INCREMENTAL FINITE-STATE PARSING

Salah Aït-Mokhtar, Jean-Pierre Chanod
Rank Xerox Research Centre
6, Chemin de Maupertuis
F-38240 Meylan, France
Ait@grenoble.rxrc.xerox.com
Chanod@grenoble.rxrc.xerox.com

Abstract

This paper describes a new finite-state shallow parser. It merges constructive and reductionist approaches within a highly modular architecture. Syntactic information is added at the sentence level in an incremental way, depending on the contextual information available at a given stage. This approach overcomes the inefficiency of previous fully reductionist constraint-based systems, while maintaining broad coverage and linguistic granularity. The implementation relies on a sequence of networks built with the replace operator. Given the high level of modularity, the core grammar is easily augmented with corpus-specific sub-grammars. The current system is implemented for French and is being expanded to new languages.

1 Background

Previous work in finite-state parsing at sentence level falls into two categories: the constructive approach or the reductionist approach.

The origins of the constructive approach go back to the parser developed by Joshi (Joshi, 1996). It is based on a lexical description of large collections of syntactic patterns (up to several hundred thousand rules) using subcategorisation frames (verbs + essential arguments) and local grammars (Roche, 1993). It is, however, still unclear whether this heavily lexicalized method can account for all sentence structures actually found in corpora, especially due to the proliferation of non-argumental complements in corpus analysis.

Another constructive line of research concentrates on identifying basic phrases such as in the FASTUS information extraction system (Appelt et al., 1993) or in the chunking approach proposed in (Abney, 1991; Federici et al., 1996). Attempts were made to mark the segments with additional syntactic information (e.g. subject or object) (Grefenstette, 1996) using simple heuristics, for the purpose of information retrieval, but not for robust parsing.

The reductionist approach starts from a large number of alternative analyses that get reduced through the application of constraints. The constraints may be expressed by a set of elimination rules applied in a sequence (Voutilainen, Tapanainen, 1993) or by a set of restrictions applied in parallel (Koskenniemi et al., 1992). In a finite-state constraint grammar (Chanod, Tapanainen, 1996), the initial sentence network represents all the combinations of the lexical readings associated with each token. The acceptable readings result from the intersection of the initial sentence network with the constraint networks. This approach led to very broad coverage analyzers, with good linguistic granularity (the information is richer than in typical chunking systems). However, the size of the intermediate networks resulting from the intersection of the initial sentence network with the sets of constraints raises serious efficiency issues.

The new approach proposed in this paper aims at merging the constructive and the reductionist approaches, so as to maintain the coverage and granularity of the constraint-based approach at a much lower computational cost. In particular, segments (chunks) are defined by constraints rather than patterns, in order to ensure broader coverage. At the same time, segments are defined in a cautious way, to ensure that clause boundaries and syntactic functions (e.g. subject, object, PP-Obj) can be defined with a high degree of accuracy.

2 The incremental parser

2.1 Overview

The input to the parser is a tagged text. We currently use a modified version of the Xerox French tagger (Chanod, Tapanainen, 1995). The revisions are meant to reduce the impact of the most frequent errors of the tagger (e.g. errors between adjectives and past participles), and to refine the tagset.

Each input token is assigned a single tag, generally representing the part-of-speech and some limited morphological information (e.g the number, but not the gender of nouns). The sentence is initially represented by a sequence of wordform-plus-tag pairs.

The incremental parser consists of a sequence of transducers. These transducers are compiled from regular expressions that use finite-state calculus operators, mainly the Replace operators (Karttunen, 1996). Each of these transducers adds syntactic information represented by reserved symbols (annotations), such as brackets and names for segments and syntactic functions. The application of each transducer composes it with the result of previous applications.

If the constraints stipulated in a given transducer are not verified, the string remains unchanged. This ensures that there is always an output string at the end of the sequence, with possibly underspecified segments.

Each transducer performs a specific linguistic task. For instance, some networks identify segments for NPs, PPs, APs (adjective phrases) and verbs, while others are dedicated to subject or object. The same task (e.g. subject assignment or verb segmentation) may be performed by more than one transducer. The additional information provided at each stage of the sequence is instrumental in the definition of the later stages of the sequence. Networks are ordered in such a way that the easiest tasks are addressed first.

