[*First International Conference on Language Resources & Evaluation*, Granada, Spain, 28-30 May 1998]

Evaluation of Terminology Extractors: Principles and Experiments

D. Bourigault & B. Habert
LLI - Université Paris 13 - db@lli.univ-paris13.fr
LIMSI & UMR 9952 - École Normale Supérieure de Fontenay St Cloud - bh@ens-fcl.fr

Abstract

We claim that the evaluation of terminology extraction systems cannot rely entirely on the comparison of their results with a reference list, as the very existence of such a list made up independently of any application is highly questionable and as these systems do not yield only actual terms but potential (or candidate) terms. We propose instead to align their results on various corpora, so as to isolate the divergences and the convergences and to understand their rationale. This paper compares two noun phrase (NP) extraction softwares. It proved necessary to translate the NP parses produced by these softwares into a pivot grammar. We developed a Frontier to Root Transducer based on syntax-directed translation schemas used for standardization and optimization in the compiling process. These tried and tested compiling techniques provide an efficient and nevertheless declarative way of obtaining mappings between very different parses ("skeleton parses" / detailled constituent structures) and to evaluate the underlying softwares.

1 Context: Computational Terminology

The role of terminology has increased significantly over the last decade. Various efforts have been made for the development of tools and methods dedicated to the automatic processing of terms. The need for terminology resources has become particularly acute owing to the globalization of scientific and technical exchanges and the concurrent development of international communication networks. The stakes are high: scientific and technical exchanges between language and/or specialists communities must be made more efficient (translation), the increasing volume of electronic texts now available on communication networks must be mastered and managed (accessing information).

But the costs related to the gathering of, access to, and maintenance of terminology resources are high and will continue to increase dramatically if these tasks are accomplished using conventional methods. This is why research on computerised tools for terminology acquisition and use is of major interest. In the French-speaking world, computational terminology is a growing, though young, area of research. More and more research systems are developed for the acquisition of terminology data from corpus *(e.g.* `Termino`, `Lexter`, `Nomino`, `Ana`, `Faster`, `Acabit`). These "Terminology Extraction Systems" (henceforth TES) are corpus processing tools which extract, by means of morpho-syntactic, statistical or hybrid methods, sequences of words (most of the time NPs) which are potential or "candidate" terms (henceforth CT).

2 Evaluating Terminology Extraction Systems: Theoretical Background

Terminology extraction systems have not been intensively used in real world applications yet. But conceptors and users of such tools need already to evaluate them. Trying to adapt tried and tested techniques and experimental frames directly from the information retrieval community seems promising. However, we do not think it is the most appropriate method. In particular, it is our view that it is not a tenable option to aim at an absolute evaluation of terminology softwares by comparing their results with a reference list which would make up "the" terminology of the subject field, for the following reasons: first the very existence of such a list made up independently of any application is highly questionable (section 2.1) ; secondly what comes out of the computer are not (supposed to be) terms but candidate terms (section 2.2).

2.1 Terminology Products and "Reference Lists"

With the development of the information society, the extension of text resources accessible on networks and the demands of multilinguism, the research in terminology have to take into account the diversification and the extension of terminology. The "classical" terminology databank in which each term is described according to a more or less standard predefined pattern (the "terminology record") is not an appropriate answer to these needs. Rather than talking about "the" terminology of a domain, one should consider instead different kinds of "terminology products" matching different kind of uses, such as multilingual lexical resources for computer-aided translation systems, terminological indexes for electronic documentation, thesauri for full text browsers, lexical resources for NPL systems, ontologies for knowledge-based systems ...

Though the main component of these terminology products is a description of the vocabulary used by the specialists of the field, nevertheless, for the same given field and for the same corpus, the list of the described lexical elements (the *base list* chosen by an analyst) and the way they are described (the *mode of description:* definitions, links with other terms, contexts etc.) mainly depend on the function assigned to the terminology product. It is as a matter of fact essential to take into account cognitive factors (ergonomic, pragmatic ones) related to the actual use of the product. In these conditions, it seems extremely tricky to work out what could be considered as a reference list. For that reason, it would be a little bit reductionist to base the evaluation of terminology extraction softwares on the comparison between this reference list and the candidate terms extracted by these softwares.

2.2 From candidate terms to terms

Before developing a method for evaluating terminology extraction softwares, one has to think about the way these softwares are being used or should be used. The report of experience concerning the use of such softwares are still quite rare and there is still a lot of work to do to develop a method of use really appropriate.

