Visualizing Sentence Parse Trees with WordBricks

Marina Purgina

Maxim Mozgovoy

The University of Aizu
Tsuruga, Ikki-machi, Aizuwakamatsu,
Fukushima, Japan
+0242-37-2664
{d8172102, mozgovoy}@u-aizu.ac.jp

Abstract—**We address the task of visualizing machine-readable natural language parser output in a graphical form, convenient for users. While there are numerous available tools implementing similar functionality, their resulting diagrams are typically aimed at language specialists, and are arguably difficult to understand for inexperienced second language learners. In contrast, our system uses a specialized language learning system (WordBricks) as a visualization module, and is deliberately designed to help language learners to understand the structure of natural language sentences. We hope that the resulting software tool (available for the Android mobile platform) will be beneficial for regular use in a variety of language learning scenarios.**

Keywords—natural language processing; mobile-assisted language learning; syntactic parsing; visualization.

I. INTRODUCTION

Typically, natural language processing (NLP) tools provide machine-readable output, not suitable for subsequent manual analysis. Therefore, the task of visualizing structured linguistic information for the convenience of a user is widely discussed in literature [1, 2]. Linguistic visualizations help to understand language data and interact with it, and serve as an aid in explaining language phenomena [21]. One of the important subtasks of visualization in NLP is graphical representation of the output generated by language parsers. This problem is addressed in a number of research projects [3–5].

However, it seems that these efforts are by large ignored in language education. Most practical textbooks describe sentence structure with informal diagrams or plain-text explanations. Only relatively advanced books that are specifically dedicated to language syntax (such as [6]) provide sentence trees in the form typically used in theoretical linguistics.

We believe that visual explanations of linguistic structures can be helpful for a learner. At least, some authors indeed employ informal diagrams to describe certain language phenomena, and pictures/visual aids in general are widely used, and considered as efficient teaching materials [7–9]. Unfortunately, common types of sentence structure diagrams require certain effort and experience to be understood by the learners. In principle, these diagrams were not designed to be teaching aids, being primarily targeted at professional linguists.

We see two main issues with traditional parse tree diagrams, causing difficulties for beginners: (a) diagrams do not preserve word order in the original sentences; and/or (b) diagrams are cluttered with arrows and auxiliary captions (see Fig. 1).

Our aim is to remedy the situation by substituting standard parse trees with simplified block diagrams that preserve most linguistic information. The proposed representation keeps original linear structure of sentences and uses shapes and colors instead of additional visual elements to reduce the amount of onscreen elements (see Fig. 2). We are assuming that preserving linear structure of the sentences should really support the learning process and give the learners better understanding of sentence structure. In particular, the learner is still able to see the sentences in their natural form while understanding its structure and inter-word relationships. “Linearization” is an important property that is sometimes stated as a deliberate design goal of a sentence structure diagram [3].

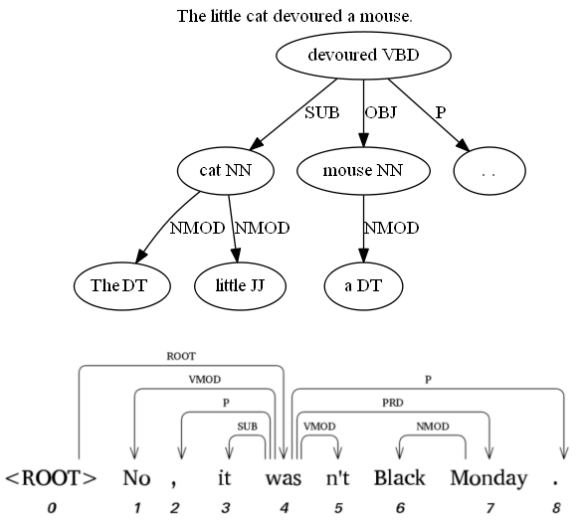

Figure 1. Parse trees obtained with
AT&T GraphViz (above) [11] and ZPAR (below) [4].

This approach for representing parse trees is implemented in our earlier system WordBricks that is designed as a mobile-assisted instrument for second language learning, and thus specifically aimed at language learners [10]. The system has been tested and evaluated in a real classroom environment, and demonstrated promising results. In these experiments we used WordBricks to visualize exercises and sentences taken from the actual textbook used by the students. In the present work, we use WordBricks as a visualization module that can display a user-supplied sentence as a simplified parse diagram.

Figure 2. Example of a complete sentence in WordBricks.

Word relations in WordBricks are based on the dependency grammar formalism [DEPENDENCY GRAMAR]. However, we tried to design lightweight graphical elements that can be compactly displayed on mobile devices and be aesthetically appealing to the users, which is important for educational software.

