Full Text Parsing using Cascades of Rules: an Information Extraction Perspective

Fabio Ciravegna and Alberto Lavelli
ITC-irst Centro per la Ricerca Scientifica e Tecnologica
via Sommarive, 18
38050 Povo (TN)
ITALY
{cirave|lavelli}@irst.itc.it

Abstract

This paper proposes an approach to full parsing suitable for Information Extraction from texts. Sequences of cascades of rules deterministically analyze the text, building unambiguous structures. Initially basic chunks are analyzed; then argumental relations are recognized; finally modifier attachment is performed and the global parse tree is built. The approach was proven to work for three languages and different domains. It was implemented in the IE module of FACILE, a EU project for multilingual text classification and IE.

1 Introduction

Most successful approaches in IE (Appelt et al., 1993; Grishman, 1995; Aone et al., 1998) make a very poor use of syntactic information. They are generally based on shallow parsing for the analysis of (non recursive) NPs and Verbal Groups (VGs). After such step regular patterns are applied in order to trigger primitive actions that fill template(s); meta-rules are applied to patterns to cope with different syntactic clausal forms (e.g., passive forms). If we consider the most complex MUC-7 task (i.e., the Scenario Template task (MUC7, 1998)), the current technology is not able to provide results near an operational level (expected F(1)=75%; the best system scored about 50% (Aone et al., 1998)). One of the limitations of the current technology is the inability to extract (and to represent) syntactic relations among elements in the sentence, i.e. grammatical functions and thematic roles. Scenario Template recognition needs the correct treatment of syntactic relations at both sentence and text level (Aone et al., 1998). Full parsing systems are generally able to correctly model syntactic relations, but they tend to be slow (because of huge search spaces) and brittle (because of gaps in the grammar). The use of big grammars partially solves the problem of gaps but worsens the problem of huge search spaces and makes grammar modifications difficult (Grishman, 1995). Grammar modifications are always to be taken into account. Many domain-specific texts present idiosyncratic phenomena that require non-standard rules. Often such phenomena are limited to some cases only (e.g., some types of coordinations are applied to people only and not to organizations). Inserting generic rules for such structures introduces (useless) extra complexity into the search space and - when applied indiscriminately (e.g., on classes other than people) - can worsen the system results. It is not clear how semantic restrictions can be introduced into (big) generic grammars.

In this paper we propose an approach to full parsing for IE based on cascades of rules. The approach is inspired by the use of finite-state cascades for parsing (e.g., (Abney, 1996) uses them in a project for inducing lexical dependencies from corpora). Our work is interesting in an IE perspective because it proposes:

- a method for efficiently and effectively performing full parsing on texts;
- a way of organizing generic grammars that simplifies changes, insertion of new rules and especially integration of domain-oriented rules.

The approach proposed in this paper for parsing has been extended to the whole architecture of an IE system. Also lexical (lexical normalization and preparsing), semantic (default reasoning and template filling) and discourse modules are based on the same approach. The system has been developed as part of FACILE (Ciravegna et al., 1999), a successfully completed project funded by the European Union. FACILE deals with text classification and information extraction from text in

the financial domain. The proposed approach has been tested mainly for Italian, but proved to work also for English and, as of the time of this writing, partially for Russian. Applications and demonstrators have been built in four different domains.

In this paper we first introduce the adopted formalism and then go into details on grammar organization and on the different steps through which parsing is accomplished. Finally we present some experimental results.

2 Representation and Rules

Every lexical element a in the input sentence w is abstractly represented by means of elementary objects, called **tokens**. A token T is associated with three structures:

- $[T]_{dep}$ is a dependency tree for a, i.e. a tree representing syntactic dependencies between a and other lexical elements (its dependees) in w.
- $[T]_{feat}$ is a feature structure representing syntactic and semantic information needed to combine a with other elements in the input.
- $[T]_{lf}$ is a Quasi Logical Form (QLF) providing a semantic interpretation for the combination of a with its dependees.

Rules operate on tokens, therefore they can access all the three structures above. Rules incrementally build and update the above structures. Lexical, syntactic and semantic constraints can then be used in rules at any level. The whole IE approach can be based on the same formalism and rule types, as both lexical, syntactic and semantic information can be processed uniformly.

