Automatic Processing of Various Levels of Linguistic Phenomena: Selected Papers from the NooJ 2011 International Conference

Automatic Processing of Various Levels of Linguistic Phenomena: Selected Papers from the NooJ 2011 International Conference

Edited by

Kristina Vučković, Božo Bekavac
and Max Silberztein

CAMBRIDGE SCHOLARS
PUBLISHING

Automatic Processing of Various Levels of Linguistic Phenomena:
Selected Papers from the NooJ 2011 International Conference,
Edited by Kristina Vučković, Božo Bekavac and Max Silberztein

This book first published 2012

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): 1-4438-3711-3, ISBN (13): 978-1-4438-3711-8

TABLE OF CONTENTS

Part II: Syntax

Part III: Applications

EDITORS PREFACE

NooJ is a linguistic development environment that allows linguists to formalize a wide gamut of linguistic phenomena: orthography and typography, inflectional and derivational morphology, lexicons for simple words, multiword units as well as discontinuous expressions, local and structural syntax, transformational syntax and semantics. For each type of linguistic phenomena, NooJ provides linguists with specific computational devices (e.g. finite-state machines, context-free and contextual grammars as well as recursive transition networks) associated with tools designed to help linguists test, debug, accumulate, combine and manage large sets of resources.

Every year since it was first released in 2002, NooJ has been regularly enhanced with new features. Linguists, social scientists and more generally researchers who are interested in corpus processing have contributed to its development and participated in the annual NooJ conference. The 2011 conference was no exception and the participation of the European Meta-Net CESAR project (which announced that NooJ will soon be available as an Open Source) is testimony to the success of NooJ.

The present volume contains a selection of 22 articles, based on a selection from the 39 papers that were presented at the *International NooJ conference* which was held from June 13th to 15th 2011 in Dubrovnik, Croatia.

For its new v3 version, NooJ's linguistic engine has been significantly rebuilt, bringing a clean unification mechanism that helps design grammars that can be used both for syntactic analysis and transformational generation (see Max Silberztein's paper "Variable Unification in NooJ v3"). The volume is then organized in three parts: "lexical" articles deal with the formalization of vocabularies and present solutions to various lexical and morphological problems; "syntactic" articles present implementations of sets of syntactic grammars and their application (language learning, machine translation or disambiguation); "corpus" articles present various applications of the analysis of corpora in Medicine and in the social sciences.

The articles in Part I involve the construction of dictionaries for simple words, multiword units or discontinuous expressions as well as the development of morphological (inflectional or derivational) grammars:

—François Trouilleux's article “Le DM, a French Dictionary for NooJ” presents a new, open source electronic dictionary that covers the standard French vocabulary as well as its inflectional morphology;

—Yury and Sviatlana Hetsevich's article “Overview of Belarussian and Russian electronic dictionaries and their adaptation for NooJ” provides a showcase of how to port existing dictionaries into the NooJ formalism;

—Maria Georganta and Lena Papadopoulou’s article “Towards an ancient Greek NooJ module” presents the project of building a NooJ module capable of parsing ancient Greek texts, based on specific lexical and morphological resources;

—Krzysztof Bogacki and Ewa Gwiazdecka’s article “Derivational structure of Polish verbs and the expansion of the dictionary” shows how they formalized verbs’ derivation in Polish, using a combination of syntactic, derivational and morphological grammars;

—Zoe Gavriilidou et alii’s article “Processing Greek frozen expressions with NooJ” describes a set of dictionaries capable of describing multiword units and discontinuous expressions consistently;

—Liana Khachatryan’s article “Formalization of proper names in the western Armenian press” presents a project that aims at constructing large-coverage dictionaries for Proper names, using both dictionaries and local grammars;

—Umut Demirhan and Mustafa Aksan’s article “Tagset for NooJ Turkish Module” presents a new morphological analyser for Turkish that can be used to perform complex queries on parts of speech and affixes;

—Finally, Ranka Stanković et alii’s article “On the compatibility of lexical resources for NooJ” compares the dictionaries available for the various languages formalized with NooJ, and proposes a framework based on XML that would help build a standard for lexical resources.

Part II of the volume is dedicated to the formalization of the various syntactic constructs in texts, as well as the construction of syntactic mechanisms that help disambiguate lexemes and structures:

—Božo Bekavac and Krešimir Šojat’s article “Syntactic patterns of verb definitions in Croatian WordNet” presents a set of ten syntactic grammars that can be used to parse definitions in WordNet;

—Tereza Jurić et alii’s article “Direct speech recognition in text” shows us how syntactic grammars can be used to recognize and extract direct speech citations in newspapers and literary works;

—Sara Librenjak's article "Using NooJ in SLA: helping learners by extracting and organizing Japanese grammar" shows how simple NooJ grammars can be used efficiently to help students recognize certain terms and expressions in Japanese proficiency tests;

—Ümit Mersinli's article "Disambiguating Turkish with NooJ" studies ambiguities in the Turkish National Corpus and presents a set of NooJ grammars that can help disambiguate texts very efficiently;

—Danijela Merkler et alii's article "Disambigation of homographic adjectives and adverb forms in Croatian texts" presents a syntactic module that can disambiguate Croatian adjectives with high accuracy;

—Peter Machonis' article "Sorting NooJ out to take multiword expressions into account" shows us how to formalize discontinuous expressions by using NooJ dictionaries and grammars;

—Simonetta Vietri's article "Transformation and frozen sentences" shows how she formalized a lexicon-grammar table for Italian frozen expressions so that NooJ can not only recognize frozen expressions in texts automatically, but also produce all their paraphrases.

