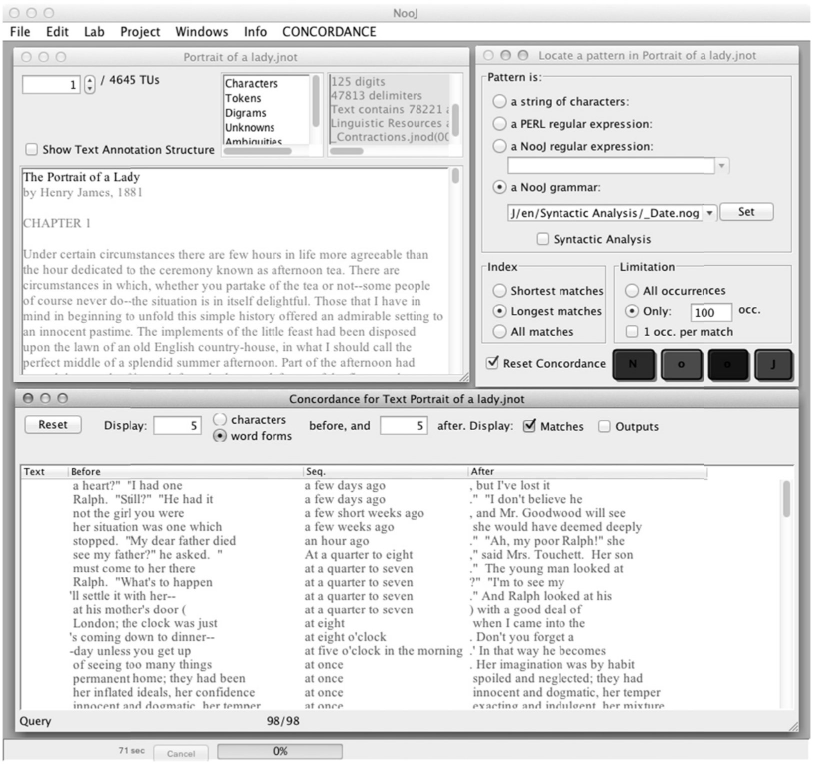
— graphical grammars (.nof, .nom and .nog formats)
— compiled dictionaries (.nod format)
— texts and corpora (.not and .noc formats)
— projects (.nop formats).

However, the Java version of NooJ cannot directly process binary files compiled in a .NET or Mono version of NooJ, and uses its own set of binary files.

Protecting the NooJ community

Although the Java version of NooJ is now open source, it was not possible to simply open access to all linguistic resources, including those posted on www.nooj4nlp.net (at the "resources" page) without their authors' permission. Indeed, several linguistic modules available for download do not contain the source file for the dictionaries, or they contain grammars that have been locked (i.e. protected).

Of course, it would not be acceptable to just publish the Java methods used to access these protected and locked files without giving away the resources. That is why the Java version of NooJ does not access binary and locked files constructed or compiled with .NET-NooJ.


NooJ
File Edit Lab Project Windows Info CONCORDANCE
Portrait of a lady.jnot
Locate a pattern in Portrait of a lady.jnot
Concordance for Text Portrait of a lady.jnot
Set
N o o J
Reset
Cancel

1 / 4645 TUs

Characters
Tokens
Digrams
Unknowns
Ambiguities

125 digits
47813 delimiters
Text contains 78221
Linguistic Resources
_Contractions.jnod(0C

☐ Show Text Annotation Structure

The Portrait of a Lady
by Henry James, 1881

CHAPTER 1

Under certain circumstances there are few hours in life more agreeable than the hour dedicated to the ceremony known as afternoon tea. There are circumstances in which, whether you partake of the tea or not--some people of course never do--the situation is in itself delightful. Those that I have in mind in beginning to unfold this simple history offered an admirable setting to an innocent pastime. The implements of the little feast had been disposed upon the lawn of an old English country-house, in what I should call the perfect middle of a splendid summer afternoon. Part of the afternoon had