The general form of a **rule** is a triple

$$\langle \gamma\alpha\delta, \Gamma_T, \Gamma_A \rangle,$$

where

- $\gamma\alpha\delta$ is a non-empty string of tokens, called the rule **pattern**; α is called the rule **core** and is non-empty, γ, δ are called the rule **context** and may be empty;
- Γ_T is a set of boolean predicates, called rule **test**, defined over tokens in the rule pattern;
- Γ_A is a set of elementary operations, called rule **action**, defined over tokens in the sole rule core.

The postfix, unary operators "*" (Kleene star) and "?" (optionality operator) can be used in the rule patterns.

A basic data structure, called **token chart**, is processed and dynamically maintained. This is a directed graph whose vertices are tokens and whose arcs represent binary relations from some (finite) basic set. Initially, the token chart is a chain-like graph with tokens ordered as the corresponding lexical elements in w, i.e. arcs initially represent lexical adjacency between tokens. During the processing, arcs might be rewritten so that the token chart becomes a more general kind of graph.

For a rule to apply, a path σ must be found in the token chart that, when viewed as a string of tokens, satisfies the two following conditions: (i) σ is matched by $\gamma\alpha\delta$; and (ii) all the boolean predicates in Γ_T hold when evaluated on σ.

When a rule applies, the elementary operations in Γ_A are executed on the tokens of σ matching the core of the rule. The effect of action execution is that $[T]_{dep}$, $[T]_{feat}$ and $[T]_{lf}$ are updated for the appropriate matching tokens.

Rules are grouped into **cascades** that are finite, ordered sequences of rules. Cascades represent elementary logical units, in the sense that all rules in a cascade deal with some specific construction (e.g., subcategorization of verbs). From a functional point of view, a cascade is composed of three segments:

- **s1** contains rules that deal with idiosyncratic cases for the construction at hand;
- **s2** contains rules dealing with the regular cases;
- **s3** contains default rules that fire only when no other rule can be successfully applied.

3 Parsing

The parsing model is strongly influenced by IE needs. Its aim is to build the *sufficient IE approximation* (SIEA) of the correct parse tree for each sentence, i.e. a complete parse tree where all the relations relevant for template filling are represented, while other relations are left implicit. The parser assumes that there is one and only one possible correct parse tree for each sentence and therefore also only one SIEA.

Parsing is based on the application of a fixed sequence of cascades of rules. It is performed in three steps using different grammars:

- chunking (analysis of NPs, VGs and PPs);
- subcategorization frame analysis (for verbs, nominalizations, etc.);
- modifier attachment.

[ACME]$_{np}$	[ha deciso]$_{vg}$,	[informa]$_{vg}$	[una nota]$_{np}$,	[di]$_{compl}$
ACME	*has decided,*	*tells*	*a press release,*	*to*
[iniziare]$_{vg}$	[l'emissione]$_{np}$	[di obbligazioni]$_{pp}$	[per 12 milioni di Euro]$_{pp}$	
start	*the issue*	*of bonds*	*for 12 million (of) Euro*	
[in modo da]$_{compl}$	[diversificare]$_{vg}$	[il proprio impegno]$_{np}$	[nel mercato]$_{pp}$.	
so to	*diversify*	*its obligation*	*in the market.*	

Figure 1: The Italian sentence used as an example.

The first two steps are syntax driven and based on generic grammars. Most rules in such grammars are general syntactic rules, even if they strongly rely on the semantic information provided by a foreground lexicon (see also Section 3.2). During modifier attachment, mainly semantic patterns are used. At the end of these three steps the SIEA is available.

We use deterministic dependency parsing[1] operating on a specific linear path within the token chart (*the parsing path*); at the beginning the parsing path is equal to the initial token chart. When a rule is successfully applied, the parsing path is modified so that only the head token is visible for the application of the following rules; this means that the other involved elements are no longer available to the parser.

3.1 Chunking

Chunking is accomplished in a standard way. In Figure 1 an example of chunk recognition is shown.[2]

3.2 A-structure Analysis

A-structure analysis is concerned with the recognition of argumental dependencies between chunks. All kinds of modifier dependencies (e.g., PP attachment) are disregarded during this step.