2.2 Non-monotonicity

The replace operators allow one not only to add information but also to modify previously computed information. It is thus possible to reassign syntactic markings at a later stage of the sequence. This has two major usages:

- assigning some segments with a default marking at some stage of the process in order to provide preliminary information that is essential to the subsequent stages; and correcting the default marking later if the context so requires
- assigning some segments with very general marking; and refining the marking later if the context so permits.

In that sense, our incremental parser is non-monotonic: earlier decisions may be refined or even revised. However, all the transducers can, in principle, be composed into a single transducer which produces the final outcome in a single step.

2.3 Cautious segmentation and syntactic marking

Each transducer defines syntactic constructions using two major operations: segmentation and syntactic marking. Segmentation consists of bracketing and labeling adjacent constituents that belong to a same partial construction (e.g. a nominal or a verbal phrase, or a more primitive/partial syntactic chain if necessary). Segmentation also includes the identification of clause boundaries. Syntactic marking annotates segments with syntactic functions (e.g. subject, object, PPObj).

The two operations, segmentation and syntactic marking, are performed throughout the sequence in an interrelated fashion. Some segmentations depend on previous syntactic marking and vice versa.

If a construction is not recognized at some point of the sequence because the constraints are too strong, it can still be recognized at a later stage, using other linguistic statements and different background information. This notion of delayed assignment is crucial for robust parsing, and requires that each statement in the sequence be linguistically cautious. Cautious segmentation prevents us from grouping syntactically independent segments.

This is why we avoid the use of simplifying approximations that would block the possibility of performing delayed assignment. For example, unlike (Abney, 1991), we do not systematically use longest pattern matching for segmentation. Segments are restricted by their underlying linguistic indeterminacy (e.g. post-nominal adjectives are not attached to the immediate noun on their left, and coordinated segments are not systematically merged, until strong evidence is established for their linkage).

2.4 Incremental parsing and linguistic description

The parsing process is incremental in the sense that the linguistic description attached to a given transducer in the sequence:

- relies on the preceding sequence of transducers
- covers only some occurrences of a given linguistic phenomenon
- can be revised at a later stage.

This has a strong impact on the linguistic character of the work. The ordering of the linguistic descriptions is in itself a matter of linguistic description: i.e. the grammarian must split the description of phenomena into sub-descriptions, depending on the available amount of linguistic knowledge at a given stage of the sequence.

This may sound like a severe disadvantage of the approach, as deciding on the order of the transducers relies mostly on the grammarian's intuition. But we argue that this incremental view of parsing is instrumental in achieving robust parsing in a principled fashion. When it comes to parsing, no statement is fully accurate (one may for instance find examples where even the subject and the verb do not agree in perfectly correct French sentences). However, one may construct statements which are true almost everywhere, that is, which are always true in some frequently occuring context.

By identifying the classes of such statements, we reduce the overall syntactic ambiguity and we simplify the task of handling less frequent phenomena. The less frequent phenomena apply only to segments that are not covered by previous linguistic description stages.

To some extent, this is reminiscent of the optimality theory, in which:

- Constraints are ranked;
- Constraints can be violated.

Transducers at the top of the sequence are ranked higher, in the sense that they apply first, thus blocking the application of similar constructions at a later stage in the sequence.

If the constraints attached to a given transducer are not fulfilled, the transducer has no effect. The output annotated string is identical to the input string and the construction is bypassed. However, a bypassed construction may be reconsidered at a later stage, using different linguistic statements. In that sense, bypassing allows for the violation of constraints.

2.5 An example of incremental description: French Subjects

As French is typically SVO, the first transducer in the sequence to mark subjects checks for NPs on the left side of finite verbs.

Later in the sequence, other transducers allow for subject inversion (thus violating the constraint on subject-verb order), especially in some specific contexts where inversion is likely to occur, e.g. within relative or subordinate clauses, or with motion verbs. Whenever a transducer defines a verb-subject construction, it is implicitly known at this stage that the initial subject-verb construction was not recognized for that particular clause (otherwise, the application of the verb-subject construction would be blocked).

