Formalising Natural Languages with NooJ

Formalising Natural Languages with NooJ

Edited by

Anaïd Donabédian, Victoria Khurshudian
and Max Silberztein

CAMBRIDGE
SCHOLARS
PUBLISHING

Formalising Natural Languages with NooJ,
Edited by Anaïd Donabédian, Victoria Khurshudian and Max Silberztein

This book first published 2013

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): 1-4438-4733-X, ISBN (13): 978-1-4438-4733-9

CONTENTS

EDITORS' PREFACE

NooJ is a linguistic development environment that provides tools for linguists to construct linguistic resources that formalise 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 that can produce the texts that these linguistic resources describe, as well as a rich toolbox that allows 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 makes NooJ's approach different from most other computational linguistic tools that typically offer a unique formalism to their users. Silberztein's article "NooJ computational devices" compares the different tools NooJ offers with the theoretical grammars described by Chomsky-Schützenberger's hierarchy.

Since it was released in 2002, NooJ has been enhanced with new features every year. Linguists, researchers in Social Sciences and more generally all professionals who analyse texts have contributed to its development and participated in the annual NooJ conference. Since 2011, the European project Meta-Net CESAR brought a new interest in NooJ as well as a new set of projects, both in linguistics and in computer science. The present volume contains 18 articles selected from the 32 papers presented at the International NooJ 2012 Conference which was held from June 14th to 16th at the Institut NAtional des Langues et Civilisations Orientales (INALCO) in Paris. These articles are organised in three parts: "Vocabulary and Morphology" contains five articles; "Syntax and Semantics" contains six articles; "NooJ Applications" contains six articles. In this volume, we decided to add a new part: eight short papers that present prototype NooJ modules developed by graduate students and could serve as bases for more ambitious projects.

The articles in the first part involve the construction of dictionaries for simple words, multiword units as well as discontinuous expressions as well as the development of morphological grammars:

—Thierry Declerck and Karlheinz Mörth's article "Porting Persian Lexical Resources to NooJ" shows how to extract information from various resources (TEI, Wiktionary, Wikipedia) and formalise it in order to construct a dictionary that NooJ can process automatically.

—Farida Aoughlis' article "Towards a Tamazight Module for NooJ" describes the formalisation of the conjugation of a class of Tamazight (Berber) Verbs.

—Traditionally in Belarusian or in Russian, accents are not explicitly written. Yury Hetsevich et al.'s article "Accentual Expansion of the Belarusian and Russian NooJ Dictionaries" shows how the authors have added the accent information explicitly to the Belarusian and Russian dictionaries used by NooJ.

—Hélène Pignot and Michèle Lardy's article "Formalising a Dictionary of Seventeenth Century English with NooJ" describes the process of writing a dictionary that will allow researchers (students as well as researchers in Literature studies and History) to parse seventeenth century English texts.

—Ümit Mersinli and Yasemin Erköse's article "A Deverbal Noun Generator for Turkish" shows how to formalise Turkish Noun derivation in order to generate all the derived forms for a given noun, while blocking the incorrect ones.

—Kristina Vučković et al.'s article "Derivation of Adjectives from Proper Names" formalises the production of possessive adjectives derived from proper names in Croatian.

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

—Francois Trouilleux's article "A Description of the French Nucleus VP Using Co-occurrence Constraints" shows how to implement Bès' *Properties* formalism with NooJ grammars.

—Simonetta Vietri's article "The Annotation of the Predicate-Argument Structure of Transfer Nouns" presents a set of lexical, syntactic and semantic resources that can be used to annotate sentences that express a transfer in Italian texts automatically.

—Krzysztof Bogacki and Ewa Gwiazdecka's article "Disambiguating Polish Verbs of Motion" presents a method to disambiguate verbs

of motion in Polish automatically, using both syntactic and semantic information.

—Zoe Gavriilidou et al.'s article "Numeral-Noun and Numeral-Adjective Construction in Greek" presents a set of morphological and syntactic local grammars that formalise the use of numerals in Greek.

—Ines Boujelben et al.'s article "Rule-based Approach for Semantic Relation Extraction between Arabic Named Entities" describes a set of grammars that can be used to detect relations between entities in Arabic texts automatically.

—Ivelina Stoyanova and Rositsa Dekova's article "Analysis of Translational Asymmetries in Verb Argument Structures" shows how to detect differences in argument distribution between two languages by applying NooJ to parallel (bilingual) corpora.

The articles in the third part describe applications of NooJ:

—Bea Ehmann et al.'s article "The Russian Linguistic Resources in Space Psychological Research" shows how NooJ was used as a tool by psychologists to perform content analysis of the Mars-500 crew communication.

—Saida Ben Kacem and Slim Mesfar's article "Towards an On-line Concordance Service" shows how they encapsulated NooJ's technology into a WEB service in order to construct a WEB server that provides users with a sophisticated concordance processor.