More precisely: let w be the input sentence. A **dependency tree** for w is a tree representing all predicate-argument and predicate-modifier syntactic relations between the lexical elements in w. The **A-structure** for w is a tree forest obtained from the dependency tree by unattaching all nodes that represent modifiers. A-structures are associated with the token that represents the semantic head of w (T_{sent} in the following). A-structure analysis associates to T_{sent}:

- $[T_{sent}]_{dep}$: the A-structure spanning the whole sentence;
- $[T_{sent}]_{feat}$: its feature structure;
- $[T_{sent}]_{lf}$: its QLF.

A-structure analysis is performed without recursion by the successive application of three sequences of rule cascades:[3]

- The first sequence of cascades performs analysis of basic (i.e., non-recursive) sentences. It does so using the subcategorization frames of available chunks. Three cascades of rules are involved: one for the subcategorization frames of NP and PPs, one for those of VGs, and one for combining complementizers with their clausal arguments.
- The second sequence of cascades performs analysis of dependencies between basic sentences. This sequence processes all sentential arguments and all incidentals by employing only two cascades of rules, without any need for recursion. This sequence is applied twice, i.e. it recognizes structures with a maximum of two nested sentences.
- The third sequence of cascades performs recovery analysis. During this step all tree fragments not yet connected together are merged.

Tokens not recognized as arguments at the end of A-structure analysis are marked as modifiers and left unattached in the resulting A-structure. They will be attached in the parse tree during modifier attachment (see Section 3.3).

We adopt a highly lexicalized approach. In a pure IE perspective the information for A-structure analysis is provided by a foreground lexicon (Kilgarriff, 1997). Foreground lexica provide detailed information about words relevant for the domain (e.g., subcategorization frame, relation with the ontology); for words outside the foreground lexicon a large background lexicon provides generic information. The term subcategorization frame is used here in restricted sense: it

[1] Even if we use dependency parsing, in this paper we will make reference to constituency based structures (e.g., PPs) because most readers are more acquainted with them than with dependency structures.

[2] In this paper we use literal English translations of Italian examples.

[3] The order of the cascades in the sequences depends on the intrasentential structure of the specific language coped with.

PATTERN	TEST	ACTION	Matched Input
T_1	$[T_1]_{feat}$.cat=NP	Dependant($[T_1]_{dep}$,$[T_3]_{dep}$) $[T_1]_{feat}$.subcat.int-arg=$[T_3]_{feat}$ $[T_1]_{lf}$.patient=$[T_3]_{lf}$.head	**"the issue"**
T_2*	$[T_2]_{feat}$.cat=Adjunct		
T_3	$[T_3]_{feat}$.cat=PP $[T_3]_{feat}$=$[T_1]_{feat}$.subcat.int-arg		**"of bonds"**

Figure 2: The rule that recognizes **[the issue of bonds]**$_{np}$.

generally includes subject, object and - possibly - one indirect complement. The information in the foreground lexicon allows to classify known tokens as arguments of other known tokens with high reliability. Let's go back to our example. The first sequence of cascades recognizes:

- **[the issue of bonds]**$_{np}$: the rule in Figure 2 recognizes **[of bonds]**$_{pp}$ as internal argument of **[the issue]**$_{np}$; the rule uses the information associated with **issue** in the foreground lexicon, i.e. that it is the nominalization of "to issue". The subcategorization frame of such verb specifies its arguments and their semantic restrictions. The syntactic rule adds the condition that - being a nominalization - the internal argument is realized as a PP marked by **of**;
- **[ACME has decided]**$_{ip}$: **[ACME]**$_{np}$ is the external argument of **[has decided]**$_{vg}$;
- **[tells a press release]**$_{ip}$: **[a press release]**$_{np}$ is the external argument of **[tells]**$_{vg}$;
- **[start the issue of bonds for 12 million Euro]**$_{ip}$: **[issue]**$_{np}$ is the internal argument of **[start]**$_{vg}$; **[for 12 million Euro]**$_{pp}$ is a modifier (as it is not subcategorized by other tokens) and is unattached in the A-structure;
- **[diversify its obligation in the market]**$_{ip}$: **[its obligation]**$_{np}$ is the internal argument of **[diversify]**$_{vg}$; **[in the market]**$_{pp}$ is a modifier;
- **[to start the issue of bonds for 12 million Euro]**$_{cp}$: **[to]**$_{compl}$ gets its argument **[start ...]**$_{ip}$;
- **[so to diversify its obligation in the market]**$_{cp}$: **[so to]**$_{compl}$ gets its argument **[diversify ...]**$_{ip}$.

