LiveTree: An Integrated Workbench for Discourse Processing

Gian Lorenzo Thione, Martin van den Berg, Chris Culy, Livia Polanyi
FX Palo Alto Laboratory
3400 Hillview Ave, Bldg. 4
Palo Alto, CA 94304
{thione|vdberg|culy|polanyi}@fxpal.com

Abstract

In this paper, we introduce LiveTree, a core component of LIDAS, the Linguistic Discourse Analysis System for automatic discourse parsing with the Unified Linguistic Discourse Model (U-LDM) (X et al, 2004). LiveTree is an integrated workbench for supervised and unsupervised creation, storage and manipulation of the discourse structure of text documents under the U-LDM. The LiveTree environment provides tools for manual and automatic U-LDM segmentation and discourse parsing. Document management, grammar testing, manipulation of discourse structures and creation and editing of discourse relations are also supported.

1 Introduction

In this paper, we introduce LiveTree, a core component of LIDAS (the Linguistic Discourse Analysis System) for automatic discourse parsing with the Unified Linguistic Discourse Model (U-LDM) (Polanyi et al, 2004). The U-LDM is a theory of discourse structure and semantics that has as its goal assigning the correct interpretation to natural language utterances.

1.1 Overview of LiveTree

LiveTree is an integrated workbench for supervised and unsupervised creation, storage and manipulation of the discourse structure of text documents under the U-LDM. LiveTree does not support speech, dialog or interaction annotation (Bernsen et al. 2002, 2003 and over view of systems in Bernsen et al. 2002). The LiveTree environment provides tools for manual and automatic U-LDM segmentation and discourse parsing. Like RSTTool, LiveTree provides support for segmentation, marking structural relations among segments, and creating and editing discourse relations (O'Donnell 2003). Similar to the D-LTAG system described in Forbes et al (2003) LiveTree is an experimental discourse parser implementing a theory of sentential and discourse relations. However, LiveTree is also a complete document handling and manual and automatic discourse parsing system. Various applications are supported as web services. Accordingly, LiveTree serves as both the user interface and theory development environment for PALSUMM, a text summarization system built on top of LIDAS (See Section 5 below) In this paper, we describe the resources LiveTree workbench provides for discourse level theoretical development as well as document handling, manual and automatic text annotation and parsing.

1.2 LiveTree Functionalities

LiveTree's Java architecture shown in Figures 1 is modular and highly extensible. LiveTree is made up by: (1) a Model Manager which provides interfaces for manipulation, storage and retrieval of actual documents and discourse representations; (2) a Module Manager, which handles and provides access to the main GUI and to all installed modules; and (3) a Service Manager providing a polling interface for all active LiveTree Services. Manager components rely on stubs which can be implemented and extended from outside the framework's core.

The LiveTree Module Manager and all installed LiveTree Modules lie on top of a general GUI Layer handling the main LiveTree window, which includes a menu bar, a tool bar, a status bar and four docking areas. The status bar is used for messages to the user and notification of status for asynchronous services. The menu bar

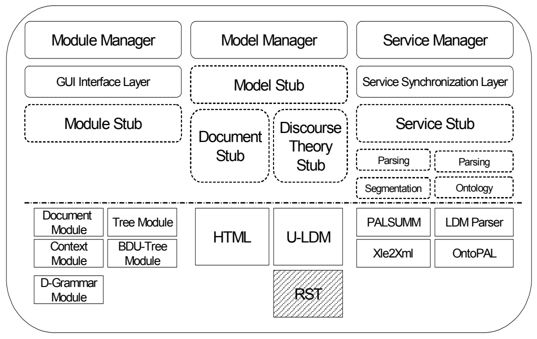
	- ***Framework/Architectural*** o Module Manager Architecture with independent dockable floating windows for each independent module o Service Manager with multi-threaded support for asynchronous services, notification management and SOAP/web-service support o Generalized Model Manager for support of multiple discourse theories/models o Scripting Engine for automatic batch execution of actions and commands
- ***Default Modules*** o Document Module; BDU-Tree Module; DPT Module; Search Module; Content Module; Grammar Module; Segments Module.	- ***Default Services*** o Discourse Parsing Service; U-LDM Model Persistence Service; Xle2Xml Syntax Parsing Service; Discourse Segmentation Service; OntoPAL

Figure 1: LiveTree Architecture.
Core system and several implemented modules and services

and the toolbar allow rapid access to general functions and to module-specific actions, including hiding and showing module windows. Every module is assigned a window which can be resized, docked or hidden/shown.