Part III of the volume is dedicated to various applications of NooJ for the analysis of corpora in Medicine and Social Sciences:

—Ines Boujelben et alii's article "Transformational analysis of Arabic sentences" describes a module that can recognize symptoms described in biomedical texts and then produce a corresponding standard short paraphrase;

—Thierry Declerck et alii's article "Integration of ontological semantic resources in NooJ" presents a system capable of parsing folk tale texts in order to compute co-reference resolutions and annotate ontology elements;

—Marie-Thérèse Gambin et alii's article "In the pursuit of a lost manuscript: Ptolemy's Planisphaerium" shows how the authors used NooJ to recognize and process variations and errors in different historical versions of the text;

—Bea Ehmann et alii's article "Narrative Psychological Application of semantic role labelling" describes a system capable of automatically adding psychosemantic annotations to Hungarian texts for the purpose of narrative psychological content analysis;

—Vanja Štefanec and Ivana Đuranić's article "Automatic syllabification in written Croatian with NooJ" describes the construction of an automatic hyphenation system, using a complete set of morphological grammars;

—Finally, Svetla Koeva et alii's article "Applications of Bulgarian-English parallel corpus for exploring translational asymmetries" presents techniques for extracting information from parallel corpora that can be

used to study translational asymmetries, and suggests several potential developments in NooJ that would enhance its multilingual parallel corpora capabilities.

We think that the reader will appreciate the importance of this volume, both for the value of each linguistic formalization and its underlying methodology, as well as for the potential for new applications of a linguistic-based corpus processor in the social sciences.

The Editors

VARIABLE UNIFICATION IN NOOJ V3

MAX SILBERZTEIN

Abstract

NooJ's linguistic engine integrates all its parsers (from the lexical to the syntactic level) with its morphological and paraphrase generators. In particular, both NooJ's syntactic parser and NooJ's transformational generator use the same single syntactic grammar. Because of this new level of integration, we had to design a new mechanism to manage variables and compute their value.

Introduction

One of NooJ's specificities among the Natural Language Processing frameworks is that NooJ allows linguists to formalize many levels of linguistic phenomena, and offers for each level one or more specific formalisms and computational devices. NooJ uses an annotation mechanism (stored in each Text Annotation Structure, or TAS) that integrates every single piece of linguistic information; this integration makes it possible to combine morphological constraints in syntactic rules, for instance. Perhaps the most striking application of this total integration is that NooJ can perform transformations automatically without requiring linguists to "program" transformation rules.

NooJ's transformations are different from Chomsky's. Chomsky's transformations are oriented derivations performed on sentences that can be combined to produce, in one direction, a deep structure (or the D-structure in the government and binding theory) or in the other direction a surface structure (or S-structure).[1] NooJ's transformations are symmetric like Harris' transformations, but their nature differs fundamentally from Harris'[2] in the sense that they are an implicit property of a syntactic

[1] See (Chomsky, 1965).
[2] See (Harris, 1981).

grammar rather than constituting a level of grammar that needs to be explicitly represented or "programmed" by linguists.[3]

In order for a given syntactic grammar to be able to be used both by NooJ's syntactic (structural) parser (Silberztein, 2010) and by its transformational parser/generator, I had to profoundly redesign the way NooJ's linguistic engine was managing variables and computing their values.

In its v2 version, NooJ computed the value of each occurrence of a given variable without keeping it in memory. This feature allowed a particular variable to obtain different values along the parsing path. In consequence, grammar designers could set and use a given variable in a loop: the variable would obtain a series of different values; as many values as times the parser would visit the loop.

For instance, consider the following grammar (Figure 1), used to translate the English noun phrase *the big beautiful table* into its French equivalent *la grande belle table*:[4]

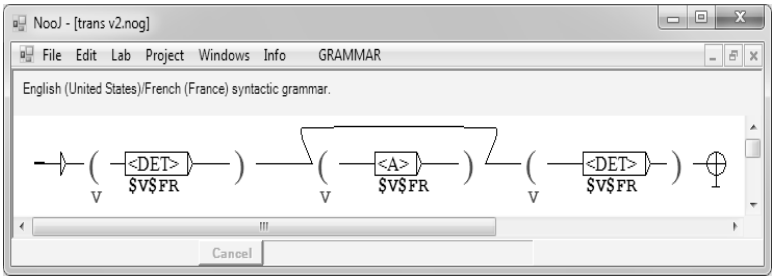

Figure 1: Variable $V has multiple values

Note how the same variable $V obtains four values during the parsing of the English noun phrase: $V="the", then $V="big", then $V= "beautiful", then $V="table". Each of these words is translated correctly because its translation (VFR) is used exactly where the variable $V is set (*i.e.* $V is in sync with VFR).

