

LFG Parsebanker:

A Toolkit for Building and Searching a Treebank as a Parsed Corpus

Victoria Rosén, Paul Meurer and Koenraad de Smedt
University of Bergen and Unifob AKSIS

TREPIL

Summary: We present the LFG Parsebanker, a comprehensive toolkit for interactive incremental construction of a treebank as a parsed corpus. This [web-based](#) toolkit offers an environment for batch and interactive parsing, versioning, inspection of structures, discriminant-based disambiguation, and statistics. It also has a structural search facility with innovations compared to TIGERSearch.

Overview

In the TREPIL project we have developed methods and tools for the semi-automatic construction of treebanks as parsed corpora. Our approach to treebanking aims for efficiency by combining automatic parsing and computer-assisted manual disambiguation. Parsing is implemented in XLE into which any compatible LFG grammar can be uploaded.

Through the automatic identification of [discriminants](#), the manual work can remain focused on simple choices even if annotations can be detailed and complex. This method has been successfully applied in previous research contexts, while we have adapted the use of discriminants to LFG structures.

We present an updated version of the LFG Parsebanker, a comprehensive toolkit that provides the user with automated services such as treebank navigation, computation of discriminants, structural search, versioning, etc. The toolkit can be used through a web browser and offers the following webpage views:

1. A [treebank overview](#) page provides corpus selection and navigation in the selected corpus. A partial screenshot from the overview page is shown. This view lists all sentences in the corpus together with information such as the sentence ID, the number of parse solutions, the number of discriminants chosen out of the number of total discriminants, the number of chosen analyses, and the sentence length. The annotator may choose to display additional information; among the possibilities are parse time, version information, whether the annotation process is finished, and annotator comments. This view also offers sorting of sentences depending on their various properties.

Id	Solutions	Disc.	Chosen	Words	Versions	Sentence
1	4	1/44	1	8	1.1	Sofie Amundsen var på vei hjem fra skolen.
2	6	2/114	1	9	1.1	Det første stykket hadde hun gått sammen med Jorunn.
3	12	3/201	1	5	1.1	De hadde snakket om roboter.
4	2+14	1/26	1	11	1.1	Jorunn hadde ment at menneskets hjerne var som en komplisert datamaskin.
5	7	2/183	1	10	1.1	Sofie var ikke helt sikker på om hun var enig.
6	64	6/256	1	10	1.1	Et menneske måtte da være noe mer enn en maskin?
7	16	4/178	1	8	1.1	Ved det store matsenteret hadde de skilt lag.
8	30+30	3/91	1	17	1.1	Sofie bodde i enden av en vidstrakt villabebyggelse og hadde nesten dobbelt så lang skolevei som Jorunn.

2. [XLE-Web](#) is an interface to the XLE parser on a web page. This interface presents the parse results for a single sentence. It includes a display of packed structures and shows the computed discriminants. The user can disambiguate the sentence by selecting or rejecting discriminants and thereby retaining or rejecting sets of corresponding analyses.
3. The [parsebanking](#) page offers views and disambiguation as in XLE-Web, but also additional parsebank management operations, such as subcorpus selection. This page also contains a search window.
4. The [discriminant statistics](#) page presents a frequency list of chosen discriminants for a subcorpus. Each discriminant is listed with its type, the number of times it has been chosen (i.e. marked as good) and the number of times it has been rejected (i.e. marked as bad).
5. [Administration](#) and user administration pages.

Most of these components are implemented in Common Lisp and use XML, XSLT and Javascript to serve the interface web pages. C-structure trees (and graphs) are drawn using Scalable Vector Graphics (SVG) and MySQL is used to store the parsebank, although the system is also compatible with other databases.

LFG Searchtool

The current version of the LFG Parsebanker offers a new advanced search facility that allows the user to [search for structural characteristics in both c-structures and f-structures](#). This search tool is modeled after TIGERSearch, which has been reimplemented and extended to cover f-structures, which are not trees, but directed graphs, possibly with structure sharing and cycles. It has a query language that is suitable for c- and f-structures and is more convenient to use than the generic TIGERSearch syntax, to the extent that a graphical interface is not deemed necessary.