When multiple modules are docked in the same docking area they are arranged in a tabbed interface which allows easy access and maximizes display real-estate. Finally, the GUI Layer administers contextual pop-up menus in a general, module-independent fashion: any module can register a number of actions bound to a specific context (e.g. a sentence, a node, a sub-tree, etc.) and at the user's request, the GUI Layer polls the Module Manager for appropriate actions from every installed module. LiveTree's clean and intuitive interface is independent of the specific modules installed and allows for seamless integration of custom modules not part of the current implementation.

Table 1 gives a comprehensive overview of LiveTree features as well as identifying the modules or services that provide them.

2 Document Handling

The Model Manager (MM) is the main access point to models, defined as the synchronized unions of a document and its (annotated) discourse structure. The MM requires that appropriate LiveTree Services provide functionalities needed for persistent storage and retrieval of annotated documents. As long as documents are not modified externally, their discourse representations can also be retrieved from a persistent XML format encoding U-LDM tree structure, visualization parameters, surface and deep node content along with other user-defined annotations.

The Document Module (DM) enables full document creation, modification and annotation at the document, region/selection, and sentence level. The DM provides the visual representation of an HTML document[1] and preserves the text organization, formatting, and non-textual information (figures, tables, etc.) of HTML source documents. The DM also provides visual feedback capabilities including highlighting and hiding/showing sections of documents. The Document Stub Interface provides the mapping between a document's content and notions of paragraphs, sentences, units and spans. In the current implementation, a document is divided in paragraphs according to standard notions; paragraphs are then tokenized in sentences using simple heuristics and sentences are segmented into Basic

[1] Currently, only HTML document formats are supported. Other data formats can also be supported by implementing the Document Stub Interface (DSI) appropriately,

Feature	Description	Modules & Services
Document Handling	Support for HTML Documents (Import, Export, Create, Edit, Print, Tokenize in sentences)	Document Module
Discourse Segmentation (Automatic & Manual)	Support for LDM Discourse Segments (Automatic Sentence Segmentation; Manual Editing of Segments; Manual Sentence Segmentation; Inspect Segments' Syntax Content)	BDU-Tree Module, Content Module, Xle2Xml Syntax parsing Service, Discourse Segmenter
Discourse Structure Creation *Document and Sentence Level* (Automatic & Manual)	Support for LDM DPT and BDU Trees (Automatic Discourse Parsing; Sub-tree Attachment via Drag 'n Drop; Editing including Node Type Editing and Content Editing; Node/Sub-tree Removal; Node-Specific Notes Editing; Expand/Collapse Sub-Trees; Export to JPG; Printing; Extensible Semantic Composition)	BDU-Tree Module, DPT Module, Content Module, Notes Module, Discourse Parsing Service
Semantic Content Inspection	Support for Feature Structure-like Semantic Content of LDM Nodes (Node Specific via mouse selection; F-Structure graphical view; In Place Editing; Grammar Condition Querying)	BDU-Tree Module, DPT Module, Content Module
Search	Full Text and RE search on: Document content, Node Surface Content, Nodes Semantic Content, Node-Specific Notes ; Online retrieval of matching sources	Search Module, Document Module, DPT Module
U-LDM Rule Editing	Grammar Editor: reusable conditions; easy-to-use GUI	Grammar Module
Discourse Grammar Testing (Manual & Scripted)	Support for Manual Grammar Testing (Check for rule enablement between attachment point and M-BDU selected from actual subtrees; Support Scripted Testing with XML Based Language)	Grammar Module, DPT Module, Discourse Parsing Service
Persistence Support	Implements and supports serialization and deserialization in LiveTree XML format of LDM Annotation for documents.	LDM Persistence Service
Other Functionalities	Tree Structure Zooming and Panning, Print Preview Functionalities, Copy/Cut/Paste for text and trees)	Tree Module, Document Module, BDU-Tree Module

Table 1: Overview of LiveTree features and the modules or services that provide them.