This type of grammar is useful in processing sequences of text "word by word" and has other advantages as well.

[3] See (Silberztein, 2011).

[4] For the sake of discussion, this grammar is overly simplified: one would need to add agreement constraints on the noun's gender and number: the French translation for *table* is feminine singular, thus the words *la*, *belle* and *grande* must be in feminine singular.

For instance, variables can be encapsulated safely inside embedded graphs and/or loops because each variable's value is always computed locally: no fear that a given variable would be used inadvertently in a different graph for different purposes. As a result, graphs in a NooJ grammar were largely independent objects so they could — at least in theory — be shared and distributed as opaque "black boxes".[5]

The way NooJ v2 handled variables looked very much like a great original idea!

A Dead End

Unfortunately, the very fact that NooJ was not unifying all values of each occurrence of a given variable prevented grammar designers from combining agreement constraints and propagating them along a parsing path. For instance, consider the following grammar (Figure 2):

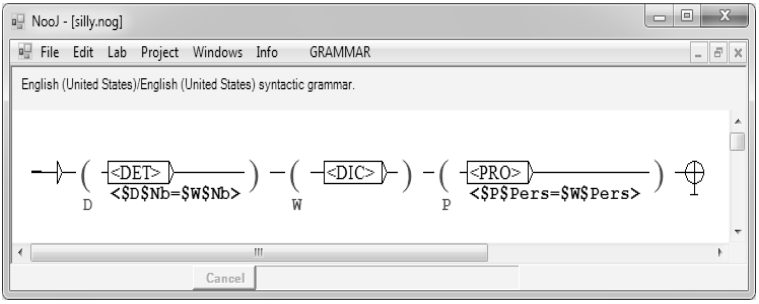

Figure 2: Two inconsistent constraints

This grammar contains two constraints:

- <DNb=WNb> (assuming that $W is a noun) checks that it agrees in number with its determiner,
- <PPers=WPers> (assuming that $W is a verb) checks that it agrees in person with the following pronoun.

[5] Graphs are not totally insulated: if a variable is set in a graph *A*, it can still be seen in another graph *B*, provided that its value was computed in *A* before it was used in *B*… As a matter of fact, graphs were never intended to be autonomous objects that could be distributed and shared independently from the grammar that encapsulates them. In NooJ — as opposed to INTEX, see (Silberztein, 1993) — the atomic piece of grammar is the grammar, not the graph.

In other words, $W takes the role of both a noun (in the first constraint) and a verb (in the second constraint): it does not make any sense, and such an inconsistent grammar should not produce any result.

However, because $W is computed independently each time NooJ needs its value, an ambiguous word that could be both a noun (that agrees in number with the preceding determiner) or a verb (that agrees in person with the following pronoun) will indeed be recognized by this grammar, *e.g.* in the following sequence:

… some cooks he …

the word *cooks* can be either a plural form of the noun *cook* (in which case it does agree with the plural determiner *some*), or a conjugated form of the verb *to cook*, in which case it agrees with the following pronoun *he*.

As grammars contain more and more lexical constraints and longer and longer sequences (which include ambiguous words), the parsing results become less and less reliable. For instance, consider the following grammar (Figure 3):

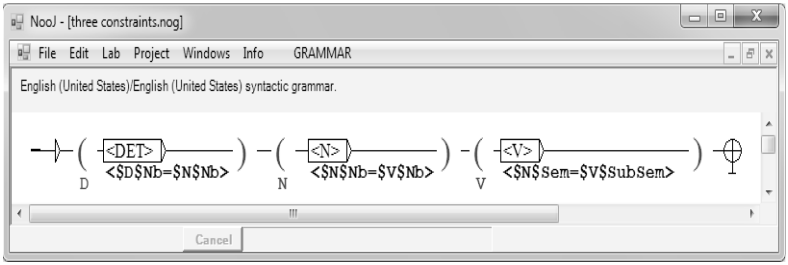

Figure 3: Three constraints

This grammar recognizes sequences constituted by a determiner, a noun and a verb: this useful grammar can be applied to large corpora when one wants to index and retrieve sequences such as “the director runs…”, “his car stopped”, etc.

The constraint <DNb=NNb> checks that the determiner agrees in number with the noun; the constraint <NNb=VNb> checks that the noun agrees in number with the verb; the constraint <NSem= VSubSem> checks that the noun’s semantic class corresponds to the verb’s subject’s semantic class. This grammar correctly recognizes sequences such as:

… A woman says …

("a" agrees in number with "woman"; "woman" agrees in number with "says"; "woman" is a proper agent for the verb *to say*). However, it will also recognize a large number of incorrect sequences, in particular if the middle word stored in variable $N is actually ambiguous. For instance, if $N is associated with three lexemes: one lexeme could agree with the determiner (but not with the verb), another lexeme could agree with the verb (but not with the determiner), and the last one could agree semantically with the verb (but not in number)…

In conclusion: although grammars such as the one in Figure 3 are seemingly simple and very natural from a linguistic point of view, the way NooJ v2 interprets them make them unreliable at best; their application to large corpora typically produces considerably too many false results.