Our own experience of the subject comes from various experiences in which the software Lexter (Bourigault, 1993) is used. Lexter has been developed at the Research and Development Division of the French Electricity Board (EDF/DER), in the framework of a collaboration between EDF/DER and the French National Scientific Research Centre (CNRS). Lexter performs a morpho-syntactical analysis of a corpus of French texts on any technical domain and yields a network of NP which are likely to be terminological units (CT). These results are then validated by a human analyst who is in charge of building a terminological product (a thesaurus for an indexing system, a terminological index for electronic documentation, a specialized dictionary, among others). Lexter is being used in several R&D projects, at the same time by EDF/DER and in some CNRS laboratories. The users' observations tell us more about the way the results from Lexter are used in the real context of an application.

Making a terminology is a work consisting in building up, modelling knowledge and which is based on the analysis of a reference corpus and on the discussion with experts. Using the results of a terminology extraction software to make a terminology doesn't come down to tick *yes* or *no* in front of the extracted candidate terms. One has to make up the description of each entry: definitions, links with other terms (synonymy, hyperonymy, etc.), links with relevant contexts, equivalents in other languages, etc. To built up this description, the user has to look at the contexts of occurrences of the candidate terms. The user needs an ergonomic and user-friendly validation interface which enables her/him to read the contexts from which the software has extracted the CT. The use of such an interface allows a transversal reading of the corpus - for a given CT, all text units from which it has been extracted - which complements a classic linear way of reading. If the software proposes syntactic kind of links between CT, as Lexter does, the navigation in the network of CT also allows another alternative way of reading.

According to the experiments, the rates of candidate terms which are not accepted, in the end, as elements of the base list depends, as we mentioned it earlier, of the kind of terminology product built up. It can vary from 30% to 70%. But one mustn't think too quickly that this figure is an evaluation of the noise, like one would do with classic information retrieval applications. The really irrelevant terms the analyst eliminates correspond to syntactic analysis errors of the software and are very easily detected: therefore they don't disturb the building up of the terminology. The study of the contexts of the CT which won't be accepted in the end is not always useless, as it allows the analyst to progress towards a better understanding of the field.

Thus, the evaluation based on a comparison to a reference list presents some limits. As we have just seen , one must take into account the phase of human validation necessary to transform the list of CT into a quality terminology. We should be working on this validation and construction phase, adapting the principles and procedures well known in software engineering. Generally speaking, the issue is to size up as accurately as possible the costs and benefits derived from putting the software product into professional practice, bearing in mind that using the software will itself modify practices.

Therefore, one needs a long-draw-out reflection. From this perspective, it appeared to us that comparing the results of two NP extractors developed by different teams and for different applications could be an initial study full of lessons to be learned concerning the research of a general methodology to evaluate terminology extraction softwares.

3 Aligning NP Extractors

3.1 Context

For several years, EDF/DER has been leading researches on the application of Natural Language Processing to the field of Electronic Document Management. Two main areas are concerned: automatic text indexing and semi-automatic terminology construction. To index the documents written by its staff, EDF/DER uses several software tools. Some of them were developed by GSI-Erli company in the framework of the Eureka GRAAL project (Sabbagh and team, 1994; HP, 1996). These tools make use of NLP techniques to analyse French (or English) texts and yield a standardized representation with keywords belonging to a thesaurus. One of these tools (AlethIP-NP) is a NP extractor whose function is to extract, from the input text, NPs which will then be compared with the entries of a thesaurus (controlled indexing). As an answer to the needs for building and updating the thesaurus exploited by the automatic text indexing system mentioned above, the terminology extraction software Lexter is used as well.

Lexter and AlethIP-NP are two NP extractors available at EDF/DER. They were developed by different teams for different purposes. Before maintaining the investment on such rather similar tools, EDF/DER managers needed a fair evaluation on the complementary features and results of these extractors. They funded a project for an empirical comparative evaluation of these two softwares, whose goal was to explain their differences and to identify their respective weak and strong points. In this paper, we report the initial work we did in this project, namely the translation of Lexter and AlethIP-NP parses into a pivot grammar.

3.2 How to compare NP parses resorting to different grammars ?

To make an assessment of Lexter and AlethIP-NP without resorting to the "reference list" paradigm, an empirical approach was adopted: comparing the NP parses produced by both softwares for each sentence of two test corpora. We chose two middle-size length test corpora (around 200 000 words), *i.e.* large enough to exhibit all the different cases on which one software could differ with the other, but short enough to allow for a qualitative analysis of the results: an automatic comparison brings out the relevant discrepancies which can be carefully examined. The first test corpus is the *Guide for Regional Electrical Networks Planning.* This corpus (henceforth the GDP corpus) is the reference document used by regional planners

who forecast the short and middle term evolution of electric networks. It is one of the corpora on which Lexter was used to build a terminological index. The second test corpus is a set of scientific paper abstracts written by EDF/DER research-engineers. This corpus (henceforth the ABS corpus) was extracted from the general collections of EDF/DER internal reports on which AlethIP-NP is currently used for automatic indexing.