—Danijela Merkler and Zeijko Agic's article "Sentiscope: A System for Sentiment Analysis in Daily Horoscopes" shows how they used NooJ to perform sentiment analysis in daily horoscopes written in Croatian.

—Lena Papadopoulou and Giannis Anagnostopoulos' article "A Model Procedure for the Enrichment and the Evaluation of the Greek NooJ Module" shows how to use available pre-tagged corpora to enrich and evaluate the quality of NooJ dictionaries.

—Héla Fehri et al.'s article "Specific NooJ Resources for the Recognition and the Translation of Arabic Sports Organization Names" presents an automatic system capable of recognizing and translating sports entities automatically.

—Mei Wu's article "The Auxiliary Verbs in NooJ's French-Chinese MT system" describes the formalisation of the French auxiliary verbs described in the LVF dictionary and its application in a French to Chinese Machine Translation system.

The short papers in the fourth part describe prototypes that were constructed with NooJ:

—Mohamed Mahdi Boudabbous et al.'s paper "Using NooJ Grammar to enrich AWN Semantic Relation" shows how to extract semantic relations between nouns by applying a set of NooJ local grammars to a corpus; these relations can then be used to enrich the Arabic WordNet ontology.
—Maximiliano Duran's paper "Formalising Quechua Noun Inflection" describes the first effort to formalise the Quechua vocabulary.
—Charles Faivre's paper "Towards a NooJ Module for Malagasy" presents a NooJ prototype capable of parsing a small text in Malagasy, using a dictionary and a morphological grammar.
—Iskander Keskes' paper "Discourse Segmentation of Arabic Texts Using Cascading Grammars" presents a prototype capable of segmenting Arabic texts automatically by applying three typographical NooJ local grammars in cascade.
—Liana Khachatryan's paper "An Armenian Grammar for Proper Names" presents a set of local grammars that can be applied to texts written in Western Armenian in order to find proper names automatically.
—Mirko Spasić et al.'s paper "Porting NooJ to Multiple Platforms" shows how the Pupin Institute's team has ported NooJ both to Mono and Java, allowing NooJ to run on practically all current Operating Systems.
—Sandrine Traïdia's paper "Formalising the Isafe Constructions in Sorani Kurdish" presents a set of grammars that can be used to disambiguate this ambiguous particle.
—Masako Watabe's paper "A NooJ Module for Rromani" presents a prototype of a set of linguistic resources capable of processing nouns and their suffixation in Rromani.

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. Note that NooJ and its linguistic resources are free and will soon be published 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 pedagogical tool by over 30 research centres and universities in 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 Albanian, Arabic, Armenian, Belarusian, Berber, Chinese, Croatian, French, Greek, Italian, Kurdish, Malagasy, Persian, Polish, Quechua, Rromani, Russian, Turkish as well as on seventeenth century English 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 for the intrinsic value of each linguistic formalisation and the underlying methodology, as well as for the potential for new applications of a linguistic-based corpus processor in the Social Sciences.

—The Editors

NOOJ COMPUTATIONAL DEVICES

MAX SILBERZTEIN

Introduction

NooJ's linguistic development environment provides tools for linguists to construct linguistic resources that formalise 7 types of linguistic phenomena: typography, orthography, inflectional and derivational morphology, local and structural syntax, and semantics. NooJ also provides a set of parsers that can process any linguistic resource for these 7 types, and apply it to any corpus of texts in order to extract examples, annotate matching sequences, perform statistical analyses, etc.[1]

NooJ's approach to Linguistics is peculiar in the world of Computational Linguistics: instead of constructing a large single grammar to describe a particular natural language (e.g. "a grammar of English"), NooJ users typically construct, edit, test and maintain a large number of local (small) grammars; for instance, there is a grammar that describes how to conjugate the verb *to be*, another grammar that describes how to state a date in English, another grammar that describes the heads of Noun Phrases, etc. NooJ then takes charge of combining all the local grammars together, even—it is actually the most frequent case—when these local grammars have a very different nature across the 7 types of linguistic phenomena.[2]

Henceforth, I use the terms *letter* and *word* when giving examples of orthographical or morphological linguistic resources: an orthographical grammar is typically used to describe sequences of *letters* that constitute

[1] NooJ also contains a set of generators that can produce the sequences of texts that these linguistic resources describe. Combining a parser with a generator allows users to develop software applications such as an automatic paraphrase generator (that parses a given sentence and produce all its paraphrases) and an automatic translation system (that parses a given sentence in one language and produces the corresponding translation in another language).

[2] Thanks to its annotation engine (cf. Silberztein 2006).

certain *words*. I will also use the terms *Atomic Linguistic Units* (ALU)[3] and *phrases* when giving examples of syntactic linguistic resources: a syntactic grammar describes sequences of *ALUs* that constitute *phrases*.

