Author manuscript, published in IEEE Transactions on Visualization and Computer Graphics, 18(12), pp. 2639-2648, Dec 2012.
DOI: 10.1109/TVCG.2012.226

Facilitating Discourse Analysis with Interactive Visualization

Jian Zhao, Fanny Chevalier, Christopher Collins, and Ravin Balakrishnan

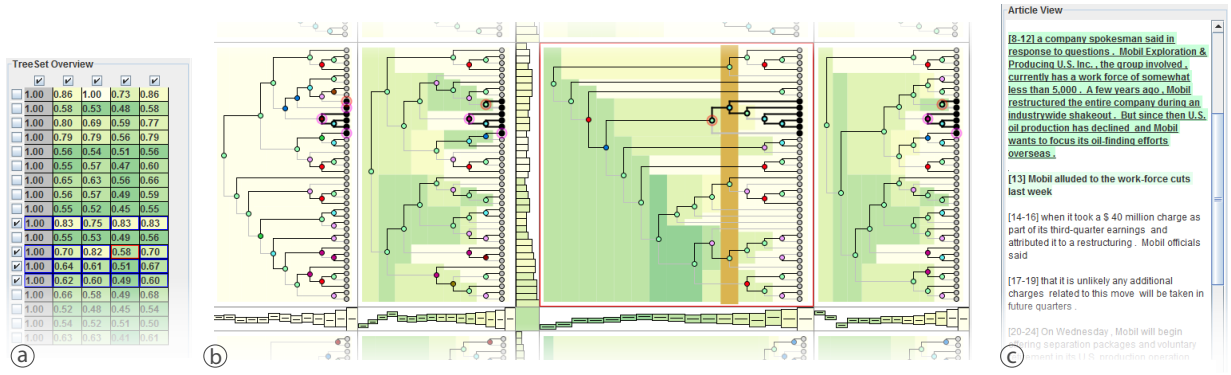

Fig. 1. The DAViewer, a visualization system for discourse analysis. (a) The overview panel shows similarity statistics about a collection of discourse trees; (b) the detail panel shows the full structure of the discourse trees; (c) the text panel is coordinated with the visualizations to allow detailed analysis of the correspondence of rhetorical structure and text content.

Abstract—A discourse parser is a natural language processing system which can represent the organization of a document based on a rhetorical structure tree—one of the key data structures enabling applications such as text summarization, question answering and dialogue generation. Computational linguistics researchers currently rely on manually exploring and comparing the discourse structures to get intuitions for improving parsing algorithms. In this paper, we present DAViewer, an interactive visualization system for assisting computational linguistics researchers to explore, compare, evaluate and annotate the results of discourse parsers. An iterative user-centered design process with domain experts was conducted in the development of DAViewer. We report the results of an informal formative study of the system to better understand how the proposed visualization and interaction techniques are used in the real research environment.

Index Terms—Discourse structure, tree comparison, computational linguisitics, visual analytics, interaction techniques.

1 Introduction

Natural Language Processing (NLP) has become a vitally important area of computer science research, as the results of research in this field are quickly put to wide use in systems such as text categorization, automatic translation, topic extraction, and summarization. A large portion of current efforts in this area involve computational linguistics researchers working to improve statistical parsing algorithms to gain ever higher accuracy and recall. Of particular interest is the subfield of automated discourse analysis, which aims to analyze the semantic structure and relationships within a text document. While parsing a sentence for its grammatical structure may be familiar to many readers, discourse parsing crosses sentence boundaries, extracting relationships within an entire document. These discourse structures are the foundation of many text-based algorithms such as certain types of summarization [19], question answering [5] and dialogue generation [24]. Yet, accurate discourse analysis remains a challenge due to the complex and nuanced nature of language, and research in this area is still very active. In this paper we present DAViewer, an interactive visualization tool for computational linguists to visually explore, compare, evaluate, and annotate automatically generated discourse structures with the aim of improving the underlying parsing algorithms (Figure 1).

In order to develop robust discourse parsers, researchers typically rely on a corpus—a large collection of human labeled documents—as a reference (called a *gold standard*) for training and evaluating algorithms. Several techniques have been proposed to make discourse parsers under a widely accepted framework, Rhetorical Structure Theory (RST) [18], which represents the organization of text as a tree structure after dividing it into non-overlapping text chunks (Figure 2a). While such an abstracted representation offers an helpful support for analysis, there are no adequate visual exploration tools to assist NLP researchers in discourse studies: in practice, researchers display or print out static representations of the discourse tree structures in the form of indented text chunks (Figure 2b). This makes the exploration and comparison process tedious, and particularly inefficient for the task of comparing the outputs of several variations of an algorithm.

Our visualization system, DAViewer is designed as an interactive tool to augment the manual analysis process, supporting the verification of hypotheses and discovery of insights about existing parsers in order to inspire the development of improved parsing algorithms. We developed DAViewer using a user-centered process, starting by the identification of the particular domain problems and challenges from which we derived design requirements for the visualization tool. In close collaboration with computational lingusitics experts at every stage of the development, we implemented and iteratively refined a functional prototype to address our target users' needs. The result-

- *Jian Zhao is with University of Toronto, Canada. E-mail: jianzhao@dgp.toronto.edu.*
- *Fanny Chevalier is with University of Toronto, Canada; and OCAD University, Canada. E-mail: fanny@dgp.toronto.edu.*
- *Christopher Collins is with University of Ontario Institute of Technology, Canada. E-mail: christopher.collins@uoit.ca.*
- *Ravin Balakrishnan is with University of Toronto, Canada. E-mail: ravin@dgp.toronto.edu.*

Manuscript received 31 March 2012; accepted 1 August 2012; posted online 14 October 2012; mailed on 5 October 2012.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

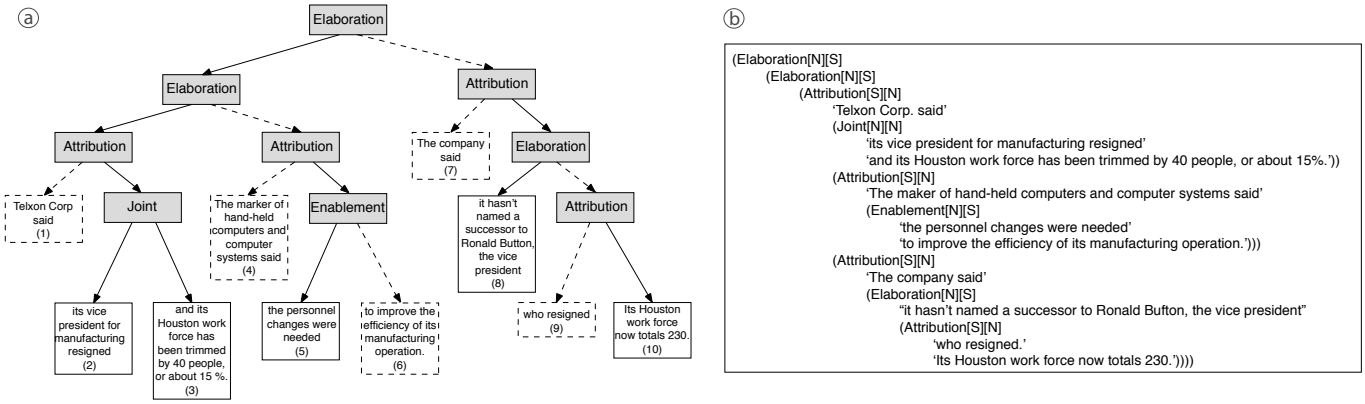

Fig. 2. Example of a typical discourse tree structure: (a) node-link representation of the hierarchical binary tree and (b) indented text format.

ing interface is built around a table of discourse tree visualizations, interactively coordinated with other visualization components including the texts under analysis and detailed information about selected objects. DAViewer is designed in such a way that computational linguistics researchers can actively integrate the visual exploration of intermediate results into their research process and use these results to further develop and refine robust algorithms in discourse studies. We conducted a formative study with a domain expert over a period of four weeks and found that our tool dramatically sped up some comparison and analysis tasks which were otherwise difficult to carry out. We also collected use case scenarios thus far unsupported using the traditional workflow. Several months after our study, our expert is still regularly using the tool for her research.

