View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by Archive Ouverte a LUniversite Lyon 2

Complex Annotations with NooJ

Max Silberztein

To cite this version:

Max Silberztein. Complex Annotations with NooJ. Blanco, Silberztein. Proceedings of the 2007 International NooJ Conference, Jun 2007, Barcelone, Spain. Cambridge Scholars Publishing, p. 214, 2008. <hal-00498042>

HAL Id: hal-00498042

https://hal.archives-ouvertes.fr/hal-00498042

Submitted on 6 Jul 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Complex Annotations with NooJ

Max Silberztein
LASELDI, Université de Franche-Comté
max.silberztein@univ-fcomte.fr

Abstract: NooJ associates each text with a Text Annotation Structure, in which each recognized linguistic unit is represented by an annotation. Annotations store the position of the text units to be represented, their length, and linguistic information. NooJ can represent and process complex annotations, such as those that represent units inside word forms, as well as those that are discontinuous. We demonstrate how to use NooJ's morphological, lexical, and syntactic tools to formalize and process these complex annotations.

1. Introduction

When parsing a text or a corpus, NooJ builds a Text Annotation Structure (TAS) in which each linguistic unit is represented by a corresponding annotation. An annotation stores the position in the text of the text unit to be represented, its length, and linguistic information, cf. (Silberztein 2007).

Typically, NooJ's annotations represent prefixes (e.g. "dis-"), affixes and suffixes (e.g. "ization"), simple words (e.g. "table"), multi-word units (e.g. "round table"), frozen expressions (e.g. "take ... into account") as well as named entities such as dates (e.g. "on Monday, October the third"), person names ("Dr John W. Smith"), addresses, locations, numerical expressions, etc.

NooJ adds annotations to the TAS automatically at various stages of the analyses. At the lexical level, its lexical parser typically applies dictionaries to the text to produce annotations. NooJ's dictionaries are an enhanced, unified version of INTEX's dictionaries: because it handles morphological and derivational morphology, there is no longer a need for DELAF-type inflected forms dictionaries. NooJ handles simple words and multi-word units in a unified way, therefore there is no longer any distinction between DELAS and DELAC dictionaries. Finally, lexical entries can be associated with spelling or synonymous variants: no need for DELAV-type dictionaries. Cf. (Silberztein 2004).

Morphological and lexical annotations are ambiguous in general. After NooJ has performed a lexical analysis of the text, the TAS represents ambiguities between simple words, such as "fly" (Noun or Verb), between a sequence of simple words and a multi-word unit, such as "round table" (a table or a meeting), between various morphological analyses, between frozen expressions and sequences of multi-word units and simple words, etc.

For instance, NooJ's lexical parser automatically annotates the text "He cannot take the round table into account" with various linguistic units, among them two simple words: **<he,PRO>** and **<the,DET>**, two morphological units: **<can,V>** and **<not,ADV>**, one multi-word unit: **<round table,N>** and one discontinuous verbal expression **<take into account,V>**.