Let us give a flavor of the problem with an input sentence (#7880 in GDP corpus). Lexter's results are underlined and AlethIP-NP's ones are in **bold face:**

<SEQ NUM="00769-07880"> <TXT>( ) : Productible hydraulique : Ce terme designe la* **quantite maximale** *d'***energie electrique** *que l'***ensemble des apports corriges** *(c'est-a-dire : en dehors de toute* **gestion humaine** *)* **constates pendant l' intervalle de temps considere** *permettrait de produire dans les* **conditions les plus favorables de disponibilite des groupes** *. </TXT>*

Note that the only NP which is obviously a term *(ce terme designe ...) — productible electrique* - was not extracted: *productible* was wrongly tagged as an adjective, and the sequence of two adjectives did not constitute a valid NP. In this example, Lexter's sequences are generally included within AlethIP-NP's ones[1]. However *quantite maximale d' energie electrique*, which Lexter considers as a unit, is split into two NP by AlethIP-NP. The parse trees differ as well. AlethIP-NP produces for instance the following parse for *conditions les plus favorables de disponibilite des groupes: [gNP [gNP [frNoun CONDITION][gCompAdj [frAdv LES-PLUS][adjStd FAVORABLE]]][gPP [frPrep DE][gNP [frNoun DISPONIBILITE][gPP [frPrep DE][gNP [gDet [frArt LES]][frNoun GROUPE]]]]]]*, whereas for the sub-group *disponibilite des groupes*, Lexter yields: *[H [NounFS DISPONIBILITE][E [Prep DE][Det LES][H [NounMP GROUPE]]]]*. As they rely on the same tagger, AlethIP-NP and Lexter use a pre-terminal tagset of about the same size (twenty tags), half of these tags corresponding to "guesses" made by the tagger (for instance `xxAdj` for a word which is believed to be an adjective). However, each parser changes some tags for its own purposes[2] and uses an idiosyncratic terminal and non-terminal tagset. Mapping the two tagsets into a pivot one would not be sufficient to compare the parse trees. As a matter of fact the main differences concern the description of the syntactic structure. These parsers produce different bracketings, resorting to very different principles, as seen on the same NP: *gestion hebdomadaire de le parc hydraulique national*[3]. Lexter gives "skeleton" dependency trees, with only two non-terminal tags: H, for *Head;* E, for *Expansion*, which gathers modifiers and arguments (see figure 1). AlethIP-NP proposes more classical and detailled constituent structures[4] (see figure 2).

It was therefore necessary to find a way to translate these very different parses into a pivot grammar. Choosing a pivot grammar is not an easy task at all. During the experiment, we successively adopted different trade-offs between a grammar richer than Lexter's one in order to have more precise information available for the comparison, and a grammar poorer than AlethIP-NP's one to make the assessment of the results easier for the human analyst. Since the very beginning of the experiment, we knew that we would have to proceed by trial and error, before choosing a definitive pivot grammar. This is the reason why we decided to develop an efficient transducer, which could allow us to quickly tune the best pivot format (at least according to our wishes and current purposes). This transducer is described in section 3.4.

3.3 A sentence to sentence "maximal-length" NP comparison

In order to accurately identify the main differences between the two softwares, it is not sufficient to compare their results at the global level of the entire corpus. Knowing that a given NP was extracted by the first software *n* times more than by the second one is not that helpful. Instead we need to examine the contexts (the sentences) from which the second software did not extract this NP. We have to identify the precise reasons for such a difference with respect to the way both softwares process the same sentence, in terms of grammar rules applying or not. That's why we chose a sentence to sentence comparison.

Furthermore, we chose to compare only the parses of the "maximal-length" noun phrases extracted by both softwares from each sentence, that is NP which are not themselves constituents of larger NP within the same sentence. Since both softwares propose as outputs these maximal-length noun phrases together with their NP subconstituents, the identity of the maximal-length NP parses implies the identity of all the sub NP extracted.