The result after the application of the first sequence is:
[ACME has decided]$_{ip}$
[tells a press release]$_{ip}$
[to start the issue of bonds for 12 million Euro]$_{cp}$
[so to diversify its obligation in the market]$_{cp}$

The second sequence recognizes that **[ACME has decided]**$_{ip}$ still needs a sentential argument, i.e. a subordinate clause introduced by **to** (such information comes from the foreground lexicon). Such argument is found after the incidental **[tells a press release]**: it is the **CP** headed by **[to start]**. The result of the second phase is:
[ACME has decided, tells a press release, to start the issue of bonds for 12 million Euro]$_{sentence}$
[so to diversify its obligation in the market]$_{cp}$

Finally, the recovery sequence collapses the two constituents above into a single sentence structure; the **CP** is considered a clausal modifier (as it was not subcategorized by anything) and is left unattached in the A-structure.

At the end of the A-structure recognition T_{sent} is the token associated with **[has decided]**. $[T_{sent}]_{dep}$ is integrated with the search space for each unattached modifier (see Section 3.3).

The way A-structures are produced is interesting for a number of reasons.

First of all generic grammars are used to cope with generic linguistic phenomena at sentence level. Secondly we represent syntactic relations in the sentence (i.e., grammatical functions and thematic roles); such relations allow a better treatment of linguistic phenomena than possible in shallow approaches (Aone et al., 1998;

Kameyama, 1997).

The initial generic grammar is designed to cover the most frequent phenomena in a restrictive sense. Additional rules can be added to the grammar (when necessary) for coping with the uncovered phenomena, especially domain-specific idiosyncratic forms. The limited size of the grammar makes modifications simple (the A-structure grammar for Italian contains 66 rules).

The deterministic approach combined with the use of sequences, cascades and segments makes grammar modifications simple, as changes in a cascade (e.g., rule addition/modification) influence only the following part of the cascade or the following cascades. This makes the writing and debugging of grammars easier than in recursive approaches (e.g., context-free grammars), where changes to a rule can influence the application of any rule in the grammar.

The grammar organization in cascades and segments allows a clean definition of the grammar parts. Each cascade copes with a specific phenomenon (modularity of the grammar). All the rules for the specific phenomenon are grouped together and are easy to check.

The segment/cascade structure is suitable for coping with the idiosyncratic phenomena of restricted corpora. As a matter of fact domain-oriented corpora can differ from the standard use of language (such as those found in generic corpora) in two ways:

- in the frequency of the constructions for a specific phenomenon;
- in presenting different (idiosyncratic) constructions.

Coping with different frequency distributions is conceptually easy by using deterministic parsing and cascades of rules, as it is just necessary to change the rule order within the cascade coping with the specific phenomenon, so that more frequently applied rules are first in the cascade. Coping with idiosyncratic constructions requires the addition of new rules. Adding new rules in highly modularized small grammars is not complex.

Finally from the point of view of grammar organization, defining segments is more than just having ordered cascades. Generic rules (in s2) are separated from domain specific ones (in s1); rules covering standard situations (in s2) are separated from recovery rules (in s3). In s2, rules are generic and deal with unmarked cases. In principle s2 and s3 are units portable across the applications without changes. Domain-dependent rules are grouped together in s1 and are the resources the application developer works on for adapting the grammar to the specific corpus needs (e.g., coping with idiosyncratic cases). Such rules generally use contexts and/or introduce domain-dependent (semantic) constraints in order to limit their application to well defined cases. S1 rules are applied before the standard rules and then idiosyncratic constructions have precedence with respect to standard forms.

Segments also help in parsing robustly. S3 deals with unexpected situations, i.e. cases that could prevent the parser from continuing. For example the presence of unknown words is coped with after chunking by a cascade trying to guess the word's lexical class. If every strategy fails, a recovery rule includes the unknown word in the immediately preceding chunk so to let the parser continue. Recovery rules are applied only when rules in s1 and s2 do not fire.