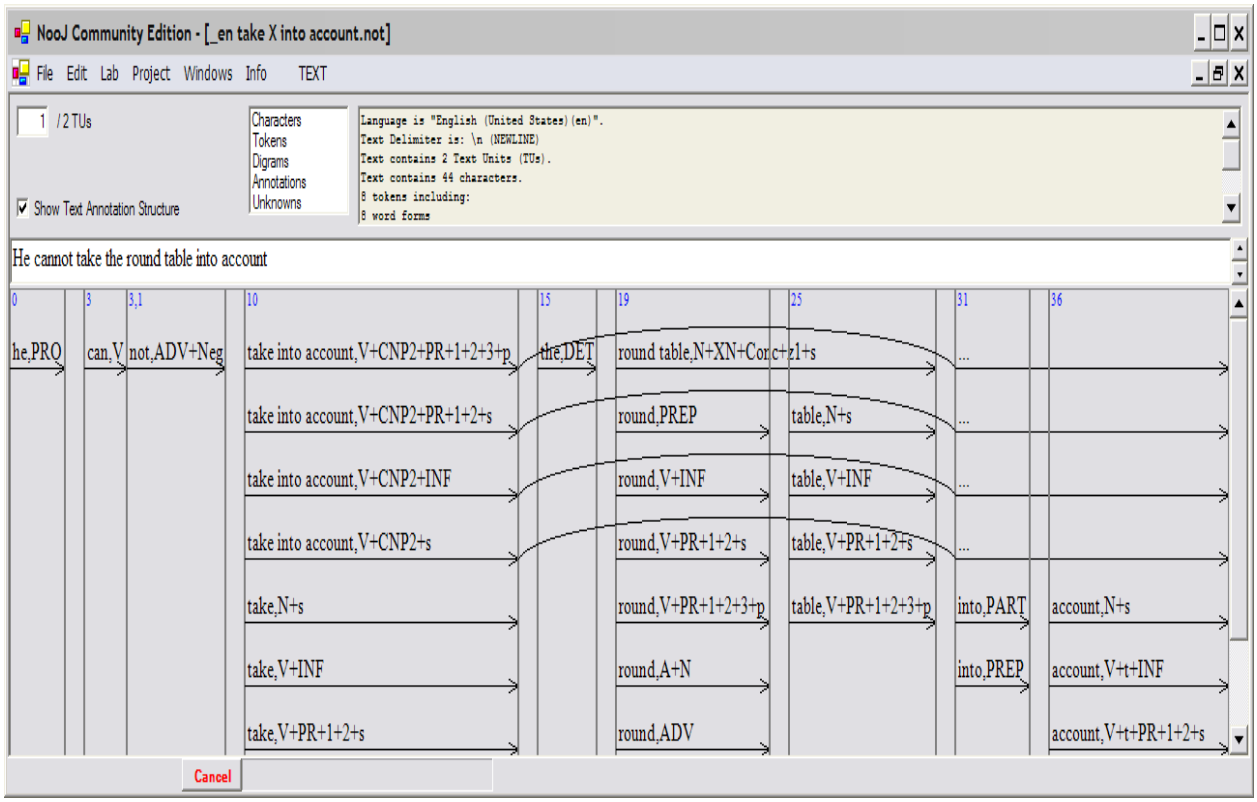

Figure 1: Lexical Analysis of a Simple Text

Each annotation is defined by its position (index) in the text. Here, the annotation **<the,DET>** occurs at position 15, and its length is 3 (See Silberztein 2007).

2. Morphology

When an annotation represents a theoretical linguistic unit that does not occur *as is* in the text, its position or its length might be a decimal number. For instance, in the previous example, the annotation **<can,V>** occurs at position 3, and has a "length" 0.1; the annotation **<not,ADV>** occurs at position 3.1, and has a "length" 5.9. This decimal topology allows word forms to be split into any number of linguistic units (which is crucial when processing agglutinative languages), and always guarantees a perfect synchronization between the TAS and the original text, which is crucial for further processing.

For instance, NooJ's query "**<V> <ADV>**" (a verb followed by an adverb) would match both texts "cannot" and "can never" in the exact same way: NooJ's syntactic parser processes sequences of annotations, and does not even know that originally, these two sequences of two linguistic units were processed differently by NooJ's lexical parser.

This characteristic is much more than an elegant trick: it allows linguists to design grammars at the syntactic or semantic levels without having to deal with a large number of orthographic and morphological idiosyncrasies. For instance, let us consider an English syntactic grammar **VG** that recognizes complex verb groups. This grammar may include the following rule:

<V+Modal> <ADV>* <V+INF>

(a modal verb followed by an optional sequence of adverbs, followed by an verb at the infinitive). Thanks to NooJ's lexical parser that represents morphological linguistic units and 'regular' lexical units in a unified way, this single rule represents and automatically recognizes the following four sequences of text:

can never come, cannot come,
will always come, won't always come

Being able to take care of complex tokenization problems once and for all at the lexical and morphological levels, independently from the forthcoming higher levels of analyses (syntactic and semantic), is crucial for formalizing a number of languages, such as Asian and Germanic languages.

Finally, note that many NLP applications benefit from the perfect synchronization of the linguistic information and the original text. For instance, NooJ's users can use the TAS in order to color the original text, which could have a complex file structure (such as an HTML page). This synchronization also allows users to import XML documents, annotate them, and then export the TAS to build a new XML document, which in turn can be re-imported into NooJ for further analyses. Cascades of linguistic processing can further be automatized with NooJ's companion program "noojapply.exe".

The synchronization also allows users to perform partial replacements in texts, such as translations into another language of certain terms or expressions, or production of a formal notation for expressions (such as a semantic PROLOG-type notation), etc.

Formalization

In order to tell NooJ that a word form or a set of word forms must be associated with more than one linguistic unit, one can build a morphological grammar or a dictionary.