NooJ users handle three types of objects: the linguistic phenomenon they are studying constitutes a *language*; they construct a *grammar* to describe it, and the *parser* tests membership of a given word (or phrase) to the language. Because the linguistic phenomena to be described are very different in nature, NooJ provides different types of grammars to formalise them and parsers to apply the grammars to texts.

The Chomsky-Schützenberger Hierarchy

Chomsky (1957) presented a mathematical model for grammars in which grammars are sets of rules of the form: $\alpha \rightarrow \beta$, where β and β are sequences of symbols and sequence β is to be replaced with sequence β. By applying a number of rules in sequence to a given word, an automaton can automatically test whether or not this word belongs to the language described by the grammar. Depending on the constraints on rules (e.g. sequence β may contain only one symbol, or more than one symbol), the grammars are more or less powerful, i.e. they can describe more or less complex languages. The hierarchy of languages is the following:

> Set of regular languages $\subset$ Set of context-free languages $\subset$ Set of context-sensitive languages $\subset$ Set of (any) languages

Each type of language corresponds to a type of grammar that can describe it, and to a type of automaton that can test if a given word belongs to a language according to its grammar.

Chomsky (1957) then argued that certain phenomena in natural languages are not regular, hence regular grammars cannot describe natural languages. Since then, researchers in Computational Linguistics have designed a large number of tools to formalise grammars. Today, the most famous of these tools are XFST,[4] GPSG,[5] LFG[6] and HPSG.[7]

---

[3] NooJ's ALU are the elements of the vocabulary of a language. ALUs are simple words (e.g. *table*), prefixes or suffixes (e.g. *dis-* in "dismount", *-ation* in "demonstration"), multiword units (e.g. *red tape* when meaning "bureaucracy") as well as discontinuous expressions (e.g. *to take X into account* in "John took the meeting into account").

[4] (Cf. Karttunen et al. 1997)

[5] (Cf. Gazdar et al. 1985)

Language	Grammar	Automaton
Regular	Regular	Finite-State Automaton
Context-Free	Context-Free	Push-down Automaton
Context-Sensitive	Context-Sensitive	Linear Bounded Automaton
Any language	Unrestricted	Turing machine

Table 1: The Chomsky-Schützenberger Hierarchy

All these tools align themselves with the Chomsky-Schützenberger hierarchy: for instance XFST's parser uses Regular Grammars with a very efficient parser; GPSG uses modified Context-Free Grammars; LFG's grammars are more powerful than CFGs, however its parsers are less efficient; HPSG's grammars are the most powerful, however HPSG has inefficient parsers that makes it unsuitable for Corpus Linguistics applications.

The NooJ Approach

This hierarchical approach has two problems:

—It is not because there are complex phenomena in a natural language that we need to describe all linguistic phenomena with a powerful grammar. In practice, most morphological, lexical and syntactic phenomena can be described with Regular Grammars easily and very efficiently. Why should we have to use a complex formalism (and an inefficient parser) to process them?

NooJ answers this question by providing different types of grammars and parsers: with NooJ, a linguist can describe spelling variants of a term with a Regular Grammar (RG), then use a Context-Free Grammar (CFG) to compute the structure of a complex sentence, and then a Context-Sensitive Grammar (CSG) to check the agreement between a noun phrase and its pronoun and apply an Unrestricted Grammar (UG) to produce paraphrases of a given sentence. NooJ's architecture allows all these linguistic phenomena to be combined in a Text Annotation Structure (cf. Silberztein 2006).

---

[6] (Cf. Kaplan, Bresnan 1982)
[7] (Cf. Pollard, Sag 1994)

—There seems to be confusion between the sets of languages and the languages themselves. Although it is true for instance that the set of all RGs is included inside the set of all CFGs, it does not mean that Regular languages are somehow "smaller" than Context-Free languages.

In fact, any language (be it Context-Free, Context-Sensitive or even unrestricted) is included in a Regular Language. NooJ uses this fundamental property to provide a two-component approach: in order to describe any language L, a NooJ user constructs both an RG that describes a Regular language R that is a superset of language L, and a "filter" component that filters out all the words of R that do not belong to L. We will see that this approach is very natural linguistically; it also makes NooJ parsers very efficient because they can parse any language with finite-state automata.

Regular Grammars

In order to formalise Regular languages, NooJ users can enter Regular Expressions or Finite-State Graphs. Here is a typical NooJ Regular Expression:

(a | the) (very* big | <E>) table

The "|" character corresponds to the disjunction operator, the "*" character corresponds to the Kleene operator; <E> represents the empty string; parentheses can be used to set priorities (by default, concatenation has priority over disjunction). Here is a typical NooJ Finite-State Graph:

