

A Flexible Language Acquisition Tool Kit for Natural Language Processing

Svetlana Sheremetyeva

Department of Computational Linguistics
Copenhagen Business School,
Bernhard Bangs Alle 17 B,
DK-2000, Denmark
lanaconsult@mail.dk

Abstract

The paper describes a Flexible Language Acquisition Tool (FLAT) kit, - a suite of multipurpose interactive tools for developing NLP systems and/or training computational linguists. The kit facilitates the use of linguistic expertise in developing NLP applications and allows for maintaining and improving a system output without extra programming effort. It can be used for any language based on ANSI character set without reengineering and has an element providing for its integration in any NLP application, thus dramatically reducing the complexity and costs of development process.

Introduction

Speeding up acquisition of high quality linguistic knowledge resources while reducing acquisition effort and cost has always been in focus of NLP researchers and developers. A lot of work has been made in the past ten years or so to develop software whose goal is to facilitate future development of NLP systems such as, to name just a few, GATE (Cunningham et al., 2002), tools for Spanish morphological lexicons (Coni et al., 1997), a thesauri and terminology maintenance framework (Fischer et al., 1996), a syntactic analyzer toolkit (Ibrahim and Cummins, 1989). On the other hand, since increasing the size of knowledge resources or their depth does not necessarily lead to a commensurate improvement of output quality in an application it is important to determine when enhancing a resource component is no longer profitable. This would allow for a considerable reduction in acquisition time, effort and cost.

We attempt to contribute to the problem by suggesting a Flexible Language Acquisition Tool (FLAT) kit that facilitates the use of linguistic expertise in developing NLP applications and allows for maintaining and improving a system output without extra programming effort. Quick-response user interfaces make it easy for a linguist to experiment with different kinds (and sizes) of lexical and grammatical knowledge to decide on what is a must for a given application.

The current version of the FLAT kit is a 32-bit Windows application developed to run in a number of operating environments: Windows 95/98/2000/NT. It can be used for languages based on ANSI character set and includes an element providing for its integration in any other application, thus reducing the complexity and costs of development process. In this paper we only focus on selected components of the FLAT kit.

FLAT Architecture

The FLAT kit is telescopically created of the following program modules: Lexicon Creator, Tagger, Chunker, Predicate/Argument Analyzer, Transfer module, and Generator. While the first module, - Lexicon Creator, can be used as a stand-alone tool, every next tool is built by successively integrating the listed modules thus covering the main steps of uni- or multilingual NLP processing.

The most characteristic feature of the FLAT kit is that all processing modules are equipped with interpreters (compilers) with extremely user-friendly control and interactive interfaces for acquisition/updating linguistic knowledge and tracing complete or partial processing.

Lexicon Creator

Lexicon Creator is a FLAT core component and allows for flexibility in descriptions of lexical items in the format of supertags that is quite popular nowadays. For example, Joshi and Srinivas (1994) who seem to coin this term use elementary trees of Lexicalized Tree-Adjoining Grammar for supertagging lexical items. Gnasa and Woch (2002) use grammatical structures of the ontology as supertags. A supertag used in (Sheremetyeva, 2003) combines morphological and semantic information described by the following feature structure:

```
Tag  
[ POS  
[Noun  
[object[plural, singular]  
process[-ing, other[plural, singular]]  
substance [plural, singular]  
other [plural, singular]]]]]
```

Within this feature space such word as, e.g., “rotation” can be coded as “noun”, “singular” “no *-ing* ending”, “process”. Figure 1 shows a screen shot of the Lexicon Creator interface with a set of tags used in AutoPat¹. A new deeper tag “Adjective meaning shape” (AdjSh) is being added to code such words as “round”, “circular”, etc. The basic principle for this tool is that the user can easily update both the list of lexemes and their descriptions making the latter as “shallow” or “deep” as required. Among other functionalities of Lexicon Creator are multi-parameter search and import functionalities. The user can import any lists of words from external text files and tag them all, in groups or individually. In addition to that the tool is pipelined to the tagger, so as to automatically import new words after tagging.