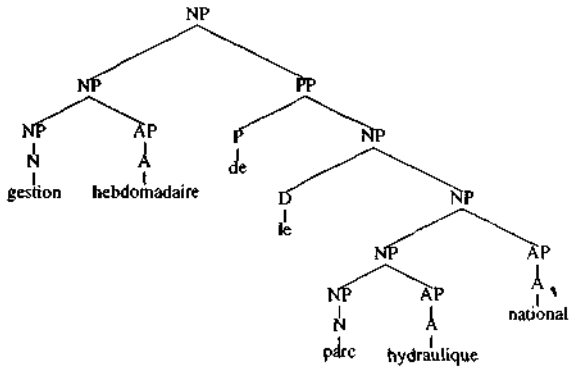

Figure 3: Transformed parse tree for *gestion...*

The overall comparison process is then as follows:

1. Lexter and AlethIP-NP process the GDP and ABS corpora. Only the maximal-length NP parses are kept, together with the number of the sentence they are extracted from, in order to permit the sentence to sentence comparison ;

[1]The sequence **ensemble des apports corriges constates pendant 1'intervalle de temps considere** is even analysed by AlethIP-NP as a single NP, in spite of the parenthesis *(c'est-a-dire, en dehors de toute gestion humaine).*

[2]Lexter for instance has different tags for the gender and number inflections of nouns and adjectives (`AdjFS` / `AdjFP` / `AdjMS` / `AdjMP`).

[3]*Weekly management of national hydraulic power.*

[4]Note the different bracketing obtained after tag removal for the PP *de les groupe:* `[ [ DE] [ LES] [ [ GROUPE] ] ]` for Lexter/ [[DE] [[[LES]] [GROUPE]]] for AlethIP-NP.

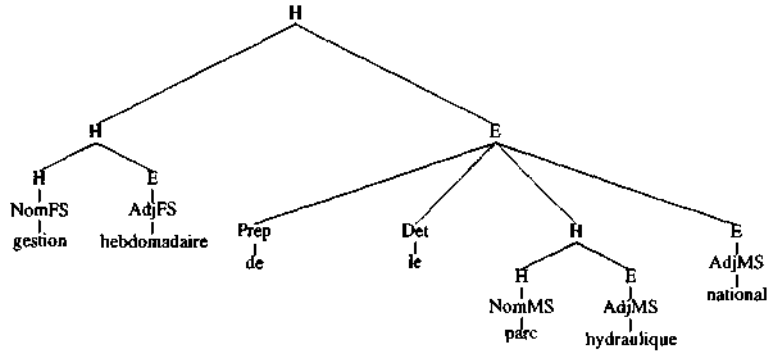

Figure 1: Lexter parse tree for *gestion hebdomadaire...*

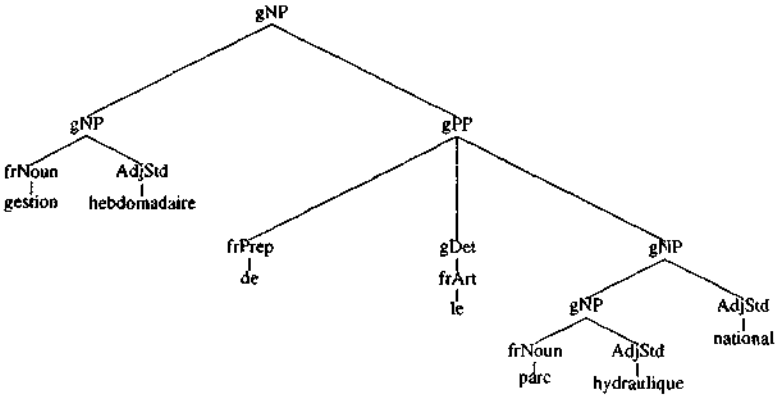

Figure 2: AlethIPGN Parse tree for *gestion hebdomadaire...*

2. The transducer rewrites the parses produced by the softwares into the pivot grammar. In figure 3 is the translated version of the parses given in figure 1 and figure 2 ;

3. A comparison program aligns sentence to sentence the translated NP parses. It compares each NP parse produced by Lexter successively to each parse produced by AlethIP-NP - and then makes the same comparison with the sequences of lemmas if necessary (see section 4).

3.4 A Frontier To Root Pushdown Transducer

We choose to rely on tried and tested techniques used in compiling: pushdown transducers (Aho and Ullman, 1972a; Aho and Ullman, 1972b, ch. 3 and 9). They are obtained by providing a pushdown automaton with an output. They are useful for standardizing the results of the parsing process (for instance mapping arithmetic expressions in infix notation into equivalent expressions in Polish notation) or for making them more efficient (Wilhelm and Maurer, 1995, ch. 11). We developed a variant of a frontier to root transducer (henceforth FRT). As the output of Lexter is rather similar to a "skeleton parse", that is, a simple bracketing of the words in the analyzed NP with a part-of-speech tagging of these words, it was necessary to use a bottom-up syntax-directed translation schema: the non-terminal symbols were too few (2 !) to allow for a top-down translation.