3.3 Modifier attachment

The aim of modifier attachment is to find the correct position for attaching relevant modifiers in the parse tree and to add the proper semantic relations between each modifier and its modifiee in $[T_{sent}]_{lf}$. $[T_{sent}]_{dep}$ and $[T_{sent}]_{lf}$ are used to determine the correct attachments and are also modified during this step. Modifier attachment is performed in two steps: first all the possible attachments are computed for each modifier (its **search space**, SP). Here mainly generic syntactic rules are used. Then the correct attachment in the search space is determined for each modifier, applying domain-specific rules. The rules always modify both $[T_{sent}]_{dep}$ and $[T_{sent}]_{lf}$. Only modifiers relevant for the IE task are attached in the proper position. Other modifiers are attached in a default position in $[T_{sent}]_{dep}$.

Initially modifiers are attached in the lowest position in $[T_{sent}]_{dep}$. No semantic relation is hypothesized between each modifier and the rest of the sentence in $[T_{sent}]_{lf}$. Given:

- T_n: a modifier token derived from chunk n,
- T_{n-1}: the token, derived from chunk $n-1$, immediately preceding n in the sentence,

in right branching languages (such as Italian) the lowest possible attachment for T_n is in the position of modifier of T_{n-1}.

Afterwards, the possible SP for each modifier is computed. The SP for a modifier T_n is a path in the token chart connecting T_n with other elements in $[T_{sent}]_{dep}$ that - from a syntactic point of view - can be modified by T_n. The initial SP for T_n is given by the path in $[T_{sent}]_{dep}$ connecting T_{n-1}

PATTERN	TEST	ACTION	Matched Input
T_1	$[T_1]_{lf}$.head=TO-INCREASE	Dependant($[T_1]_{dep}$,$[T_5]_{dep}$) $[T_1]_{lf}$.increased-by=$[T_5]_{lf}$.head	**"an increase"**
T_2*	$[T_2]_{feat}$.cat=Adjunct		
T_3	$[T_3]_{lf}$.head=PROFIT $[T_3]_{feat}$.cat=PP $[T_3]_{feat}$.marked=''of''		**"of profits"**
T_4*	$[T_4]_{feat}$.cat=Adjunct		
T_5	$[T_5]_{lf}$.head=PERCENTAGE $[T_5]_{feat}$.cat=PP $[T_5]_{feat}$.marked=''of/by''		**"of/by 20%"**

Figure 3: An example of modifier attachment rule.

with T_{sent}. Then rules are applied to filter out elements in SPs according to syntactic constraints (e.g., NPs or PPs can be modified by a relative clause, but VGs can not).

After SPs have been computed, modifiers are attached using a sequence of cascades of rules.

A first cascade, mainly composed by generic syntactic rules, attaches subordinates (e.g., relative clauses). Many of these rules are somehow similar to A-structure recognition rules. They are truly syntactic rules recognizing part of the subcategorization frame of subordinated verbs, using semantic information provided by the foreground lexicon. Note however that they are applied onto SPs not on the parsing path (as A-structure rules are).

Other cascades are used to attach different types of modifiers, such as PPs. Such rules mainly involve semantic constraints. For example, the rule shown in Figure 3 can recognize `una crescita dei profitti del 20%` (lit. an increase of profits of/by 20%).[4]

[4]Generally rules involve two elements (i.e. the modifier and the modifiee), taking into account intervening elements (such as other adjuncts) that do not have further associated conditions. The example above, instead, is more complex as it introduces constraints also on one of the intervening adjuncts (i.e., on T_3). Such domain-oriented rule solves a recurring ambiguity in the domain of company financial results. As a matter of fact `of/by 20%` could modify both nouns from a syntactic and semantic point of view. The rule allows to solve the ambiguity attaching `of/by 20%` to `increase`.

Rules for modifier attachment are easy to write. The SP allows to reduce complex cases to simple ones. For example the rule in Figure 3 also applies to:

- `an increase in 1997 of profits of/by 20%`
- `an increase, news report, of profits of/by 20%`
- `an increase, considering the inflation rate, of profits (both gross and net) of/by 20%`

Patterns are usually developed having in mind the simplest case, i.e. a sequence of contiguous chunks in the sentence (such as in `[an increase] [of profits] [of/by 20%]`) that can be interleaved by other non relevant chunks.