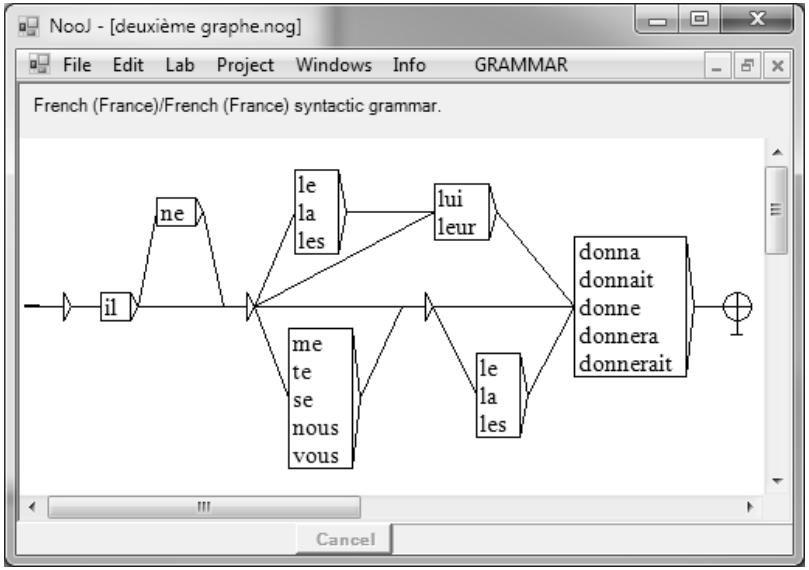


Figure 1: A Finite-State Graph

NooJ graphs are sets of nodes; nodes are labelled by a regular expression; any two nodes can be connected; there is one initial node and one terminal node. This graph represents the French sequences of preverbal particles that occur between the subject pronoun *il* and the transitive verb *donner*.

Context-Free Grammars

In the Chomsky-Schützenberger hierarchy, Context-Free Grammars (CFG) contain rules such as: **A→βBβ**, where A and B are auxiliary symbols. NooJ CFGs contain auxiliary symbols as well; here is a typical NooJ CFG:

```
Main = :NP (looks at | sees) :NP ;
NP = (the | a) (cat | dog) ;
```

In NooJ, auxiliary symbols are prefixed with the special character ":". NooJ graphs can also include auxiliary nodes, i.e. nodes labelled with an auxiliary node (which is displayed in yellow, see below) that link to an embedded graph. NooJ recursive graphs are also equivalent to CFG.

Enhancements

NooJ also contains a few enhancements over pure Regular Grammars and Context-Free Grammars.

Symbols

NooJ allows users to use lexical and syntactic symbols that function like abbreviations. For instance, in the English module, the lexical symbol <be> is equivalent to the following regular expression:

am | are | is | was | were | being | been

In the French module, the lexical symbol <manifester> [to demonstrate] matches any conjugated form of the verb *manifester*, as well as all inflected forms of its derived forms *manifestation* [demonstration] and *manifestant* [demonstrator].

In the French module, the syntactic symbol <CONJC> (coordinating conjunction) is equivalent to the following Regular Grammar:

mais | ou | et | donc | or | ni | car

In the same manner, the syntactic symbol <V> matches over 30,000 verbal forms in English and over 300,000 in French: all the conjugated forms of lexical entries associated with the category code "V", *i.e.* Verbs. Any lexical property can be used in a lexical or syntactic symbol. For instance, symbol <N-Hum+Medic-m+p> matches all the nouns (N) that are not Human (-Hum), belong to the Medical domain (+Medic), are not masculine (-m) and are in the plural (+p).

Adding a lexical or a syntactic symbol to NooJ is straightforward: just add a code in a dictionary. It can be argued that NooJ symbols are more than mere abbreviations because they make the alphabet of NooJ grammars potentially infinite in size.

Order

NooJ grammars allow the use of a few operators that can help reduce the size of grammars drastically. For instance, let's say we want to describe sentences that have one subject, one verb, one direct object and one indirect object, e.g. *Eva gave the pencil to Joe*. In languages that have cases (*e.g.* nominative, accusative and dative), these four components

might be written in any order. The corresponding grammar would then look like:

> Verb Nominative Accusative Dative | Verb Nominative Dative Accusative | Verb Accusative Nominative Dative | Verb Accusative Dative Nominative | Verb Dative Nominative Accusative | Verb Dative Accusative Nominative | Nominative Verb Accusative Dative | Nominative Verb Dative Accusative | Accusative Verb Nominative Dative | Accusative Verb Dative Nominative | Dative Verb Nominative Accusative | Dative Verb Accusative Nominative | Nominative Accusative Verb Dative | Accusative Nominative Verb Dative | Nominative Dative Verb Accusative | Dative Nominative Verb Accusative | Accusative Dative Verb Nominative | Dative Accusative Verb Nominative | Nominative Accusative Dative Verb | Nominative Dative Accusative Verb | Accusative Nominative Dative Verb | Accusative Dative Nominative Verb | Dative Nominative Accusative Verb | Dative Accusative Nominative Verb

A grammar that recognizes sentences with *n* components contains *n*! terms: 4! = 24, 5! = 120, 6! = 720, etc. As soon as we add circumstantial complements, the size of the grammar would literally explode… By contrast, consider the following NooJ recursive graph:

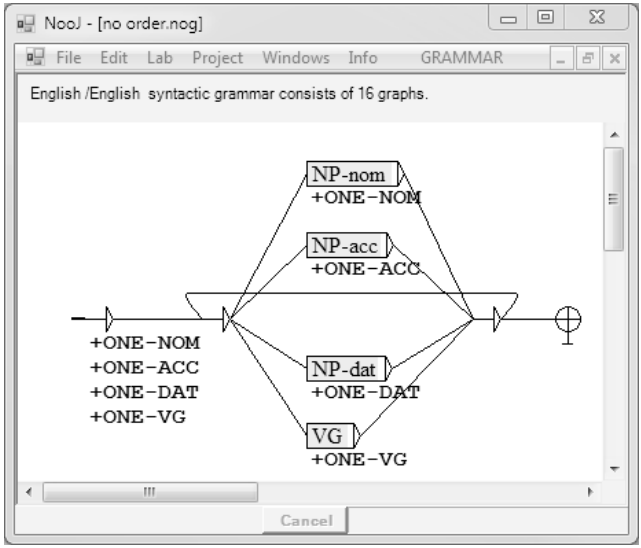


Figure 2: Check that each argument occurs only once

This graph recognizes any number of noun phrases in the nominative (NP-nom), in the accusative (NP-acc), in the dative (NP-dat) as well as any number of verbs (VG). The operator +ONE then filters out all the sequences that do not contain exactly one occurrence of each component. NooJ contains other operators (EXCLUDE, ONCE, UNAMB) that also aim to simplify grammars.

Context-Sensitive Grammars

In the Chomsky-Schützenberger hierarchy, Context-Sensitive Grammars (CSG) accept rules such as: $C\beta \rightarrow C\beta$, where C is a context that needs to occur for β to be replaced with β. NooJ's CSGs are RGs or CFGs that contain variables and constraints.[8] For instance, consider the Context-Sensitive language $a^n b^n c^n$ that contains all the words that are sequences of a number of "a", followed by the same number of "b" followed by the same number of "c". In NooJ, we can describe this language with the following CSG:

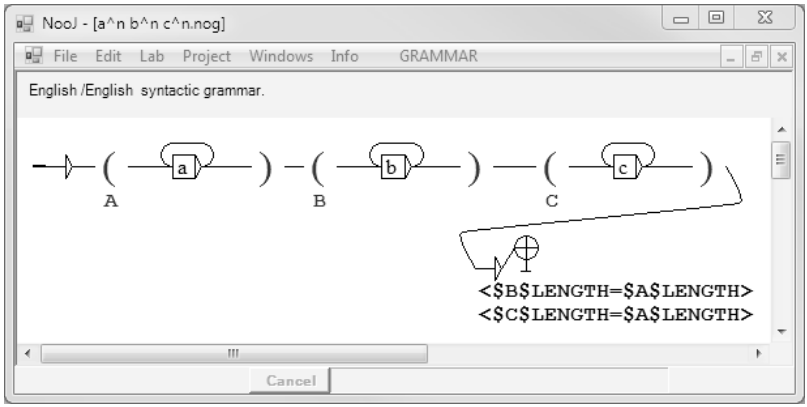


Figure 3: Context-Sensitive Grammar

The finite-state graph part of this grammar recognizes any sequence of "a" followed by any number of "b" followed by any number of "d". Then, the sequence of "a" is stored in variable $A, the sequence of "b" is stored

[8] NooJ's parser uses a simple and efficient unification mechanism (cf. Silberztein 2011).

in variable $B and the sequence of "c" is stored in variable $C. Finally, the two constraints check that the length of the three sequences are identical.

There are a number of linguistic phenomena that need to be formalised by CSGs. For instance, it is much easier and natural to describe the agreement with a CSG than with a RG.

Consider the following RG that describes certain noun phrases in French. The four quasi-identical paths in the graph correspond to each of the four possible types of noun phrases in French: masculine singular, feminine singular, masculine plural and feminine plural. For each of these types, we need to ensure that the determiner, the noun and the two adjectives agree both in number and in gender.