-- Morphological grammars are finite-state transducers that *recognize* a set of word forms and then *produce* the corresponding sequence of annotations. For instance, the following morphological grammar recognizes the word form "*cannot*" and then produces the sequence of two annotations: "**<can,V> <not,ADV+Neg>**":

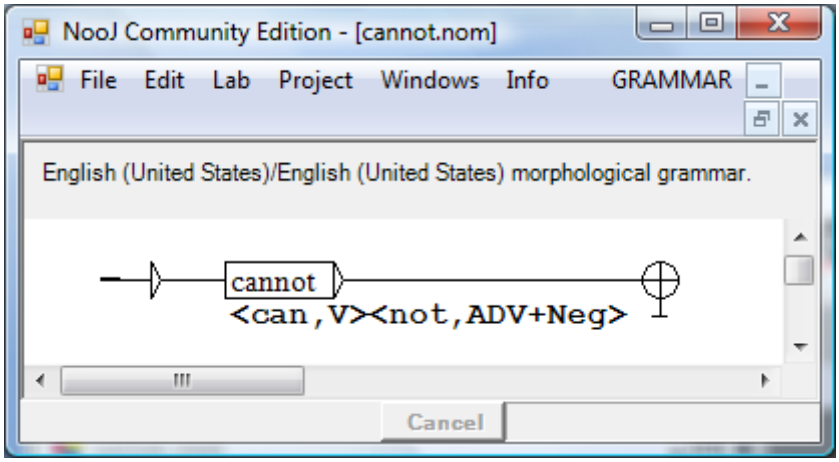

Figure 2: A Morphological Grammar

-- Sets of word forms are sometimes better represented in dictionaries. For instance, the set of English contractions is represented in the English dictionary "Contractions.dic" stored in NooJ's English module:

```
I'm,<I,I,PRO><am,be,V+PR+1+s>+UNAMB
you're,<you,you,PRO+2+s><are,be,V+2+s>+UNAMB
he's,<he,he,PRO+3+m+s><is,be,V+PR+3+s>+UNAMB
it's,<it,it,PRO+3+n+s><is,be,V+PR+3+s>+UNAMB
...
```

After having applied this dictionary to texts, each sequence "I'm" will be internally represented in the TAS as a sequence of the two annotations "**<I,PRO> <be,V+PR+1+s>**".

NooJ's English module contains a dozen dictionaries and morphological grammars, including one to tokenize contracted negations (e.g. "don't"), one to recognize prefixed verbs (e.g. "disrespect"), various grammars to recognize derivations (e.g. "Frenchify"), as well as derivations of proper names (e.g. "Bushization"), etc.

2. Frozen expressions

Although a number of NLP applications -- most taggers and corpus linguistic tools -- simply ignore multi-word units and frozen expressions, it is crucial to represent, recognize and process them. For instance, consider the following English text sample ([1]):

Battle-tested Japanese industrial managers here always buck up nervous newcomers with the tale of the first of their countrymen to visit Mexico, a boatload of samurai warriors blown ashore 375 years ago. From the beginning, it took a man with extraordinary qualities to succeed in Mexico, says Kimihide Takimura, president of Mitsui group's Kensetsu Engineering Inc. unit.

Underlined sequences are multi-word units or frozen expressions that should be processed as atomic units, rather than word-by-word: for instance, "Battle-tested" translates in French as "aguerris", and not (word-by-word) as "? testés par une bataille". "industrial managers" must *not* be analyzed in the same way as thousands of sequences such as "industrial food", "industrial soap", "industrial diamond", by a productive rule such as:

an industrial <N> is a <N> that was made with industrial methods

An *industrial manager* is not a "*manager made with industrial methods*": he/she is a manager *in* the industrial world.

In NooJ, simple words and multi-word units are processed in a unified way: they are stored in the same dictionaries, their inflectional and derivational morphology is formalized with the same tools, and their annotations are undistinguishable from those of simple words. This

[1] The tagging of this text is commented on in [6], but there is no mention of its MWUs.

unifies the treatment of linguistic units, whether simple words or multi-word units. Thus, the simple syntactic query:

<N> <ADV>* <V>

(a noun, followed by an optional sequence of adverbs, followed by a verb)

recognizes the two following text sequences:

... meeting suddenly stopped ...
... round table all of a sudden was ...

In the second example, <N> matches “round table” and <ADV> matches “all of a sudden”.

