Hyperbolic Dependency Tree Visualization for Parser Evaluation

Le Wang* Yue Zhang† Lei Shi*

Abstract

We present a novel tool to visualize sentence dependency trees in the hyperbolic layout, and to provide visual support for comparative evaluation of parsing errors. Compared with the traditional flat tree view in NLP tools, our hyperbolic visualization tool can be more convenient for showing sentence structures and long-range word dependencies. Our tool integrates a hyperbolic view with the flat view, and supports the corpus-level error analysis. The tool supports the effective dependency parser evaluation by combining statistical analysis of error distributions, visual analysis of an individual dependency tree, and an integrated online interface.

1 Introduction

Dependency parsing is the task of automatically analyzing the syntax of natural language sentences according to dependency grammars [3, 7]. For example, Figure 1 shows the dependency structure of a sentence, where a directed link from "includes" to "ads" represents a verb-object dependency. Dependency parsing is useful for a wide range of tasks, including semantic role labeling, information retrieval, question answering and machine translation. Improving the accuracy of dependency parsers has been a major research topic in the field of natural language processing(NLP).

Visualization has been widely used to help corpus linguists study grammers, and to help computational linguists analyze dependency parser outputs and to find their weaknesses. There are quite a few NLP tools that focus on visualizing the dependency parsing results or integrate the visualization as an important feature, such as MaltEval [5], MaltDriver [2] and ViZPar [6]. In particular, the most popular tool, MaltEval, provides several error statistics by computationally comparing outputs of a statistical parser and the corresponding manually annotated gold trees.

MaltEval uses the flat tree representation in Figure 1, which is adopted by most existing NLP tools. However, in light of the visualization objective and the tree comparison task, this flat view design is not optimized for three major reasons. First, the resulting view normally fills a narrow rectangle with a large aspect ratio, while the general recommendation for visualizations is to place them in a square room best fit for modern interfaces. Second, the flat view is more appropriate for the n-gram context, where the dependent words are also close in the sentence, while a dependency parse tree can have quite a lot of long-range dependencies. Third, on the task of visual error comparison, the parsing errors can happen on the long-range dependency links, some of which cannot be shown in a single flat view without scrolling in the interface. This poses an additional cost in the linguistic analysis process.

The main innovation of this paper is to introduce the *hyperbolic radial tree layout* [4], which has been well studied in the visualization community, to the task of comparing dependency trees. On this task, the hyperbolic tree view has the following advantages. First, the radial layout ensures a nearly circular shape in the final view, which can be accommodated in a square panel without significant distortions or scalings. Second, words having direct dependency links are placed close to each other. Thus, the dependency context and structure are better revealed through the visualization. Third, in most cases, the parse tree of one sentence can be displayed in a single view, and all the details of the output v.s. gold parse trees can be compared side-by-side without need to scroll the panel.

In addition to the hyperbolic layout, we incorporate a few useful features to the visual design, such as word indexing and visual error comparison, by learning from the classical design and the specific task. Building over this visualization design, we have developed a dependency parser evaluation system that integrates the corpus-level parser error statistics with the sentence-level visual parsing result comparison. Notably, the system is implemented in JavaScript and can be accessed via a web browser. Compared to MaltEval, which is based on Java and requires PC-side installations, our system enjoys the best of the ubiquitous mobile computing trend in heterogeneous devices. An online demo is available at http://211.147.15.14/hyperbolic/index.html .

2 Comparison of Parse Tree Layouts

Here we make a comparison between four common visualization techniques for dependency trees, including the flat view (Figure 1), which has been used by most existing NLP tools, the hyperbolic view we introduce (Figure 2), the hierarchical tree view (Figure 3), which is used by ParseViz [1], and the force-directed layout for general graphs (Figure 4). Each figure shows the dependency structure of a sentence, in which the number inside a circle represents the index of the corresponding word in the sentence. In the flat view, the label on each arrow represents the type of the dependency relation. The flat view shows a given sentence in one line by the natural word order, while the other views show a tree structures more saliently.

