The AGFL Grammar Work Lab

Cornelis H.A. Koster & Erik Verbruggen
Dept. Comp. Sci.
University of Nijmegen (KUN)
The Netherlands
{kees,erikv}@cs.kun.nl

Abstract

The AGFL Grammar Work Lab is the first parser generator for natural languages to be brought under the GNU public license. Apart from its linguistic uses, it is intended for the production of parsers which are to be embedded in application systems. In particular, the AGFL system comes with a free grammar and lexicon of English, allowing the construction of user interfaces and applications involving Natural Language Processing.

We give a brief description of the AGFL formalism and its use in transducing English text to Head/Modifier frames and discuss some possible applications.

1 Introduction

A large part of the capacity of computers is devoted to the capture, storage, analysis, transformation and production of human-readable documents, in the form of publications, correspondence and web-documents. Therefore, a growing number of applications is dependent on, or could benefit from, some form of linguistic analysis of documents.

In particular, Natural Language Processing (NLP) is an important enabling technology for future web-based applications: from classification of web-pages, filtering and narrowcasting to more intelligent search machines and services based on the automatic interpretation of the contents of documents. As is the case in Information Retrieval (IR) in general, the state-of-the-art in search machines on the web is based mainly on the use of keywords, and only some limited linguistic techniques are used to enhance recall: stop lists, stemming, some ontologies, and the simplest of phrase recognition techniques. An example is the Linguistix software library, incorporated in commercial search machines like Altavista and Askjeeves, which performs tagging, lemmatization and fuzzy semantic matching.

No use is made of syntax analysis or semantic analysis and the discourse structure of documents is largely ignored, although their use might yield an important increase in precision. The great success of the present statistical techniques combined with such "shallow linguistic techniques" [Sparck Jones, 1998] has led to the idea that deep linguistics is not worth the trouble.

What is worse, it is very hard to find resources to build applications using deeper linguistic techniques like parsing. In applied linguistic communities like the `corpora` list, many groups appear to be in need of parsers and lexica for natural languages, and requests for freely accessible linguistic resources are frequently posed. But such resources are just not available, or just not free.

There is a definite need for parsers and lexica in the public domain, so that people developing say a question answering system do not have to start by reinventing the wheel. The extraordinary success of one such resource, the [Princeton WordNet], may be due to its public availability rather than to superb quality, but it has had tremendous impact, and it is improving over time.

In this article, we make a plea for linguistic resources in the public domain and announce the public availability of the AGFL Grammar Work Lab, and the EP4IR grammar and lexicon of English. We describe how to use a parser, generated by the AGFL system from EP4IR, in practical applications.

2 The problem area

Academic research groups that have developed parsers and lexica are unable to sell these as products and market them. Such resources may have cost many manyears for their development, but that does not mean anyone is willing to pay the same price to obtain them. Furthermore, being academics they are not in a position to offer maintenance.

There are a number of repositories of linguistic resources, but they are either proprietary or they make the resources available at a low price for research purposes only, while the conditions for commercial use are very vague (write us). The low price hardly covers the cost of distribution and certainly is not enough to cover maintenance. The gold wagon is expected to come in from industry.

As a consequence, in building NLP applications many industrial corporations prefer to develop their own resources from scratch rather than being dependent on others. Research whose results might become economically interesting can not be based on such resources.

In fact, the situation is remarkably like that for software in the eighties, and the same solution should be considered:

> Basic linguistic resources like grammars, lexica, parsers, corpora and ontologies should be made freely available in the public domain, especially if they have been developed with public money. Their users should be invited to contribute improvements, thus enabling a low-cost form of maintenance.

Where have we heard this before?

3 AGFL under GPL

The purpose of this article is to announce the availability of the AGFL Grammar Work Lab under the GNU Public Licence, making it publicly and freely available as a tool for linguistic research and for the development of NLP-based applications. The AGFL system is the *first parser-generator for natural languages* available under the GPL.

The run-time system for the generated parsers has been brought under the Lesser GPL, so that parsers by the system may be included in other systems (even commercially) under very liberal conditions.