Pattern is:
- ○ a string of characters:
- ○ a PERL regular expression:
- ○ a NooJ regular expression:
- ● a NooJ grammar: J/en/Syntactic Analysis/_Date.nog

☐ Syntactic Analysis

Index: ○ Shortest matches [✓ Longest matches] ○ All matches

Limitation: ○ All occurrences [✓ Only: 100 occ.] ☐ 1 occ. per match

☑ Reset Concordance

Display: 5 ○ characters [✓ word forms] before, and 5 after. Display: ☑ Matches ☐ Outputs

Text	Before	Seq.	After
	a heart?" "I had one	a few days ago	, but I've lost it
	Ralph. "Still?" "He had it	a few days ago	." "I don't believe he
	not the girl you were	a few short weeks ago	, and Mr. Goodwood will see
	her situation was one which	a few weeks ago	she would have deemed deeply
	stopped. "My dear father died	an hour ago	." "Ah, my poor Ralph!" she
	see my father?" he asked. "	At a quarter to eight	," said Mrs. Touchett. Her son
	must come to her there	at a quarter to seven	." The young man looked at
	Ralph. "What's to happen	at a quarter to seven	?" "I'm to see my
	'll settle it with her--	at a quarter to seven	." And Ralph looked at his
	at his mother's door (	at a quarter to seven	) with a good deal of
	London; the clock was just	at eight	when I came into the
	's coming down to dinner--	at eight o'clock	. Don't you forget a
	-day unless you get up	at five o'clock in the morning	.' In that way he becomes
	of seeing too many things	at once	. Her imagination was by habit
	permanent home; they had been	at once	spoiled and neglected; they had
	her inflated ideals, her confidence	at once	innocent and dogmatic, her temper
	innocent and dogmatic, her temper	at once	exacting and indulgent, her mixture

Query 98/98

71 sec 0%

The Java version of NooJ

For dictionaries, texts and corpora the situation is the following: users of Java-NooJ must compile NooJ dictionaries, texts and corpora from their source origins. The resulting files will have extensions .jnod (dictionaries), .jnot (annotated texts) and .jnoc (annotated corpora).

For grammars the situation is the following: the latest version of .NET-NooJ (v4) allows any unlocked graphical grammar (.nof, .nom or .nog format) to be exported into the Java-NooJ open file format; reciprocally, .NET-NooJ can now read any grammar that was created using Java-NooJ. The double compatibility allows users to create, edit and exchange grammars that were developed on both .NET (v4) and Java versions of NooJ.

However, in order to protect copyright, grammars that were created with .NET-NooJ and locked by their author cannot be read by Java-NooJ.

In conclusion:

— Mono and .NET versions of NooJ can create, read and share any NooJ files.

— all .dic, .def, .txt, unlocked .nof, .nom and .nog files are compatible with the new, open source Java version of NooJ;

— locked grammars, and .noc, .nod and .not compiled files cannot be accessed by Java-NooJ;

— Java-NooJ has three new binary file formats: .jnoc, .jnod and .jnot to store compiled corpora, dictionaries and texts.

The open source version of NooJ

The Java version of NooJ is available under the Affero-GPL license[10]. That means that anyone can download it and use it in any way they wish, including for commercial applications. However, there are two legal constraints:

— any software application that uses part of NooJ sources becomes automatically available under the same Affero-GPL license;

— the previous constraint includes any web service or web application that runs on a server.

The latter constraint would force any entity which uses NooJ's technology to offer services via a web server to publish the source of the application offering these services.

We believe these two constraints offer good protection for the NooJ community: there is no possibility that NooJ can "fork out", i.e. that two or more incompatible versions of NooJ get developed independently from different actors or competitors because any modification or enhancement of NooJ will instantly be available to the whole community and thus can be imported back to the NooJ "main" version.