The contributions of this research are fourfold. First, we contribute design requirements for discourse analysis visualization, which were obtained incrementally through an iterative, four-stage user-centered design process. Second, we present an iteratively refined implementation of DAViewer with interactions and visualizations particularly tailored for NLP researchers in discourse analysis. Third, our tool introduces two new tree representations that can be applied to other domains—compact dendrograms in column or row views and a dendrogram version of icicle plots. Finally, we contribute the results of an informal formative study in a real work setting with a domain expert.

2 Background

2.1 Discourse Analysis

Discourse analysis in the NLP community is mostly driven by the RST framework [18]. In RST, the discourse structure is a binary tree[1] built from non-overlapping text chunks called elementary discourse units (EDUs). The EDUs are segmented from the text document, and serve as the leaves of the discourse tree. Then the discourse parsing algorithm successively combines EDUs to create internal nodes, each of which corresponds to a rhetorical relation between its branches (e.g., Attribution, Cause, etc.). Hernault *et al*. describe the 18 relations which are used in the parsers tested in this research, and widely used by the discourse analysis community [15]. Under each relation, the children can be either *nucleus* or *satellite*. With respect to the parent relation, the text associated with a nucleus branch is considered more prominent or important than the text associated with a satellite branch.

There are two ways to combine the segmentation and parsing algorithms. They can be coupled, so that the segmentation of the text into EDUs and the creation of the parse tree are interdependent and co-optimized. Or, the segmentation can take place in a separate, initial step, and the parsing algorithm is then forced to build the tree from these EDUs. The second (decoupled) method is more common and is used in this work. By forcing the EDUs to be the same across parsing algorithms, it allows for easier structural comparison of the trees generated by different parsers.

[1]While the original tree is not necessarily binary, a widely accepted practice in discourse analysis consists of converting an n-ary tree into a binary tree [15].

A popular corpus used in this area of research is the RST Discourse Treebank (RST-DT) [4], which consists of 385 documents transcribed from the *Wall Street Journal* and annotated under the RST framework. Figure 2a shows a typical rendering of a discourse tree, generated over the text of an article, with rhetorical relations shown in gray-filled boxes, and EDUs depicted with white boxes labeled with sequence numbers (i.e. the EDU's position as it appears in the text). Solid or dashed lines indicate whether the branch (or EDU) is nucleus or satellite respectively.

Several attempts have been made to develop discourse parsers using the RST framework. Many discourse parsers rely on a bottom-up approach for the tree building, i.e., linking EDUs and internal nodes with a parent relation, level by level until encountering the single top root node. The HILDA discourse parser [11], further improved by Feng *et al*. [12], is an instantiation of this approach to parsing, and is used in this work. Our external expert is working to improve parsing built with the HILDA parser as a starting point.

Despite the efforts of computational linguists, the generation of highly accurate discourse structures over a text document remains an open research question in NLP. In order to get a better understanding of the flaws and strengths of existing and newly created algorithms, the common practice consists of a close analysis of the discourse trees generated by different parsers: the comparison of a generated tree with the gold standard reveals how the obtained result differs from the optimal solution; and the comparison between generated trees helps analysts to identify the impact of tuned parameters of the same algorithm, or differing performances of multiple algorithms. In particular, linguists are interested in answering the following questions:

- Q1 At a given intermediate level in the discourse tree, is the text separated into meaningful groups (chunks)? Do the nucleus branches capture the prominent content of the text?
- Q2 Where are the errors in the generated discourse tree? Do errors at low levels propagate to upper levels of the tree structure?
- Q3 What are the structural differences between branches generated by two parsing algorithms over the same EDUs?
- Q4 Does the algorithm generate common parsing structures (or errors) across all the documents in the corpus?
- Q5 How consistently does each algorithm perform across documents in the corpus? Which types of documents are more problematic?

With the lack of efficient visualization systems to address their needs, linguists struggle to answer such questions effectively, as they are currently working with a collection of indented text encoding the tree structure, as shown in Figure 2b.

2.2 Visualization and Computational Linguistics

There is a growing body of research in the area of text visualization, with most efforts aimed at content analysis—revealing keywords in a document, topics in a corpus, and changes in streaming text data. In this work, we are interested in a specific subset of language-related visualizations: those works whose aim is to improve our understanding of linguistic phenomena or computational linguistic algorithms.

Visualization has been used to answer fundamental questions in linguistics. For example, the Dichronlex diagram is used to reveal changes in language constructs over time [28]. Pilz *et al*. [23] use multidimensional scaling to plot the similarities of spelling variants to understand the propagation of spelling changes through time and geography. Structured Parallel Coordinates can be used to understand common patterns in corpus linguistics [9]. Finally, Constellation is a graph-based visualization targeted at helping refine algorithms that determine semantic networks [21].

More relevant to this work are visualizations designed to better understand, and even to improve, computational linguistic algorithms. The DerivTool is designed for computational linguistic researchers to interactively correct problems in a translation system by directly editing the translation model learned from training data [10]. The lattice visualization of Collins *et al*. [7] and the Chinese Room visualization of Albrecht *et al*. [1] both use visualization to reveal a collection of closely ranked translation hypotheses considered by an automated translation algorithm, and allow a user to select the most reasonable alternative. Finally, the Bubble Sets visualization was designed to support machine translation researchers, using a participatory approach similar to the one we adopted here [8]. Bubble Sets are overlays on parse trees to improve their usefulness in the task of diagnosing translation errors in order to improve translation algorithms. Similarly, the discourse tree visualizations of DAViewer are designed to support the discovery of errors in discourse trees in order to inspire improvements to discourse parsing algorithms.

2.3 Visualization of Tree Structures

A breadth of techniques have been proposed for the visual comparison of trees, that can be classified under three main categories: side-by-side views [2, 6, 22], merged views [29, 30] and animation [27]. For an exhaustive review see the survey by Graham *et al*. [13].

Side-by-side views can be based on visual cues to convey the relationships. Explicit links can be drawn between the matched nodes, and the alpha channel of the links tuned to indicate the similarity measure between the corresponding branches, as for example in the syntax trees by Chevalier *et al*. [6]. However, such an explicit linking can lead to a cluttered view due to the numerous lines. Side-by-side views can leverage interaction through dynamic queries: for instance, TreeJuxtaposer [22] allows the user to interactively visualize similar subtree structures by highlighting a query pattern. A system particularly related to our work was developed by Bremm *et al*. [2] for the comparison of phylogenic trees. In that work, several visualization techniques are combined as coordinated views, including a tabular view of the trees of interest and a similarity matrix view indicating similarity scores between the trees of the dataset.

An alternative to side-by-side views is to use merged views in which two trees are combined in a single visualization that encodes the differences. Similarity matrices fall into this category. The nodes of the trees to compare correspond to rows and columns, and the cells of the matrix indicate the similarity between the nodes. Van Ham [30] uses such an approach for software analysis. These techniques constitute a powerful and space-efficient way for comparing the different nodes with one another. This is, however, to the detriment of making the hierarchical structure apparent. Union Tree [29], which integrates two trees into a single treemap visualization based on a structural match, and colour-codes the differences between the nodes are another example of merged views. Similarly, Candid Tree [17] visualizes structural differences between two trees by merging them into a single, colour-coded, node-link representation. While such approaches make differences salient, they are limited to the comparison of two trees at a time.

Finally, a third approach is to use animations to convey changes between two different trees [27]. However, users may lose track of the overall difference since the positions of many nodes are varying during the animation [13]. In addition, while animations can help keep track of changes while smoothly transitioning between trees, only one of the trees is visible at any given time, making comparison difficult.

3 Iterative User-Centered Design

We used a user-centered approach over the course of four months. Here, we describe our methodology and the design requirements.

3.1 Methodology

We followed a standard user-centered design process as described in [20], involving two experts: an expert in both computational linguistics and visual design, and a co-author of this paper; and an external researcher, from a university computational linguistics group whose focus is discourse analysis. While both were engaged in the iterative design stages of the prototype, only the external expert was involved in the formative study.