Conceptually this step is very similar to that used by shallow parsing approaches such as in (Grishman, 1997). Note however that rules are not applied on a list of contiguous chunks, but on the search space (where the parse tree and related syntactic relations are available). Parse-tree based modifier attachment is less error prone than attachment performed on flat chunk structures (as used in shallow parsing). For example it is possible to avoid cases of attachments violating syntactic constraints, as it would be the case in attaching

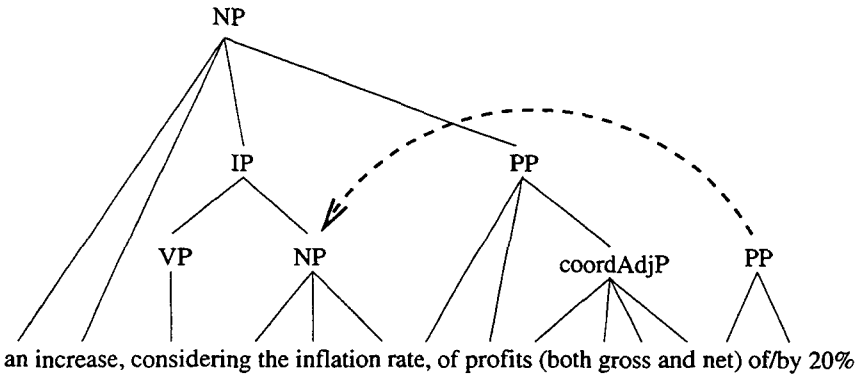

Figure 4: Violation of syntactic constraints

(in the third example above) `profit` to `increase` and 20% to `inflation rate` (see Figure 4).

At the end of modifier attachment the final parse tree is available in $[T_{sent}]_{dep}$, together with $[T_{sent}]_{feat}$ and $[T_{sent}]_{lf}$. The syntactic information in $[T_{sent}]_{dep}$ and $[T_{sent}]_{feat}$ is useful in the steps following parsing because, for example, the availability of syntactic relations increases the accuracy in determining discourse relations. As a matter of fact at discourse level it is possible to adopt strategies to compute salience for coreference resolution that take into account both the syntactic relations among constituents and the argumental structure of the sentence. Shallower approaches do not produce anything similar either to $[T_{sent}]_{dep}$ or to $[T_{sent}]_{feat}$. They generally adopt heuristics such as linear ordering and recency of basic chunks: such heuristics have been shown not as effective as those based on full syntactic relations, even if for some languages they represent an acceptable approximation (Kameyama, 1997).

The obtained final parse tree is very close to the SIEA mentioned at the beginning of Section 3. In this tree all the A-structures are correctly built and all the modifiers are attached. Modifiers relevant for the IE application are attached in the correct position in the tree and a valid semantic relation is established with the modifiee in $[T_{sent}]_{lf}$. Irrelevant modifiers are attached in a default position in the tree (the lowest possible attachment) and a null semantic relation is established with the modifiee. The only difference between the produced tree and the SIEA is in the A-structure, where all the relations are captured (and not only those relevant for the domain). Modeling also the argumental structures of irrelevant constituents can be useful in order to correctly assign salience at discourse level. For example when interesting relations involve elements that are dependees of irrelevant verbs.

4 Remarks and Conclusions

The approach to parsing proposed in this paper was implemented in the IE module of FACILE (Ciravegna et al., 1999), a EU project for multilingual text classification and IE. It was tested on four domains and in three languages. In particular for Italian one application (about bond issues) has been fully developed and two others have reached the level of demonstration (management succession and company financial results). For English a demonstrator for the field of economic indicators was developed. A Russian demonstrator for bond issues was developed till the level of modifier attachment. The approach to rule writing and organization adopted for parsing (i.e., the type of rules, cascades, segments, and available primitives and rule interpreter) was extended to the whole architecture of the IE module. Also lexical (lexical normalization and preparsing), semantic (default reasoning and template filling) and discourse levels are organized in the same way.

Provided that the approach to parsing proposed in this paper is strongly influenced by IE needs, it is difficult to evaluate it by means of the standard tools used in the parsing community. Approximate indications can be provided by the effectiveness in recognizing A-structures and by the measures on the overall IE tasks.