Discourse Units following the U-LDM discourse segmentation conventions discussed below.

3 LiveTree Support for Discourse Annotation

The LiveTree Workbench supports manual and automatic, supervised and unsupervised annotation practices for each step in the analysis process. In addition, our default implementation includes a completely integrated interface for writing, testing and debugging U-LDM Discourse grammar rules which are used for automatically constructing the discourse representation for individual sentences and entire texts.

3.1 U-LDM Parsing Steps

The U-LDM specifies rules both for segmenting sentences into Basic Discourse Units (BDUs) and then for combining the BDUs into an Open-Right Discourse Parse Tree (DPT) that captures structural relations among constituent structures. The U-LDM discourse parsing process can be summarized as follows:

- Identify potential Basic Discourse Units (BDUs) within sentence from output of analysis of sentence documents from the Xerox Linguistic Environment (XLE) Lexical Functional Grammar (LFG) parser using sentence-segmentation rules.
- Construct a set of Open-Right BDU-trees representations which map onto top-level coordinated structures within the sentence, using syntactic information from the XLE parse and sentential discourse rules to identify the relationships among BDUs.

- Attach the BDU-trees, each one as a single unit, to the DPT by computing the relationship between the node corresponding to the root of a BDU-Tree to accessible DCUs aligned along the right edge of the DPT using rules of discourse relations. There are 3 possible macro-types of relation:
  Coordination: new unit continues development of previous unit
  Subordination: new unit provides additional information about previous unit
  N-ary: new unit bears a special logical, rhetorical or genre based relationship to previous unit
- Once a BDU-tree is attached, its leaves become terminal nodes of the DPT and nodes on its right edge become therefore accessible for attachment in the next iteration of the process.

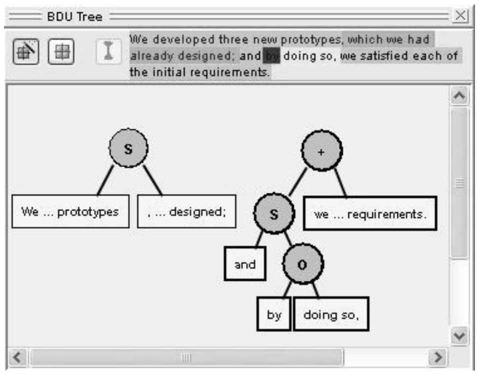
3.2 Live Tree Modules for U-LDM discourse annotation

Live Tree Modules (LTM) provide extensive manual and automatic capabilities for annotating documents with U-LDM discourse tags. They are local to the framework and provide user-directed functionalities, relying on mutual interaction through the LiveTree GUI. Two modules in LiveTree's current implementation contribute primarily to discourse annotation (besides the DM): the BDU-Tree Module and the DPT Module.

3.2.1 Discourse Segmentation

A critical task for U-LDM analysis is to account for the availability for update of appropriate discourse contexts or subcontexts introduced in earlier text. Thus, discourse segmentation under the U-LDM requires the identification of discourse units within the sentence that can function as possible attachment points as well as segmenting sentential units and non-sentential structures such as titles from other units. The U-LDM may match incoming discourse utterances with target contexts which are in syntactically subordinated positions within a previous sentence. In order to construct the appropriate representation of the rhetorical or semantic structure of discourse we must therefore keep sub-sentential units available for attachment at independent nodes along the right edge of the DPT.

For discourse segmentation, the U-LDM depends on the syntactic analysis of constituent sentences. Initially, sentences are divided up into discourse segments reflecting syntactic encodings of minimal units of meaning or function. Subsequently, some segments are identified as Basic Discourse Units (BDUs). Only those discourse segments that are of a type that can be independently continued are BDUs. Operator segments are one example of non-BDU segments. Gerunds, nominalizations, auxiliary and modal verbs or clefts are verb based constituents but not segments because they do not independently establish an interpretation context for update by subsequent units (Polanyi, 2004).



