NooJ: A Linguistic Annotation System For Corpus Processing

Max Silberztein
LASELDI
Université de Franche-Comté
Besançon, 25000 France
max.silberztein@univ-fcomte.fr

1 Introduction

NooJ is a new corpus processing system, similar to the INTEX software,[1] and designed to replace it. NooJ allows users to process large sets of texts in real time. Users can build, accumulate and manage sophisticated concordances that correspond to morphological and syntactic grammars organized in re-usable libraries.

One characteristic of NooJ is that its corpus processing engine uses large-coverage linguistic lexical and syntactic resources. This allows NooJ users to perform sophisticated queries that include any of the available morphological, lexical or syntactic properties. In comparison with INTEX, NooJ uses a new technology (.NET), a new linguistic engine, and was designed with a new range of applications in mind.

2 A new software architecture

NooJ's architecture is based on the .NET "Component programming" technology, which goes a step beyond the Object-Oriented approach (Silberztein 2004). This architecture gives it several advantages, including:

(1) it allows NooJ to read any document that can be managed on the user's computer. For instance, on a typical MS-Windows computer, NooJ can process corpora in 100+ file formats, including all variants of ASCII, ISO and Unicode, HTML, RTF, XML, MS-WORD, etc.

[1] Cf. (Silberztein 1999a) for a description of the INTEX toolbox, and (Silberztein 1999b) for a description of its application as a corpus processing system. See various INTEX WEB sites for references and information on its applications, workshops and communities: http://intex.univ-fcomte.fr and the NooJ WEB site for a description of NooJ: http://www.nooj4nlp.net.

(2) it allows other .NET applications to access all NooJ's public methods via its software component library. For instance, a programmer can easily run a NooJ method to extract sequences of texts that match a NooJ grammar from a document that is currently opened in the current application (e.g. MS-WORD).

3 A new linguistic engine

As a corpus processing system, NooJ's most important characteristic is its linguistic engine, which is based on an annotation system. An annotation is a pair (*position, information*) that states that at a certain position in the text, a sequence is associated with a certain piece of information. NooJ processes texts that are *annotated*; annotations are stored in each text's annotation structure which is synchronized with the text buffer. Text annotations that are represented as XML tags can be easily imported to NooJ; for instance, importing the XML text:

```
<N Hum> Mr. John Smith </N>
```

will produce an annotated text in which the sequence "Mr. John Smith" is annotated with the tag "N+Hum" (annotation category "N"; property "Hum"). NooJ also provides several powerful tools to annotate texts:

-- NooJ's morphological parser is capable of analyzing complex word forms, such as Hungarian words and Germanic compounds, as well as tokenizing Asian languages. The morphological parser annotates complex word forms as sequences of annotations. For instance, the contracted word form "don't" is associated with a sequence of two annotations: <do,V+Aux+PR> and <not,ADV+Neg>.

-- NooJ's lexical parser can process the inflection of large dictionaries for simple and compound words. For instance, the English dictionary contains 100,000+ simple words and 70,000+ compound nouns. NooJ contains large-coverage dictionaries for Arabic, Armenian, Chinese, Danish, English, French, Hungarian, Italian and Spanish. In general, running NooJ's lexical parser results in adding multiple lexical annotations to a text. The annotation system can represent all types of lexical ambiguities, such as between compounds and sequences of simple words (e.g. "round table"), overlapping or embedded compounds (e.g. "round table mat"), etc.

-- NooJ's local grammars are Recursive Transition Networks; they allow users to recognize certain sequences of texts, and to associate them with annotations. NooJ's graphical editor contains a dozen development tools to edit, test and debug local grammars, to organize them in libraries, and to apply them to texts, either as queries or to add (or filter out) annotations.

NooJ's query system and parsers can access any previously inserted annotation. For instance, the following query includes references to word forms (e.g. "mind") as well as to two annotations (written between brackets):

```
(the + these) <N+Hum> <lose>
their (mind + temper)
```

<N+Hum> matches all sequences in the text that are associated with an "N" annotation with property "Hum"; these annotations might have been added by NooJ's lexical parser (e.g. for the word "director"), or by a local grammar used to recognize human entities (e.g. for the sequence "head of this company"). Similarly, <lose> matches all sequences of the text that are associated with an annotation whose lemma is "lose"; these annotations might have been added by the lexical parser (for all conjugated forms of "to lose", e.g. "lost"), or by a local grammar that recognizes compound tenses, e.g. 'have not yet lost". When all resulting matching sequences, e.g. "These men have not yet lost their mind", have been indexed, they can be annotated, and their annotation is then instantly available either for other queries or for further cascaded parsing.

Annotated texts can be used to build complex concordances, annotate or color texts, perform a syntactic or semantic analysis, etc.

NooJ's linguistic engine, dictionaries and grammars are multilingual; that should allow users to implement translation functionalities.

4 Conclusion

Although NooJ has just come out and its technology is quite new, it is already being used by several research teams in a variety of projects. See the proceedings of the "Eight INTEX/NooJ workshop" at NooJ's WEB site: http://www.nooj4nlp.net.

5 Demo

Participants will use NooJ in order to build a named-entity recognizer from the ground up. Participants will learn how to apply a simple query to a corpus and build its corresponding concordance. Then I will demonstrate the building of a local grammar with NooJ's graphical editor, followed by a presentation of the organization of local grammars in re-usable libraries that can be shared and integrated into larger grammars.

References

Silberztein Max. 1999. INTEX: a finite-state transducer toolbox. In Theoretical Computer Science #233:1, pp. 33-46.

Silberztein Max. 1999. Indexing large corpora with INTEX. In Computer and the Humanities #33:3.

Silberztein Max, 2004. NooJ: an Object-Oriented Approach. In *INTEX pour la linguistique et le traitement automatique des langues*. C. Muller, J. Royauté, Max Silberztein eds. Cahiers de la MSH Ledoux, Presses Universitaires de Franche-Comté., pp. 359-369.