However, the fact that frozen expressions are potentially discontinuous in texts makes it more complicated to formalize them. Two operations are necessary:

(1) We need to give NooJ a way to recognize each occurrence of the frozen expression, even when the frozen expression is discontinuous. For that purpose, we create a syntactic grammar.

(2) We need to produce an annotation for the sequences recognized in (1), in which all the components of the frozen expression are linked together.

The following is a simple grammar that can be used for that purpose:

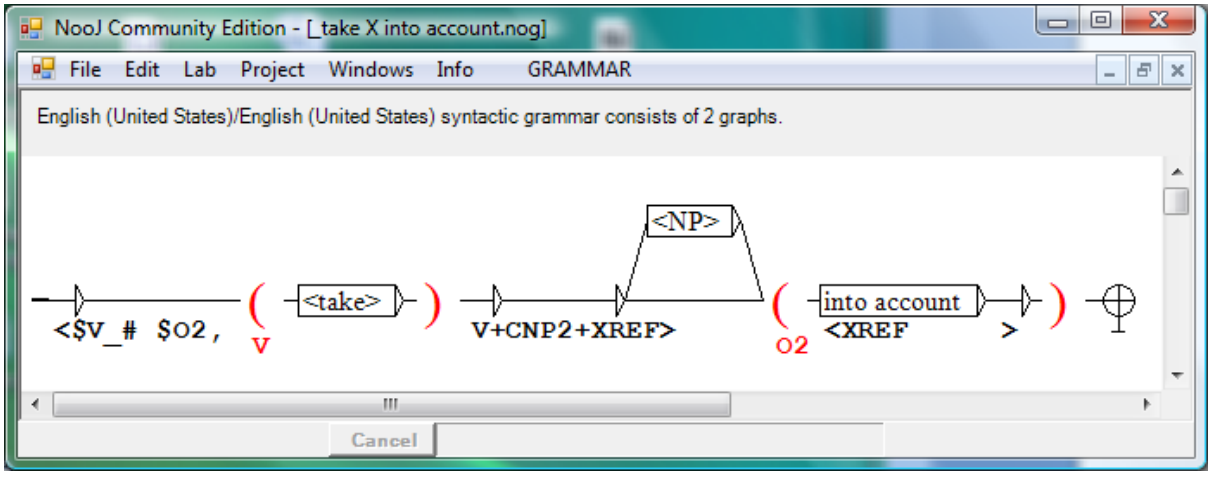

Figure 3: Grammar for *take NP into account*

This grammar recognizes text sequences such as:

... took the meeting into account ...

We assume here that noun phrases have already been annotated as <NP> by a previous analysis of the text ([2]). In the output of the grammar (written in bold, under the nodes), the variables **$V** and **$O2** refer to the text that was matched between the red parentheses named

[2] When loading a corpus or text in NooJ, the command **Linguistic Analysis** applies all lexical and syntactic resources that are selected in **Info > Preferences**. Syntactic resources are typically local grammars applied in cascade, and used to annotate larger and larger chunks of the text.

V and **O2**. The morphological operator "_" is used to lemmatize the content of variable **$V**, i.e. "took" produces "take". The concatenation operator "#" is used to concatenate the lemmatized value **$V_** and the value of **$O2** to get the result "take into account".

Thus, applying this grammar to the previous text produces the sequence of two annotations:

```
<take into account,V+CNP2+XREF> <XREF>
```

The keyword **XREF** is used to link together all the components into a single, discontinuous annotation. The resulting annotation resembles the one that was displayed in figure 1. The result of applying this grammar to the text is that the discontinuous expression has been successfully represented as a single unit in the TAS.

3. Lexicon-Grammars

The previous grammar successfully represents and recognizes occurrences of the single expression "to take <NP> into account". However, as the number of expressions to be represented becomes extremely numerous (tens of thousands), it is not efficient to build one grammar per expression. In that case, we need to use a combination of a dictionary and a grammar.

The 'INTEX' method

The software INTEX (see http://intex.univ-fcomte.fr) could already link together a dictionary and a grammar in order to formalize frozen expressions and recognize them automatically in texts. More specifically, INTEX uses:

(1) a "lexicon-grammar" table in which all frozen expressions of a certain structure are listed and described. For each possible structure of a frozen expression, there is a corresponding lexicon-grammar table (See Gross 1984, Gross 1993).