The system comes with a number of grammars and lexica for free, in particular the EP4IR (English Phrases for Information Retrieval) grammar of English. Linguists and Computer Scientists alike are invited to use the AGFL system and the accompanying EP4IR grammar and lexicon of English for whatever purpose they like, including commercial purposes, as long as the GPL is adhered to. Linguists are invited to make and share improvements to the free grammars and lexica, or add new grammars and lexica in the same spirit.

4 Affix Grammars over a Finite Lattice

The AGFL formalism (Affix Grammars over a Finite Lattice) [Koster, 1992] is a notation for Context-Free grammars enriched with finite set-valued features, acceptable to linguists of many different schools. For a computer scientist this means: syntax rules are procedures with parameters and a nondeterministic execution, like that of PROLOG.

No natural language can reasonably be described by a deterministic grammar, so that deterministic parser generators like YACC are useless for realistic NLP. Nondeterminism (ambiguity!) is an essential property of language, so that a completely different kind of parser generator is needed. The AGFL system is such a system.

For the interested reader we give two examples to convey some of the flavor of AGFL. The notation of AGFL is reminiscent of that of PROLOG, with which it is distantly related. This may help in reading (and understanding) the examples.

4.1 Example: noun phrases

The first example is a fragment of a rather simplistic grammar for english noun phrases. Each rule is exemplified by one or more examples.

```
NUMB :: sing; plur.
```

The feature expressing the number can take on two values: `sing` or `plur`.

```
RULE noun phrase (NUMB):
   noun part (NUMB).
 # EX the previous president

RULE noun phrase (plur):
   noun part (NUMB1), coordinator,
      noun phrase (NUMB2).
 # EX the president and his wife
```

These two rules express that a noun phrase consists of one or more noun parts combined by coordinators. In the latter case it is always plural.

```
RULE noun part (NUMB):
   determiner (NUMB), noun group (NUMB);
 # EX the red bag
   noun group (NUMB).
 # EX software engineering
```

This rule has a number of alternatives, separated by semicolons. The number of the determiner has to agree with that of the noun group.

```
RULE noun group (NUMB):
   noun (NUMB);
 # EX bag
   adjective, noun group (NUMB);
 # EX red bag
   noun group (NUMB1), noun (NUMB).
 # EX software engineering
```

Obviously, the last rule is ambiguous for a noun phrase consisting of three or more nouns, like software engineering conference. Other sources of ambiguity are found in the attachment of preposition phrases (not described here) and in lexical ambiguities (e.g. time as noun and verb). AGFL provides a number of mechanisms (penalties, lexical frequencies, syntactic probabilities) to help in finding the most probable analysis (rather than the set of all analyses).

4.2 Example: transduction

The second example shows the recognition of certain sentence patterns and their *transduction* to Head/Modifier pairs `[head, modifier]`. The transduction mechanism allows the production of a (compositional) translation instead of a parse tree. For each alternative of the rule, a transduction can be specified preceded by a slash.

```
NUMB:: sing | plur.
PERS:: first | secnd | third.
TRAN:: intrans | trans.

RULE sentence:
   subject(NUMB, PERS),
     verb part (intrans, NUMB, PERS) /
      "[", subject, ",", verb part, "]";
```

The *transitivity* feature of the verb determines whether it takes an object. Assuming a suitable transduction for `subject` and `verb part`, this rule would transduce I am freezing to `[I, freeze]`.

```
RULE sentence:
   subject(NUMB, PERS),
     verb part (trans,NUMB,PERS), object /
   "[",subject,",[",verb part,",",object,"]]".
```

Under similar assumptions, this should transduce I was attending a software engineering conference to `[ I, [attend, [conference, software engineering]]]`.

5 The EP4IR grammar

The "English Phrases for IR" (EP4IR) grammar is a reasonably complete grammar of English, concentrating on the description of the noun phrase and the verb phrase. The grammar is provided with a large lexicon, providing detailed Part-Of-Speech information. The grammar is quite robust against incorrect input and unknown words. The EP4IR grammar and lexicon were developed in the [PEKING project] for Information Retrieval applications, and they are released along with the AGFL system.