This kind of transducer (Gecseg and Steinby, 1984, ch. IV, p. 139-143) rewrites a tree from its leaves to its root. The one we developed uses rules such as the following ones :

(1) $[E[Adj\ a_1]] \Rightarrow [AP(A\ a_1]]$
(2) $[H[Noun\ a_1]] \Rightarrow [NP[N\ a_1]]$
(3) $[H[N\ P\ a_1][AP\ a_2]] \Rightarrow (NP[NP\ a_1][AP\ a_2\}]$
(4) $[H[NP\ a_1][PP\ a_2]] \Rightarrow$
$[NP[NP\ a_1][PP\ a_2]]$
(5) $[E[Prep\ a_1][Det\ a_2][NP\ a_3]] \Rightarrow$
$[PP[P\ a_1][NP[D\ a_2][NP\ a_3]]]$

These rules relate a source (on the left-hand side) and a target subtree. When the source subtree is present at the current node of the frontier, it is replaced with the target subtree. The indices show the parts matched in the source subtree which must be transferred as such in the target subtree. For instance, according to rule *5*, an E node dominating a Prep node, a Det node and an NP node, can be replaced by a PP node dominating a P node and a NP node dominating a D node and another NP node (a level of embedding is added). The sub-trees dominated by Prep, Det and NP are then copied under the P node, the D node and the second NP node respectively.

The frontier is examined from left to right as many times as necessary. At the beginning, the frontier (in boldface in the example below) consists in the leaves of the tree. When a source subtree matches at the current node of the frontier (in a box in the example), the root of the corresponding target subtree replaces at the frontier the nodes which were dominated by the source tree (the frontier is partially reduced). The nodes *below* the frontier (in italics in the example) cannot be modified any more.

The rewriting process stops either when there is only one node remaining at the frontier (in this case, the whole tree has been transformed) or when it is not possible any more to transform the current tree (in that case, the tree is partially transformed).

Let us examine the work of this transducer on t_1, a sample of Lexter output on GDP corpus (see figure 1) with the leaves (in boldface) as initial frontier and the current node in a box:

[H [H [H [Noun [**gestion**]][E [Adj **hebdomadaire**]]] [E [Prep **de**][Det **le**][H [H [H [Noun **parc**]][E [Adj **hydraulique**]]][E [Adj **national**]]]]]

The source-tree of rule 2 matches at the left corner of the frontier: [H [Noun **gestion**]] is replaced with [**NP**$_{(2)}$ *[N gestion]]*[5]. Thanks to the index a_1, the terminal symbol *gestion* is copied in the target subtree which is inserted in t_1. Note that the frontier has changed: *gestion* and its father are no longer accessible. The FRT shifts to the next node at the frontier. The new frontier of t_1 is now:

[H [H [**NP**$_{(2)}$ *[N gestion*]][E [Adj **hebdomadaire**]]]] [E [Prep **de**][Det **le**][H [H [H [Noun **parc**]][E [Adj **hydraulique**]]][E [Adj **national**]]]]]

Scanning the rest of this new frontier leads to the use of rules 2 and 1 on the rest of the frontier. As the tree is not totally transformed, the frontier is examined again from the left-hand side. The current item is then at the beginning of the changed frontier:

[H [H [**NP**$_{(2)}$ *[N gestion]]* [**AP**$_{(1)}$ *[A hebdomadaire]]]* [E [Prep **de**][Det **le**][H [H [**NP**$_{(2)}$ *[Noun parc]]* [**AP**$_{(1)}$ *[A hydraulique]]]* [**AP**$_{(1)}$ *[A national]]]]]*

During the second traversal of the (changing) frontier, the rule 3 yields the NP for *gestion hebdomadaire* and for *parc hydraulique.* A third traversal makes an NP with rule 3 from *parc hydraulique* and the AP *national.* Here is the situation *after* this third traversal:

[H [[**NP**]$_{(3)}$ [*NP$_{(2)}$ [N gestion]] [AP$_{(1)}$ [A hebdomadaire]]]* [E [Prep **de**][Det **le**] [**NP**$_{(3)}$ *[NP$_{(3)}$[NP$_{(2)}$ [Noun parc]] [AP$_{(1)}$ [A hydraulique]]] [AP$_{(1)}$ [A national]]]]]*

A fourth traversal makes a PP, and adds an embedding level inside it (a new NP dominates the determiner and the NP *parc hydraulique national):*

[H [[**NP**]$_{(3)}$ [*NP$_{(2)}$ [N gestion]] [AP$_{(1)}$ [A hebdomadaire]]]* [**PP**$_{(5)}$ *[Prep de][NP [Det le] [NP$_{(3)}$ [NP$_{(3)}$[NP$_{(2)}$ [Noun parc]] [AP$_{(1)}$ [A hydraulique]]] [AP$_{(1)}$ [A national]]]]]*