¹ AutoPat is an authoring system for patent claims (Sheremetyeva, 2003).

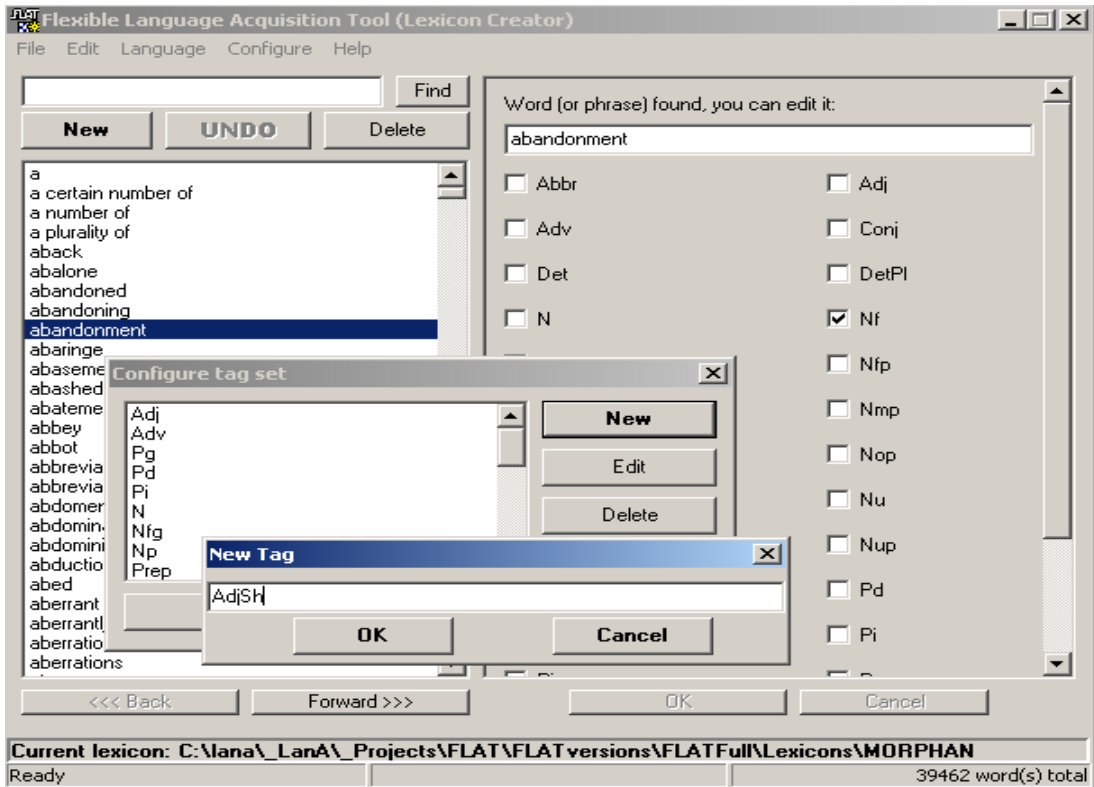
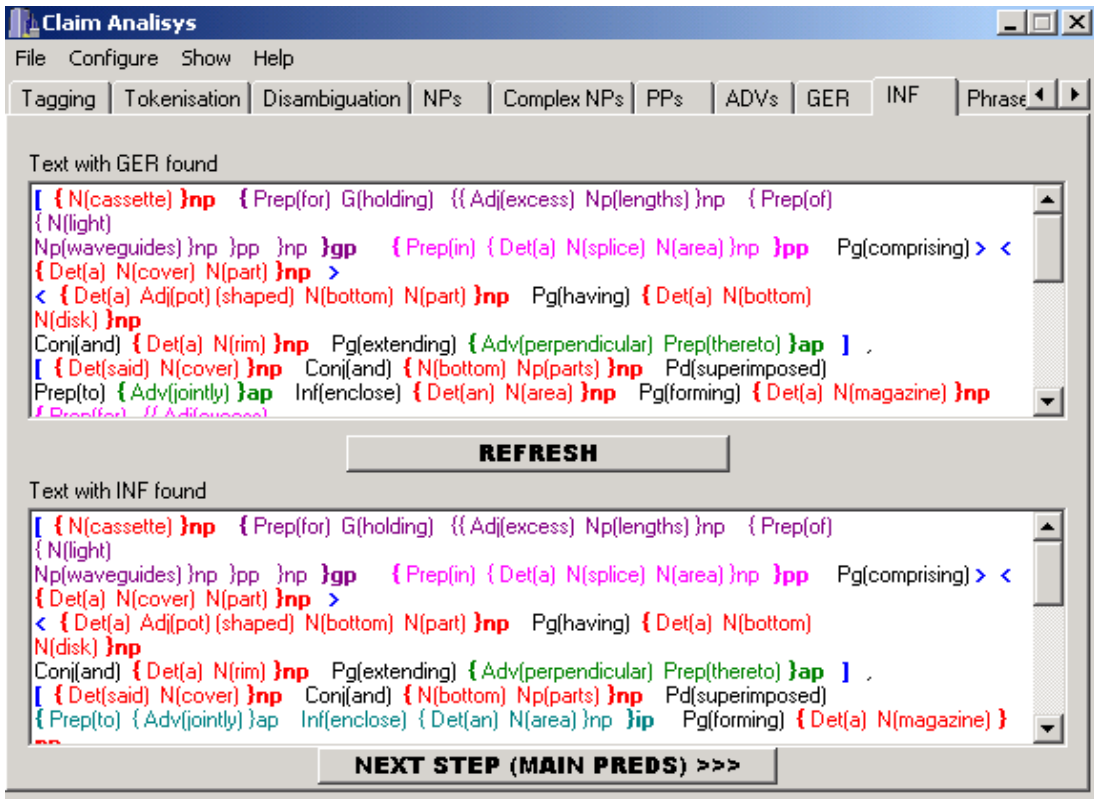