Further down in the sequence, transducers may allow for verb-subject constructions outside the previously considered contexts. If none of these subject-pickup constructions applies, the final sentence string remains underspecified: the output does not specify where the subject stands.

It should be observed that in real texts, not only may one find subjects that do not agree with the verb (and even in correct sentences), but one may also find finite verbs without a subject. This is the case for instance in elliptic technical reports (esp. failure reports) or on cigarette packs with inscriptions like *Nuit gravement à la santé*[1].

This is a major feature of shallow and robust parsers (Jensen et al., 1993; Ejerhed, 1993): they may provide partial and underspecified parses when full analyses cannot be performed; the issue of grammaticality is independent from the parsing process; the parser identifies the most likely interpretations for any given input.

An additional feature of the incremental parser derives from its modular architecture: one may handle underspecified elements in a tractable fashion, by adding optional transducers to the sequence. For instance, one may use corpus specific transducers (e.g. sub-grammars for technical manuals are specially useful to block analyses that are linguistically acceptable, but unlikely in technical manuals: a good example in French is to forbid second person singular imperatives in technical manuals as they are often ambiguous with nouns in a syntactically undecidable fashion). One may also use heuristics which go beyond the cautious statements of the core grammar (to get back to the example of French subjects, heuristics can identify any underspecified NP as the subject of a finite verb if the slot is available at the end of the sequence). How specific grammars and heuristics can be used is obviously application dependent.

3 Architecture

The parser has four main linguistic modules, each of them consisting of one or several sequenced transducers:

[1] *Seriously endangers your health.* This example represents an interesting case of deixis and at the same time a challenge for the POS tagger as *Nuit* is more likely to be recognized as a noun (*Night*) than as a verb (*Endangers*) in this particular context.

- Primary segmentation
- Subject tagging
- Segment expansion (Optional)
- Other syntactic functions tagging

The input text is first tagged with part-of-speech information using the Xerox tagger. The tagger uses 44 morphosyntactic tags such as NOUN-SG for singular nouns and VERB-P3SG for verb 3rd person singular.

The morphosyntactic tags are used to mark AP, NP, PP and VP segments. We then use the segmentation tags and some additional information (including typography) to mark subjects which, in turn, determine to what extent VCs (Verb Chunks) can be expanded. Finally, other syntactic functions are tagged within the segments.

Marking transducers are compiled from regular expressions of the form `A @-> T1 ... T2` that contains the left-to-right longest match replace operator `@->` . Such a transducer marks in a left-to-right fashion the maximal instances of A by adding the bracketing strings T1 and T2.

4 Primary Segmentation

A segment is a continuous sequence of words that are syntactically linked to each other or to a main word (the Head). In the primary segmentation step, we mark segment boundaries within sentences as shown below where NP stands for Noun Phrase, PP for Preposition Phrase and VC for Verb Chunk (a VC contains at least one verb and possibly some of its arguments and modifiers).

Example:

```
[VC [VC Lorsqu' [NP on NP] tourne VC] [NP le
commutateur NP] [PP de démarrage PP] [PP sur la
position PP] [AP auxiliaire AP] , [NP l'
aiguille NP] retourne alors [PP à zéro PP] VC]
./SENT
```
[2]

All the words within a segment should be linked to words in the same segment at the same level, except the head. For instance, in the NP *le commutateur* (*the switch*), *le* should be linked to *commutateur* (the head) which, in turn, should be linked to the verb *tourne*, and not to the verb *retourne* because the two words are not in the same segment. The main purpose of marking segments is therefore to constrain the particular linguistic space that determines the syntactic function of a word.

[2] *Turning the starter switch to the auxiliary position, the pointer will then return to zero.*

As one can notice from the example above, segmentation is very cautious, and structural ambiguity inherent to modifier attachment (even postnominal adjectives), verb arguments and coordination is not resolved at this stage.

In order to get more robust linguistic descriptions and networks that compile faster, segments are not defined by marking sequences that match classical regular expressions of the type [Det (Coord Det) Adj* Noun], except in simple or heavily constrained cases (APs, Infinitives, etc). Rather, we take advantage of the fact that, within a linguistic segment introduced by some grammatical words and terminated by the head, there is no attachement ambiguity and therefore these words can be safely used as segment delimiters (Bès, 1993). We first mark possible beginnings and endings of a segment and then associate each beginning tag with an ending if some internal constraints are satisfied. Hence, the main steps in segmentation are:

- Tag potential beginnings and ends of a segment
- Use these temporary tags to mark the segment
- Remove the temporary tags.