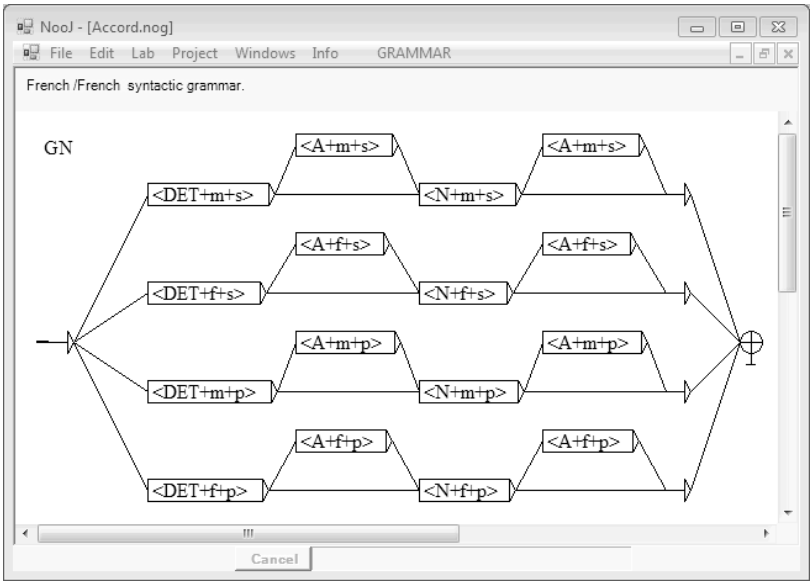


Figure 4: A redundant Finite-State Graph

The same phenomenon can be described with the following CSG:

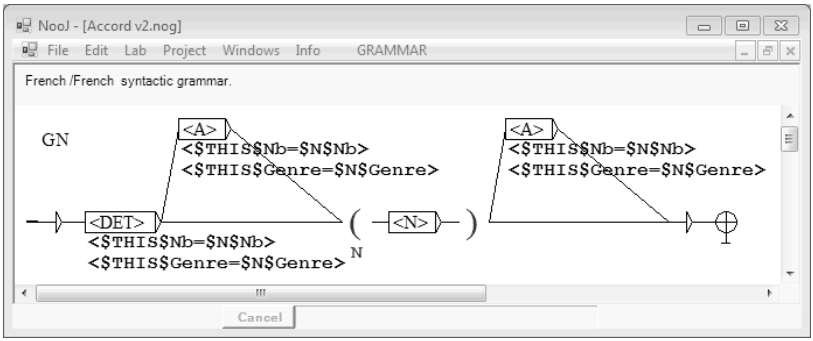


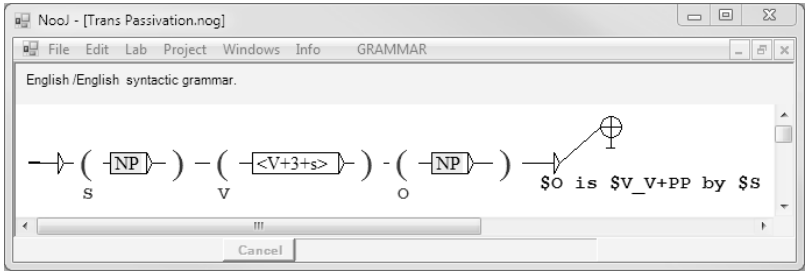
Figure 5: An equivalent Context-Sensitive Graph

In this graph, we describe the Noun Phrase only once, and we add agreement constraints for each component. There are a large number of linguistic phenomena (such as morphological and syntactic reduplications, productive derivations, etc.) that can be described with NooJ CSG very naturally.

Unrestricted Grammars

In the Chomsky-Schützenberger hierarchy, Unrestricted Grammars (UG) include rules such as: $\beta \rightarrow \beta$, where there is no constraint whatsoever on the content of β nor β. These rules correspond to a special "replacement" mode in NooJ, which basically can perform any number of replacements to a given text.[9] The most linguistically useful unrestricted grammars are transformational grammars, which take a sentence as an input and produce another sentence as the output. For instance, consider the following graph:

[9] NooJ can parse a sentence, and then apply a generator that produces all the corresponding paraphrases (cf. Silberztein 2010).



The CFG recognizes direct transitive sentences such as "John eats an apple", then stores the subject in variable $S, the verb in variable $V and the object in variable $O. Finally, the grammar produces the output $O is $V_V+PP by $S, in which $V_V+PP takes the verb "eats" and then produces its Past Participle (PP) form "eaten". The resulting sentence is then "the apple is eaten by John".

Conclusion

NooJ proposes several types of grammars to formalise seven types of linguistic phenomena: Regular Grammars, Context-Free Grammars, Context-Sensitive Grammars as well as Unrestricted Grammars. NooJ's architecture makes it easy to combine large numbers of local (small) grammars of any type. This "engineering approach", where users are encouraged to develop a large number of linguistic resources rather than a single large (powerful) grammar brings also various advantages, such as the possibility of designing very efficient parsers.

References

Chomsky, Noam. 1957. *Syntactic Structures*. Mouton: The Hague.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Oxford: Blackwell, and Cambridge, MA: Harvard University Press.

Kaplan, Ronald, and Joan Bresnan. 1982. "Lexical-Functional Grammar: A formal system for grammatical representation". In *The Mental Representation of Grammatical Relations*, edited by Joan Bresnan, 173–281. Cambridge: The MIT Press.

Karttunen, Lauri, Tamás Gaál and André Kempe. 1997. Xerox Finite-State Tool. Technical report, Xerox Research Centre Europe.