Figure 1. A screenshot of the Lexicon Creator interface.



Picture 3. A screenshot of the Chunker interface

Analysis tools

FLAT analysis tools include Tagger, Chunker and Predicate/Argument Analyzer. Figures 2 shows a screen shot of the interface for tracing analysis process at different phases, -tagging, disambiguation and chunking. It contains the main menu, a set of bookmarks for different processing steps and a control screen divided into two windows with instruction buttons under each of them. The lower window of the screen traces a certain analysis phase, the upper window displaying the analysis trace of the previous phase, which makes it convenient to spot any errors. At the set up stage the upper window is empty and interactive. The user can either download a text from an external file or type it directly into the upper interactive window. It is possible to save traces of individual processing stages in different text files. Processing traces are presented in a very illustrative form, e.g., in addition to brackets different colors are used for every type of a chunk.

Tagger assigns and disambiguates supertags of a selected lexicon. It is possible to perform the tagging of the same text based on different lexicons to decide on the right one for a given application. The tagger reports immediately whether the text is covered by a lexicon listing unknown words (if any) in a pop-up window. Multiple tags are disambiguated according to the disambiguation rules written by the developer in Disambiguation Rule Interpreter. Any changes that might be made in a lexicon or in disambiguation rules are immediately traced in the control interface thus allowing for operative testing. Tagger can directly be used for lexical acquisition from relevant corpora due to

its functionality to import unknown lexemes to a FLAT lexicon.

Chunking is carried out by matching the strings of supertags against patterns in the right hand side of the rules written by the developer in the Chunking Rule Interpreter. The chunking procedure is a succession of processing steps itself starting with the simple-noun-phrase procedure, followed the complex- noun-phrase procedure, which integrates simple noun phrases into more complex structures (those including prepositions and conjunctions). Then the prepositional-, adverbial-, infinitival- and gerundial-phrase procedures switch on in turn. The order of the calls to these component procedures in the chunking algorithm is established to minimize the processing time and effort. The ordering is based on a set of heuristics.