From the grammar and lexicon, an English parser can be generated automatically using the AGFL system, which produces as its output not parse trees but Head/Modifier frames, more or less as described above. The following picture illustrates the generation of a parser and its use.

The HM frame representation is a very good starting point for diverse applications, and the transduc-

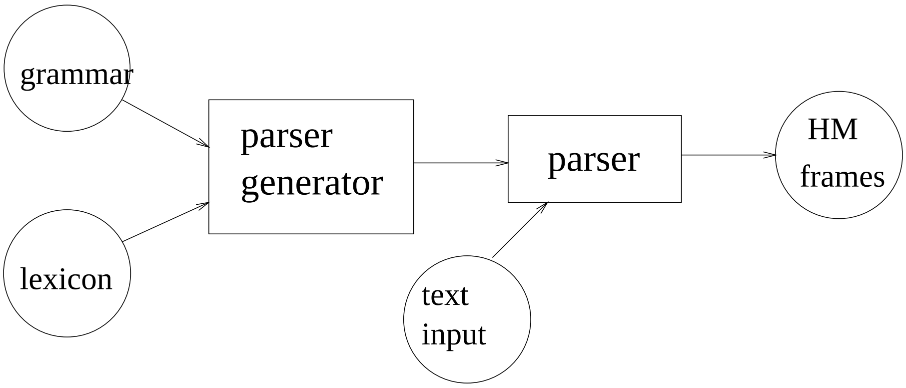

Figure 1: Generating and using a parser

tion can be adapted with relative ease to yet other applications.

The grammar still many details of the language, which will hopefully seduce linguists to propose additions and improvements.

5.1 Head/Modifier frames

The frames generated when parsing some text represent only the major relations expressed in the text:

relation	example
subject relation	`[N:script,V:calls]` `[P:it,V:reduced]`
object relation	`[V:performed,N:research]` `[V:interrupted,P:it]`
attr/pred relation	`[N:divergent,N:function]` `[N:analysis,A:syntactic]`
preposition relation	`[V:interacts,with N:user]` `[N:interaction,with N:user]` `[A:relative,to N:counter]`

Notice the fact that each word is *typed* as Noun, Pronoun, Adjective or Verbform and that, except for the prepositions, *all other words are eliminated*: adverbs, determiners, quantifiers. What is left is really the bare bone structure of the text. The goal is to get rid of embellishments and variations which add no value for retrieval purposes.

The types may be used in further processing of the frames, or they may be removed by some postprocessing.

A demo version of the parser/transducer is available at the website of the [AGFL project]; trying it out may help in understanding the HM frame representation.

5.2 Nested frames

Due to the recursive nature of phrases, the frames transduced from them may be nested, e.g. (omitting the types):

IBM sponsored this conference
$\Rightarrow$
`[IBM,[sponsored,conference]]`

this conference was sponsored by IBM
$\Rightarrow$
`[IBM,[sponsored,conference]]`

Every PhD student gets a reduced price for the ETAP conference
$\Rightarrow$
`[[student,PhD],[get,[price,reduced|for [conference,ETAP]]]]`

Notice the transformation from passive to active voice in the second example. Nested frames can be unnested using a component included with the grammar, the *unnester*, which for the last example yields

```
[N:student,N:PhD] [N:student,V:gets]
[V:gets,N:price] [P:it,V:reduced]
```

```
[V:reduced,N:price][N:price,
for N:conference][N:conference,N:ETAP]
```

5.3 Application examples

We shall describe only a few of the many possible examples, leaving it to the reader to contrive others.

- Information Retrieval
  The grammar was developed for IR applications, in which the traditional keywords are to be replaced by frames obtained from phrases. All frames are first unnested, so that each document is represented by a bag of frames without nesting. The frames are also morphologically normalized, using a lemmatizer and the typing information provided. Furthermore, semantically related frames will be clustered together (see [Koster et al., 1999]).