Effectiveness in recognizing A-structures was experimentally verified for Italian on a corpus of 95 texts in the domain of bond issue: 33 texts were used for training, 62 for test. Results were: P=97, R=83 on the training corpus, P=95, R=71 on the test corpus. In our opinion the high precision demonstrates the applicability of the method. The lower recall shows difficulties of building complete foreground lexica, a well known fact in IE.

Concerning the effectiveness of the IE process, in the Italian application on bond issues the system reached P=80, R=72, F(1)=76 on the 95

texts used for development (33 ANSA agency news, 20 "Il Sole 24 ore" newspaper articles, 42 Radiocor agency news; 10,472 words in all). Table 4 shows the kind of template used for this application. Effectiveness was automatically calculated by comparing the system results against a user-defined tagged corpus via the MUC scorer (Douthat, 1998). The development cycle of the template application was organised as follows: resources (grammars, lexicon and knowledge base) were developed by carefully inspecting the first 33 texts of the corpus. Then the system was compared against the whole corpus (95 texts) with the following results: Recall=51, Precision=74, F(1)=60. Note that the corpus used for training was composed only by ANSA news, while the test corpus included 20 "Il Sole 24 ore" newspaper articles and 42 Radiocor agency news (i.e., texts quite different from ANSA's in both terminology and length). Finally resources were tuned on the whole corpus mainly by focusing on the texts that did not reach sufficient results in terms of R&P. The system analyzed 1,125 word/minute on a Sparc Ultra 5, 128M RAM (whole IE process).

issuer	a template element
kind of bond	a label
amount	a monetary amount
currency	a string from the text
announcement date	a temporal expression
placement date	a temporal expression
interest date	a temporal expression
maturity	a temporal expression
average duration	a temporal expression
global rate	a string from the text
first rate	a string from the text

Table 1: The template to be filled for bond issues.

Acknowledgments

Giorgio Satta has contributed to the whole work on parsing for IE via cascades of rules. The authors would like to thank him for the constant help and the fruitful discussions in the last two years. He also provided useful comments to this paper. The FACILE project (LE 2440) was partially funded by the European Union in the framework of the Language Engineering Sector. The English demonstrator was developed by Bill Black, Fabio Rinaldi and David Mowatt (Umist, Manchester) as part of the FACILE project. The Russian demonstrator was developed by Nikolai Grigoriev (Russian Academy of Sciences, Moscow).

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLI '96 Robust Parsing Workshop.*

Chinatsu Aone, Lauren Halverson, Tom Hampton, and Mila Ramos-Santacruz. 1998. SRA: description of the IE^2 system used for MUC-7. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, http://www.muc.saic.com/.

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, and Mabry Tyson. 1993. FASTUS: A finite-state processor for information extraction from real-world text. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, Chambery, France.

Fabio Ciravegna, Alberto Lavelli, Nadia Mana, Luca Gilardoni, Silvia Mazza, Massimo Ferraro, Johannes Matiasek, William J. Black, Fabio Rinaldi, and David Mowatt. 1999. FACILE: Classifying texts integrating pattern matching and information extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, Stockholm, Sweden.

Aaron Douthat. 1998. The message understanding conference scoring software user's manual. In *Proceedings of the Seventh Message Understanding Conference (MUC-7)*, http://www.muc.saic.com/.

Ralph Grishman. 1995. The NYU system for MUC-6 or where's syntax? In *Sixth message understanding conference MUC-6.* Morgan Kaufmann Publishers.

Ralph Grishman. 1997. Information extraction: Techniques and challenges. In M. T. Pazienza, editor, *Information Extraction: a multidisciplinary approach to an emerging technology.* Springer Verlag.

Megumi Kameyama. 1997. Recognizing referential links: An information extraction perspective. In Mitkov and Boguraev, editors, *Proceedings of ACL/EACL Workshop on Operational Factors in Practical, Robust Anaphora Resolution for Unrestricted Texts*, Madrid, Spain.

Adam Kilgarriff. 1997. Foreground and background lexicons and word sense disambiguation for information extraction. In *International Workshop on Lexically Driven Information Extraction*, Frascati, Italy.

MUC7. 1998. *Proceedings of the Seventh Message Understanding Conference (MUC-7).* SAIC, http://www.muc.saic.com/.