4.1 AP Segmentation

Adjective phrases are marked by a replacement transducer which inserts the **[AP** and **AP]** boundaries around any word sequence that matches the regular expression (RE):

```
[ (ADVP) ADJ ( COMMA [ (ADVP) ADJ
  COMMA ]+ ) ( COORD (ADVP) ADJ ) ]
```

ADVP stands for adverb phrase and is defined as:

```
[ ADV+ [[COORD|COMMA] ADV+]* ]
```

4.2 NP Segmentation

Unlike APs, NPs are marked in two steps where the basic idea is the following: we first insert a special mark wherever a beginning of an NP is possible, i.e, on the left of a determiner, a numeral, a pronoun, etc. The mark is called a temporary beginning of NP (TBeginNP). The same is done for all possible ends of NP (TEndNP), i.e. nouns, numerals, pronouns, etc. Then, using a replacement transducer, we insert the **[NP** and **NP]** boundaries around the longest sequence that contains at least one temporary beginning of NP followed by one temporary end of NP:

```
[TBeginNP ~$[TEndNP] TEndNP ] @->
        BeginNP ...  EndNP
```

This way, we implicitly handle complicated NPs such as *le ou les responsables* (*the-SG or the-PL person(s) in charge*), *les trois ou quatre affaires* (*the three or four cases*), etc.

4.3 PP Segmentation

Once NP boundaries are marked, we insert on the left of any preposition a temporary PP beginning mark (TBeginPP = <**PP**):

```
<PP Avec ou <PP sans [NP le premier
          ministre NP[3]]
```

Then the longest sequence containing at least one TBeginPP followed by one EndNP is surrounded with the [PP and PP] boundaries using the RE:

```
[TBeginPP ~$[EndNP|TVerb] EndNP] @->
         BeginPP ...  EndPP
```

which eventually leads to:

```
[PP Avec ou sans le premier ministre PP]
```

4.4 VC Segmentation

A VC (Verb Chunk) is a sequence containing at least one verb (the head). It may include words or segments (NPs, PPs, APs or other VCs) that are possibly linked as arguments or adjuncts to the verb. There are three types of VCs: infinitives, present participle phrases and finite verb phrases. We first mark infinitives and present participle segments as they are simpler than finite verb phrases–they are not recursive, they cannot contain other VCs.

4.4.1 Infinitives

The infinitive phrases are recognized using the regular expression:

```
[ (PREPO) (NEG) (ADVP) PC* INF
  [(ADVP PastPartV+) | PastPartV*]]
```

e.g.: *sans même prévenir* (*without even warning*):

```
[VC [NP Mr NP] [NP Guilhaume NP] supprime VC]
[PP des émissions PP] [VC sans même prévenir
VC] [NP leurs responsables NP]
```

4.4.2 Present Participle Segments

The present participle phrases are recognized using the regular expression:

```
[ (EN) (NEG) PC* PrePart
[(ADVP PastPartV+) | PastPartV*]]
```

e.g.: *en dénonçant* (*while denouncing*)

```
[VC en dénonçant VC] [NP les provocations
NP] [ADJ mensongères ADJ]
```

[3] *With or without the prime minister.*

4.4.3 Finite Verb Segments

Here we use the basic idea described in the NP marking: temporary beginnings (TBeginVC) and ends (TEndVC) of VC are first marked.

Temporary beginnings of VCs are usually introduced by grammatical words such as *qui* (relative pronoun), *lorsque*, *et* (coordination) etc. However, not all these words are certain VC boundaries: *et* could be an NP coordinator, while *que* (tagged as CONJQUE by the HMM tagger) could be used in comparatives (e.g. *plus blanc que blanc*). Therefore, we use three kinds of TBeginVC to handle different levels of uncertainty: a certain TBeginVC (TBeginVC1), a possible BeginVC (TBeginVC2) and an initial TBeginVC (TBeginVCS) automatically inserted at the beginning of every sentence in the input text. With TBeginVCS, we assume that the sentence has a main finite verb, as is usually the case, but this is just an assumption that can be corrected later.