Through regular consultations with our experts, we gathered and refined a list of requirements, and built a series of prototypes: over the course of four months, we maintained a weekly meeting with the experts, during which they were presented with the latest prototypes for feedback on further requirements using several methods including interviews, meetings, observations, exchanging emails, and phone calls. A series of prototypes were deployed, including sketches, paper prototypes, an initial implementation running on a small dataset, a refined high-fidelity prototype with full functionality, and a deployable system. Details of the four stages of our design process follow.

Problem identification and quick prototyping. The first step was aimed at identifying the problems and challenges faced by the target users, along with the exploration of possible design alternatives to address their needs. Through a series of interviews focusing on the frustrations and issues NLP analysts encounter in daily research life with their current workflow, we derived a set of basic functionalities and design requirements for a visual exploration tool (Q1-5 and Section 3.2). Several design alternatives where then explored, critiqued, and refined with the help of the experts using sketches and paper prototypes.

Initial implementation with core functionality. After agreeing on a general design concept and priorities, the second phase consisted of implementing, iteratively testing, and refining an initial interactive prototype that included a number of key functions. We conducted this iterative process over a period of six weeks, using a small sample of the whole data provided by the external expert for testing: the annotated gold standard and the results of the HILDA parser over six documents. During this phase, several design problems were identified regarding the colour coding of discourse trees, improper visual cues and interaction issues preventing a fluid exploration. Our experts also requested additional functions such as filtering and querying the tree structures and displaying summary statistical information about the discourse trees.

Refined prototype. At this stage, a full version of the tool with a refined interface and complete set of functions was be released to the expert for use in her real work setting, using a journaling technique to record usage and potential areas for improvement. Only the external expert was involved in this two-week long testing phase, which allowed us to identify a number of minor bugs. More importantly, our expert reported that she sometimes felt that the visualizations failed at providing adequate information about how the text was separated into groups under a level of the tree (Q1). In response, we designed and added an alternative representation of the tree structure based on the icicle plot [16]. Because icicle plots are space-filling, the extent of EDUs under the nodes at each level is readily apparent.

System release and field test. This final phase served as a field formative study: a final system, including logging instrumentation of user actions for reliable quantitative usage analysis, was deployed to the expert user for a longer time period (several weeks). We report on this last phase in more detail in Section 6.

3.2 Design Requirements

Through the iterative process with our expert users, we learned a lot about the process of research to improve discourse analysis. Gathered from and validated by our observations, and in response to the questions enumerated in Section 2.1, we defined the following specific design requirements for analytic tools to support discourse analysis:

R1 **Discourse tree representation.** While several approaches for representing the discourse trees could be considered, our experts explicitly requested that the core visualization resemble the representation they currently use for analysis of the parsing structure (see Figure 2a). In particular, immutable constraints are the presentation of leaf nodes in the same order as the associated text chunks appear in the document, and the explicit visual encoding of nodes' type (satellite or nucleus) and relation. More importantly, the visualization should facilitate the identification of clusters at any level of the tree, a flaw of the node-link diagram currently used in practice. Finally, our users mentioned that it was naturally preferable that the different trees that share the same set of EDUs are aligned to facilitate structure comparison (Q3).

R2 **Errors, distributions and statistical information.** In order to gain better intuitions for designing new algorithms, linguists must discover the pros and cons of different parsers. Statistical information, such as the distribution of relationship types appearing in a tree or the similarity scores assigned to branches of the tree should be readily available, as they provide an important overview of the discourse structure and performance of the parsers (Q2).

R3 **Article views and textual contexts.** In order to observe whether the parsing algorithm groups EDUs into meaningful units when building the discourse tree, it is important to be able to see how EDUs are grouped under a relation (Q1). At any level of a discourse tree, the tool should support separating the source text into proper chunks (groups of EDUs) according to the branching pattern. It is also important to be able to view the text of individual EDUs in isolation. Finally, to support close reading of the documents under analysis, a standard paragraph-based layout of the document should be supported.

While the above requirements are described in the context of the specific NLP application domain, we note that the general ideas behind (R1) and (R2) are not exclusive to linguists. Indeed, representing the tree structure and node details in a similar fashion to that traditionally used by the target users is likely to be valid in other domains involving tree comparison. Similarly, the analysis of distributions, statistical information and errors are recurrent themes for systems that operate under uncertainty. However, the general tree visualizations tools that could support these requirements would still need to be adapted to effectively satisfy the specific domain constraints. To best support R1-3, we propose a highly tuned design for representing the discourse tree structures, the core visualization component of the DAViewer system, that we describe in Section 4.

In addition to the above requirements, the visual exploration tool should support general requirements including dynamic queries such as filtering and querying, to allow for a focused attention on specific types of relations or structural patterns (R4), and which effect should instantaneously be reflected in all of the coordinated views to guarantee visual consistency across the different visualization components for a fluid and effective exploration (R5); flexible data management, to assist researchers to do long-term progressive research during which the datasets may expand (R6); and finally, annotation, to support documentation of the findings for future reference (R7).

4 Discourse Tree Representations

4.1 Expanded views

We designed two main types of representations to visualize the discourse trees: an adapted version of the icicle plot [16] (Figure 3a), and a dendrogram—a branching node-link diagram particularly suited to reflect relationships (Figure 3b), that resembles the traditional representation as used by our target users (Figure 2a).

Tree structure and node details. In both representations, the nodes are colour-coded according to the assigned relation label from the 18 described by Hernault *et al.* [15]. The specific hues were selected from 18 of the 22 most distinguishable colours in Green-Armytage's colour alphabet [14]. The hue of the links between the nodes and their parent relation indicates the nuclearity of the children

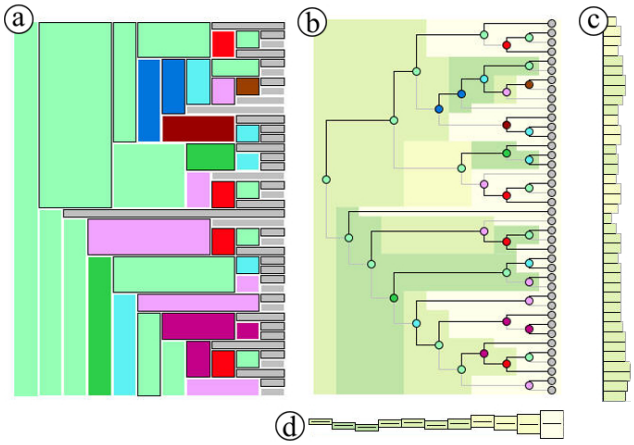

Fig. 3. Discourse tree representations: (a) icicle plot view, (b) dendrogram view, (c) vertical compact view, and (d) horizontal compact view.

nodes in the node-link dendrogram: black indicates a nucleus (the most prominent nodes), and grey indicates a satellite. In the icicle-dendrogram, the colour of the outline of the children node indicates this property. These representations are interactively linked to the text panel which displays the content of the document (see Section 5.2.2), thus allowing for easy access of the text chunks associated to the nodes of current focus (R3).

Representing the clusters. We added the icicle-based plot after our experts reported that it was difficult to see the clustering of the EDUs at a specific level of the tree (Q1). With the node-link dendrogram, one has to follow the lines to accurately rebuild the different groupings. This can be challenging with large or high trees. While it is well accepted that the dendrogram eases the readability of hierarchical structures when the leaves must be aligned, the traditional icicle plot is generally better than node-link diagrams for the task of identifying clusters [16]. Our icicle visualization is a hybrid representation: we display nodes in the form of rectangles as in the traditional icicle, to make the embedding relation apparent. However, our layout mimics the dendrogram in that we align all the leaves at the same rightmost level, and we expand each rectangle's width up to the level where the corresponding node is grouped in turn. In this way, when looking at the icicle-dendrogram in columns, one can clearly see the clustering of EDUs at each intermediary stage of the bottom-up grouping process.

Showing the errors. In order to inspire the development of improved parsing algorithms, it is essential for linguists to develop a good understanding of the specific flaws of the existing algorithms compared to the gold standard. In particular, our experts are interested in identifying precisely which step(s) of the greedy bottom-up process fail, and to what extent such errors have an impact on the steps that follow (Q2). To facilitate such analysis, we build an icicle-dendrogram where nodes are colour-coded according to the similarity scores of the internal nodes using a yellow-to-green palette. This dendrogram is used as a background on top of which the node-link dendrogram is overlaid (Figure 3b), thus serving as a heatmap where errors are made more salient because of the darker colour.