These layouts are different in the following aspects: First, in terms of the space efficiency, given its flat shape, the flat view is the weakest, while the others are stronger. The hyperbolic and force-directed layouts can make full use of the square space as these aspect ratios are close to one, while the aspect ratio of the hierarchical layout can be significantly larger. Compared with the force-directed layout, the hyperbolic layout is more space efficient by folding peripheral nodes non-linearly.

Second, in terms of the dependency structure, the flat view represents dependency relations as polygonal lines with an arrow, while the other views represent related words using directed tree edges. According to the connectedness law in the Gestalt theory, human has a tendency to perceive any uniform, connected line or area, as a single unit. Thus, the other three views are better choices in showing the dependency structure. For example, consider the dependency relations of the root verb "includes". In the flat view, the words which depend on "includes" are difficult to find due to the intricate polygonal lines, let alone the words beyond the flat view. In the other views, the dependency relations of each word, such as "includes", are easy to interpret as they are directly connected by the tree edges. In the flat view, the full tree structure of a sentence cannot be perceived. On the other hand, the other layouts all show the tree structure, in which the hierarchical view explicitly models the top-down tree structure, at the expense of space efficiency.

Third, the viewing of long-range dependencies is important for linguistic analysis. In the flat view, long-range dependencies, espe-

*SKLCS, Institute of Software, Chinese Academy of Sciences, e-mail: {wangl,shil}@ios.ac.cn. This work is supported by China National 973 project 2014CB340301, NSFC project No. 61379088.

†Singapore University of Technology and Design, e-mail: yue_zhang@sutd.edu.sg

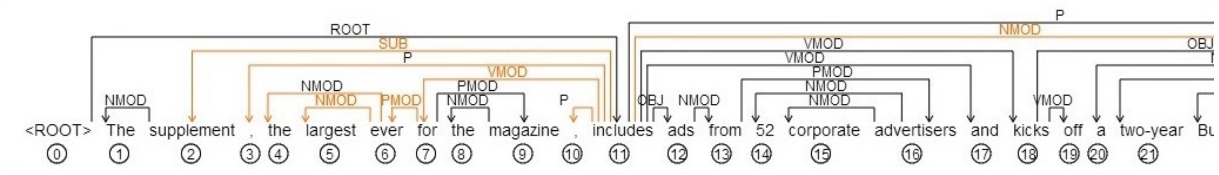
Figure 1: The flat tree view of the dependency structure (a scroll bar is necessary).

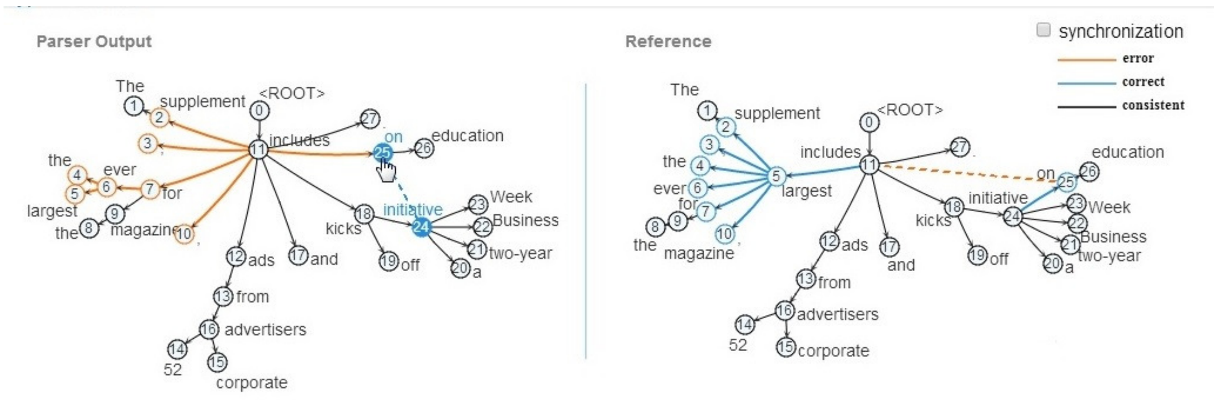

Figure 2: Hyperbolic tree visualization comparing the parser output with the reference.

