Towards a Finite-State Parser for Swedish

Beáta Megyesi & Sara Rydin
Computational Linguistics
Department of Linguistics
Stockholm university
[bealsara]@ling.su.se

Abstract

In this study, we describe a method for parsing part-of-speech tagged unrestricted texts in Swedish using finite-state networks. We use the Xerox Finite-State Tool because of its expressiveness and power for writing and compiling regular expressions and relations. The parser is divided into four modules: i) contiguous phrase structure marker, ii) phrasal head marker, iii) syntactic function tagger, and iv) non-contiguous group boundary recognizer. The aim is to develop a parser that can be used as a light/shallow parser for marking phrase structure and, when needed, to label syntactic functions. We believe that modularity is necessary since different NLP tasks require various levels of analysis. The parser for Swedish is under development, but present-day results are promising.

1. Introduction

In several Natural Language Processing (NLP) tasks, such as information retrieval, information extraction, speech technology, machine translation, etc., full or partial information about phrasal and/or syntactic structures is needed. The main interest in these tasks lies in detecting the constituent structures and sometimes their syntactic functions in a robust and fast way. In this study, our aim is to develop a parser for Swedish part-of-speech tagged texts, based on finite-state techniques using the Xerox Finite-State Tool (Karttunen et al, 1997).

Finite-state techniques have been shown to be very useful for parsing unrestricted texts for several languages, such as English, Finnish, French, German, Swedish, etc. Under certain circumstances, these parsers are robust, fast and accurate. There are mainly three approaches that have been applied for the construction of finite-state parsers: constructive, reductionist, and the combinations of these.

Briefly, the *constructive* approach is based on lexical description of large collections of syntactic patterns using subcategorisation frames such as verbs and their arguments, and local grammars (Abney, 1996). The *reductionist* approach, on the other hand, starts from a large number of alternative analyses that get reduced through the application of constraints where the constraints may be expressed by a set of elimination rules (Voutilainen & Tapanainen, 1993) or by a set of restrictions applied in parallell (Koskenniemi et al, 1992). The *hybrid* method merges the constructive and the reductionist approaches. It is developed by Aït-Mokhtar and Chanod (1997) who built an incremental finite-state shallow parser for French in a modular way. The parser

Proceedings of NODALIDA 1999, pages 115-123

makes incremental decisions throughout the parsing process. Syntactic information is added at the sentence level depending on the contextual information. They achieve broad coverage and include richer information than typical chunking systems.

A common procedure for building finite-state parsers from part-of-speech tagged texts is to first mark contiguous groups, e.g. noun or verb groups, then mark the heads within the groups and lastly, to extract patterns between non-contiguous group bounderies. However, Grefenstette (1996) points out that several parsers mix non-finite-state methods with finite-state procedures for different modules. He shows that the entire parser can be built easily within a finite-state framework by using finite-state transducers.

Finite-state transducers are finite-state machines that take an input and produce an output with each state transition. They generate or accept regular relations, i.e. sets of pairs of strings where each string has an upper and lower language. They can be written as regular expressions and can be used for introducing extra symbols into an input string, i.e. for labelling entities (groups) in a text. The labels, then, can be used for deriving further information from the text, such as extracting non-contiguous syntactic n-ary dependencies. By composing a sequence of transducers and dividing the parsing task into a sequence of partial tasks, such as contiguous group labelling, head marking, and the detection of non-contiguous group boundaries, Grefenstette presents a robust and fast light parser.

In this study, we use the Xerox finite-state tool (XFST), for constructing the parser. The reason for our choise is that the XFST is very convenient to use since it allows powerful and elegant linguistic descriptions by different operators for a high level of abstraction.