Pollard, Carl, and Ivan Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Silberztein, Max. 2003. *NooJ Manual*, available at http://nooj4nlp.net.

—. 2006. "NooJ's Linguistic Annotation Engine". In *INTEX/NooJ pour le Traitement Automatique des Langues*, edited by S. Koeva, D. Maurel and M. Silberztein, 9-26. Les Cahiers de la MSH Ledoux. Presses Universitaires de Franche-Comté.

—. 2010. "Automatic Transformational Analysis and Generation". In *Proceedings of the NooJ 2010 International Conference and Workshop*, 221-231. Komotini: Thrace University Ed.

—. 2011. "Variable Unification with NooJ v3". In *Automatic Processing of Various Levels of Linguistic Phenomena*, edited by Kristina Vučković Božo Bekavac, and Max Silberztein, 1-13. Newcastle upon Tyne: Cambridge Scholars Publishing.

PART ONE:
VOCABULARY AND MORPHOLOGY

PORTING PERSIAN LEXICAL RESOURCES TO NOOJ

THIERRY DECLERCK AND KARLHEINZ MÖRTH

Introduction: A Small Trilingual Dictionary

The starting point of our on-going experiments was a manually assembled digital Persian-English-German dictionary, which has been used in digital language learning. An example for an entry (encoded in TEI P5[1]) is given below in Figure 1.

Structure of our Dictionary Source

The encoding system used to prepare our source dictionary is TEI P5. While the TEI dictionary module is the *de facto* encoding standard for dictionaries digitised from print sources—as such, "TEI for dictionaries" has a meanwhile long-standing tradition—using this system for machine readable dictionaries in the context of NLP applications is a rather new idea which has been discussed repeatedly in the more recent past.[2]

In order to make the TEI dictionary module usable for NLP purposes, it was necessary to tighten the many combinatorial options of TEI P5. Practically, this was done by a document type definition allowing only a limited set of combinations of elements. While a high degree of flexibility is a necessary prerequisite for encoding a wide range of different digitised print dictionaries, imposing restrictions on the system is inevitable when you want to use it in software applications. All of the work on this TEI P5 schema has been carried out with an eye to other relevant standards in the field such as LMF (Lexical Markup Framework; ISO-24613:2008) and MAF (Morpho-syntactic annotation Framework). This customisation of the TEI P5 dictionary module encoding system that was meant to function

[1] http://www.tei-c.org/Guidelines/P5/

[2] The workshop "Tightening the representation of lexical data, a TEI perspective" at the TEI Members' Meeting 2011 (Würzburg, Germany) had a considerably large audience.

as a multi-purpose system targeting both human users and software applications has already been put to use successfully in various lexicographic projects of our institute and proved to be a solid bedrock for our lexicographic work (references to be added in the final, non anonymous version).

```
    <entry xml:id="ketaab_001">
      <form type="lemma">
        <orth xml:lang="fa-Arab">کتاب</orth>
        <orth xml:lang="fa-x-modDMG">ketāb</orth>
      </form>
  <gramGrp><gram type="pos">noun</gram></gramGrp>
  <form type="inflected" ana="#n_pl">
        <orth xml:lang="fa-Arab">کتب</orth>
        <orth xml:lang="fa-x-modDMG">kotob</orth>
      </form>
      <form type="inflected" ana="#n_pl">
        <orth xml:lang="fa-Arab">کتاب ها</orth>
        <orth xml:lang="fa-x-modDMG">ketāb-hā</orth>
      </form>
      <etym><lang>Arabic</lang></etym>
      <sense>
        <cit type="translation" xml:lang="en">
          <quote>book, scripture</quote></cit>
      </sense>
      <sense>
        <cit type="translation" xml:lang="en">
          <quote>letter</quote></cit>
      </sense>
</entry>
```

Figure 1: Example of a Persian entry in our small dictionary, using TEI encoding

Expanding the Original Dictionary through Merging with Additional Language Resources

In need of more entries and more detailed linguistic information we have started to direct our efforts in enriching our dictionary with language data from freely accessible sources.

There are as of today only very few freely available Modern Persian language resources. We can have access to the Hamshahri Collection (Ale Ahmad 2009), a corpus of newspaper texts,[3] and to the Bijankhan Corpus (Amiri 2007), a tagged corpus made up of both newspapers and other texts.[4] A very useful Treebank has been built on the basis of the latter one (Ghayoomi 2012) and been made available recently.[5] We do not know of any larger corpora of Modern Persian. Tools to add word class (POS) and lemma information to digital texts have remained scarce (Raja 2007). The largest lexical resource in the public domain is definitely the Persian language version of the collaborative Wiktionary project. In short, Persian is still a less-resourced language, particularly with regard to its comparatively large number of speakers.[6]

To conduct our experiments in automatic enhancement of our Persian lexicographic database, we made use of four different sources, from each of which particular types of data were drawn. These resources are the Persian language version of Wikipedia, the Persian Wiktionary (in Persian *Wiki-wāže*), the English Wiktionary and the above mentioned Persian Treebank.