In INTEX, each lexicon-grammar table is represented as an MS-EXCEL-type file in which each frozen expression is listed in one line. A set of relevant linguistic properties, such as the distribution of the expressions' arguments (Subject =: NHum, Object =: NConc, etc.) and their structural variants (e.g. +Passive) are listed in columns; each cell contains either a string (e.g. "into account") or a binary code ("+" or "-") to express that an expression has a property or not.

For instance, INTEX's table C1d.xls (see figure below) describes a set of 1,663 French frozen expressions in which a verb is frozen with its direct object complement (as in the English expressions "to lose one's mind", "to catch a cold", etc.). The two first columns encode the distributional class of the subject of the expression (Human or Non-Human); the third column marks the expressions that are obligatorily negative (e.g. "il n'a pas inventé l'eau chaude", literally "he has not invented warm water"); the fourth column encodes a potential pre-verbal particle, the sixth column encodes the fact that the object complement is optional or not, etc.

N0 = : Nhum	N0 = : N-hum	Nég	PPV	V	N0 V	DET	N0 V Dét N1	N	N1 = : Npc	Passif
+	-	-	<E>	<abandonner>	+	la	-	partie	-	+
+	-	-	<E>	<abandonner>	+	la	-	partie	-	+
+	-	-	<E>	<abandonner>	-	la	+	pose	-	+
+	-	-	<E>	<abandonner>	-	le	-	domicile conjugal	-	+
+	-	-	<E>	<abandonner>	-	le	-	terrain	-	+
+	-	-	<E>	<abandonner>	-	les	-	lieux	-	+
+	-	-	<E>	<abjurer>	-	la	-	foi catholique	-	+
-	+	-	<E>	<abolir>	-	le	-	temps	-	+
+	+	-	<E>	<accélérer>	-	le	-	mouvement	-	+
+	+	-	<E>	<accélérer>	-	le	-	mouvement	-	+
+	-	-	<E>	<accompagner>	-	le	-	pas	-	+
+	-	-	<E>	<accomplir>	-	le	-	rêve de ":Poss-0" vie	-	+
+	-	-	<E>	<accrocher>	-	le	-	linge	-	+
+	-	-	<E>	<accuser>	-	le	-	coup	-	+
+	+	-	<E>	<accuser>	-	les	-	années	-	-

Figure 4: A lexicon-grammar table for French frozen expressions in INTEX

(2) a "meta-grammar" in which the syntactic contexts of the frozen expressions are formalized. INTEX's meta-grammar is a specific type of syntactic grammar strongly linked to the above-mentioned table. Typically, the meta-grammar contains paths that represent elementary sentences such as NP V NP, NP V PREP NP, etc. as well as complex sentences that contain potential adverbial insertions, e.g. NP ADV V NP, verbal constructions with auxiliary, modal and aspectual verbs, as well as transformed structures (extraction, passive forms, etc.). The meta-grammar is linked to the table through the use of special variables that instantiate the grammar's lexical symbols. For instance, the meta-grammar's variable "@5" refers to the content of the table's 5th column (in Figure 4: the verb of the frozen expression); variable "@3" has a Boolean value that encodes expressions that must be recognized only in a negative construction, etc.

From the table and its meta-grammar, INTEX could automatically compile an autonomous syntactic grammar that can be applied to texts to automatically recognize occurrences of frozen expressions. (Vietri 2004) shows how to formalize with INTEX several Italian frozen expressions' lexicon-grammar tables.

Problems

This method involved using two objects that were specifically designed for the single task of frozen expression recognition: the lexicon-grammar table was incompatible with other

INTEX dictionaries, and the meta-grammar was incompatible with other INTEX syntactic grammars.

Because it was not possible to merge the meta-grammars with other grammars, such as the one for noun phrases, elementary sentences, coordination, etc., each meta-grammar had to be autonomous, i.e. it had to formalize every syntactic context in which a frozen expression could occur. But most frozen expressions are used in syntactic contexts that mimic free ones. For instance, the syntactic structures of the two sentences “John lost his mind” (frozen) and “John lost his hat” (free) are identical. Surely there might be a way to describe these two sentence structures with one unified tool! Unfortunately, INTEX forced linguists to describe these two structures twice, and, even more problematically, with two very different and incompatible tools.

Finally, each lexicon-grammar table / meta-grammar association had to be compiled into a finite-state transducer in which all the components of each frozen expression were represented: the resulting finite-state transducer was huge, typically hundreds of thousands of states, which is significantly larger than even the largest FSTs that encode INTEX’s dictionaries.