XFST is a general-purpose Unix application for computing with finite-state networks. Simple automata and transducers can be easily created by a set of operations from text files, binary files, regular expressions and other networks. Thus, XFST can read finite-state networks and compile them from regular expressions and text files. The networks can be simple finite-state automata or finite-state transducers and can be combined by various operations. In addition to the usual operators[1] (e.g. concatenation, union, optionality, Kleene star, Kleene plus, complement, intersection, relative complement, crossproduct, composition, etc.) XFST also supports some special operators for high level abstraction: restriction, replacement, and left to right longest match replacement. The *restriction* operator is very useful when writing constraints to exclude unwanted analyses. The rule A => B _ C expresses that A must appear in the context of B _ C, i.e. between B and C. The *replacement* operator replaces a string with another string with or without regard taken to context. For example, the rule A -> B|| L_R replaces A by B between a certain left and right context where A and B denote regular languages and the expression as a whole denotes a relation. The *longest match* operator is a special kind of replacement operator. It imposes a unique factorisation on every input. It can also be constrained by context and generalised for parallel replacement. For instance, the rule A @-> B ... C forces the transducer to locate and pick out maximal instances of the regular language A, leaving the entire string unchanged and inserting B and C around the selected A strings as markers.

2. The Parsing Method

The Swedish parser is based on the hybrid approach using a cascade of finite-state transducers. The parser consists, in its present form, of four modules: the phrase structure module, the phrasal head module, the syntactic function module and the non-contiguous group boundary module. The thought behind the modular architecture is to facilitate the work during development, to allow different uses of the parser and to reflect the different linguistic knowledge that is built into the parser.

First, modularity is important during the development of the parser as the modules (and the rules in each module) are ordered. Because information about the ordering of rules/modules and the separation of the linguistic knowledge are clearly specified, the detection of sources of incorrect analysis is facilitated. It is, because of the modularity, not only possible to use all four modules in the parser but to separate and run only the first module, the first and the second module, and so on. The first module would give us a phrase structure analyzed text and the sequential addition of the other three modules would introduce more and more information to the analysis.

Secondly, modularity can also be useful in regular use. There are, for example, NLP-tasks where only the information given by the first module is wanted; information about noun phrase and verb phrase boundaries can be used to identify events and entities in information extraction. There are also times when the syntactic function of the noun phrases is needed; information about the object can, in word sense disambiguation, be used to disambiguate the verb.

There is one last reason for modularity that is purely technical. With four modules, the whole set of rules is compiled into four separate finite-state networks. If all this information would be merged into one module, the compiled finite-state network would be quite large, the compilation would be time-consuming and the insertion of additional rules and the altering of the rule order would be more complex.

In the following, the finite-state networks describing the phrase and syntactic structure of the language are presented. As mentioned above, the parser consists of four networks, where each network is a composition of simple finite-state automata and/or finite-state transducers. Within each network, the transducers are composed and ordered in such a way that the easiest tasks are addressed first.

Each module marks up specific linguistic information by the use of reserved symbols, i.e. symbols that cannot be found in the natural language text files that are analyzed. The reserved symbols used by the parser consist of brackets and labels, shown in table 1 below.

Brackets	Label	Example	Comment
[/]	NP, PP, VP, AP	[NP	Left hand side phrase structure tag for NP
*	ActV, CopV, PasV InfV, HeadN, PrepN	*PrepN	Tag for head within a NP in a PP
{*** / ***}	Subj, Obj, Advl, PredF	{***Subj	Left hand side tag for subject
[/]	PVP	[PVP	Left hand side tag for particle verb phrase

Table 1. Symbols (tags) inserted by the parser

The corpus used to train and test the parser is the Stockholm-Umeå Corpus, so-called SUC (Ejerhed et al, 1992) annotated with the PAROLE tag set. A plus sign, a part-of-speech tag and an appropriate number of morphological tags follow each token in the text:

"...svenska+AQP0PN0S städer+NCUPN@IS..."

Before describing the modules that the parser consists of, the reader should be reminded that the parser is under development. This means that the description of Swedish in no way is fully correct or exhaustive. The goal has been to see how suitable XFST or finite-state networks are for building a parser for Swedish.