- Information Analysis and Modelling
  An important step in most analysis techniques is to start out from an informal (i.e. verbal) description of the problem domain or the problem, and scan it for nouns and verbs: the nouns are candidate objects or classes and the verbs are candidate methods [Abbott, 1983]. Other elements, like the adjectives can also be used [Graham, 1994]. The HM frame representation obtained by using EP4IR can be used straight away for this purpose. The same technique can be used for the *validation* of an existing Object-Oriented Analysis model [Frederiks et al, 1996].

- Question answering system
  A question answering system of any kind will need more information than is included in the HM frames, e.g. quantifiers and determiners. Of course the grammar does already recognize these constructs, so by modifying three or four lines in the transduction they will also be expressed. Luckily, the grammar already knows how to parse questions.

Feel free to use the AGFL system for your purposes according to the GPL/LGPL license. Let us know if you have a nice application. For more complicated projects you might consider collaborating with the authors of this paper.

6 Disclaimer

Nobody is perfect. The currently available release 2.0 of the AGFL system, resulting from a total revision of the formalism and its implementation, is only the first step. It still has to be improved, in particular with respect to its speed, but we are working on that. A version generating much faster parsers is in the pipeline. In the mean time, the system is “solidly under way, but may not yet be 100% finished”. The same holds for the accompanying grammars and lexica.

7 Summary and conclusions

There are many good reasons to bring the AGFL Grammar Work Lab into the GNU family:

- there is at present no parser generator for linguistic grammars under GPL
- AGFL can fill a niche that will make GNU attractive to a large number of linguistic users who now live in a Microsoft-dominated world
- AGFL is a well-developed and stable system, which merits availability in the public domain
- a university like ours (the University of Nijmegen) is not in a position to distribute and maintain the system on a commercial basis
- The GPL conventions provide a rational framework for its distribution and use
- the open availability of the source text will invite contributions by others, improving the AGFL software and the associated grammars and lexica, which will ease the maintenance problem.

It is our expectation that the availability of a parser generator for natural language parsers in the public domain will enable not only the development of many new applications, but that the good example of making the software system and the associated grammars and lexica freely available will inspire others to contribute grammars and lexica to the public domain. For computer scientists, this argument may be all too familiar, but for linguists this is a wholly new approach!

8 Acknowledgements

The AGFL formalism was first implemented between 1991 and 1996 with funding from the Dutch national research organization NWO. In the period 2000/2001, with financial support from the [NLnet foundation], the formalism has been revised in the light of experience and been brought under the GNU public licence.

Out of the many people who have contributed to the AGFL project we feel particularly obliged to Arend van Zwol, Arjan Knijff and Caspar Derksen who have spent years of their life on this elusive ideal parser generator.

9 Availability

Further information, the EP4IR grammar and all the software can be found at the [AGFL project] website.

References

[Abbott, 1983] R. J. Abbott: *Program design by informal English descriptions*; CACM 26(11), pg 882-94.

[AGFL project] `http://www.cs.kun.nl/agfl/`.

[Frederiks et al, 1996] P.J.M. Frederiks, C.H.A. Koster, and Th.P. van der Weide. Validation of Object-Oriented Analysis Models using Informal Language. Technical Report CSIR9609, Computing Science Institute, University of Nijmegen, Nijmegen, The Netherlands, May 1996. http://citeseer.nj.nec.com/frederiks96validation.html

[Graham, 1994] Ian Graham, *Object Oriented Methods*, AddisonWesley, 1994

[Koster, 1992] Cornelis H.A. Koster (1992), Affix Grammars for Natural Languages. In H. Alblas and B. Melichar, editors, *Attribute Grammars, Applications and Systems*, volume 545 of *Springer LNCS*, pp. 469-484.

[Koster et al., 1999] C.H.A. Koster, C. Derksen, D. van de Ende and J. Potjer, Normalization and matching in the DORO system. Proceedings BCS-IRSG 1999 colloquium, Glasgow University.

[NLnet foundation] `http://www.nlnet.nl/`

[Sparck Jones, 1998] K. Sparck Jones (1998), Information retrieval: how far will *really* simple methods take you? in: Proceedings TWTL 14, Twente University, the Netherlands, pp. 71-78.

[PEKING project] see `http://www.cs.kun.nl/peking`.

[Princeton WordNet] http://www.cogsci.princeton.edu/ wn/