In contrast to the TIGERSearch system, whose treebanks and search indices are static, our system allows for [dynamic index updating](#), such that the index always represents the current state of the dynamic treebank. The index data is stored in a database, but in addition, a representation of the index is kept in memory for fast querying.

The query language

For many common query types it is a problem that they are complicated (or impossible) to express or that they perform unsatisfactorily when using the basic TIGERSearch syntax. Therefore, an abbreviated node description syntax has been devised, and special search language constructs have been implemented that translate to efficient code. Among the new constructs are:

- An [abbreviated syntax](#) for c-structure nodes: inner nodes can be specified using the bare node label, whereas leaf nodes are represented by the quoted surface string. For example, query (1) matches all configurations where an NP node dominates a surface node "barna":
(1) NP > * "barna"

- A [rule operator](#) '→' to search for c-structure configurations:
(2) NP → N PP
(3) NP → (N #x:PP) & #x → (. P NP .*)

Query (2) matches all NPs having exactly an N and a PP child, in that order. Query (3) matches the same configuration, where in addition the PP node should have a P node and an NP node as second and third children. The left hand side of the rule operator can be a regular expression over node labels.

- A [path operator](#) '>(...)' to search for paths through f-structures (coded as regular expressions over attributes). Examples:

- (4) #f >(COMP TNS-ASP TENSE) "past"
- (5) #f₁ >(TOPIC & OBJ) #f₂
- (6) #f₁ >((COMP | XCOMP)+ (OBJ | OBJth)) #f₂

Example (4) searches for a simple path from an f-structure to an atomic value, whereas (5) shows a query involving structure sharing: the f-structure #f₂ should be the value of both the TOPIC and the OBJ attribute of #f₁. Functional uncertainty can be expressed using the Kleene star or plus operators, as illustrated in (6).

- A [projection operator](#) '>>' relating c-structure nodes and f-structures they project to:

- (7) NP >> #f

This query finds all NP nodes and the f-structures they project to. The projection operator can be combined with the path operator:

- (8) #c >>(OBJ PRED) "vei"

The query interface

Queries can be input both on the overview page and on the sentence page. Long-running queries may be stopped at any time, and a list of links to matching sentences is updated dynamically while a query runs. Matches can be inspected by clicking on a sentence number in the match list. In the c-structure, matching nodes are highlighted, whereas in the f-structure, matching paths are shown.

The screenshot shows the search interface for the sentence "Ilden" ser vi jo. The query entered is "#x >(TOPIC & OBJ) #y". The interface displays 3 matches: #577, #672, #927. The selected match is #927. The c-structure tree shows the sentence structure with nodes for QUOTE, NP, QUOTE, Vfin, and S. The f-structure graph shows the relationships between nodes, with TOPIC and OBJ highlighted in red. The interface also includes a list of discriminants and a comment on the sentence.

The detailed view shows the c-structure tree and the f-structure graph. The c-structure tree has a root node IP, which branches into QUENOM and I'. QUENOM branches into QUOTE and NP. NP branches into N, which is the word "ilden". I' branches into Vfin and S. Vfin is the word "ser". S branches into PRON and ADVprt. PRON is the word "vi" and ADVprt is the word "jo". The f-structure graph shows the relationships between nodes, with TOPIC and OBJ highlighted in red. The TOPIC node is associated with the value "ilden" and the OBJ node is associated with the value "jo".

The screenshot illustrates a view of the parsebanking page with the results of a search. The treebank "sofie" has been searched with the query in (5) above, which specifies that the same f-structure value must fill both the TOPIC and the OBJ functions, in other words that the sentence has a [topicalized object](#). This treebank has three matches for the query, listed by their sentence ID numbers under the query window. The sentence displayed is "Ilden" ser vi jo. ("The fire", we see after all.) The attributes that share the same value are highlighted in the f-structure display.

The toolkit is [language independent](#) and can be used with any LFG grammar implemented in XLE. It has been licensed to several members of the Parallel Grammar Project.