A temporary end of VC (TEndVC) is then inserted on the right of any finite verb, and the process of recognizing VCs consists of the following steps:

- Step 1: Each certain TBeginVC1 is matched with a TEndVC, and the sequence is marked with **[VC** and **VC]**. The matching is applied iteratively on the input text to handle the case of embedded clauses (arbitrarily bound to three iterations in the current implementations).
- Step 2: The same is done with the TBeginVCS (inserted at the beginning of a sentence).
- Step 3: If there is still a TEndVC that was not matched in (1) or (2), then it is matched with a possible TBeginVC2, if any, and the sequence is marked with [VC and VC].
- Step 4: Any TBeginVC that was not matched in (1), (2) or (3) is removed.

Verb Segmentation Example:

Initial input

```
Lorsqu' [NP on NP] appuie [PP sur l'
interrupteur PP] [PP de feux PP] [PP de
détresse PP] , [NP tous_les indicateurs NP] [PP
de direction PP] clignotent simultanément et
[NP un triangle NP] [AP rouge AP] clignote [PP
dans l' interrupteur PP] ./SENT[4]
```

[4] *When the hazard warning switch is pressed all the direction indicators will flash in unison and the switch will flash a red triangle.*

Temporary tagging of VC boundaries

<VCS <VC1 Lorsqu' [NP on NP] appuie VC> [PP sur l' interrupteur PP] [PP de feux PP] [PP de détresse PP] , [NP tous_les indicateurs NP] [PP de direction PP] clignotent VC> simultanément <VC2 et [NP un triangle NP] [AP rouge AP] clignote VC> [PP dans l' interrupteur PP] ./SENT

VC marking

[VC [VC Lorsqu' [NP on NP] appuie VC] [PP sur l' interrupteur PP] [PP de feux PP] [PP de détresse PP] , [NP tous_les indicateurs NP] [PP de direction PP] clignotent VC] simultanément [VC et [NP un triangle NP] [AP rouge AP] clignote VC] [PP dans l' interrupteur PP] ./SENT

5 Marking Syntactic Functions

The process of tagging words and segments with syntactic functions is a good example of the non-monotonic nature of the parser and its hybrid constructive-reductionnist approach. Syntactic functions within non recursive segments (AP, NP and PP) are addressed first because they are easier to tag. Then other functions within verb segments and at sentence level (subject, direct object, verb modifier, etc.) are considered.

Potential subjects are marked first: an NP is a potential subject if and only if it satisfies some typographical conditions (it should not be separated from the verb with only one comma, etc.). This prevents the NP *Jacques*, for example, from being marked as a subject in the sentence below:

[VC [NP le président NP]/SUBJ [PP du CSA PP], [NP Jacques NP] [NP Boutet NP] , a décidé VC] [VC de publier VC] [NP la profession NP] [PP de foi PP] ./SENT [5]

Then constraints are applied to eliminate some of the potential subject candidates. The constraints are mainly syntactic: they are about subject uniqueness (unless there is a coordination), the necessary sharing of the subject function among coordinated NPs, etc. The remaining candidates are then considered as real subjects. The other syntactic functions, such as object, PP-Obj, verb modifier, etc. are tagged using similar steps.

[5] *The CSA president, Jacques Boutet, decided to present his profession of faith.*

6 Expanding Verb Segments

Because primary segmentation is cautious, verb segments end right after a verb in order to avoid arbitrary attachment of argument or adjunct segments (NPs, PPs and APs on the right of a verb). However, experiments have shown that in some kinds of texts, mainly in technical manuals written in a "controlled language", it is worth applying the "nearest attachment" principle. We expand VCs to include segments and to consider them as arguments or adjuncts of the VC head. This reduces structural ambiguity in the parser output with a very small error rate. For instance, expanding VCs in the sentence given in the previous section leads to the following structure:

[VC [NP le président NP]/SUBJ [PP du CSA PP], [NP Jacques NP] [NP Boutet NP] , a décidé [VC de publier [NP la profession NP] [PP de foi PP] VC] VC] ./SENT

Nevertheless, as this principle leads to a significant number of incorrect attachments in the case of more free-style texts, the VC expansion network is optionally applied depending on the input text.