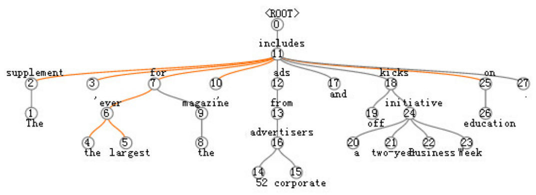
Figure 3: Hierarchical Layout.

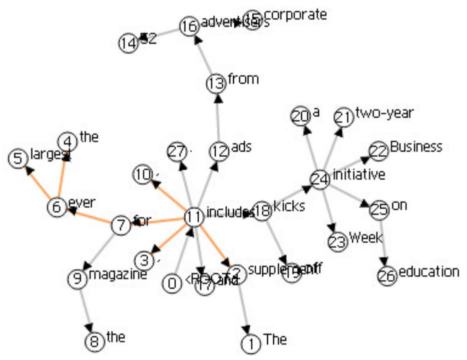
Figure 4: Force-directed Layout.

cially dependencies outside the view, cannot be displayed directly. The hyperbolic and force-directed layouts show both long-range and short-range dependencies in the same way using a short arc.

In summary, the flat view is strong in showing the actual word order, while the force-directed and hyperbolic views are better in showing the dependency syntax structure. The hierarchical layout can be viewed as a compromise between the flat view and the force-directed view, which keeps the word order but is less space efficient. The hyperbolic view guarantees to show a full tree in limited space by condensing peripheral nodes. Given that the error analysis typically studies one node and its immediate context at a time, we recommend the hyperbolic layout to visualize the dependency structure. To compensate for the lack of natural word order in this view, we associate it with the flat view in our system interface.

3 System Interface and Visual Design

The input to our system is a parser output corpus and optionally a gold-standard reference. Our system indexes a corpus by the error distribution. A sentence can be selected according to the different type in parsing errors in the corpus, including the part-of-speech (POS) of incorrectly parsed words, the label of incorrectly parsed dependency relations and the incorrectly parsed word. Each sentence is visualized in the flat and hyperbolic views, juxtaposing dependency parsing output and the reference.

In both views, we use red and blue to represent the incorrect relations and the corresponding correct relations, respectively. For example, in the flat view in Figure 1 and the hyperbolic view in Figure 2, the relation between “includes” and “supplement” is red, representing that the dependency relation of word “supplement” is incorrect. Correspondingly, the correct relation, “largest” to “supplement”, is shown as a blue edge in the right panel.

The hyperbolic view also supports the following interactions: First, the user can drag and drop any word into the center, in order to see its dependency context. In Figure 2, the word “includes” is moved into center. In addition, the user can rotate the output and references trees synchronously by selecting the check box “synchronization”. Second, the user can check the incorrect and reference dependency relations of any word by moving the mouse over the word in the view. For example, when mouse is hovered over the node “on” whose index is 25 in the parser output view, as shown in Figure 2, the correct dependency relation, “initiative”→“on”, is shown by a dotted blue edge. At the same time, in the reference view, the incorrect dependency relation, “includes”→“on”, is shown in a dotted red edge.

References

[1] Parseviz. `http://www.ark.cs.cmu.edu/parseviz/`.
[2] M. Ballesteros and R. Carlini. Maltdiver: A transition-based parser visualizer. In *IJCNLP*, 2013.
[3] S. Kübler, R. McDonald, and J. Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2009.
[4] J. Lamping, R. Rao, and P. Pirolli. A focus+ context technique based on hyperbolic geometry for visualizing large hierarchies. In *CHI*, 1995.
[5] J. Nilsson and J. Nivre. Malteval: an evaluation and visualization tool for dependency parsing. In *LREC*, 2008.
[6] I. Ortiz, M. Ballesteros Martínez, and Y. Zhang. Vizpar: A gui for zpar with manual feature selection. *SEPLN*, 2014.
[7] Y. Zhang and J. Nivre. Transition-based dependency parsing with rich non-local features. In *ACL*, 2011.