Note, however, that the Affero-GPL license only applies to the NooJ software and it does not concern any of the many linguistic resources developed with NooJ. In other terms, NooJ users are free to develop their

[10] See http://www.gnu.org/licenses/agpl-3.0.html.

resources as they wish, using any of the three versions of NooJ, including Java-NooJ. They will still be able to decide for themselves how they wish to distribute their own resources.

NooJ v4 new functionalities

Since Java-NooJ was released by the Mihajlo Pupin Institute, Héla Fehri has taken control of the Java source and, during a "bug hunting" 3-month mission, has done an excellent job of making Java-NooJ more robust. She has already fixed a dozen simple and complex problems. The latest version of Java-NooJ has been used in several tutorial sessions without any major problems.

In parallel to the work on the Java version, I have updated NooJ to make its technology converge with Java-NooJ, and at the same time, make it more useful for industrial needs[11]. Version v4 brings the following set of new functionalities:

— The definition of multiword units has been extended to enable the processing of sequences of delimiters (such as "$" or "…"), as well as terms starting with a delimiter (such as ".NET"). In particular, these objects are now processed as ALUs and are accepted as valid NooJ lexical entries:

```
.NET,TERM+OS
$,DOLLAR,TERM+Currency
…,PUNCTUATION+EndOfSentence
```

— One consequence of the generalised multiword unit definition is that a grammar now can produce an output containing delimiters. For instance, if a grammar's output is the following:

```
<E>/$BRACKET
```

and the value of variable $BRACKET is ”<”, NooJ now clearly distinguishes the "<" character produced in the output from the meta-character "<" used to insert new annotations in the TAS. Similarly, all output keywords used by NooJ's syntactic parser to trigger some special behavior (e.g. EXCLUDE, <ONCE>), now have been unified and regarded as first-order features (e.g. +EXCLUDE, +ONCE).

[11] Many thanks to Stéphanie Brizard and her team at Invoxis for their rigorous testing and their suggestions on how to improve the NooJ experience.

— The fact that any character or sequence of characters can now constitute a valid lexical entry has pushed us to implement a robust management of special characters. Indeed, we now need to be able to process strings such as "," as bona fide lexical entries… We have introduced a double mechanism of character protection. Consider the following lexical entries:

```
\,,PUNCTUATION+Type=comma
",",PUNCTUATION+Type=comma
"abc",PUNCTUATION+Type=comma
"a\"b",PUNCTUATION+Type=quote
```

The backslash \ (escape) character is used to protect the character following it: the first lexical entry from the above examples is a comma. Double quote characters are used to protect a sequence of characters: the second entry is also a comma and both lexical entries are equivalent. In the same manner, the third lexical entry represents the exact word form "abc", i.e. the two strings "ABC" and "Abc" are not recognized by this entry. The last lexical entry contains the sequence "`a"b`", i.e. the lowercase letter "a", followed by a double quote character, followed by a single letter "b".

— The complex inheritance system that allowed linguists to link different lexical entries has been reviewed. It was inefficient because it potentially doubled the number of dictionary lookups that had to be performed for each token of a text, and could not represent differences between variants and their standard form, such as when a standard form has a property that it does not share with its variants. For instance, the dm dictionary[12]contains the following two lexical entries:

```
combatif,combattif,A+Opt
combattif,A+Rec
```

The first entry indicates that *combatif* is an optional (+Opt) spelling variant of *combattif*; the second lexical entry indicates that *combattif* is the recommended (+Rec) spelling for this term. Obviously, we don't want the optional variant *combatif* to inherit the feature +Rec from the lexical entry *combattif*. In the new system, NooJ simply links the *combatif* variant to its lemma *combattif*, without copying the lexical properties of the *combattif* lexical entry. It is now even possible to have only the first lexical entry without the second one. One drawback of the new system, compared to the old system, is that any potential property for the standard form *combattif*

[12]Cf. (Trouilleux 2012).

(as for example: +Hum, +Emotion +Aggressive) that we want to associate with its variants, such as *combatif,* now would need to be copied explicitly into every lexical entry for each of its variants.