7 Performance

As mentioned above, the parser is implemented as a sequence of finite state networks. The total size of the 14 networks we currently use is about 500 KBytes of disk space. The speed of analysis is around 150 words per second on a SPARCstation 10 machine running in a development environment that we expect to optimize in the future. As for linguistic performance, we conducted a preliminary evaluation of subject recognition over a technical manual text (2320 words, 157 sentences) and newspaper articles from *Le Monde* (5872 words, 249 sentences). The precision and recall rates were respectively 99.2% and 97.8% in the first case, 92.6% and 82.6% in the case of the newspaper articles. This difference in performance is due to the fact that, on the one hand, we used the technical manual text to develop the parser and on the other hand, it shows much less rich syntactic structures than the newspaper text. We are currently conducting wider experiments to evaluate the linguistic accuracy of the parser.

8 Parsing Samples

Below are some parsing samples, where the output is slightly simplified to make it more readable. In particular, morphosyntactic tags are hidden and only the major functions and the segment boundaries appear.

A l'interprétation des sentiments présidentiels s'ajoute l'atmosphère de surenchère politique qui précède tout congrès du Parti socialiste.

```
[VC [PP A/PrepN> l'/DET> interprétation
PP]/PPObj [PP des/PrepN> sentiments PP]/PPObj
[AP présidentiels AP]/<NM s' ajoute|MV VC] [NP
l'/DET> atmosphère NP]/<SUBJ [PP de/PrepN>
surenchère PP]/PPObj [AP politique AP]/<NM [VC
[NP qui NP]/SUBJ précède VC] tout/VM [NP
congrès NP]/OBJ [PP du/PrepN> Parti PP]/PPObj
[AP socialiste AP]/<NM ./SENT
```

Les députés azerbaidjanais ont adressé à Moscou un ultimatum exigeant la levée de l'état d'urgence, et le retrait des troupes, faute de quoi ils reconsidéreraient " l'acte d'union " intégrant l'Azerbaidjan à l'URSS.

```
[VC [NP Les/DET> députés NP]/SUBJ [AP
azerbaidjanais AP]/<NM ont adressé|MV VC] [PP
à/PrepN> Moscou PP]/PPObj [NP un/DET>
ultimatum NP]/OBJ [VC exigeant VC] [NP la/DET>
levée NP]/OBJ [PP de/PrepN> l'/DET> état
PP]/PPObj [PP d'/PrepN> urgence PP]/PPObj , et
[NP le/DET> retrait NP]/N [PP des/PrepN>
troupes PP]/PPObj , [VC faute_de_quoi [NP ils
NP]/SUBJ reconsidéreraient VC] [NP l'/DET> acte
NP]/OBJ [PP d'/PrepN> union PP]/PPObj [VC
intégrant VC] [NP l'/DET> Azerbaidjan NP]/OBJ [PP
à/PrepN> l'/DET> URSS PP]/PPObj ./SENT
```

A l'heure, vendredi soir, où les troupes soviétiques s'apprêtaient à pénétrer dans Bakou, la minuscule République autonome du Nakhitchevan, territoire azéri enclavé en Arménie à la frontière de l'Iran, proclamait unilatéralement son indépendance, par décision de son propre Soviet suprême.

```
[VC [PP A/PrepN> l'/DET> heure PP]/PPObj , [NP
vendredi NP]/N [NP soir NP]/<NM , [VC où [NP
les/DET> troupes NP]/SUBJ [AP soviétiques
AP]/<NM s' apprêtaient VC] [VC à pénétrer VC]
[PP dans/PrepN> Bakou PP]/PPObj , [NP la/DET>
[AP minuscule AP]/NM> République NP]/SUBJ [AP
autonome AP]/<NM [PP du/PrepN> Nakhitchevan
PP]/PPObj , [NP territoire NP]/N [AP azéri
AP]/<NM [AP enclavé AP]/<NM [PP en/PREP
Arménie PP]/PPObj [PP à/PrepN> la/DET>
frontière PP]/PPObj [PP de/PrepN> l'/DET> Iran
PP]/PPObj , proclamait|MV VC] unilatéralement/VM
[NP son/DET> indépendance NP]/OBJ [PP par/PrepN>
décision PP]/PPObj [PP de/PrepN> son/DET>
[AP propre AP]/NM> Soviet PP]/PPObj
[AP suprême AP]/<NM ./SENT
```

9 Conclusion

The incremental finite-state parser presented here merges both constructive and reductionist approaches. As a whole, the parser is constructive: it makes incremental decisions throughout the parsing process. However, at each step, linguistic contraints may eliminate or correct some of the previously added information. Therefore, the analysis is non-monotonic and handles uncertainty.