We developed three new prototypes, which we had already designed; and by doing so, we satisfied each of the initial requirements.

Figure 2: A segmented sentence and the BDU-Trees corresponding to its two coordination-chunks.

In LiveTree, the BDU-Tree Module shown in Figure 2 provides the visual interface and annotation tools for sentence segmentation. The top section of the BDU-Tree window is composed of two areas: a small toolbar, and the sentence/segment viewer. A simple toggle-button interface allows the user to select between automatic or manual segmentation. In automatic mode, an external Segmentation Service (part of LIDAS) is polled and a set of segments retrieved. Segments are automatically colored, and segments embedding other segments are represented by non-contiguous spans of text associated by the same highlighting color. In manual mode segmentation is performed by dragging the divider (the rightmost button in the toolbar) to the desired span boundaries and, if necessary, assigning non-contiguous spans to the same segment using drag-n-drop.

3.2.2 BDU Tree Construction

In LIDAS operating in automatic mode, BDU-Trees are constructed from segmented sentences by mapping the LFG f-structure representations of sentential syntax produced by the XLE onto appropriated sentence-level discourse attachments. The resulting structure is a BDU-Tree, a DPT of an individual sentence. Although automatic BDU-Tree parsing can only be performed on automatically generated segments, LiveTree supports manual construction of BDU-Trees regardless of how segmentation occurred.

In manual mode, segments can be dragged from the Sentence Viewer area to the bottom section of the window. When dropped, these become BDU nodes and the content of the node can be manually annotated. To create the relationship between two nodes the user drags one node over the other as attachment point and selects a preferred relation from a pop-up menu. If syntactic/semantic annotations are present they are correctly percolated and composed throughout the BDU-Tree.

BDU Trees can be easily edited and manipulated for correcting or changing annotations, and for improving results generated by automatic BDU-Tree parsing. Nodes can be removed, their associated annotations inspected and modified, and the type of relation node changed. When the type of a relation node is changed, the annotations of all nodes dominating the changed relation are updated and the correct syntactic/semantic information percolated through the tree in accord with the new relation type. Nodes and whole sub-trees can be detached and reattached at a different point using simple mouse gestures.

3.2.3 Discourse Parse Tree Construction

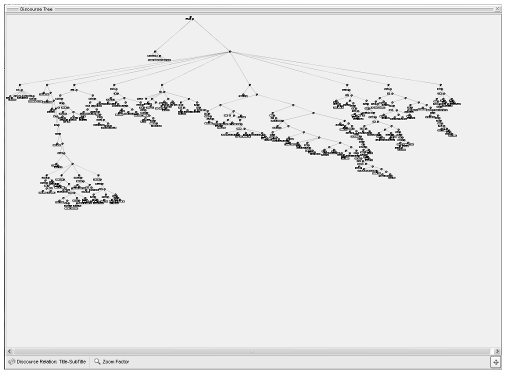
U-LDM discourse parsing is a three step process: (1) segmentation, (2) BDU-Tree Parsing, and (3) DPT parsing. LiveTree supports automatic and manual modes at all three stages enabling multiple annotation scenarios.

For example, users can segment and annotate a document entirely by hand, or, alternatively, rely on automatic segmentation and BDU-Tree parsing while manually completing the more error-prone stage of DPT parsing. Another option is to bootstrap the annotation at every stage using LiveTree automatic resources and then manually correct mistakes and undesired choices (supervised mode). A Discourse Segmentation Service and a Discourse Parsing Service using two separate discourse grammars provide automation. The user interfaces of the BDU-Tree Module and of the DPT module allow for manual and supervised annotation.

3.3 Discourse Relations under the U-LDM

Automatic DPT parsing is rule based. Lexical information (synonym, antonym, hypernym, discourse connectives), semantics (involving genericity, modality, cardinality, temporal interpretation etc.), and syntactic information (including topicalization, grammatical function promotion/demotion, etc.) are used by weighted ordered discourse grammar rules to determine both the site of attachment and the relationship obtaining among the nodes. Rules may combine different sources of evidential information. LiveTree provides a complete rule development and testing environment used for both theoretical investigation and automatic parsing.