— A number of tools and functionalities have been added to facilitate the development of large resources in an industrial environment. For instance, it is now possible to edit project (.nop) files: add, remove and/or compile new resources inside a project that has already been created. The interface in Info > Preferences has been enhanced so that resources can be picked from any source folder (it is no longer limited to a specific folder in My Documents), etc.

Perspectives

Java-NooJ is still in its infancy, but it already has been proved useful in teaching NooJ and linguistics thanks to its simplified user experience. Because its source is available to anyone for studying, it has been used as an educational tool to teach Computational Linguistics techniques to students in Computer Science. Moreover, the very fact that its source is publicly available has opened up new possibilities for cooperation with groups of scientists who have been working on specific NLP tools that can now be interfaced with NooJ's development environment and specific functionalities.

NooJ v4 is now evolving rapidly to become a powerful industrial development tool that can process new types of linguistic objects (e.g. non-alphabetical terms) in a more robust way (e.g. special treatment of special characters) and offers new tools (e.g. the new project editor) critical to an engineering approach. However, NooJ v4 is more complex to use than Java-NooJ.

At this point nobody really knows precisely how the two versions will evolve, but the Affero-GPL policy should guarantee that the two versions will always be compatible, and that any new functionality in one version can be made available in the other.

References

Ehmann B., Balazs L., Shved D., Bénet V., Gushin V., 2012. The Russian Linguistic Resources in Space Psychological Research. In *Formalising Natural Languages with NooJ*. A. Donabédian, V. Khaskarian, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Soussi R., Mesfar S., Faget M., 2014 (in this volume). Towards a NooJ module within Armadillo database to manage museum collection. In

Formalizing Natural Languages with NooJ: Selected papers of the NooJ2013 International Conference. S. Koeva, S. Mesfar, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Pignot H., Lardy M, 2013. Formalising a dictionary of 17th Century English with NooJ. In *Formalising Natural Languages with NooJ*. A. Donabédian, V. Khaskarian, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Sidhom S., Lambert P., 2014 (in this volume). Project Management within Economic Intelligence: using NooJ as a Diagnostic Tool for nanometrology clustering. In *Formalizing Natural Languages with NooJ: Selected papers of the NooJ 2013 International Conference*. S. Koeva, S. Mesfar, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Silberztein M., 2003. *NooJ Manual*. Available at: www.nooj4nlp.net.

Silberztein, M., 2005, NooJ's Dictionaries. In Actes de la *2nd Language and Technology Conference*, Poznan.

Silberztein, M., 2013, NooJ Computational Devices. In *Formalising Natural Languages with NooJ*. A. Donabédian, V. Khaskarian, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Spacic M., Milosevic U., Kovacevic N., Stanojevic M., 2013. Porting NooJ to Multiple Platforms. In *Formalising Natural Languages with NooJ*. A. Donabédian, V. Khaskarian, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

Trouilleux F., 2012. Le DM, a French Dictionary for NooJ. In Automatic Processing of Various Levels of Linguistic Phenomena: Selected Papers from the NooJ 2011 International Conference. K. Vuckovic, B. Bekavac, M. Silberztein Eds. Cambridge Scholars Publishing: Cambridge.

PART I:

DICTIONARIES AND MORPHOLOGICAL GRAMMARS

A NEW TAMAZIGHT MODULE FOR NOOJ

FARIDA AOUGHLIS, KAMAL NAÏT-ZERRAD,
HAMID ANNOUZ, FERROUDJA KACI
AND MOHAMMED SAID HABET

Abstract

In this paper we present the Tamazight linguistic module and describe its main components. The NooJ linguistic resources are the electronic dictionaries (verbs, nouns), the rules which formalize Tamazight inflectional and derivational descriptions, and the morphological grammars. We show the application of these linguistic resources, dictionaries and grammars to text.