Persian Wikipedia

The Persian edition of Wikipedia is quite sizeable. It belongs to the first category of Wikipedia editions, which should contain more than 100,000 articles. In February 2012 the main page of Persian Wikipedia indicated a number of 170,000 articles. According to the number of Wikipedia articles, those concerning the Persian language number 24.

---

[3] Downloadable at http://ece.ut.ac.ir/dbrg/hamshahri/

[4] Downloadable at http://ece.ut.ac.ir/dbrg/bijankhan/

[5] For more about conditions, see:
http://hpsg.fu-berlin.de/~ghayoomi/PTB/TermCon.html

[6] Numbers given in various publications range from 80 to 110 million native speakers. In terms of numbers of speakers, the Ethnologue site assigns Persian rank number 34 among the languages of the world (http://www.ethnologue.com/ethno_docs/distribution.asp?by=size).

The data gleaned from this resource yield above all candidates for new dictionary entries (but without linguistic information) and one translation equivalent. The list created from this resource was particularly helpful with regard to lexical items that could not be found in printed bilingual Persian dictionaries.[7] The associated Wikipedia categories can also be very helpful for the semantic classification of the potential entries.

Wiktionary

Wiktionary is the lexicographic counterpart of the encyclopaedic project Wikipedia. It is currently (Feb. 2012) available in 158 languages, although only a small number of these versions is sufficiently large to be useful. An additional argument often raised against these dictionaries is that they are not edited by professional lexicographers but by enthusiastic volunteers. The most regrettable drawback of this project is that the content of the Wiktionary database is formatted in a lightweight mark-up system commonly used in Wiki applications. This system is neither standardised nor very structure-oriented. To acerbate the situation, it is often applied in a considerably inconsistent manner, which makes extracting structured information a really challenging task.

But on closer inspection, many of the larger versions of Wiktionary turn out to be quite valuable treasure troves, and it seems worthwhile to develop programs that transform the Wiktionary formats into a more structured representation. Thus both computational linguists and lexicographers have used those steadily growing language resources in various experiments to pursue monolingual as well as multilingual studies by means of computational methods. But to our knowledge nobody has done this for Persian so far, and converting Wiktionary into a standard representation like TEI has not been done often either.[8]

Persian Wiktionary

The Persian Wiktionary is ranked 30 if one considers the number of entries: 68,582 (as of Nov. 2011). As far as data extraction is concerned, this resource turned out to be the hardest part of our work. Data that could be gained thereof are: particular etymologies, references to sources whence the information given in particular entries was taken, some

[7] Regrettably, Persian lexicography has been in a deplorable state for quite some time. The majority of noteworthy publications were produced a couple of decades ago.

[8] http://wiktionary-export.nataraj.su/en/about.html

morphological data, some domain-specific semantics (the Wiktionary "Categories" associated with entries) and many translation equivalents.

English Wiktionary

The English Wiktionary contains a great number of (partial) entries in languages other than English.[9] One of those is Persian. In addition to the data taken from the Persian language Wiktionary, we also made use of this data as it was adapted to the English system of Wiktionary-encoding which appears to have been applied in a much more consistent manner. We could extract ~5500 Persian language entries from the English Wiktionary which proved to hold a large amount of reusable information. In addition to several of the above listed data categories, many of these entries contained the pronunciation of the Persian lemmas transcribed in the system of the International Phonetic Association (IPA). On the character level, most of these were neatly encoded making use of Unicode. Furthermore, most of the entries were furnished with labels semantically categorizing the lemmas, which could easily be used for domain-specific NLP applications. We stress here that the semantic categories used in the English Wiktionary are consistently used over all the languages present in the lexicon, so that a kind of cross-lingual Wiktionary-Net can be built. A screen-shot of the entry for the Persian word "Republic" is shown below in order to illustrate the type of information we can extract from the XML dump of the English Wiktionary, and include in our dictionary.[10]

From the XML dump, we can not only extract morpho-syntactic information, domain information ("fa:Forms of Government") and the English translation ("republic"), but we can also draw all the other Persian entries that refer to anything labelled "form of government" or to the higher category of "government", and via the English entry for "republic" to all other available translations (ca. 30 languages).

---

[9] In fact the English Wiktionary edition contains entries for more than 400 languages, so that out of this source, more language specific wiktionaries could be created than there are actually officially listed. What is meant by "English Wiktionary" is in fact that all entries listed are explained and described in the English language.

[10] We should like to stress here that this lexical data has also been mapped onto a computational lexicon, within the linguistic development platform NooJ (www.nooj4nlp.net), creating a totally new resource for this platform. This new resource will very soon be made available on the NooJ resource web page (http://www.nooj4nlp.net/pages/resources.html). The full-form lexicon will also be made available to non NooJ users, in the TEI format.