When a BDU-Tree is available for attachment, linguistic information available at DCUs along the right edge of the DPT is compared with evidence retrieved from the incoming BDU-Tree to identify semantic information that acts as an “anaphoric anchor” for information in the incoming BDU-Tree by examining the content of the root node (M-BDU). Each attachment rule is checked against information available at the M-BDU and at the available DCUs and an ordered set of attachment sites and associated relations, as specified by the winning rules, is generated. Local semantic, lexical and syntactic information is percolated up through the tree according to the constraints of the discourse relations at each dominating node.



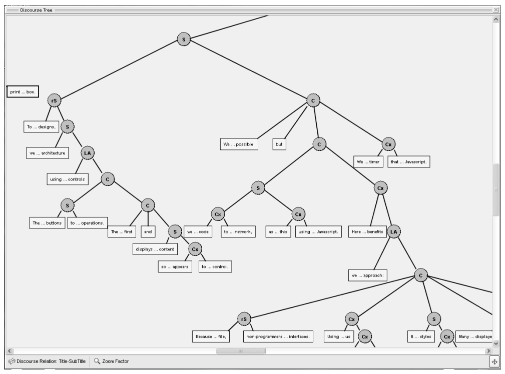


Figure 3: Two views of a document's discourse structure. Trees and subtrees can be modified, rearranged and moved through simple drag 'n drop operations.

If multiple attachments are possible, ambiguous parses ordered by likelihood are generated. In LiveTree operating in automatic mode, the system proposes a preferred structure. Dispreferred structures can be obtained by operating in supervised mode.

3.4 The DPT Module

The DPT Module shown in Figure 3 provides the visual representation and manipulation interface for U-LDM Discourse Trees. Advanced viewing capabilities help the user analyze large complex discourse structures: sub-trees can be collapsed and expanded; zooming and panning capabilities and fit-to-page printing are fully supported. Trees and sub-trees can be moved, rearranged, and removed with the same editing functions available as in the BDU-Tree Module. In addition, automatic layout capabilities enable even the most graphically complex discourse structures to be displayed clearly.

4 Discourse parsing with LiveTree

In order to create an DPT, a user can work in different modes. In Fully Automatic (Unsupervised) mode, the user simply selects a document for full processing. The document is tokenized, each sentence is automatically segmented, and passed to the parser. The discourse parsing service automatically creates BDU-Trees from each sentence and as trees are created they are attached to the emerging DPT. The user can then revise the structure and make changes[2]. In Incremental Automatic (Supervised) mode, the user is prompted for corrections at selected stages of the process. For example, after a sentence is selected by the system for processing, automatically segmented[3], and parsed into BDU-Trees, the user can rearrange nodes, change relationships between nodes, and if necessary, even merge multiple BDU-Trees into one. The BDU-Tree(s) might then be automatically attached to the DPT and the user prompted again to correct any mistakes. When the parsing process is supervised in this way, the number of overall mistakes is often reduced because attachments occur on incrementally checked structures thus maintaining the correct open right edge at all times.

Finally, in Manual DPT Parsing mode, BDU-Trees can be dragged from the BDU-Tree module to the DPT module and manually attached to the DPT however the BDU-Trees were computed. The decision of how to combine manual and automatic processing is made by the user.

[2] For large documents problems often arise as parsing mistakes build on themselves as the right edge changes and large structures are harder to examine and manipulate.

[3] Optionally the user can correct any segmentation mistake at this stage, though this interrupts the automatic mode and the attachment of the sentence must be completed manually, since the necessary syntactic information is no longer attached to the segments. Of course, this information which was likely to have been incorrect anyway, thereby necessitating correcting the segmentation.

4.1 Discourse Grammar Writing and Testing

LiveTree incorporates facilities for writing, accessing and testing discourse grammars both at the sentence and at the document level. Rules are edited via a dialog window which allows the user to create new rules by reusing macros and conditions previously used in other rules. Access to all defined types of discourse relations is permitted and it is easy to set priorities and preempting relations among existing rules.