4.2 Compact views

When many trees have to be compared, as it is the case for our users, space limitation can become an issue. We designed two new compact representations of the discourse trees. In a method similar to how Table Lens [25] allows for compact graphical representations of symbolic data for space efficiency, we propose compacted rows and columns, while providing a vertical (Figure 3c) or horizontal (Figure 3d) informative representation that summarizes important characteristics of the compacted discourse trees (R2).

Vertical compact view. In this view, each bin corresponds to a leaf node, i.e., an EDU. The width of a bin encodes the depth of its associated leaf in the tree, i.e., the number of intermediate clusters the EDU belongs to on the path to the root. The colour of a bin is mapped to the average similarity score of its corresponding leaf node

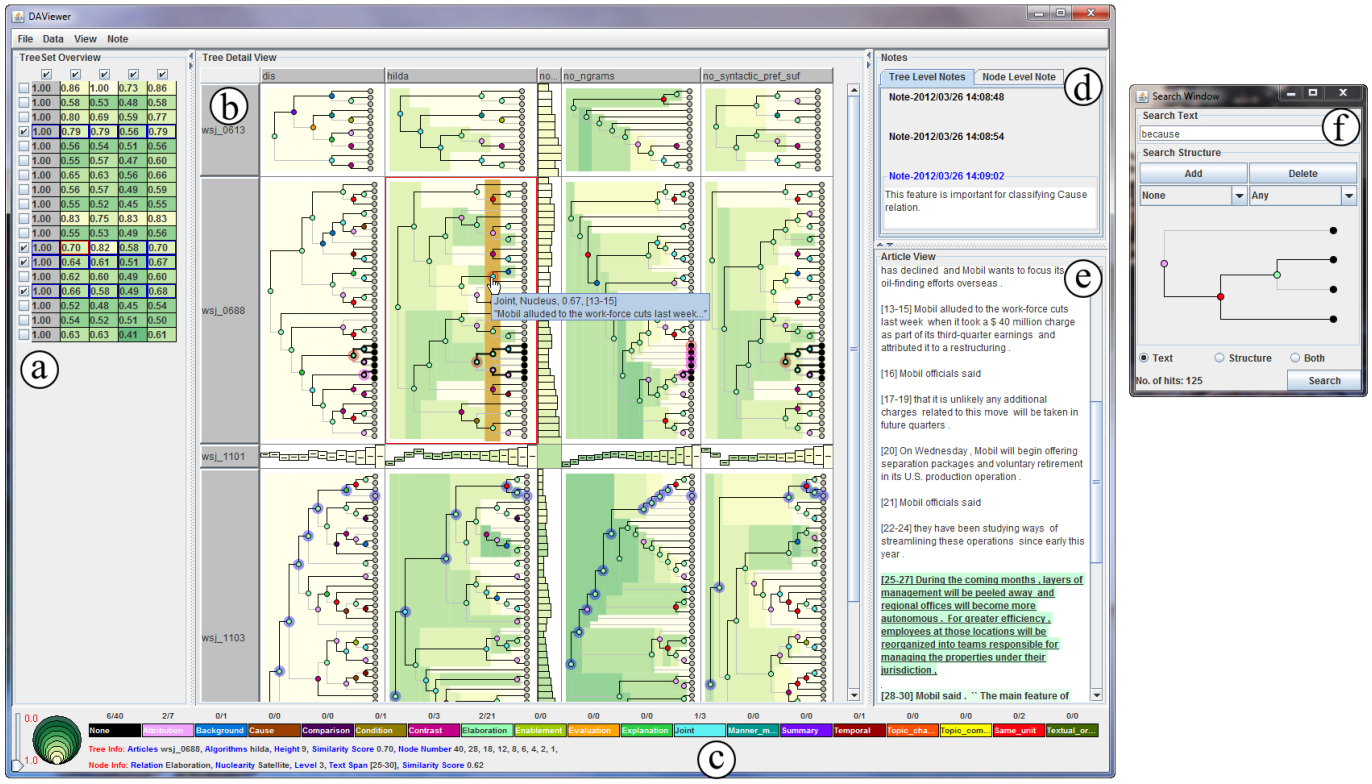

Fig. 4. The DAViewer interface is comprised of: (a) an overview which displays the overall performance of all the parsers over all the documents in the dataset, (b) a detail panel which visualizes the discourse tree structures of the focused algorithms and documents as node-link or icicle dendrograms, (c) a status panel which provides the basic properties of the currently selected items as well as an interactive legend for filtering operations, (d) an annotation panel which allows users to edit notes, (e) a text panel which shows the content of the active document as parsed by the focused algorithm, and (f) a search window which for querying based on keyword and structure over the data in the detail panel.

and all nodes on the path from the leaf to the root. Hence, the compact representation conveys the average error an EDU contributes across the levels. For example, a wide dark bin indicates an early grouped EDU whose initial error strongly impacts all its upper levels.

Horizontal compact view. In this view, each bin corresponds to a level of the compacted tree. The height of a bin encodes the number of nodes at the associated level. The colour is mapped to the average similarity score of the set of nodes. Hence, the compact representation allows a viewer to identify at which stage of the clustering process the algorithm starts to fail and whether such errors propagate to upper levels or not. We also use the vertical axis to represent the centroid vertical position of the nodes belonging to the level, and center the bins accordingly. A horizontal black mark in the center of the bin is used as a visual cue of the centroid location. This gives an indication of whether the EDUs are clustered evenly across the document (reflected by a mark close the center), or if the groups of EDUs are more concentrated on one or the other side (the higher the bin, the more independent clusters at the beginning of the document).

The two compact representations shown in Figure 3c,d can be viewed as two marginal perspectives of the node link view shown in Figure 3b along the vertical and horizontal dimensions. While they have initially been designed to help answer question Q2 through the presentation of trees at different levels of abstraction, the proposed compact representations, as a design concept, can be generalized for hierarchical tree structures in general, as we later discuss in Section 7.

5 The DAViewer Interface

We developed DAViewer, a complete interactive visualization tool for computational linguists to explore, compare, evaluate, and annotate the results of parsers in discourse studies. We designed DAViewer around the core discourse tree visualizations introduced above, and refined the interface through the four discrete stages of the user-centered design methodology as described in Section 3.1.

The DAViewer interface is composed of five interactively coordinated views (Figure 4) including (a) an overview panel of the entire dataset, (b) a detail panel, (c) a status panel, (d) an annotation panel, (e) an article panel, and (f) a search window. Dynamic brushing and linking techniques are applied for interactively coordinating information displayed in the different panels (R5).

5.1 Overview: See the Whole Dataset

The overview panel (Figure 4a) consists of a matrix view representing the entire available dataset—a collection of discourse trees, each tree resulting from the computation of a given parsing algorithm (columns in the matrix) applied to a given document (rows in the matrix).

The overview is useful to determine at a glance how the different algorithms perform on the same document—which amounts to comparing the results along a row of the matrix, and how the same algorithm performs across a set of documents—which amounts to comparing the results along a column of the matrix (Q5), as compared to the reference algorithm (i.e., the gold standard, or otherwise user-defined). One column is set as the reference and shown in grey. In Figure 4a, the leftmost column serves as the reference. Each non-reference cell is colour-coded according to a similarity score between the tree in that cell, and the current reference tree in that row (i.e., we compare trees for the same document). The numeric scores are displayed in each cell, and also mapped to a yellow-to-green scale [3]. Therefore, the matrix plays the role of a heatmap, that the user can use to quickly identify the trees that differ the most from the current referent, and thus requiring deeper investigation to identify the problems.