Introduction

Tamazight (Berber) is a branch of the Hamito-Semitic linguistic family which consists of Berber, Semitic, Cushitic (East Africa), ancient Egyptian and, with a degree of more distant relatives, the “Chadic” group (Hausa). Tamazight is spoken in an area ranging from Morocco, where it is spoken by 50% of the population according to (Boukous, 1995), to Egypt, moving from Algeria, where it is used by 25 % of the inhabitants. Our study will focus on a variety of Tamazight (Kabyle), spoken in Kabylia, in northern Algeria. Tamazight language presents particular morphological phenomena mainly related to its inflectional and agglutinative morphology.

In this work, we focus on the description of dictionaries as well as the construction of morphological grammars for verbs and nouns. It should be stated that categories in the Amazigh language are related (Chaker, 1991). Berber syntactic units are organised into four large sets:verbs, nouns, connectors or relationals (prepositions, conjunctions and coordination), and various determinants.

As is the case with Semitic languages, the Berber verb (or noun) is composed of a root and a scheme. The root consists of one or more meaningful consonants (Naït-Zerrad, 1995b). The scheme provides the nouns and the verbs, for instance:

- In mmesgallen ("*they swore*"):
  The root is GL, G is the first and L the second radical consonant; since it is formed of two consonants, it is called biconsonant (Mammeri, 1990).

- Root ZDM ($C_1C_2C_3$) provides the following words:
  No Scheme: $C_1C_2C_3$[1] →zdem, (verb = "*to collect wood*").
  Scheme *an* +C_1C_2 + *a* + C_3: anC_1C_2aC_3 →anezdam (agent name = "*who collects wood*").
  Scheme *a* + C_1C_2+*a*+ C_3: C_1C_2aC_3 → azdam (verbal action name = "*collecting wood*"), (Naït-Zerrad, 1995b).

Verb

Verb stems incorporate theme, aspect and person indices. The verb is conjugated according to gender, number and person (Naït-Zerrad, 1995a).

Following André Basset works (Chaker, 1991), the majority of Berber scholars adopt a common Berber ternary system with three basic themes (intensive, aorist and preterit) marked by a set of vowel and / or consonant alternations. There are two other themes: negative preterite and intensive negative aorist.

A verb can be decomposed as follows :

Verbal form = radical or theme + affix(es).

The radical is composed of a root and a vocalic scheme, so we have:

Verbal form = lexical root + vocalic scheme + affix(es)

The scheme consists of one or more vowels that indicate the tense or the aspect of the verb. Affixes (prefixes and / or suffixes) serve as person or aspect marks. For example, for the verb aru[2] "*write*":

- turamt = t-u-r-a-mt, "*you wrote*"
  Root: *r*

[1] C1: 1st consonant, C2: 2nd consonant, etc…
[2] In "aru", the voyel "u", represents a weak radical consonant "w" (tirawt).

Scheme: *u-a* (preterit or past)
Radical or theme: *ura*
Affixes or person indices: t + *mt,* 2nd person, feminine, plural.

Verbal Themes

We describe the verbal themes (aorist, intensive and preterit):

Aorist

We found out:

- Imperative aorist (anaḍ urmir): aorist simple form corresponds to 2nd person of singular imperative,
  Krez, "*Plow!*"

- ad + aorist (ad + urmir):
  Used in particular to shape the future,
  Ad kerzeɣ, "*I will plow*"

- Negative optative : a wer + aorist
  a wer yekrez ! *"May he not plow!"*

Intensive

We found out:

- Intensive aorist (urmir ussid): This theme has an imperfective meaning,
- Itett *"he usually eats* " / *"he actually eats* ",
  Kerrzeɣ, "*I usually plow*",
- Negative intensive aorist: Ur kerrzeɣ ara, "*I don't usually plow*",
- Intensive imperative (anaḍ ussid): Kerrez, "*plow usually!*"

Preterite (or Past)