Clearly, the benefits of computing variables' values dynamically without keeping track of the resulting values are negated by the fact that linguists cannot combine constraints in grammars reliably. We need to make sure a given variable in a grammar holds the same value everywhere it is used, i.e. we need to unify all the values of each variable.

NooJ v3's Variable Management Mechanism

In the new v3 version, Nooj uses a single variable space to make sure that every variable in a given grammar has only one value (more precisely: one lexical unit). Thus the new engine processes the grammars in Figure 2 and Figure 3 reliably.

In order to allow grammar designers to use variables in loops (such as in Figure 1), I have added the special variable $THIS which always refers to the current lexical unit and thus "simulates" the way v2 bound variables to their values. This time, however; there is no need to define the variable $THIS: it always refers to the current lexical unit. The grammar shown in Figure 1 could then be rewritten as the following grammar (Figure 4):

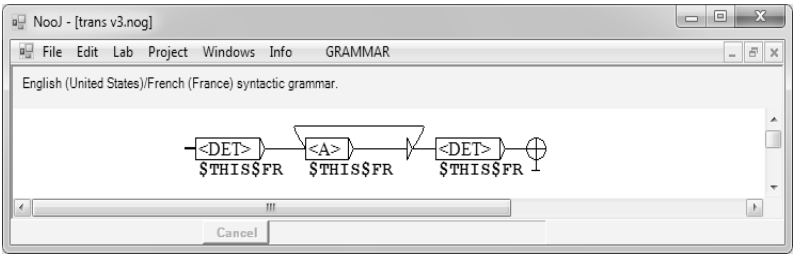

Figure 4: Variable $THIS can be used in loops

$THIS takes four different values when parsing the noun phrase *the beautiful red table*, and $THIS$FR computes their translation correctly.

However, some v2 grammars will still need to be rewritten to v3, as it is no longer possible to use the same variable name in different graphs of a single grammar to refer to different objects.

This is unavoidable; I hope NooJ users who have relied on the possibility of using one single variable name to refer to different objects in one grammar will forgive me! The new design has allowed us to escape from the dead end of the dynamic computation of each variable's occurrence. In the meantime, I believe that the new design enforces the unity of every variable's value and thus will help us build more robust grammars that are more natural from a linguistic point of view.

In the short term, the new design has already allowed us to add new, exciting functionalities to NooJ's transformational module.

Variables' Multiple References

Version v3 already takes advantage of the fact that each variable has indeed one unique value in order to allow grammar designers to simplify each grammar's graph massively. Consider, for instance, the following grammar (Figure 5), which represents simple transitive sentences:

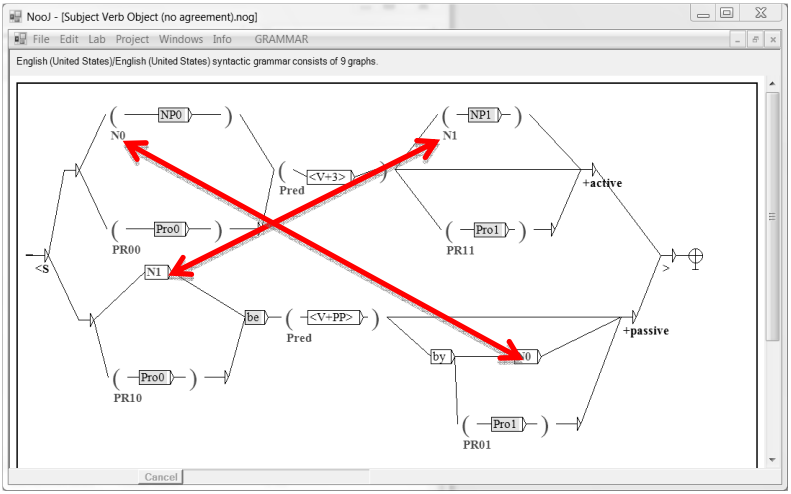

Figure 5: A syntactic grammar

In this graph, note how variable $N0 is defined at the top left of the graph, and is used in the *input* of the grammar at the bottom right of the graph. One may see the reference to $N0 as a mere "abbreviation" of its definition, and indeed, this abbreviation allows grammar designers to draw more compact and elegant graphs.

But NooJ's new variable system does much more than that: it actually keeps a live link between all the occurrences of a variable and its unique value: technically, if one applies the grammar of Figure 5 to the sentence *John sees Mary*, NooJ's syntactic parser sets "John" to variable $N0 and "Mary" to $N1. Then, the paraphrase generator uses these same exact values when it explores the paths at the bottom of the graph to produce the passive sentence *Mary is seen by John* (as opposed to *John is seen by Mary*). In other words: the new variable mechanism allows us to design parameterized grammars, i.e. grammars that contain variables set during a syntactic parsing.