In our current implementation, the similarity measure relies on the element-based measure proposed by Bremm *et al.* [2], but other similarity measures could be considered. The overview displays the scores associated with the root nodes, conveying the similarity scores between the whole tree structures. Low similarity scores generally indicate parsing errors. For example, one can easily tell from Figure 4a,

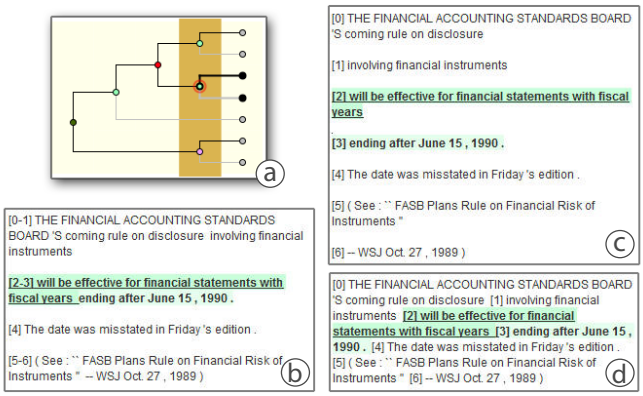

Fig. 5. Different text display formats: (a) in the discourse tree, the currently active level is indicated by a dark brown ribbon. In the text panel, the content display can be set to (b) hybrid, to reveal the EDUs grouping at that level, (c) separated, to show individual EDUs, or (d) continuous to present the text as a single paragraph.

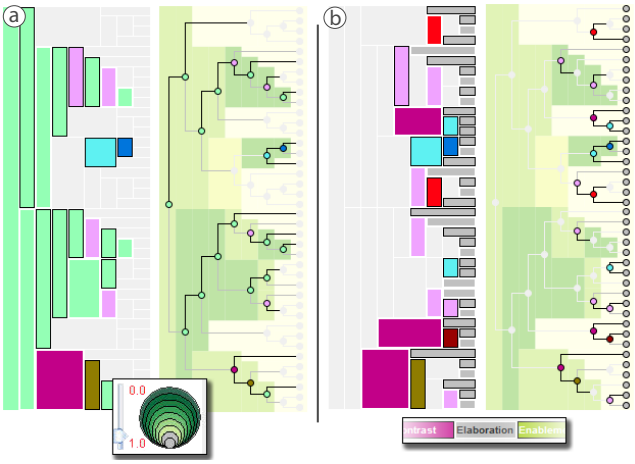

Fig. 6. Filtering operations of interactive legends: (a) filtering out nodes above score 0.8 and (b) filtering out nodes with the *Elaboration* relation.

that the last document (last row) is generally problematic, and that the third algorithm (third column) under performs the others on this specific document.

In practice, as the research goes, linguists develop several refined versions of the same algorithm or propose new algorithms. They may also decide to extend their gold standard by adding more annotated documents. In order to support the progressive research process of our users, DAViewer provides convenient ways of adding and/or removing entries to workspace (R6).

5.2 Drill Down: Further Explore a Set of Trees

When the user has identified problematic algorithms and documents, she can drill down into the data and access an expanded view of a set of trees, as described in Section 4.

5.2.1 Choose the Trees of Interest

In order to preserve a consistent tabular representation across the views and maintain the alignment of EDUs across algorithms (R1), the selection is algorithm- and document-based, in that the user selects the set of trees of interest by checking row (document) and column (algorithm) headers in the overview. All the cells at the intersection of the selected rows and columns are loaded in the detail panel, as shown in Figure 4b. The currently selected cells are outlined in blue in the matrix overview.

Depending on her current task, the user can interactively choose between the icicle plot view, the dendrogram view, or decide to reduce some trees to their compact representations to devote more space to the algorithms and documents of immediate interest.

5.2.2 Explore the Discourse Trees

To facilitate the visual exploration, DAViewer provides fluid interactions for visual exploration of discourse trees. The user can select a node to access detailed information of the node and its subtree in the status panel (Figure 4c). This also triggers the emphasis of the node and the branch under it, using thicker edges. In other trees in the same row, the branches corresponding to the same EDUs (leaf nodes) are also emphasized (see middle row, Figure 4b) so as to facilitate the comparison of internal structures (Q4). The associated text of the EDUs under the selected node is loaded into the text panel (Figure 4e), and emphasized in several ways (R3): (1) we apply a bold font to the selected node's content, (2) the text of the first child node is underlined, and (3) the background is coloured according to the main relation's hue, with a dark or light tone whether the text belongs to a nucleus or a satellite branch respectively (Figure 5).

As the user moves the cursor in the active table cell, a semi-transparent brown ribbon indicates the tree level where the cursor resides, as an aid to identify the nodes on the same tree level (Figure 5a). There are three ways the text can be formatted. First, clicking on an empty space within a level (as opposed to a node) selects that level, which rearranges the text into a layout we call *hybrid display*: the text is split into paragraphs according to the EDUs branching at that level (Figure 5b). This text display is particularly useful when the user wants to analyze the intermediate stages of the relation grouping (Q1, R1). The second text layout is the *separated display* (Figure 5c) which presents each EDU as a separate paragraph. Finally the *continuous display* (Figure 5d) displays the article as a single paragraph. The separated display is suited to the matching EDUs to leaves in the tree, as the EDUs remains clearly separated, while the continuous display aims to offer a more comfortable way of reading text in a continuous flow. The separated display is equivalent to choosing the leaf level in the hybrid display; the continuous display is equivalent to selecting the root level in the hybrid display.

DAViewer also supports traditional tree operations such as branch collapsing when the user double clicks on a node. It is also possible to collapse branches through batch operations using a context menu (e.g., collapse all branches under a node with similarity score below a certain threshold, or all branches up to a specific tree level).

The rich interactions in DAViewer allows for a flexible exploration of the trees. This is particularly useful for a close analysis of the relations and the text associated to a problematic node, that can help identify the causes of a mislabeled relation (e.g., a keyword causing ambiguity). When comparing how a set of EDUs are clustered across several algorithms, the user can also explore in details of parsing differences, allowing her to infer hypotheses on the impact of parameter settings on the correct identification of specific relations, as illustrated in Section 6.4.

5.3 Filter and Query: Reveal the Important

The status panel acts as an interactive legend for filtering operations (R4), including the similarity score legend (Figure 6a), to filter out nodes with specific similarity scores, the relation legend (Figure 6b), where each label is a modal button for dynamic filtering [26]. DAViewer also incorporates querying functions for node and branch searching (R4), including keyword search, structure-based querying, and the combination of the two (i.e., all branches that satisfy both queries) through a separate dialog (Figure 4f). Keywords can be entered in the text area, triggering the highlighting of all the nodes which descendant EDUs contain such a keyword. To perform a structural search, the user builds her pattern of interest either from scratch, or by editing a structure copied from the detailed view. The user can specify or omit the relation and nuclearity types associated to the nodes of the query pattern. When no value is specified, the engine searches for *any* value of such nodes. All the structures that match the query are highlighted in the detail panel with a blue halo. The structure-based query function has been explicitly requested by our users as a critical function to locate recurrent error patterns, as this task is extremely difficult to perform with their current analysis method (Q4).

5.4 Annotate: Write a Research Journal

The analysis of parsing results is a continuous and progressive process as the research progresses. To allow NLP researchers to record insights and exploration notes, DAViewer supports annotations on trees or nodes, and groups of trees or nodes through the panel as shown in Figure 4d (R7). Notes are presented as a collapsible list of items titled with the timestamp of the last addition and can be saved together with the workspace status. The note panel is fully coordinated with the other views, implying that when a note is selected, all the corresponding items in the overview and detailed view are visually emphasized.

6 Formative Study

We conducted an informal formative study with our external expert to evaluate DAViewer in a realistic work environment. This section describes the methodology, summarizes the findings, and presents two example use case scenarios.

6.1 Methodology

DAViewer is designed to be tailored to the needs of computational linguists in their research process. Our research process depends on many factors including the user's motivation, previous knowledge and particular work settings, which are not easy to evaluate in a laboratory study. To better understand the strengths and flaws of our tool in a realistic work environment, we adopted a formative study approach with our external expert user over a three week period.

The goal of the study was threefold. First, we wanted to evaluate the overall acceptance and usability of DAViewer as an analytical tool assisting computational linguists in their research process. Second, we were interested in investigating the patterns of use of the different interaction and visualization components, as a valuable guide for future improvements of our system. Third, we were hoping DAViewer would allow our user to discover new insights, and were therefore interested in collecting use case scenarios.