- Preterite (izri):
  This theme corresponds to perfective; it expresses achieved action in the past.
  Iwala, "*he saw*",
  Kerzeɣ, "*I plowed*".
- Negative preterit (izri ibaw):

Ur kr**i**zeɣ ara*, "I have not Plowed",*
Ur yur**i** ara*, "He didn't write".*

Participle

The participle does not contain person indices and is invariable in gender and number. It is found in the preterite, the aorist and in the intensive aorist. Its form is of 3rd masculine person:

- Preterite: yečča**n**, "(*those*) *who has/have eaten*",
- Negative preterite: (ur) **ne**čči (ara), "(*those*) *who has/have not eaten*",
- Intensive aorist : **i**tette**n,** "(*he) who eat(s)*",
- Intensive negative aorist: (ur) **n**tett, "(*those*) *who do not eat*",

Aorist: (ara) **yeččen***,* "(*those*) *who will eat*".

Aspect

In Tamazight the verb has an aspectual character-André Basset was the first scholarspecialising in Berber, who mentioned it. Aspect is a quantification of the process, not a temporal location.

Depending on the language, the aspect category (Chaker, 1991) can express very different semantic content:

- Momentary / durative,
- Perfective / imperfective,
- Single / repetitive, etc...

For instance, we have :

- Yečča*, "he ate*", perfective action.
- Ad yečč*, "he will eat*", imperfective action.

Person Indices

For the ordinary verbs conjugation, person indices (Table 1) are the same for all themes:

	Singular	Plural
1st P m f	---(e)ɣ	n(e)---
2nd P m	t(e)---(e)ḍ	t(e)---(e)m
2nd P f	t(e)---(e)ḍ	t(e)---(e)mt
3rd P m	i/y(e)---	---(e)n
3rd P f	t(e)---	---(e)nt

Table.1: Person indexes[3]

Noun

The noun paradigm includes forms for gender (feminine, masculine), number (singular, plural) and state (Naït-Zerrad, 2001).

Gender

- Generally, the masculine begins with the vowels a, *i* or *u* :
  axxam, "*house* ",
  imi, "*mouth*",
  uzzal, "*iron*".

- The feminine form is derived from the masculine by prefixation and suffixation using ***t***:
  aqcic, "*boy*" → **t**aqcic**t,**"*girl*".

- A masculine noun ending with *d* or *ḍ* ends with *ṭ* in the feminine:
  Aɣeggad, "*leather*" → taɣeggaṭ, "*belt*",
  Ayazid, "*chicken*" → tayaziṭ, "*hen*".

Number

- The singular form often begins with a vowel;
- The plural can be designated by:
  (1) Changing an initial vowel and adding a suffix (external plural):
  Axxam, "*house*" → **i**xxame**n,** "*house**s***"
  T**a**xxamt, "*room*"→ t**i**xxam**in,** "*rooms*".

[3] P: person, f: feminine, m: masculine.

(2) Vocalic changes (internal plural):
ay**a**z**i**ḍ, “*cock*” → **i**y**u**z**a**ḍ, “*cock****s***”

(3) Suffix+ vocalic changes (mixed plural):
Izi, “*fly*” → **izan** “*flies*”

State

The state is a characteristic of the noun in Berber. The two types of state are free state and constructed state.

Free State

The free state is used when an initial vowel is not subjected to modification.

(1) The noun is in isolation from any syntactic context :
Axxam, “*a house*”.

(2) The Noun can be the direct object of a verb :
Yečča aɣrum.
SUJ3ms.eat.PRET bread
“He ate the bread”.

(3) It can be a lexical subject before the verb:
Argaz yeffeɣ.
man.EL[4] SUJ3ms.go out.PRET
“The man went out”.

(4) It can be a theme indicator :
Imensi, yečča.
Dinner.EL SUJ3ms.eat.PRET.
“Regarding the dinner, he ate”.

Constructed State

The constructed state is shown through a formal variation of the first syllable in a specific syntactic context, as follows:

[4] EL: free state, EA: constructive state.