The fifth and last traversal ends the transduction and makes a NP from the NP *gestion hebdomadaire* and the PP *de le parc hydraulique national:*

[[**NP**]$_{(4)}$ *[NP$_{(3)}$ [NP$_{(2)}$ [N gestion]] [AP$_{(1)}$ [A hebdomadaire]]] [PP$_{(5)}$ [Prep de][NP [Det le] [NP $_{(3)}$ [NP$_{(3)}$ [NP($_{2}$) [Noun parc]] [AP$_{(1)}$ [A hydraulique]]] [AP$_{(1)}$ [A national]]]]]*

The whole tree has been translated.

The tree rewriting techniques available in Prolog III could have been used. However, we needed a very fast translation program, running on different computers (Sun workstations and PCs), with different operating systems (SunOS, Solaris, Linux). We wanted as well to ease the interface with other programs. We therefore developed this FRT in C as an elaborate filter which can be pipe-lined with other processes. The rapidity of this FRT (between 60 and 90 trees transformed per second, i.e. 16,000 trees transformed in 3 or 4 minutes) proved adequate to test a set of rules and tune them quickly.

The rules available with this FRT permit to change the labels of the pre-terminal and the non-terminal symbols, to add levels of embedding (as in rule 5 above), or to eliminate "spurious" embeddings[6]. This last facility was very important to standardize AlethIP-NP output, as this parser has a great number of unary productions, resulting in sub-trees such as [/`gDet` [/`frArt` LE]] or [/`gInc` [/`xx I`]].

It is as well possible to change the linear order of the daughters of a node, or even to add or suppress nodes. However, theses features were not relevant for our task. Note that this FRT is deterministic: only one rule can be applied in a given configuration. Additionally, the fact that trees of arbitrary depth can be matched gives more control on the translation process.

4 Comparison

In the output of this FRT on AlethIP-NP and Lexter NP parses, there are only 11 pre-terminal symbols and 5 non-terminal ones. About forty rules only[7] were necessary to obtain this syntactic alignment.

Table 1 provides some numerical information on AlethIP-NP and Lexter parses in GDP and ABS corpora and helps comparing these softwares as far as the length and the complexity of the trees before and after the use of this FRT are concerned. The NP parses extracted represent more than 30 % of the corpora. AlethIP-NP builds trees which are less numerous but longer, and more complex than the ones produced by Lexter.

Table 2 shows that the number of exact matches between the two softwares (#1) is rather important. The number of parses and sequences found only by one software (#6+#7) makes a high score as well. Surprisingly, the number of sequences which differ only in bracketing (#4) is rather small. The detailled results of the comparison were used to improve both softwares, the advantages of one of them revealing the limits of the other. The qualitative importance of the differences was assessed in function of the interest regarding the building up of a terminology (Habert et al., 1997). Below are some lessons drawn from this experiment regarding the Lexter software.

[5]The number of the matching rule is given in the resulting tree.

[6]As a matter of fact, our FRT is rather similar to the pushdown processor (PP) defined in (Aho and Ullman, 1972b, ch. 9, p. 737): "A pushdown processor is a PDT [pushdown transducer] whose output is a labeled directed graph, generally a tree or part of a tree which the processor is constructing. The major feature of the PP is that its pushdown list, in addition to pushdown symbols, can hold pointers to nodes in the output graph. Like the extended PDA [pushdown automaton], the pushdown processor can examine the top k cells of its pushdown list for any finite k and can manipulate the contents of these cells arbitrarily. Unlike the PDA, if these k cells include some pointers to the output graph, the PP can modify the output graph by adding or deleting directed edges connected to the nodes pointed to. The PP can also create new nodes, label them, create pointers to them, and create edges between these nodes and the nodes pointed to by those pointers on the top k cells of the pushdown list."

[7]44 for AlethIP-NP and 38 for Lexter

Table 1: Tree parameters in *GDP* and ABS corpora.

	AlethIP	Lexter
# trees	*15,503*	*16,514*
	16,241	18,623
# tree types b. FRT	*2,247*	*2,585*
	3,260	3,842
# tree types a. FRT	*1,883*	*780*
	2,716	1,400
# words	*65,122*	*59,886*
	81,837	75,821
max # words	*34*	*18*
	33	28
mean # words	*4.20*	*3.63*
	5.04	4.07
max # non-terminals b. FRT	*74*	*34*
	74	77
mean # non-terminals b. FRT	*7.90*	*7.42*
	9.76	8.52
max # non-terminals a. FRT	*61*	*39*
	88	74
mean # non-terminals a. FRT	*10.34*	*9.13*
	12.59	10.30

Table 2: Comparison on *GDP* and ABS corpora.