4. The NooJ method

In NooJ, by contrast, instead of compiling a finite-state transducer from a table and a meta-grammar, we use both a *standard* NooJ dictionary, and a *standard* syntactic grammar at run-time: there is no third, redundant object to compile.

-- The dictionary stores one **characteristic component** of each expression, i.e. a word form or a word sequence that occurs every time the expression occurs.

For instance, for the previous frozen expression “to take ... into account”, the word “take” and the sequence “into account” are both valid characteristic components. However, for a number of expressions such as “to be in trouble”, the verb can be replaced with a variant (e.g. “to get in trouble”) or even be deleted (e.g. “John, in trouble, decided to run away”). In those cases, the characteristic component of the verbal expression cannot be the verb *to be*; it will be the noun “trouble” or the sequence “in trouble”.

-- The characteristic component stored in the dictionary is associated with all the frozen expression’s properties, as well as with the other components of the expression. For instance, here is a lexical entry that could be used to represent the frozen expression “to take something into account”:

```
take,V+FLX=TAKE+SHum+OFree+O2=”into account”
```

V stands for verb; **FLX=TAKE** says that the lexical entry inflects according to the paradigm **TAKE**; **SHum** says that the subject of the expression is **Human**; **OFree** says that there is no distributional constraint on the direct object complement; **O2=”into account”** stores the frozen component of the expression in the property field **O2**. Thus, any reference to the variable **VO2** in the grammar will be automatically instantiated as the text “*into account*”.

In the previous example, the property field **O2** has a constant value: the sequence of two word forms. But a property field can also store a regular expression. For instance, the frozen expression “to lose one’s mind” can be described by the following entry:

```
lose,V+FLX=LOSE+SHum+O="<DET+Poss> mind"
```

In the corresponding grammar, the variable **VO** will be instantiated as “<DET+Poss> mind”, which then will match “my mind”, “your mind”, “his mind”, “her mind”, “its mind”, “our mind”, “their mind”.

Important: the dictionaries that store characteristic components of frozen expressions are exactly in the same format as regular NooJ dictionaries: the fact that all NooJ dictionaries have the same format allows linguists to unify the description of all types of linguistic units: from morphemes to frozen expressions (See Silberztein 2005).

-- This syntactic grammar describes the syntactic contexts of frozen expressions, in the exact same way as a “standard” syntactic grammar would describe the same contexts. Just like with any other syntactic grammars, the syntactic grammar that represents frozen expressions may include references to its lexical entries’ properties. Here, more specifically, when dealing with frozen expressions, it will include references to the fields that encode the various frozen components of the frozen expressions (above: **VO2** or **VO**).

The following syntactic grammar represents all the frozen expressions of the type “take X into account”:

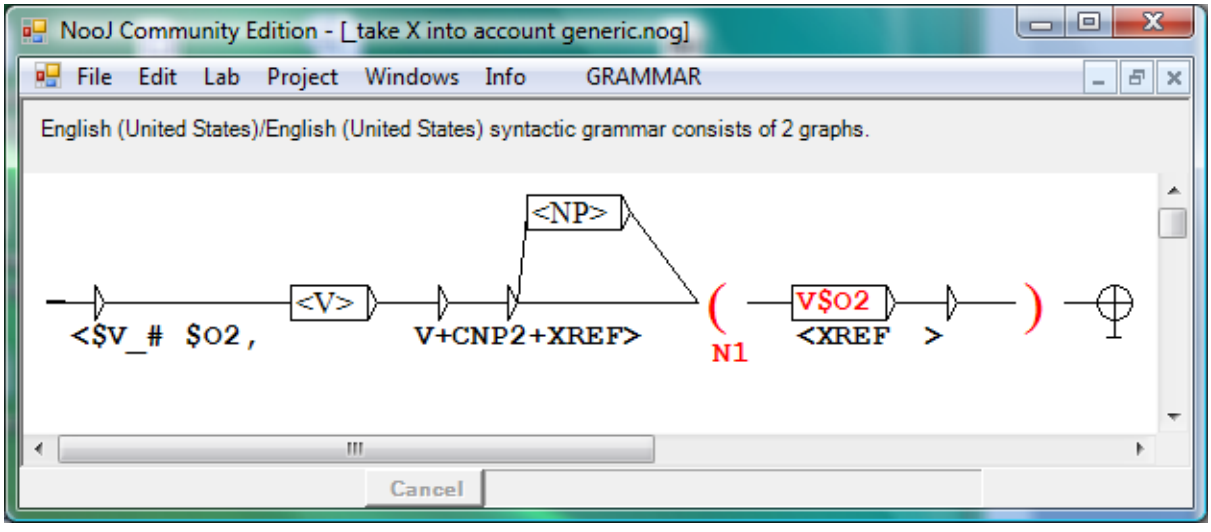

Figure 5: A Generic Syntactic Grammar for an Open Set of Frozen Expressions

Notice how similar this grammar is to the one in Figure 3: the lexical symbol **<take>** (any conjugated form of *to take*) has simply been replaced with the generic symbol **<V>** (any verbal form); the constant “into account” has been replaced with **VO2**, i.e. the value of the property field **O2** of the verb. When parsing the text *John took the problem into account*, the result is exactly the same as if we were applying the specific grammar, but the generic grammar can handle hundreds of expressions of the same type:

consider X with interest, have X in mind, set X in motion, treat X with respect

In terms of computing requirements, the generic grammar is extremely small (a few bytes), as opposed to the huge Finite-State Transducer that had to be compiled by INTEX (200,000+ states!).

But the most important benefit is that this grammar can be easily merged with a grammar that recognizes free sentences with the same structure, such as *John gave a pen to Mary*: we just need to add a <PREP> <NP> path in parallel with **VO2**. In other words, NooJ allows linguists to represent, manage, and accumulate a number of syntactic grammars for free structures as well as for frozen structures, in a unified way.

Properties' Inheritance

The previous graph can be successfully used to automatically recognize and lemmatize frozen expressions in texts, but the resulting annotation is poor: it only annotates the recognized sequences as a **V+CNP2** occurrence. We need to produce a finer analysis of each occurrence: for instance, we want to describe "took X into account" as an expression in the Preterit, and any syntactic or semantic information that was described in the lexical entry that represents the frozen expression has to be copied in the resulting annotation: the distributional selection of its subject, the possibility of having a passive form, etc.

For those purpose, we use the same special variables as the ones used in NooJ's morphological parser: $ALLF stores all the lexical entry's inflectional and derivational properties (such as tense, number), whereas $ALLS stores all the lexical entry's syntactic and semantic properties. The final syntactic grammar used to annotate all **CNP2**–type frozen expressions is thus:

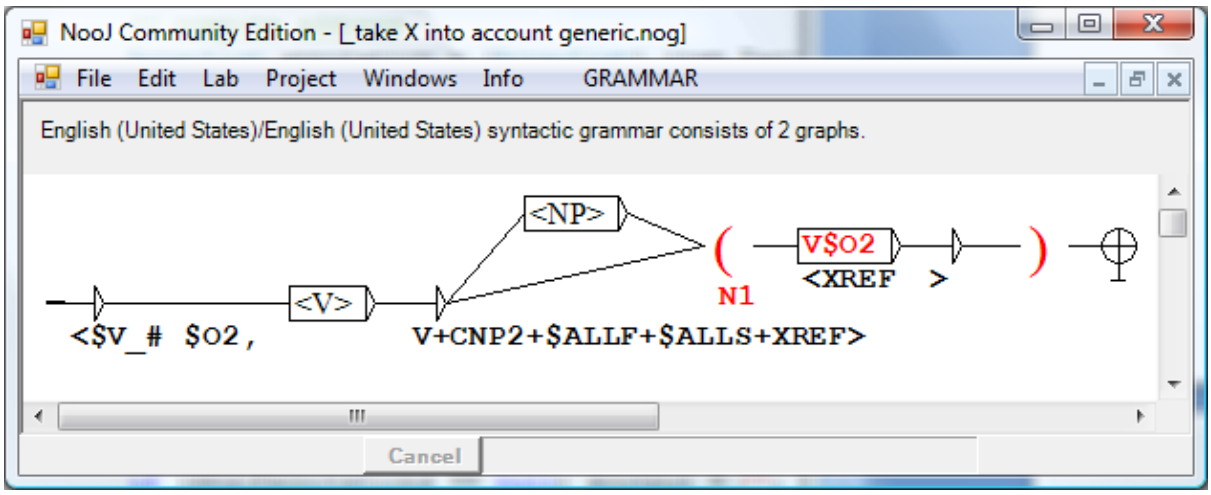

Figure 6: The Final Grammar

Applying this actual grammar to the text:

John took the problem into account

produces the following annotation:

```
<take into account,V+CNP2+Pt+SHum+OFree>
```

Artificial Ambiguities

The characteristic components of a large number of frozen expressions are entered into a dictionary, which might be merged with other NooJ standard dictionaries, or even other dictionaries that store characteristic components of other types of frozen expressions.