We released DAViewer to our external user, who was free to use the tool at will on her personal computer. Since DAViewer supports the standard file formats used by linguists, our user did not encounter any difficulty generating and loading the additional datasets to her workspace. For the study, we added a logging mechanism to collect occurrences of a set of 39 operations (e.g., select a node, set icicle view, etc.) that we classified into 8 higher level categories (e.g., dataset management, detail view manipulation, querying, etc.). Operations and categories are detailed in Figure 7b. Each operation was recorded alongside its time stamp. We also requested our user to keep an analysis diary, and maintained regular contact via email and several short informal interviews at regular intervals during the study. At the conclusion, we also conducted an additional interview to collect her feedback after the evaluation period.

In addition to the interview notes, user's diary and emails, we collected 17 log files (449 minutes in total), from which we discarded 5 log files that corresponded to sessions of less than 5 minutes long. The remaining logs corresponded to use of 432 minutes, in session ranging from 15 to 56 minutes long ($\mu = 36, \sigma = 13$). Altogether, the remaining log files contain a total of 3048 operations, with a minimum of 55 and a maximum of 456 operations per session ($\mu = 254, \sigma = 124$). In addition, our user described several use case scenarios of the insights she had discovered while using DAViewer.

6.2 Usability and Satisfaction

During the final interview concluding the formative study, our expert user reported that the visualizations offered significant benefits for analyzing the outputs of discourse parsing algorithms: *"It is great to have all the [discourse] trees available and see them visually. I can print [trees] in text files and look at them all day long, but never found it could be that clear and easy with the visual representations."* She added that *"The collapsed views are very handy, because sometimes I just want an overall comparison, not the deep deep details,"* and reported to be highly satified with the GUI design and interactions that she found to be *"handy and straightforward"*.

DAViewer proved to be very efficient to our user as it greatly increased her productivity: *"I used to draw trees by hand according to the output text files, so I could only compare 2 or 3 trees at the time and they are basically simple trees around 5 levels. With DAViewer I can compare many large trees efficiently. All trees are nicely aligned and the interactions allow me to focus on subtrees easily."* The user emphasized that her past experience of drawing trees of about 20 EDUs was "awful" (each tree taking her more than 10min to draw) and highly error-prone due to the manual process. Then she mentioned: *"For the same data, now I can spot the [tree structure] differences in 2 seconds by observing the [similarity] score heatmap."* For comparing very large trees, which was almost impossible to do in the past, the expert also found the row and column compact tree views very useful for identifying the parts to focus on first.

The visual representations of discourse trees were favoured by the user because they were efficient in showing the parsing process and comparing tree structures, though they were slightly different from the manually produced graphs. The user said, *"I know I want the tree levels aligned from the bottom, but I can't draw such layout from the output file in one round since the tree nodes are indented in the depth-first order."* Moreover, she commented that the data visualization was flexible because the separation of text and tree structure allowed her to focus on different aspects of the dataset and the interactions such as tooltips and highlights of text provided the connections conveniently.

Other valuable functions for our user were the searching and filtering capabilities, especially the structure-based query. She said, *"I used to insert debug code inside the parsers, for a verbose output, but it is tedious and almost impossible to find patterns. Now, I can do it visually by just creating what I need and clicking a button."* She indicated that a further improvement of the querying function with boolean operations would greatly enhance the tool.

In summary, our expert was enthusiastic about DAViewer, as reflected by the numerous extended sessions she conducted during the evaluation. She would like to continue using the tool in her research.

6.3 Patterns of Usage

Figure 7 summarizes the results of the operations logging during the formative study, including (a) a colour-coded time diagram of a typical working session, (b) the classification of the operations into categories, and (c) the overall distribution for each category use.

Not surprisingly, the user spent most of her time on tree exploration, of which node selections were predominant (92% of tree exploration operations). We found out in the final interview that collapsing nodes was not desirable, as it does not preserve alignment of the EDUs and it disturbs the global *"mental representation of the tree."* A design implication is that future versions should support synchronized node collapsing and coupled panning to preserve EDU alignments.

Since the detail panel was her main focus, the user naturally spent a fair amount of time adjusting the treeset shown in this panel. Nearly one third of the operations consisted of compacting and uncompacting rows and columns. The expert commented that she preferred to look at the trees first in these compact views to get a general idea about the performance of the parsers, especially when the structure was large. This reveals that our design of compact views was useful for analysis, in addition to saving screen real-estate.

The dendrogram was largely preferred over the icicle plot (79%). Our user explained that for most of the time her purposes were to identify structural differences among trees, that is easier to do with the dendrogram. However, she said that the icicle plot was more useful and more salient when observing the behaviors of parsers, i.e., how nodes are merged from bottom to top and how the whole article is partitioned at each tree level of processing.

The user also performed a large amount of queries (299 in total) and filtering operation (177 in total). Among the querying operations, 18% were keyword searches, 55% were structure-based and 27% were both, confirming the importance of structural queries in our domain of application. Moreover, the expert mentioned that this significantly increased the efficiency of spotting desired structural dissimilarities by interactively combining operations of both querying and filtering.

b

dataset management	open treeset in plain text format, import a tree or a treeset, save all workspace, load a workspace, remove row/column from the dataset, append row/column to the dataset
overview and tree subset	change reference column, load selected trees in the detail panel, add trees to the detail panel, remove trees from the detail panel
detail panel manipulation	select a row in the detail panel, select a column in the detail panel, select a tree cell, compact a tree cell, uncompact a tree cell
tree exploration	set dendrogram view, set icicle view, collapse branches up to a level, collapse branches above a score, expand all nodes, expand a branch, collapse/expand a branch (double click), select a node
text display	set continuous display, set separated display, set hybrid display
annotation	add a tree-level note, add a node-level note, delete a note
querying	add a note to a pattern, select a node in the pattern, edit pattern, search keywords, search structure, search both
interactive legend filtering	activate/deactivate a relation, change score filter threshold

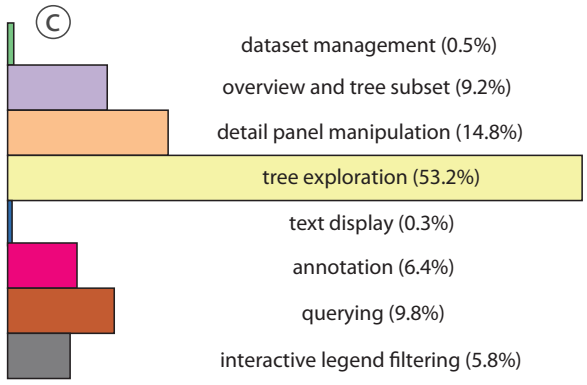

Fig. 7. Results of the logging gathered during the formative study: (a) shows the break up of a typical session of DAViewer into categories of operation as detailed in (b), and (c) summarizes the overall usage of the different types of operations.

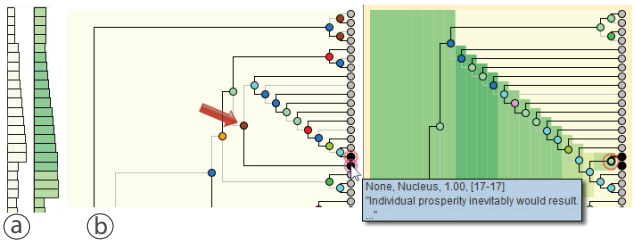

Fig. 8. Comparing HILDA with the gold standard: (a) the compact vertical views allow for the localization of the error in HILDA: (b) the incorrectly grouped nodes are shown in red outlines, (c) the red arrow points at the correct *Cause* relation.

Surprisingly, the user rarely changed the text display format, although it was an important functional requirement. In the end, the user explained that her most favoured display format, the hybrid display, was flexible enough to easily access both other displays by simply selecting the root or the leaves levels. She however did choose to keep the display fixed in rare occasions.