The situation is more complicated for variable $Pred which is defined twice in the grammar: at the top of the graph, we have $Pred = <V+3>, whereas we have $Pred = <V+PP> at the bottom of the graph. That certainly looks like an inconsistency because a variable should not have more than one value. However, note that during the parsing of any sentence, this variable will, in fact, be set only once:

- either the sentence is in the active mode (e.g. *John sees Mary*), then we get $Pred = <V+3> (e.g. $Pred = "sees"), or

- the sentence is in the passive mode (e.g. *Mary is seen by John*), then we get $Pred = <V+PP> (e.g. $Pred = "seen").

The fact that the same variable $Pred is used to hold two different values is not an issue because these values will never be alive at the same time during the syntactic parsing of any sentence. That, however, is no longer the case for NooJ's paraphrase's generator which computes both $Pred = "sees" when it produces paraphrases such as *John sees her*, and $Pred = "seen" when it produces passive sentences such as *Mary is seen by John*.

What then is the true value stored by NooJ for $Pred? The answer is in fact a linguistic one: NooJ considers both lexemes <sees,see,V+PR+3+s> and <seen,see,V+PP> to be instances of the single lexical unit <see,V>, but with different properties. In other words, both word forms "sees" and "seen" are processed by NooJ as two variants of a unique linguistic unit. When NooJ's parser matches the variable $Pred in the context of the symbol <V+3>, it produces the form "sees" whereas when the parser matches $Pred with the symbol <V+PP>, it produces the form "seen".

Note that this behavior is just a generalization of the way NooJ v2 already processed complex variables: just as VGender, VNb or VTense refer to different property values for the same lexical unit, $V in the context of the lexical symbol <V+3> and in the context of the lexical symbol <V+PP> produce two different lexeme values.[6]

In conclusion: just like the two variables $N0 and $N1, the variable $Pred takes its (unique) value when NooJ parses the input sentence, e.g. *John sees Mary*. When NooJ's paraphrase generator produces sentences in the active such as *John sees her* (thanks to the symbol <V+3>), NooJ instantiates $Pred with the lexeme value "<sees,see,V+PR+3+s>". When it produces passive sentences such as *she is seen by him*, NooJ matches the lexical unit $Pred with the symbol <V+PP>, which in effect computes the lexeme "<seen, see, V+PP>".

This mechanism is reversible when the number of linguistic units is constant: for instance, if one enters the sentence *John is seen by Mary*, then the same grammar will produce the sentence *Mary sees John*: the

[6] Internally, NooJ's parser uses the notation $V_V+PP and $V_V+3. This functionality, unique to NooJ, is made possible by the fact that all of NooJ's engines (including the morphological parser and generator) are integrated. Note that $V_V+3 produces actually three conjugated forms: "sees", but also "see" (third person plural) and "saw" (third person in the preterit). The grammar contains an agreement constraint that filters out all the forms that do not agree with the subject.

constraint <V+PR> will produce the form “sees” from the lexeme <seen, see, V+PP>. However, if one enters the sentence *he sees her*, NooJ cannot compute values for the lexical symbols <DET> and <N> (used in the NP0 and NP1 graphs): as a result, NooJ confines itself to displaying these symbols.

Agreements

The grammar shown in Figure 5 can parse any direct transitive sentence that contains a noun phrase or a pronoun followed by a verb and then a noun phrase or a pronoun, as well as its passive form. There are two levels of agreement checks that need to be discussed:

(1) One needs to take several syntactic agreement constraints into account, such as the one between the determiner and the noun and the one between the subject and the verb. For instance, the grammar must reject the incorrect sentence **The men sees the apple*

NooJ v2 was already capable of processing these types of constraints: we simply need to add them into the grammar (Figure 6).

The constraint <$THIS$Nb=$Head0$Nb> located under the node <V+3> checks that the verb’s number property is identical to the head of its subject’s noun phrase; $HeadN0 is defined in the embedded graph NP0 (Figure 7).

In the same way, the constraint <$THIS$Nb=$PR00$Nb> checks that the verb agrees with its subject pronoun.