#		*GDP*	ABS
1	Lexter parse = AlethIP-NP parse	8,530	7,621
2	Lexter parse ⊂ AlethIP-NP parse	2,179	3,387
3	AlethIP-NP parse ⊂ Lexter parse	1,138	1,561
4	Lexter seq. = AlethIP-NP seq.	501	771
5	Lexter seq. ⊂ AlethIP-NP seq.	1,861	2,584
6	AlethIP-NP seq. ⊂ Lexter seq.	330	542
7	Lexter parse alone	2,250	2,638
8	AlethIP-NP parse alone	1,979	1,738

4.1 Coordination processing

Coordination processing in real-life technical texts is extremely difficult. The configurations found, even in the restricted case of NP, are extremely varied. An important percentage of the differences found between the results of both softwares are due to a different way of processing coordination. In most cases, AlethIP just produces a coordinate sequence, with componential description, whereas Lexter has many different parsing rules, particularly for the reconstitution of the underlying coordinated components when necessary. For instance, when AlethIP extracts the only CT *(the) organisation and the regulation of electric systems*, Lexter extracts the following two CT *organisation of electric systems* and *regulation of electric systems*, which share the same expansion *electric systems* but have different heads *(organisation* and *regulation).* The experience (Daille et al., 1996) shows that coordinate structures rarely constitute terms as such whereas one or two of the underlying components can be actual terms. Therefore, the heavy investment allowed for the development of processing rules regarding coordination proves to be worthwhile.

4.2 Endogenous corpus-based disambiguation

To sort out the ambiguous cases of preposition phrase or adjective attachment, Lexter uses endogenous corpus-based learning procedures which enable it to make a more precise processing work than AlethIP. For instance, Lexter analyses correctly the CT *programme d'investissement lisse* (H: *programme d'investissement,* E: *lisse),* as it has found in another part of the corpus analysed at least one occurrence of the phrase *programme d'investissement* in a non-ambiguous situation, and no occurrence of *investissement lisse.* The fact that Lexter resorts to these procedures explains an important part of the cases, admittedly rare (see table 2, #4), where both softwares offer different analysis for the same sequence of lemmas.

4.3 Constraints related to determiners

The comparison showed that Lexter is too strict as to the determiners which can be part of terms. It rejects noun phrases including possessive determiners *(his, her, their,* etc.), demonstratives *(this, these, those,* etc.) or indefinites *(some, several,* etc.), that AlethIP quite rightly accepts. Phrases including these determiners (such as *installation of this electric line)* will never be chosen to appear in a base list. Nevertheless, as we explained it in section 2.2, building up a terminology is not just making a base list, it also consists in describing terms. But, some CT that are not kept as entries in a terminology can yet give access to interesting text contexts useful to make the description of other CT from which they can be variations. For instance, the contexts of the CT *installation of this electric line* are certainly relevant for the description of the term *installation of an electric line.* Therefore, we had to change the rules of Lexter to loosen the constraints concerning determiners.

5 Evaluation and Future Work

Obviously, comparing only NP parses makes easier the task of syntactic alignment. However, this restriction follows from the limited aim of our study. A syntax-directed translation schema like the one we developed can be

used to align parses of whole sentences, as it is based on the grammar of the input trees. Secondly, we have not tried the whole set of possible mappings between the two types of output. The "pivot grammar" which is being used for the current task of alignment lies somewhere between a "skeleton dependency representation" and a rich constituent-structure tree. It would have been possible to try both ends of the range. So far, we have neither guidelines nor methodology for building such a pivot grammar. However, it is already rather satisfactory to be able to "upgrade" a "skeleton parse" via a bottom-up syntax-directed translation schema so as to mimic a much more detailed constituent structure.

Parsed corpora can range from skeletal marking to detailled labelling- as in Susanne (Sampson, 1994), with every kind of intermediate situation - like the Penn Treebank (Marcus et al., 1993). For the evaluation we are aiming at, we aligned "skeleton parses" with more detailed ones. This situation is rather typical of the variety of formats which are likely to be found with the rapid expansion of parsed corpora. These corpora serve as input to a growing number of softwares. These software enforce as such various formats (Black, 1994). For instance, the Nijmegen Linguistic Database (vHalteren and Oostdijk, 1993; Nederhof and Koster, 1993), which is a tool for storing and investigating large numbers of tree structures of different kinds, still needs a conversion to its own format. As the establishment of standards for pre-terminal and non-terminal categories is rather unlikely (Souter, 1993), it is necessary to devise tree transducers in order to map existing and future tag sets as well as various bracketing strategies, as suggested in (Souter and Atwell, 1994). Our approach, which relies on well-tried compiling techniques, provides an efficient and nevertheless declarative way of obtaining such mappings and of re-structuring parse trees as required. It represents as well a practical means to compare parsed corpora, and thus to help in parser evaluation.