(1) A noun following a preposition (d, n, ɣer, ɣef...), exception s "*towards*" and ar "*untill*":

Yečča d weqcic.
SUJ3ms.eat.PRET with boy.EA
"*He ate with the boy*".

(2) A lexical subject following a verb:

Yečča weqcic.
SUJ3ms.eat.PRET boy.EA
"The boy ate".

(3) A noun which is an indirect object of the verb:

Yesla i teqcict mi tcennu.
SUJ3ms.listen.PRET RELI girl.EA when SUJ3FS.sing.AORI
"He heard the girl singing".

Morphological modifications consist in either keeping or suppressing an initial vowel, and adding a *w* or *u*:

(1) Masculine nouns with vowel a:
a → wa : ass.EL *"day"* → wass.EA
a → we : aqcic.EL *"boy"* → weqcic.EA
a → u : ameksa.EL *"shepherd"* → umeksa.EA

(2) Feminine nouns with vowel a:
ta → ta (no change) : tala*, "fountain".*
ta → te : tasraft.*EL, "trap"* → tesraft.*EA*
ta → t : tafunast.*EL, "cow"* → tfunast.*EA*

(3) Masculine nouns with initial i:
i →yi : iḍ.*El, "night"* → yiḍ.*EA*
i →i : igenni.*EL, "sky"* → igenni.*EA*
i →y(e) : ixxamen.*EL, "houses"* → yexxamen.*EA*

(4) Feminine nouns with initial *ti*:
ti → ti : tili *"shadow".*
ti→ te : tixxamin.*EL, "rooms"* → texxamin.*EA*
ti → t( ) : timura.*EL, "countries"* → tmura.*EA*

(5) Masculine nouns with initial u:
u → wu : uccen.EL, *"jackal"* → wuccen.EA

(6) Feminine nouns beginning with *tu* :
tu → *tu :* tuɣmest, *“tooth”* → tuɣmest.

Tamazight NooJ Module Dictionaries

The Noun Dictionary

The noun dictionary “nom.EL+EA” contains 504 entries (Kaci, 2013). Below, the example presents some entries from the noun dictionary:

```
udi,N+FLX=UDI
ugel,N+FLX=UGEL
ul,N+FLX=UL
urrif,N+FLX=URRIF
uqbel,ADV+FLX=UQBEL
ulac,ADV+FLX=UQBEL
uday,N+FLX=UDAY
ucbiḥ,N+FLX=UDAY
usu,N+FLX=UDI
ucmit,N+FLX=UDAY
uḥdiq,N+FLX=UDAY
uzzal,N+FLX=UZZAL
```

The entry udi, “*butter*”, is represented by “Udi,N+FLX=UDI”, where Udi is a noun (N), and UDI is the inflectional paradigm of the noun “*udi*”.

The inflectional paradigms are described in the inflectional file *“*nom,EL+EA.nof*”*. Eight inflectional models have been defined:

- Two for the gender (masculine, feminine)
- Two for the number (singular, plural)
- Four models for the constructed state.

Below, we give an extract of the inflectional file:

```
UDI = <E>/EL+m+s|<LW>w/EA+m+s |
 <B>awen/EL+m+p | <B>awen<LW>w/EA+m+p;
UGEL = <E>/EL+m+s | <LW>w/EA+m+s |
 <L><B><RW>an/EL+m+p | <L><B><RW>an<LW>w/EA+m+p;
UL = <E>/EL+m+s|<LW>w/EA+m+s |
 <RW>awen/EL+m+p | <RW>awen<LW>w/EA+m+p;
URRIF = <E>/EL+m+s|<LW>w/EA+m+s |
 <RW>en/EL+m+p | <RW>en<LW>w/EA+m+p;
UQBEL = <E>/EL | <LW>w/EA;
UDAY = <E>/EL+m+s | <LW>w/EA+m+s |
 <LW>t<RW>t/EL+f+s | <LW>t<RW>t/EA+f+s|
```