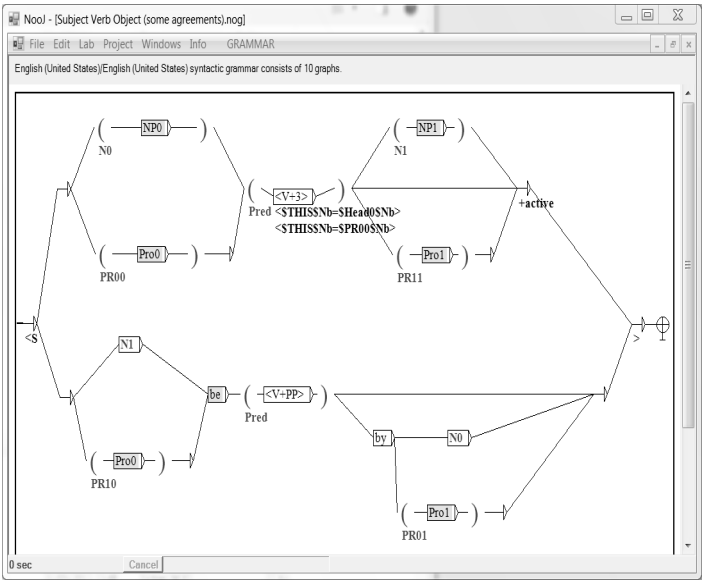

Figure 6: Agreement constraints used by the syntactic parser

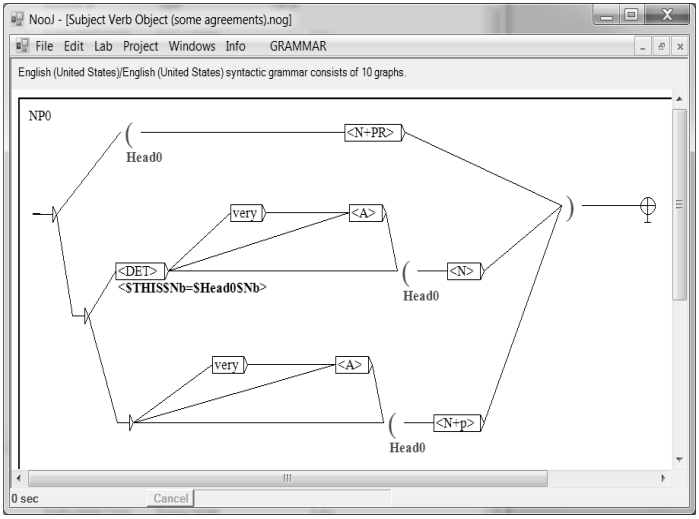

Figure 7: A simple grammar for noun phrases

(2) One also needs to take care of agreement constraints between the two noun phrases and their corresponding pronouns, which are produced by the transformational generator. For instance, when NooJ produces paraphrases for the sentence *John sees Mary*, it must not produce sentences such as *she sees Mary* nor *she sees them*. Although these latter sentences are correct syntactically, they do not constitute correct paraphrases of the original sentence and thus should not be produced by the paraphrase generator.

Thanks to NooJ v3's new variable management mechanism, though, this type of agreement constraint is actually implemented in the most straightforward way: one just needs to add the corresponding constraint, as if the variables that hold the pronouns had actual values in the input sentence. In a way, the grammar already tells us that the pronouns he and her are somehow implicitly present in the sentence *John sees Mary*: we just need to filter out the wrong pronouns and keep the one that agrees with John (i.e. *he*) and the one that agrees with Mary (i.e. *her*).

Let's turn to the final version of our grammar (Figure 8):

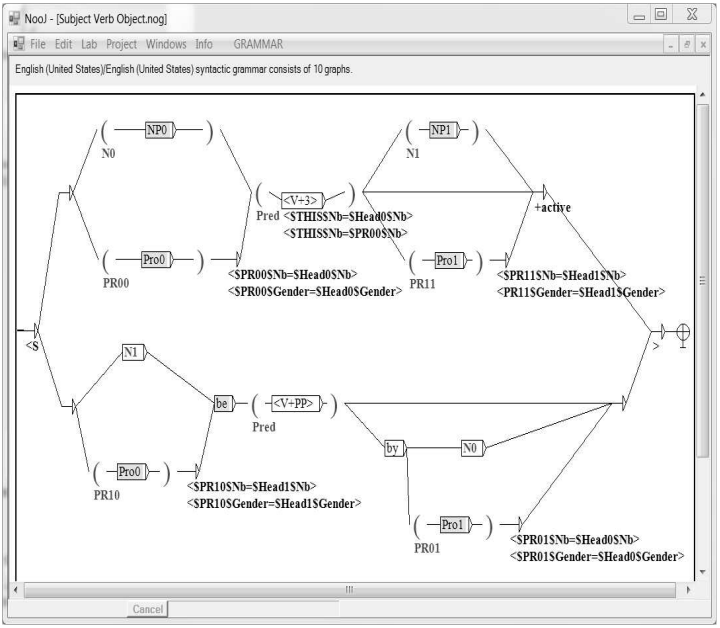

Figure 8: Agreement constraints used by the paraphrase generator

The two constraints <$PR00$Nb=$Head0$Nb> and <$PR00$Gender= $Head0$Gender> check that the subject pronoun — which is produced by the paraphrase generator — agrees in number and in gender with the subject noun phrase (i.e. *John*) of the input sentence.[7]

In conclusion, the new v3 engine allows variables to be used not only by the syntactic parser, but also by the paraphrase generator. Vietri (2012) shows how the new engine is used to produce transformations for idiomatic expressions formalized in a lexicon-grammar table.