2.1 Phrase structure module

The first finite-state network module marks phrase structure for noun phrases (NP), verb phrases (VP), prepositional phrases (PP), adverb phrases (AdvP) and adjective phrases (AP). The parsing is done in a bottom-up fashion where the deepest constituents are analyzed first. Thus, adverb phrases are detected before the adjective phrases since AdvPs may be included in APs but not the other way around. In a similar way, APs are marked before NPs.

Example 1 below shows how the adjective phrase is marked up. First, an adverb phrase (ADVP) is defined as a word (Ord[4]), a part-of-speech tag "R" and a string of morphological tags (Tagg+). Second, the adjective phrase (AP) is defined as containing an optional adverb phrase, a word (Ord), the part-of-speech tag (A) followed by morphological tags (Tagg). Last, the regular expression for the insertion of the AP tag is defined with the help of the longest match and replacement operators.

```
define ADVP [Ord R Tagg+] ;
define AP [(ADVP " ") Ord A Tagg] ;
regex AP @-> "[AP " ... " AP]" ;
```

Example 1. Definition of and insertion of tags for the adjective phrase.

Next, the noun phrases are detected. Noun phrases are presently defined as being of three different types: 1. a single pronoun (PRON), 2. an optional determiner (DET) followed by one or more optional ordinal/cardinal numerals (NUM), followed by an optional adjective phrase (AP) (the last two can optionally be in reversed order), and at least one noun, 3. an optional determiner, a possessive pronoun (POSSPRON), an optional, tagged adjective phrase and at least one noun. The definition of NPs is given below.

```
define NP [[PRON]|[(DET) ([NUM]+) (AP) ([NUM]+) [NOUN]+]|
      [(DET) POSSPRON (AP) [NOUN]+]] ;
```

Only attributes that precede the noun are included in the NP. The regular expression for the insertion of tags for noun phrases (and VP and PP as well) is similar to the regular expression given for AP (see example 1). In the rest of this section, only the outlines of the definition rules are given in order to make the examples easier to understand.

Next, prepositional phrases are defined as consisting of either a preposition (PREP) followed by a noun phrase (NP), or of a composite preposition and a tagged noun phrase.

```
define PP [[[PREP]|[PREP KONJ PREP]] NP] ;
```

Last, the verb phrase can have two different forms depending on the sentence type the verb occurs in. First, the position of the verb in regular word order is defined as follows: an optional infinitive particle (INF, equivalent to the Eng. *to*), at least one verb (VERB) and an optional verb particle (PART). Secondly, the position of the verb is given in the case of subject-verb inversion: an auxiliary verb (AUX), followed by a tagged noun phrase (NP), followed by at least one verb (VERB) and an optional verb particle (PART).

```
define VP [[(INF ) [VERB]+ (PART)]|[AUX NP [VERB]+ (PART)]] ;
```

Below, example 2 shows text annotated with phrase tags where the phrase tags are in bold face. The sentence in English is: *"Fear of the disease forced the decision to build water mains and sewage pipes"*.

[NP Skräcken+NCUSN@DS **NP]** **[PP** för+SPS **[NP** sjukdomen+NCUSN@DS **NP] PP]** **[VP** tvingade+V@IIAS fram+QS **VP]** **[NP** beslut+NCNSN@IS **NP]** om+SPS **[VP** att+CIS bygga+V@N0AS **VP]** **[NP** vattenledningar+NCUPN@IS **NP]** och+CCS **[NP** avloppsrör+NCNPN@IS **NP]** .+FE

Example 2: Output from the Phrase Structure Module[2].

2.2 Phrasal head module

The output of the phrase structure module constitutes the input to the phrasal head module. Heads are marked in two types of phrases: verb phrases and noun phrases. The phrase head information is marked in order to be used later, in the definition of syntactic functions. Grefenstette (1996) suggested the division of noun and verb phrases into subcategories, though we decided on a different division[3] for the verb types. For noun phrases there are two tags; HeadN and PrepN. Tagging of noun phrase heads are done using two composed transducers. The first transducer tags all head nouns (i.e. the last noun or pronoun in the noun phrase) as HeadN while the second transducer alters the tag HeadN to PrepN when it occurs in a prepositional phrase. Note that this is not to say that the head of the PP is the noun (which would of course be wrong), but only a way to mark the noun in the PP and thereby differentiate NP included in PPs from the poor lonely ones.