Finally, we analyzed individual log files to gain knowledge of the user's exploration flow when using DAViewer. We plotted each session as a colour-coded time diagram. An example in Figure 7a represents a typical working session. We observed a recurrent pattern across all the sessions: after loading the dataset, the user adjusts the documents and algorithms of interest in the overview. Then a long period of time is dedicated to detailed exploration: tree manipulations, interleaved with querying and filtering, and eventually the creation of a note about insights. This pattern is then repeated several times on the same treeset, until the user decides to adjust the latter for a new investigation on different trees. This usage exhibits a typical visual exploration process for hypothesis verification and discovery, that DAViewer successfully supports with its coherent coordination between the panels presenting the data at different levels of abstraction.

6.4 Use Case Scenarios

We present two use case scenarios based on our user's diaries. The dataset she used as a gold standard was a set of 20 annotated documents from the RST-DT set [4]. The parsing algorithms included the HILDA discourse parser [15] and several algorithms of her own.

6.4.1 The HILDA Parser versus the Gold Standard

In this scenario, the user wants to investigate the flaws of the HILDA parser, a popular algorithm in the domain. She starts by glancing at the overview which presents the similarity scores, and finds that overall, the parser is performing fairly well. She identifies the two documents with the highest (0.86 and 0.83) and lowest (0.52 and 0.54) scores and selects them for deeper analysis. The gold standard, and the HILDA output for these four documents are loaded in the detail panel.

The user immediately finds out that the documents causing errors are much longer than the other ones, which is reasonable, as typically, the more content, the more challenging the parsing. Likewise, the discourse trees are large and difficult to read as a whole structure. In order to get an overall idea of where the algorithm fails, the user reduces the trees to their compact representations to observe the distribution of scores, groups and so forth. From the vertical compact view, she observes that while the distribution of nodes into groups is similar to the gold standard, the scores of HILDA are very low (Figure 8a). Next she expands the tree views, and at the same time compacts the short documents since they are not the focus a the moment.

With the help of the heatmap background of the dendrogram, the user identifies where the error first occurs: HILDA groups EDUs 16 and 17 as early as the second level whereas the gold standard keeps the branches separated up to level 17 (Figure 8b). Thus she found that this first error, which propagates to the root node, is a major problem that strongly affects the overall parsing. By looking at the text, she finds that the EDU 17 says "individual prosperity inevitably would result" where the keyword "result" is a critical indicator of the *Cause* relation. Yet, the HILDA parser groups this node under an *Elaboration* relation.

To get some context, the user switches to the continuous text display to comfortably read the sentence with the problematic EDU. She finds that EDUs 7-16 as a whole are the summary of previous content, that should be grouped together in a branch under the *Cause* relation, with EDU 17, as indicated by the gold standard. The user thus selects the group of nodes and comments on her finding on the annotation panel. Meanwhile, she wonders if such errors happen elsewhere. She adds more trees with low scores to the detail panel, and through the query panel, looks for other *Causal* relation structures with branches containing the keywords "because" and "as a result". A close examination of the results reveals that HILDA incorrectly groups the nodes or mislabels the relation (*Elaboration* or *Explanation*) and adds notes each time she finds such error in the dataset for further consultation. Indeed, the above findings provide hints for improving parsers, by taking more careful consideration of the content under a *Cause* relation.

6.4.2 Comparison of the performance of different algorithms

After identifying several issues in the HILDA parser, the user wants to investigate if and how tuning different parameters affects the outputs of her own parsers. She appends the result of three variations of her algorithm to the overview matrix (referred to as algorithms A1, A2 and A3). In this scenario, she sets HILDA as the reference column, since she wants to compare where the algorithms differ in performance. Adopting a similar approach as that of the previous scenario, she first glances at the overview and finds out that the fourth column (A2, corresponding to the condition "no N-grams feature") provides the most differing results, and that the rightmost column (A3, condition "no syntactic prefix and suffix") provides a very similar parsing as that of HILDA (Figure 9a). She selects a subset of four rows (two

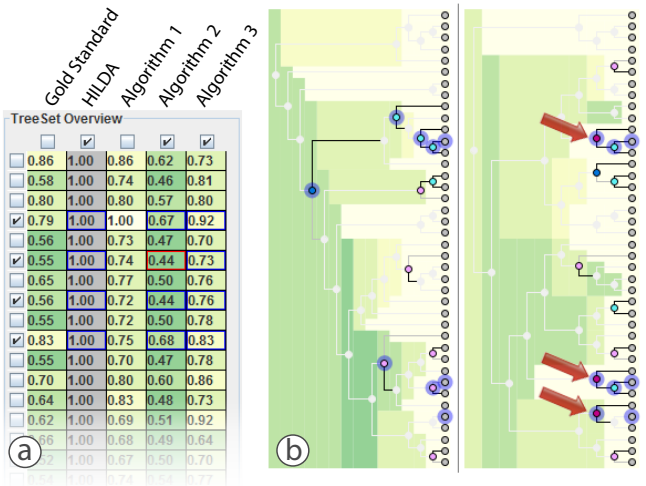

Fig. 9. Comparing different algorithms: (a) HILDA parser is set as the reference; (b) after filtering out *Elaboration* and *Same Unit* relations, the tree of algorithm 2 (left) is almost all faded out. A keyword search for "but" is applied (blue highlighted nodes). When compared with the tree of algorithm 3 (right) algorithm 2 cannot detect *Contrast* relations containing "but" whereas algorithm 3 does (red arrows).

documents with the most similar and most differing results) and three columns (HILDA, A2 and A3) for further analysis in the detail panel.

By looking at the different tree representations (i.e., compact views, dendrogram and icicle), the user discovers that the trees generated by A2 usually contain many more levels and are more skewed, indicating that the classifier cannot find clear grouping pivots. To make the differences more visible, the user decides to fade out the branches similar to those of HILDA by filtering them out through the use of the interactive similarity score legend, and switches to the icicle view to analyze the relation types. She observes that most of the remaining nodes are labeled with *Elaboration* or *Same Unit* by A2, which is unsurprising, since the two relations are the most common ones in that specific corpus. After deactivating the latter in the relation legend, she clearly observes that A2 hardly identifies any other relations, indicating that the N-grams feature, which was deactivated in A2, is essential for the relation classifier.

Our user wants next to compare the performance of a particular relation: *Contrast*. She looks for phrases indicative of this relation: "but," "on the contrast," etc., coupled with a structure-based query around the *Contrast* relation. She observes that A2 fails at identifying such relations, while it is usually well labeled by discourse parsers (Figure 9b). She further comments on her findings by adding a note that considering a certain number of EDUs as a whole (as N-grams does) is a good parser feature for identifying the real rhetorical relations.

7 Discussion

The core visualization component of DAViewer is the tabular tree detail panel (Figure 4b) with its icicle-dendrogram and compact representations of discourse trees which were revealed to be an important contribution of this paper (Figure 3). The icicle-dendrogram, when visually browsed vertically, reveals key features of hierarchical clustering as the traditional icicle plot does, which in our case reflects the greedy process of merging of text chunks from the original EDUs. Taking the benefits of dendrogram, nodes that remain unmerged across many levels, are shown saliently as rectangles with larger horizontal widths, making it easier to identify the nodes that reside on the levels covered by the width of a specific node, than when using a traditional dendrogram. In some studies of clustering algorithms such as classifiers in machine learning applications, this is useful for checking why a specific node is not combined with others in the algorithm as well as spotting the anomalies. In addition, when an icicle plot encoding the similarity score is displayed as the background of the dendrogram representation (Figure 3b), the analyst can more easily locate the structural differences, as large color-coded areas are easy to spot at a glance. Our expert user extensively relied on this background heatmap to quickly find the very first merging anomaly propagated to the top levels when comparing parsers. This visualization, as a combination of two representations of the same tree, is general enough to be applied to any other hierarchical structure, whether it is for comparison purposes, e.g., studying the difference of evolutionary relationships between organisms in phylogenetic trees, or to simply augment a dendrogram with the visualization of an additional attribute associated to the nodes, e.g., displaying the space that files and folders take on the hard drive on the background of a file system browser.