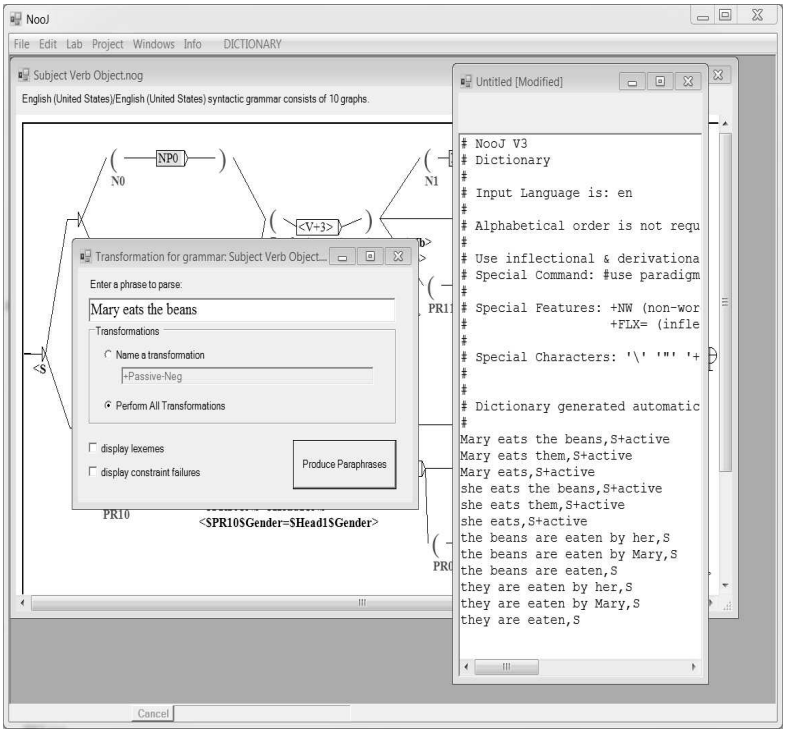

Figure 9: Producing paraphrases for the sentence Mary eats the beans

[7] I am using indices to refer to the role and to the syntactic function of each pronoun. For the input sentence *John sees Mary*, $PR00 is the pronoun associated with the agent (0) role in the subject (0) syntactic position, i.e. *he*; $PR01 is the pronoun associated with the agent (0) role in the complement (1) syntactic position, i.e. *him*; $PR10 is the pronoun associated with the theme role in the subject position, i.e. *she*; $PR11 is the pronoun associated with the theme role in the complement position, i.e. *her*.

Perspective: NooJ CESAR project

I apologize for the compatibility problems that the new variable engine might create for some NooJ users; I do believe that the "flexible" dynamic way of processing variables was indeed a dead end, and I have shown that with the new engine, not only can we design grammars that are as powerful as the ones created with NooJ's previous versions (thanks to the new $THIS variable): we can also design grammars that can be used both by NooJ's syntactic parser and by its transformational generator. I believe this makes NooJ truly different from all other Natural Language Processing Tools with which transformations have to be "programmed" explicitly.

The European META-SHARE CESAR project has chosen to use NooJ as their linguistic corpus processor, and a team at the Mihajlo Pupin Institute (Belgrade) has already started to port NooJ into the MONO framework, which will allow it to run on most operating systems, including LINUX, UNIX and Mac OSX. The upcoming v3.1 version will be based on v3.0 (the .NET version) and will incorporate the new variable management mechanism.

References

Chomsky, N. 1965, *Aspects of the Theory of Syntax*. MIT Press.

Harris, Z. 1981, *Papers on Syntax*. Henry Hiż Ed. Dordrecht : Holland, pp. 437-479.

Silberztein, M. 1993, *Dictionnaires électroniques et analyse automatique de textes: le système INTEX*. Masson Ed.: Paris. (240 pages).

—. 2010, Syntactic parsing with NooJ. In *Proceedings of the NooJ 2009 International Conference and Workshop*. Centre de Publication Universitaire : Sfax, pp. 177-190.

—. 2011, Automatic transformational analysis and generation. In *Proceedings of the NooJ 2010 International Conference and Workshop*. Thrace University: Komotini, pp. 221-231.

Vietri, S. 2012, *Transformations and frozen sentences*. Same Volume.

PART I:

LEXICONS

LE DM, A FRENCH DICTIONARY FOR NOOJ

FRANÇOIS TROUILLEUX

Abstract

This paper presents the DM, a new dictionary for French. Freely available resources are selectively used to obtain lexical lemmas, from which morphological grammars generate about 538 000 baseforms. Evaluation of the DM on corpus shows that it stands the comparison with the previous NooJ delaf dictionary.