The verb phrases have four types of head tags according to different subcategorisation frames: the active (ActV), the passive (PasV), the infinitive (InfV), and the copulative verb (CopV). The first three are easily defined and tagged on the basis of the morphological information given by the PAROLE tag set. Copulative verbs, on the other hand, are defined as one of the words 'bliva' ('become'), 'finnas' ('be', 'exist'), 'vara' ('be') and 'heta'/'kallas' ('be called'). Example 3 shows the output from the phrase head module.

> [NP ***HeadN** Skräcken+NCUSN@DS NP] [PP för+SPS [NP ***PrepN** sjukdomen+NCUSN@DS NP] PP] [VP ***ActV** tvingade+V@IIAS fram+QS VP] [NP ***HeadN** beslut+NCNSN@IS NP] om+SPS [VP att+CIS ***InfV** bygga+V@N0AS VP] [NP ***HeadN** vattenledningar+NCUPN@IS NP] och+CCS [NP ***HeadN** avloppsrör+NCNPN@IS NP] .+FE

Example 3: Phrasal head information inserted for the same sentence as in example 2.

2.3 Syntactic function module

In this module, an attempt is made to mark the syntactic functions of the phrases. We have elaborated with four kinds of syntactic functions: the subject, the object, the adverbial and the complement to the copulative verb. The annotation is done with help from the phrase tags and the head labels inserted by the previous modules. For example, NPs containing the HeadN label (in contrast to NPs with PrepN label) can be marked as subject or object. We have tried only to annotate syntactic function when fairly certain of the correctness of the result. Thus, we tried to avoid rules that would increase the recall but lower the precision substantially. Still, this module of the parser is probably the trickiest because of semantic and structural ambiguities.

Subject labeling is often dependent of both the left and the right context of the possible subject. There are several rules for the annotation of subjects and the choice among

them is done in the parser based on the word order in the sentence. Rules are primarily specified for regular word order (SV) and subject-verb inversion (VS). The scope of the subject is extended to include not only the noun phrase but also following adjacent prepositional phrases. This extension cannot be done for the object because of the PP-attachment ambiguity.

The annotation of objects is done on the basis of its left context. Here, the position of the verb and the already labeled subject are of interest since the object must follow the subject and/or the verb in a sentence. Note that the rule does not cover the case of topicalized objects, e.g. in the sentence 'Him she loved'.

The complement to the copulative verb is easily found since the copulative verb itself was annotated in the previous module by subcategorisation. Both object and verbal complements are expected to come after the verb but unfortunately, that is not always the case. Semantic features would be necessary to handle these phenomena correctly. Concerning adverbials only prepositional phrases are marked as such in the present module.

Lastly, an example of a sentence with syntactic function tags is given.

{*Subj** [NP *HeadN Skräcken+NCUSN@DS NP] **{***Advl** [PP för+SPS [NP *PrepN sjukdomen+NCUSN@DS NP] PP] **Advl***} Subj***}** [VP *ActV tvingade+V@IIAS fram+QS VP] **{***Obj** [NP *HeadN beslut+NCNSN@IS NP] **Obj***}** om+SPS [VP att+CIS *InfV bygga+V@N0AS VP] **{***Obj** [NP *HeadN vattenledningar+NCUPN@IS NP] och+CCS [NP *HeadN avloppsrör+NCNPN@IS NP] **Obj***}** .+FE

Example 4: Output from the syntactic function module for the same sentence as in example 2 and 3.

2.4 Module for Non-Contiguous Group Boundaries

At the moment, this module incorporates only information about verb particles. Most of the particles are already found by the phrasal rules in the first module, i.e. when they follow directly after the verb. Here, those particles that are not adjacent to the verb are detected. The regular expression is quite straightforward as the verb particles have a separate tag in SUC and phrases between the verb and the particle are already marked up. In the future, we plan to incorporate other long distance dependencies, for instance in non-contiguous VP idioms.