The linguistic modularity of the system makes it tractable and easy to adapt for specific texts (e.g. technical manuals or newspaper texts). This is done by adding specialized modules into the parsing sequence. This way, the core grammar is clearly separated from optional linguistic descriptions and heuristics.

Ongoing work includes expansion of the French grammar, a wider evaluation, and grammar development for new languages. We will also experiment with our primary target applications, information retrieval and translation assistance.

Acknowledgements

We would like to thank Kenneth R. Beesley and Lauri Karttunen for their editorial advice and Gregory Grefenstette for the valuable discussions we had about finite-state parsing and filtering.

References

Steven P. Abney, 'Parsing by chunks', in *Principled-Based Parsing*, eds., R. Berwick, S. Abney, and C. Tenny, Kluwer Academic Publishers, Dordrecht, (1991).

Douglas E. Appelt, Jerry R. Hobbs, John Bear, David Israel, and Mabry Tyson 'FASTUS: A Finite-State Processor for Information Extraction from Real-World Text', in *Proceedings IJCAI-93*, Chambery, France, August 1993.

Gabriel G. Bès, 'Axiomas y algoritmos en la descripciòn de las lenguas naturales', V Congreso Argentino de Lingüistica, Mendoza, 1993.

Jean-Pierre Chanod and Pasi Tapanainen, 'Tagging French - comparing a statistical and a constraint-based method', in *Proceedings of the Seventh Conference of the European Chapter of the Association for Computational Linguistics*, pp. 149–156, Dublin, (1995).

Jean-Pierre Chanod and Pasi Tapanainen. 'A Robust Finite-State Parser for French', in *ESSLLI'96*

Workshop on Robust Parsing, August 1996 12-16, Prague, Czech Republic.

Eva Ejerhed, 'Nouveaux courants en analyse syntaxique', *Traitement automatique des langues*, **34**(1), (1993).

Stefano Federici, Simonetta Montemagni and Vito Pirrelli 'Shallow Parsing and Text Chunking: a View on Underspecification in Syntax', in *ESSLLI'96 Workshop on Robust Parsing*, August 1996 12-16, Prague, Czech Republic.

Gregory Grefenstette, 'Light Parsing as Finite-State Filtering', in *Proceedings ECAI '96 workshop on "Extended finite state models of language"* Aug. 11-12, 1996, Budapest.

Karen Jensen, George E. Heidorn, and Stephen D. Richardson, eds., *Natural language processing: the PLNLP approach*, number 196 in The Kluwer international series in engineering and computer science, Kluwer Academic Publishers, Boston/Dordrecht/London, 1993.

Aravind Joshi. 'A Parser from Antiquity: An Early Application of Finite State Transducers to Natural Language Parsing', in *Proceedings ECAI '96 workshop on "Extended finite state models of language"*, Budapest, August 11-12, 1996, Budapest.

Lauri Karttunen, 'Directed replacement', in *Proceedings of the 34th Annual Meeting of the Association for Computational Linguistics*, Santa Cruz, USA, (June 1996). Association for Computational Linguistics.

Kimmo Koskenniemi, Pasi Tapanainen, and Atro Voutilainen, 'Compiling and using finite-state syntactic rules', in *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92* vol. I, pp. 156–162. Nantes, (1992).

Emmanuel Roche, *Analyse syntaxique transformationnelle du français par transducteurs et lexique-grammaire*, Ph.D. dissertation, Université de Paris 7, 1993.

Atro Voutilainen and Pasi Tapanainen, 'Ambiguity resolution in a reductionistic parser', in *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, pp. 394–403, Utrecht, (1993).