Introduction

For historical reasons, large coverage French dictionaries are available to NooJ users only in the compiled *.nod* format. This poses several problems for grammar development, e.g. constraints won't work, adding new information to a lexical entry requires redefining the whole set of information on that entry, the dictionaries cannot be used for generation…

We then decided to produce a new dictionary, called the DM, designed in the NooJ format (Silberztein 2003, 2005), on which the NooJ community will have control. This paper first presents a quantitative analysis of the freely available resources we considered. The next two sections describe the way we constructed the DM for lexical and function words, leading to a global view of the DM extension. Finally, results of morphological analysis and parsing with the DM are compared to results obtained with the *delaf.nod* dictionary.

Available resources

To build a freely available dictionary, we had to rely on resources for which free reuse is licensed. This excludes some resources, in particular the Morfetik dictionary (Mathieu-Colas 2009), which is interesting in that it makes use of several good quality resources. Our initial plan was to rely on three free resources: the DELA (Courtois 1990), Morphalou (Romary

et al. 2004) and the Le*fff* (Sagot 2010)[1]. In order to evaluate the potential contribution of each of these resources, we compared the lemmas they contain for the adjective, adverb, common noun, verb, interjection and prefix categories. Results are given in Table 1[2]. The *union* column gives the number of lemmas in the union of the three dictionaries. The central columns give the percentage of the union which is common to the three dictionaries (*int.*), common to only two (*DL*, *LM*, *MD*) and specific to each (*D*, *L*, *M*).

	int.	**DL**	**LM**	**MD**	**D**	**L**	**M**	**union**
adjectives	24.7	5.1	2.5	15.4	23.1	11.6	17.6	35124
adverbs	58.9	22.9	0.9	0.8	8.1	2.9	5.4	3704
nouns	35.5	6.1	0.3	9.4	30.8	1	16.9	82118
verbs	45.7	3.5	0.1	12.2	29.1	1.9	7.5	13271
interj.	3.4	2.3	0.3	30.1	23	10.5	30.4	352
prefixes	0	9.6	0	0	84.1	6.3	0	921

Table 1. DELA-Lefff-Morphalou comparison.

	int.	**dela**	**morph.**	**union**
adjectives	45.3	32	22.7	31064
adverbs	61.6	32	6.5	3596
nouns	45.4	37.3	17.4	81289
verbs	59	33.2	7.8	13020
interjections	37.5	28.3	34.3	315

Table 2. **DELA-Morphalou comparison.**

The prefix and interjection categories are peculiar. There are no prefixes as autonomous entries in Morphalou. For this category, the DELA is much richer than the Le*fff*, the intersection of the two being rather small, with only 88 lemmas. Interjections are dealt with disparately in the three dictionaries. The table line actually counts the following categories: *intj* for the DELA, *interjection* and *onomatopoeia* for Morphalou and *pres* for the Le*fff*. As can be seen, the intersection is not empty, but the categories do not correspond very well. 23 onomatopoeia of Morphalou are

[1] First work on the Le*fff* dates back to 2004; we use the latest version to date, extensional version 3.0.

[2] We only look at uncapitalized *simple* words, *i.e.* without any whitespace, hyphen nor apostrophe. Figures are obtained after correction of a few errors and normalization of some lemmas.

interjections in the DELA; the category *pres* of the Le*fff* includes both interjections and presentatives as *voici*, *voilà* ("here is"), which insofar as they may introduce a complement and combine with clitic pronouns should maybe be categorized differently.

Regarding the four other, more important, categories, one may note that the specific contribution of the Le*fff* is small, except for adjectives. More than 96% of the adverbs, nouns and verbs of the Le*fff* are actually present in the DELA. For adjectives, the number is only 67%, but the difference comes from the fact that the Le*fff* quite systematically codes past participles as adjectives: the "adjectives" *exhumé*, *blasphémé*, *démarré* ("started"), for instance, are specific to the Le*fff*.

The specific contribution of Morphalou is relatively small for adverbs and verbs. This is due to recent work on these two categories in the Le*fff* (cf. Sagot and Fort 2007; Tolone and Sagot 2009). On the other hand, Morphalou contains an important number of nouns and adjectives which are neither in the DELA nor the Le*fff*.

Table 2 gives a direct comparison between the DELA and Morphalou. It shows that the intersection of the two dictionaries is surprisingly small and that the DELA is clearly the bigger of the two dictionaries.

The DM Lexical Words

Lemma Selection

In view of the observations we made, we had several options for a new dictionary. A first idea could be to make the union of the three dictionaries and obtain what would probably be the biggest freely available French dictionary for NLP. The drawback of this approach would be that the dictionary will include all the errors to be found in the dictionaries. Rather than taking the union of the dictionaries, we then decided to go for the intersection. The idea is to favour precision: the presence of a word in several dictionaries is a guarantee that it does exist in French. Each dictionary is in a way validated against the others. With the choice of the intersection, this project differs from the Morfetik project (Mathieu-Colas 2009), which builds the *union* of the resources it uses; the choice of the union in this project imposes manual validation of the entries, our choice of the intersection allows automatic validation.

Having chosen to favour precision, the question remains which intersection to take. A first idea could be to take the intersection of the three dictionaries, but, as we have seen, the Le*fff* is for a very large part included in the DELA. The DELA would thus have a sort of double