3. Discussion

Presently, no extensive test or evaluation has been done on the parser since correctly labeled texts with phrase structure and syntactic information are not available. However, we tested the different modules on one text, consisting of 3000 words. The system accuracy regarding the detection of the different phrase structures seems to be good,

approximately 95%. The precision of marking syntactic features is lower approximately 60%-70%, because of syntactic ambiguity, such as PP-attachment, the scope of predicatives, complex NP structures and elliptic expressions. Recall is in both cases lower since our strategy has been to only label entities when fairly certain of the correctness of the result. As the reader realises, there is more work to do in order to develop a reliable parser.

However, we believe that the finite-state tool together with our parser architecture suits the requirements for a useful shallow parser. The advantage of our system is that it is fast, robust (in the case of the shallow parser) and modular. Because of the modularity, the user can choose between only analysing the phrase structure, that is the usual case, or adding even syntactic analyses when needed.

4. Conclusions

In this study, we presented a method for parsing part-of-speech tagged unrestricted texts in Swedish by using finite-state techniques in the Xerox Finite-State Tool. Because of the modular architecture of the parser, it can be used as a light/shallow parser for marking phrase structure and, when needed, to label syntactic functions. The different modules reflect different types of linguistic knowledge such as information on phrase structure, phrasal heads and syntactic functions. However, the parser for Swedish is under development. Due to the promising results we are planning to continue to improve upon the different modules.

Acknowledgements

We would like to thank the Department of Linguistics, Uppsala university, for giving us the opportunity to participate in the course 'Automata theory', and especially Torbjörn Lager who first introduced us to the XFST during this course.

Footnotes

[1] See Karttunen et al (1997) for a good description of the XFST operators.
[2] 'Ord' is defined as a string of accepted characters in the natural language that forms a word.
[3] Note that neither the maximal projection of the NPs ('vattenledningar och avloppsrör'), nor the PP consisting of a preposition and infinitive verb phrase ('om att bygga') are labeled in this module.
[4] Grefenstette (1996) parses verb and noun groups instead of phrases.

References

Abney, S. 1996. Partial Parsing via Finite-State Cascades. In *Workshop on Robust Parsing, 8th European Summer School in Logic, Language and Information*, 8-15 Prague, Czech Republic.

Aït-Mokhtar, S., & Chanod, J-P. 1997. Incremental Finite-State Parsing. In *Proceedings of ANLP'97*, 72-79, Washington.

Chanod, J. P. & Tapanainen, P. 1996. A robust finite-state grammar for French. *Techniqual report, Rank Xerox Research Centre*, Meylan, France.

Ejerhed, E., Källgren, G., Wennstedt, O. & Åström, M. 1992. The Linguistic Annotation System of the Stockholm-Umeå Corpus Project. *Report nr. 33,* Dept. of General Linguistics, University of Umeå.

Grefenstette, Gregory. 1996. Light Parsing as Finite-State Filtering. *Workshop on Extended Finite State Models of Language, ECAI-96*, Budapest, Hungary.

Karttunen, L., Chanod, J. P., Grefenstette, G., & Schiller, A. 1997. Regular Expressions for Language Engineering. *Natural Language Engineering* 2, 305--238, Cambridge University Press.

Koskenniemi, K. 1990. Finite-State Parsing and Disambiguation. In *Proceedings of the Thirteenth International Conference on Computational Linguistics COLING-90* 2, 229-- 232, Helsinki, Finland.

Koskenniemi, K., Tapanainen, P., & Voutilainen, A. 1992. Compiling and using finite-state syntactic rules. In *Proceedings of the Fourteenth International Conference on Computational Linguistics COLING-92* vol 1, 156-162, Nantes, France.

Voutilainen, A. & Tapanainen, P. 1993. Ambiguity resolution in a reductionistic parser. In *Proceedings of the Sixth Conference of the European Chapter of the Association for Computational Linguistics*, 394-403, Utrecht, Netherlands.