Acknowledgements

This work has been supported by the Research and Development Division of the French Electricity Board as a contract between EDF/DER and ELI/ENS de Fontenay St Cloud. We thank Marie-Luce Herviou-Picard (EDF/DER) and Richard Quatrain (EDF/DER) for the discussions we had during this contract. Bernard Lang (INRIA) gave us access to (Gecseg and Steinby, 1984). Christian Jacquemin (LIMSI) explained us the tree rewriting techniques available in Prolog III. Elie Naulleau (EDF/DER) gave us a hand to transfer our FRT on different computers. Patrick Paroubek (LIMSI) pointed out the more general relevance of the technique we used.

References

Aho, Alfred V. & Ullman, Jeffrey D. (1972). *The Theory of Parsing, Translation and Compiling.* Englewood Cliffs, New-Jersey:Prentice Hall, Inc., volume 1 : Parsing.

Aho, Alfred V. & Ullman, Jeffrey D. (1972). *The Theory of Parsing, Translation and Compiling.* Englewood Cliffs, New-Jersey:Prentice Hall, Inc., volume 2 : Compiling.

Black, Ezra (1994). An experiment in customizing the Lancaster Treebank. In Nelleke Oostdijk & Pieter de Haan (Eds.), *Corpus-based research into language* (pp. 159-168). Amsterdam: Rodopi.

Bourigault, Didier (1993). An Endogenous Corpus-Based Method for Structural Noun Phrase Disambiguation. In *Actes, 6th Conference of the European Chapter of the Association for Computational Linguistics (EACL'93)* (pp. 81-86). Utrecht.

Daille, Beatrice, Habert, Benoît, Jacquemin, Christian & Royauté, Jean (1996). Empirical Observation of Term Variations and Principles for their Description. *Terminology*, vol. 3 (2), 197-257.

Gécseg, Ferenc & Steinby, Magnus (1984). *Tree Automata*. Budapest: Akadémia Kiado.

Habert, Benoît, Herviou-Picard, Marie-Luce, Bourigault, Didier, Quatrain, Richard & Roumens, Marielle (avril 1997). Un outil et une méthode pour comparer deux extracteurs de groupes nominaux. In *1ères Journées Scientifiques et Techniques FRANC1L.* Avignon.

Herviou-Piccard, Marie-Luce (1996. *Les outils d 'indexation AlethIP issus du projet GRAAL: principes et utilisation.* Technical Report HN-46/96/022, Clamart, EDF, Direction des Études et Recherches.

Marcus, Mitchell, Santorini, Beatrice & Marcinkiewicz, Mary Ann (June 1993). Building a Large Annotated Corpus of English : The Penn Treebank. *Computational Linguistics*, vol. 19 (2), 313-330.

Nederhof, Mark-Jan & Koster, Kees (1993). A customized grammar workbench. In Jan Aarts, Pieter de Haan & Nelleke Oostdijk (Eds.), *English language corpora : design, analysis and exploitation* (pp. 163-180). Amsterdam: Rodopi.

Sabbagh, Simon & team, GRAAL (1994). GRAAL Eureka project: re-usable grammars for automatic language analysis. In *Proceedings of the Language Engineering Convention.* Paris.

Sampson, Geoffrey (1994). SUSANNE : a Domesday Book of English grammar. In Nelleke Oostdijk & Pieter de Haan (Eds.), *Corpus Based Research into Language (pp.* 169-187). Amsterdam: Rodopi.

Souter, Clive & Atwell, Eric (1994). Using parsed corpora : a review of current practice. In Nelleke Oostdijk & Pieter de Haan (Eds.), *Corpus-based research into language (pp.* 143-158). Amsterdam: Rodopi.

Souter, Clive (1993). Towards a standard format for parsed corpora. In Jan Aarts, Pieter de Haan & Nelleke Oostdijk (Eds.), *English language corpora : design, analysis and exploitation* (pp. 197-212). Amsterdam: Rodopi.

van Halteren, Hans & Oostdijk, Nelleke (1993). Towards a syntactic database : the TOSCA analysis system. In Jan Aarts, Pieter de Haan & Nelleke Oostdijk (Eds.), *English language corpora: design, analysis and exploitation* (pp. 145-162). Amsterdam: Rodopi.

Wilhelm, Reinhard & Maurer, Dieter (1995). *Compiler Design:Addison-Wesley.*