As a result, a dictionary might contain thousands of lexical entries for a relatively small number of characteristic components, such as support verbs such as "get", "have", "take", particles such as "in", "up", etc. (these verbs and particles are components of thousands of frozen expressions). In some cases, we can choose the characteristic components to minimize the number of its lexical entries: for instance, we might decide to describe the expression "to take into account" with the lexical entry "account". However, it has to be expected that we will end up processing a large number of lexical ambiguities, much larger than the ones we are accustomed to at the lexical and morphological levels (3 to 10 in Romance languages).

NooJ (as opposed to INTEX) has been designed from the ground up to handle such large degrees of ambiguities. Importantly, however, these ambiguities are artificial and unnecessary, and therefore should not pose a problem in the first case. For instance, the hundreds of lexical entries for the characteristic component *take* do not correspond to the "real" verb *to take*, but rather to a hundred independent frozen expressions or combinations of the support verb and its predicative noun.

In order to stop NooJ from processing characteristic components as highly ambiguous "ordinary" lexical entries, we have added a specific property: **FXC** (Frozen eXpression Component), that states that the current lexical entry must not be processed as an ambiguous linguistic unit. Thus, the frozen expression "take into account" is described by the following lexical entry:

```
take,V+FXC+FLX=TAKE+SHum+OFree+O2="into account"
```

Now, even if there are thousands of lexical entries for "take", NooJ's lexical parser will produce annotations only for the lexical entries that are not marked as **FXC**. Therefore, the annotations that represent frozen expressions will only be produced by a grammar.

5. Conclusion

The combination of the special property **FXC** in dictionaries and the special code **XRF** in grammars allows linguists to use standard dictionaries and standard grammars to represent and automatically annotate both free structures and frozen expressions with the same tools, even when the latter are discontinuous.

We have successfully formalized a description of over 12,000 English phrasal verb in the form of a dictionary "phrasal verbs.dic" and the corresponding syntactic grammar "phrasal verbs.nog" (see Machonis 2007, intra). Here are three entries that correspond to the phrasal verbs *to give away*, *to give back*, *to give up*:

```
give,V+PV+FXC+Part=away+N0Hum+N1Hum+N1NHum+NoPart+FLX=GIVE
give,V+PV+FXC+Part=back+N0Hum+N1Hum+N1NHum+NoPart+FLX=GIVE
give,V+PV+FXC+N0Hum+N0NHum+Part=up+N1Hum+N1NHum+FLX=GIVE
```

The particle is described in the property field **Part**; the distributional selections of the subject and the object are described (e.g. **N0Hum**), and if the particle can be omitted, the expression has the property **NoPart**.

Applying this dictionary to large texts takes exactly the same computing time as applying any other NooJ dictionary. Although the use of property field variables is likely to consume some computing power, in practice we have not noticed any difference when applying "regular" and "enhanced" syntactic grammars to large texts.

References

Gross, Maurice, 1984. Une classification des phrases figées en français. In *Linguisticae Investigationes Suplementa vol. 8 : De la syntaxe à la pragmatique*. Attal, Muller Eds. John Benjamins Publishing Company : Amsterdam, Philadelphia.

Gross, Maurice, 1993. Les phrases figées en français. In *L'information grammaticale no 59*, pp. 36-41 : Paris.

Machonis Peter, 2007. NooJ : a Practical Method for Parsing Phrasal Verbs. In the *Proceedings of the 2007 NooJ Conference*. Autonomous University of Barcelona Eds: Barcelona.

Silberztein, 2005. NooJ's Dictionaries. In the *Proceedings of 2nd Language and Technology Conference*, Poznan University: Poznan.

Max Silberztein, 2007. NooJ's Linguistic Annotation Engine. In *Formaliser les langues avec l'ordinateur : de INTEX à NooJ*. Cahiers de la MSH Ledoux. Koeva, Maurel Silberztein Eds. Presses Universitaires de Franche-Comté : Besançon.

Vietri Simona, 2004. Lemmatization of Idioms in Italian. In *INTEX pour la Linguistique et le Traitement Automatique des Langues*, pp. 173-198. Cahiers de la MSH Ledoux. Muller, Royauté, Silberztein Eds. Presses Universitaires de Franche-Comté : Besançon.