Predicate/Argument Analyzer searches for all possible predicate-pattern matches over the “residue” of “free” words in a chunked text and returns flagged predicates. It then retrieves semantic (case-roles) and syntactic dependencies (such as syntactic subject), requiring that all and only dependent elements (chunked phrases in our case) be present within the same predicate structure. The tool including Predicate/Argument Analyzer requires a FLAT predicate lexicon with deep description of lexical units including syntactic and semantic knowledge about predicates and their arguments (Sheremetyeva, 1999). A sample piece of such knowledge tuned to a patent domain is provided with the FLAT programs. The developer can reuse a lot of universal knowledge in the lexicon and Predicate/Argument Rule Interpreter and edit it as needed.

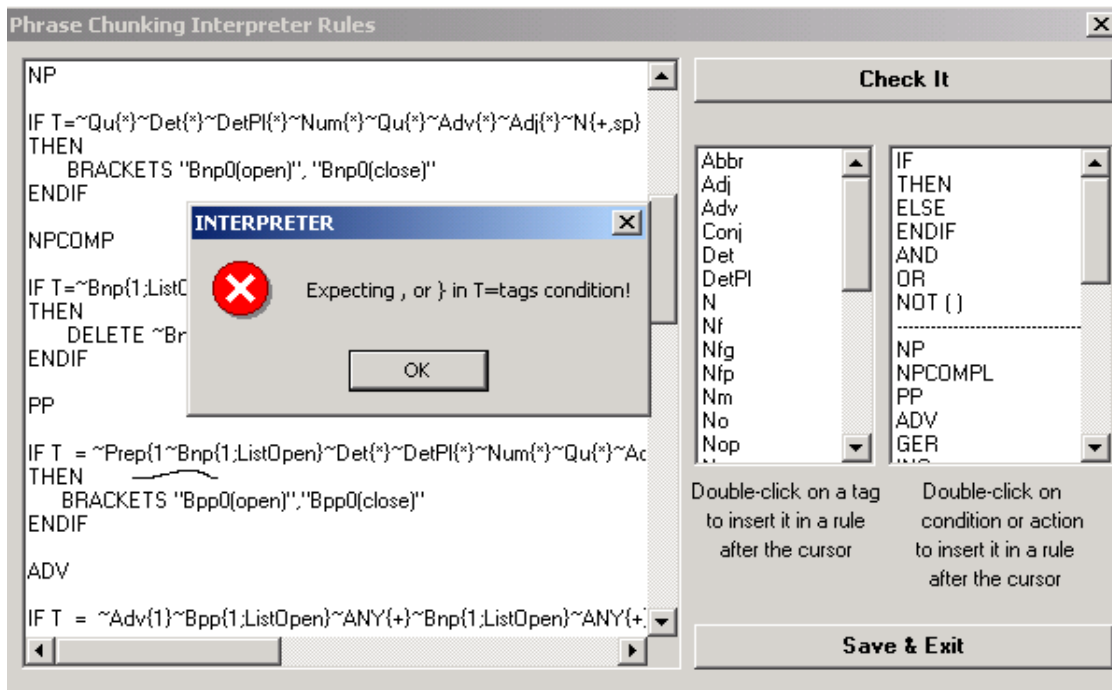


Figure 3. A screenshot of the chunker compiler interface.

Compilers

The most characteristic feature of the FLAT kit is that all processing modules are equipped with interpreters (compilers) with extremely user-friendly interactive interfaces for acquisition or updating grammar knowledge. A developer can write or update rules underlying a certain processing phase in a very simple IF-THEN-ELSE-ENDIF formalism.

The rules can use a 5-step window context with the lexeme, tag or phrase in question in the middle. The conditioning knowledge can be rather rich. It includes the knowledge specified in the lexicons and the knowledge acquired at preceding processing steps. Immediately after saving new rules an updated trace can be displayed in the control interface. The rules that can be written by a developer in every interpreter are tuned to a particular FLAT lexicon. There might be several sets of rules based on the same or different lexicons. The linguist is thus free to experiment with different kinds (and sizes) of lexical and grammatical knowledge.