Rules are tested in two ways. In scripted mode, testcase files are written specifying exemplary sentences and the discourse rule(s) to be tested, along with the expected outcome. This way, several rules and several testcases can be tested automatically at once. A report is created at the end of the process with information about the outcome of the tests. In manual mode, a rule can be selected for testing from the Grammar Module and a node or subtree from the DPT can be dragged on a candidate attachment site. The parser attempts to make an attachment using the selected rule and reports the result to the user. This mechanism has proven very useful during grammar creation providing important information to understand why expected structures are not created by the parser.

5 The PALSUMM Text Summarizer

The global discourse trees resulting from U-LDM parsing in the LiveTree Environment are used for text summarization in the PALSUMM System. PALSUMM is a hybrid sentence extraction system that uses conventional statistical methods to identify important information in a text and then marks for extraction those discourse segments in the DPT that are necessary in order to provide context for reference and proper resolution of anaphors. The goal of PALSUMM Summarization is to produce high quality readable summaries. We have tested our summarization methods using both manually annotated and automatically created U-LDM structures of Technical Reports taken from the FX Palo Alto archive of over 300 reports in more than 10 domains of computer science. These reports vary in size from a few to thirty or more pages. All 300 reports have been automatically summarized. Initial results, though hardly perfect, are encouraging.

6 Conclusion

LiveTree is a powerful and extremely flexible workbench for discourse level NLP annotation and parsing tasks. Throughout the design and implementation of LiveTree, our goal has been to support a full range of work-practices and to make sure that annotation steps were integrated in an intuitive and seamless fashion. Services and modules make use of available resources efficiently and interoperate unobtrusively. New functionalities can be easily added on top of existing ones and the service-oriented LiveTree architecture enables concurrent and asynchronous services to be executed locally or remotely as automatically generated web services. Working in LiveTree has proven very efficient without waste of user’s time. For example, a document can be parsed automatically in the background while other tasks such as manual annotation, grammar writing or testing are performed. While LiveTree has been designed an implemented as a workbench for U-LDM analysis, many of the features and aspects of the architecture could be adopted for use with other analytic frameworks.

References

Bernsen, N. O., Dybkjær, L., and Kolodnytsky, M.: *An Interface for Annotating Natural Interactivity*. In J. v. Kuppevelt and R. W. Smith (Eds.): Current and New Directions in Discourse and Dialogue, Dordrecht: Kluwer 2003. Ch. 3. pp. 35–62.

Bernsen, N. O., Dybkjær, L. and Kolodnytsky, M.: *The NITE Workbench - A Tool for Annotation of Natural Interactivity and Multimodal Data*. Proceedings of the Third International Conference on Language Resources and Evaluation (LREC-2002), Las Palmas, 2002, 43-49.

Katherine Forbes, Eleni Miltsakaki, Rashmi Prasad, Anoop Sarkar, Aravind Joshi and Bonnie Webber. 2003. D-LTAG System - Discourse Parsing with a Lexicalized Tree-Adjoining Grammar , Journal of Language, Logic and Information, 12(3).

Barbara Grosz and Candace Sidner. 1986. Attention, Intention and the Structure of Discourse. Computational Linguistics 12:175-204.

Bonnie Webber and Aravind Joshi. 1998. Anchoring a Lexicalized Tree-Adjoining Grammar for Discourse. ACL/COLING Workshop on Discourse Relations and Discourse Markers, Montreal, Canada.

William C. Mann and Sandra A. Thompson. 1988. Rhetorical Structure Theory: Towards a Functional Theory of Text Organization. Text 8(3)243-281.

Marcu, Daniel. 2000. The Theory and Practice of Discourse Parsing and Summarization. The MIT Press. Cambridge, MA.

O'Donnell, Michael. 2003. RSTTool. (http:www.waysoft.com/RSTTool.)

Livia Polanyi and Remko Scha. 1984. A syntactic approach to discourse semantics. In Proceedings of COLING 6. Stanford, CA. 413-419.

Livia Polanyi, Martin van den Berg, Chris Culy, Gian Lorenzo Thione, David Ahn. 2004. A Rule Based Approach to Discourse Parsing. Proceedings of SIGDIAL '04. Boston MA.