The row and column compact representations of trees, on the other hand, offer a legible summary of both the structural information of the tree and aggregate attributes of contained nodes. While these visualizations were designed for discourse analysis, the concept is also generalizable to exploration of other types of hierarchical structure, and their new ways of abstracting tree structures could be useful for common tasks in other domains, of which finding structural change and identifying co-evolutions in phylogenetic trees are an example. In particular, when projected to the vertical axis, the view compresses the tree along its leaves by showing the depths similar to the icicle-plot, which indicates how species generally evolve and which branches are most active; and when projected to the horizontal axis, the view compresses the tree along its levels by showing its skewness, which shows the trends of the evolution to a common ancestor.

It is worth recalling that in our study, the trees corresponding to the same document are built from an identical set of EDUs, which is motivated by the unique features of discourse analysis. Yet, the tool proposed in this paper, and all its respective components, can be extended to the comparison of trees with different leaf nodes. In our specific application domain, this means that the similarity measure needs to be adjusted accordingly. The synchronized selection would also have to be adapted, by looking into the contents of the selected text, rather than EDU identifier numbers, as is done in our current implementation. In this more general case, the alignment of EDUs in trees within the same row of our detail view would not be meaningful anymore, as the EDU contents could differ.

8 Conclusions and Future Work

This paper introduced DAViewer, an interactive visualization tool to assist computational linguists in developing insights and intuitions in the specific subfield of discourse analysis. DAViewer has been iteratively refined in a four stages user-centered approach with domain experts, based on incremental collection of user feedback and refinement of design requirements. Our system consists of coordinated components aiming to facilitate the analytic workflow in discourse analysis. DAViewer integrates novel visualizations such as the dendrogram icicle, and vertical and horizontal compact representations, which are generalizable to other application domains. A formative study was conducted to evaluate the system in real work settings and the results showed that DAViewer was useful in the research of a domain expert.

While a formative study was performed, we aim to distribute the system to the computational linguistics community, to evaluate the acceptance of DAViewer, as well as to gather more feedback for further enhancing the system. We also plan to empower the querying capabilities with boolean operations, as requested by our user. According to the results of the evaluation, we plan to implement new interaction techniques to facilitate tree exploration, including synchronized node collapsing, zooming, and panning of trees in the same row. We aim to augment DAViewer into a more complete working environment by adding functions such as live monitoring of the parsing progress and on-the-fly modifications of parser codes in a unified interface. Also, we will work on a generalization of our novel visualization techniques, and further explore their application to other domains.

Acknowledgments

We especially thank Vanessa Wei Feng, from the Computational Linguistics Group, Univeristy of Toronto, for her valuable time and feedbacks in the participation of the design and evaluation of the DAViewer visualization system.

REFERENCES

[1] J. Albrecht, R. Hwa, and G. E. Marai. The Chinese Room: Visualization and Interaction to Understand and Correct Ambiguous Machine Translation. *Computer Graphics Forum*, 28(3):1047–1054, June 2009.

[2] S. Bremm, T. von Landesberger, and M. Heb. Interactive visual comparison of multiple trees. In *Proc. of the IEEE Symp. on Visual Analytics Science and Technology*, pages 29–38, 2011.

[3] C. A. Brewer. Colorbrewer2. http://www.colorbrewer.org, 2009. Accessed March, 2012.

[4] L. Carlson, D. Marcu, and M. E. Okurowski. Building a discourse-tagged corpus in the framework of Rhetorical Structure Theory. In *Proc. of the Second SIGdial Workshop on Discourse and Dialogue*, pages 1–10, 2001.

[5] J. Y. Chai and R. Jin. Discourse structure for context question answering. In *Proc. of HLT-NAACL 2004 Workshop on Pragmatics in Question Answering*, pages 23–30, 2004.

[6] F. Chevalier, D. Auber, and A. Telea. Structural analysis and visualization of C++ code evolution using syntax trees. In *Ninth Int. Workshop on Principles of Software Evolution: in conjunction with the 6th ESEC/FSE Joint Meeting*, IWPSE '07, pages 90–97, 2007.

[7] C. Collins, S. Carpendale, and G. Penn. Visualization of uncertainty in lattices to support decision-making. In *Proc. of Eurographics / IEEE-VGTC Symp. on Visualization*, May 2007.

[8] C. Collins, G. Penn, and S. Carpendale. Bubble sets: Revealing set relations with isocontours over existing visualizations. *IEEE Trans. on Visualization and Computer Graphics (Proc. of the IEEE Symp. on Information Visualization)*, 15(6), 2009.

[9] C. Culy, V. Lyding, and H. Dittmann. Structured parallel coordinates: a visualization for analyzing structured language data. In *Proc. of the Int. Conf. on Corpus Linguistics*, pages 485–493, 2011.

[10] S. DeNeefe, K. Knight, and H. H. Chan. Interactively exploring a machine translation model. In *Proc. of the Annual Meeting of the Assoc. for Computational Linguistics, Poster Session*, 2005.

[11] D. A. duVerle and H. Prendinger. A novel discourse parser based on support vector machine classification. In *Proc. of the Joint Conf. of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing*, ACL '09, pages 665–673, 2009.

[12] V. W. Feng and G. Hirst. Text-level discourse parsing with rich linguistic features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 60–68, July 2012.

[13] M. Graham and J. Kennedy. A survey of multiple tree visualisation. *Information Visualization*, 9(4):235–252, 2010.

[14] P. Green-Armytage. A colour alphabet and the limits of colour coding. *Colour: Design and Creativity*, 5(10):1–23, 2010.

[15] H. Hernault, H. Prendinger, D. A. duVerle, and M. Ishizuka. Hilda: A discourse parser using support vector machine classification. *Dialogue and Discourse*, 1(3):1–33, 2010.

[16] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):pp. 162–168, 1983.

[17] B. Lee, G. G. Robertson, M. Czerwinski, and C. S. Parr. Candidtree: visualizing structural uncertainty in similar hierarchies. *Information Visualization*, 6(3):233–246, Dec. 2007.

[18] W. C. Mann and S. A. Thompson. Rhetorical structure theory: Toward a functional theory of text organization. *Text*, 8(3):243–281, 1988.

[19] D. Marcu. Discourse trees are good indicators of importance in text. In *Advances in Automatic Text Summarization*, pages 123–136, 1999.

[20] P. McLachlan, T. Munzner, E. Koutsofios, and S. North. Liverac: interactive visual exploration of system management time-series data. In *Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems*, CHI '08, pages 1483–1492, 2008.

[21] T. Munzner, F. Guimbretiere, and G. Robertson. Constellation: A visualization tool for linguistic queries from mindnet. In *Proceedings of the 1999 IEEE Symposium on Information Visualization*, INFOVIS '99, pages 132–135, 1999.

[22] T. Munzner, F. Guimbretière, S. Tasiran, L. Zhang, and Y. Zhou. Treejuxtaposer: scalable tree comparison using focus+context with guaranteed visibility. *ACM Trans. on Graphics*, 22:453–462, July 2003.

[23] T. Pilz, W. Luther, and U. Ammon. Retrieval of spelling variants in nonstandard texts – automated support and visualization. *SKY Journal of Linguistics*, 21:155–200, 2008.

[24] H. Prendinger, P. Piwek, and M. Ishizuka. A novel method for automatically generating multi-modal dialogue from text. *Int. J. Semantic Computing*, 1(3):319–334, 2007.

[25] R. Rao and S. K. Card. The table lens: Merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI)*, 1994.

[26] N. H. Riche, B. Lee, and C. Plaisant. Understanding interactive legends: a comparative evaluation with standard widgets. *Computer Graphics Forum*, 29(3):1193–1202, 2010.

[27] G. Robertson, K. Cameron, M. Czerwinski, and D. Robbins. Animated visualization of multiple intersecting hierarchies. *Information Visualization*, 1(1):50–65, Mar. 2002.

[28] R. Therón, L. F. Fontanillo, A. E. Marcos, and C. S. Herrero. Visual analytics: A novel approach in corpus linguistics and the Nuevo Diccionario Histórico del Español. In *Proc. of III Congreso Internacional de Lingüística de Corpus*, 2011.

[29] Y. Tu and H.-W. Shen. Visualizing changes of hierarchical data using treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 13:1286–1293, November 2007.

[30] F. van Ham. Using multilevel call matrices in large software projects. In *Proc. of the IEEE Conf. on Information Visualization*, INFOVIS'03, pages 227–232, 2003.