Figure 3 shows the interface of the phrase-chunking compiler (all other compilers have similar functionalities). The right pane is a type-in area. The right pane is designed to support rule writing. It contains two clickable menus, the menu of tags and the menu of expressions used in the rule formalism. A double-click on any of the selections transfer it into the text of a rule. The tool would not let the user leave a compiler without the "Check it" button click. In case of a mistake an error message appears with the cursor in the place where a correction should be made. This, on the one hand, makes the work less difficult and time consuming and, on the other hand, controls rule consistency and correctness of the formalism.

Conclusions

We have presented the FLAT kit, - a suite of multipurpose interactive tools for developing NLP systems and/or training computational linguists. A rich feature space that can be created with the kit tools allows for quite a good performance in solving the most difficult NLP problems such as, recovery of empty syntactic nodes, long distance dependencies, disambiguation of PP attachment and parallel structures, etc.

FLAT has been successfully used as part of developer environment for AutoPat (authoring system for patent claims) and is now being used for updating the AutoPat knowledge for AutoTrans (an MT system for patents).

The kit might be useful, for example, for creating tagged corpora due to its pipelined Lexicon creator and Tagger and flexible user-friendly import and tagging functionalities.

FLAT can also be used in teaching computational linguistics disciplines due to its "glass-box" nature as the "locked" character of many NLP commercial products does not make it possible to use them to the most advantage of students (Balkan et al., 1997). The author has a positive experience of using FLAT as training software when Teaching MT courses at Copenhagen Business School (Denmark) and Southern Ural State University (Russia) on the material of

English, Danish and Russian languages. One of the lab tasks, for example, was to acquire linguistic knowledge necessary to translate a training suite of sentences. With FLAT students had a hands-on experience in creating a lexicon for a given text, deciding on the depth of description to resolve ambiguities, learning how to write formal (programmable) grammar rules in FLAT compilers. They could do it in two languages, - English and their native (Danish or Russian), thus learning more about universal and contrastive features of natural languages, etc. Quite an intensive linguistic training was possible due to the fact FLAT is targeted to solving linguistic problems of NLP and do not require programming skills.

References

- Balkan L., D.Arnold and L.Sadler. (1997) Tools and Techniques for Machine Translation Teaching: A Survey.
http://clwww.essex.ac.uk/group/projects/MTforTeaching/index_1.html
- Cunningham H., D.Maynard, K.Bontcheva, V Tablan, and C.Ursu. (2002). The GATE User Guide.
<http://gate.ac.uk>
- Coni, J.M., J.C.Gonzalez, and A.Moreno (1997). ARIES: A Lexical Platform for Engineering Spanish Processing Tools. *Journal of Natural Language Engineering*, 3(4).
- Fischer, D., W.Mohr, and L.Rostek (1996). A Modular, Object-Oriented and Generic Approach for Building Terminology Maintenance Systems. In TKE'96: *Terminology and Knowledge Engineering*. Frankfurt.
- Gnasa M., and Jens Woch. 2002. Architecture of a knowledge based interactive Information Retrieval System.
<http://konvens2002.dfki.de/cd/pdf/12Pgnasa.pdf>.
- Ibrahim, M.H. and F.A Cummins (1989). TARO: An Interactive Object-Oriented Tool for Building Natural Language Systems, In *IEEE Intelligence*. Los Angeles.
- Joshi A.K., and Bangalore Srinivas (1994). Disambiguation of Super Parts of Speech (or Supertags): Almost Parsing.
<http://acl.ldc.upenn.edu/C/C94/C94-1024.pdf>
- Sheremetyeva S. (2003) Towards Designing Natural Language Interfaces. *Proceedings of the 4th International Conference "Computational Linguistics and Intelligent Text Processing"* Mexico City, Mexico, February 16-22.
- Sheremetyeva, S. (1999). A Flexible Approach To Multi-Lingual Knowledge Acquisition For NLG. 1999. *Proceedings of the 7th European Workshop on Natural Language Generation*. Toulouse. (France) May 13-15.