II. WordBricks: A Quick Overview

The main purpose of WordBricks is to let the user to combine words and phrases into grammatically correct constructions, exploring the possibilities of natural language grammar. This system supports two modes of operation. In the *free mode*, WordBricks lets the user to experiment with any known words and word combinations to check which constructions are admissible according to natural language grammar. In the *lesson mode*, WordBricks displays a set of predefined exercises. Each exercise consists of several disjoint bricks the user needs to combine into correct sentences.

Preliminary evaluation of WordBriks in real classrooms showed that the students who used WordBricks as a study aid scored higher on the exam tests [10]. One of the reasons of WordBricks' efficiency as a study aid lies in its simplicity as a presentation medium for language grammar. In its original version, WordBricks can only display words and constructions defined by the authors of the exercises (technically, the exercises are stored in static XML files).

WordBricks visualizes sentence structure with a combination of nested colored bricks of different shapes. Each brick is associated with a set of textual attributes that unambiguously define the shape and the color of a brick. A brick also contains a linear list of child items, consisting of words and connectors. A word is a static predefined textual element, drawn on a brick background. Each connector is a placeholder for a brick that forms a dependency relation with the connector's parent brick. Like bricks, connectors are associated with textual attributes that define their shapes. From the technical point of view, the classes for bricks and connectors are based on the standard Android framework (see Fig. 3). When a brick is inserted into a connector, the respective BrickView object is placed on the corresponding ConnectorView object. This operation is possible since both classes are inherited from the standard class RelativeLayout that can work both as a graphical element, and as a drawing canvas holding other View objects.

A brick can be placed inside another brick's connector if that connector's attributes form a subset of the set of brick attributes. Let us state once again that child bricks are displayed inside the connectors of their parents, thus the whole structure preserves the original linear form of a sentence. When WordBricks is used as a visualization module, it displays a static structure of already linked bricks, corresponding to a user-supplied sentence.

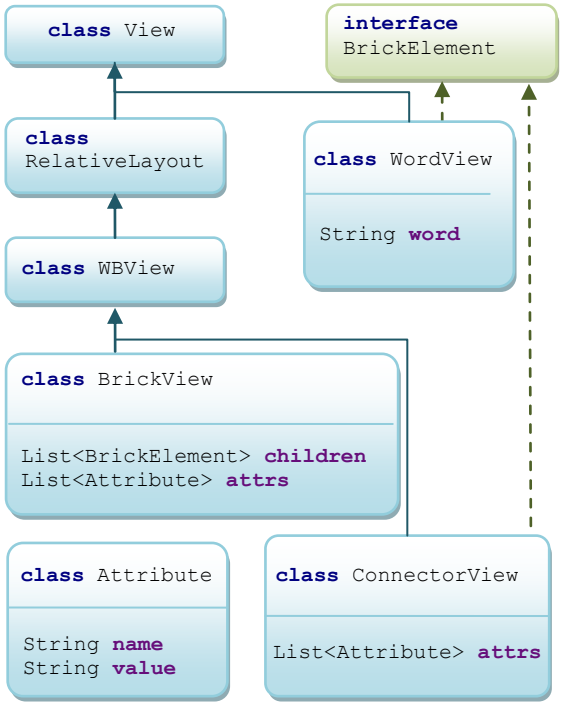

Figure 3. Class diagram of bricks visualization subsystem

Since WordBricks performs no linguistic processing of the input data, it can be adapted to a variety of natural languages and grammar formalisms. This design decision also makes it easy to use WordBricks as a visualization module for any given sentence structure, properly encoded in compatible XML documents. Each XML file corresponds to a single sentence and contains three sections. *Brick descriptions* section defines the attributes, words and connectors of each brick present in the given sentence. *Sentence tree* section describes the sentence parse tree generated by the parser (it lists all the bricks with their associated dependent elements). *Colors & shapes* section is a language-specific description that maps linguistic attributes (such as part-of-speech tags) into brick shapes and colors. Since the present version of the system works only with English sentences marked with Penn treebank-styled part-of-speech tags, this

section remains unchanged for any user-supplied sentence.

It should be noted that in the present form WordBricks can visualize *projective* relations only (i.e., each word and its descendants should form a contiguous substring of the sentence). This restriction is nearly negligible for English [12], but for languages with flexible word order such as Czech or German the proportion of non-projective structures in real texts can be higher than 23-27% [13].

III. Client-Server Architecture of the Parse Tree Visualizer

WordBricks is a lightweight mobile application, consuming little computational and memory resources. In order to preserve these attractive features, we decided to implement all linguistic processing on the remote server side. Therefore, the resulting application consists of two separate modules: a mobile WordBricks-based GUI, and a server backend. On startup, the application displays an input box prompting the user to provide any arbitrary sentence. After the user taps the OK button, the application sends a request to the server side. The server returns an XML document describing the desired brick configuration to be displayed on the screen.

The GUI module (frontend) of the system is an Android application, written in Java language. The server side (backend) is a Python CGI script, accessible via HTTP interface (see Fig. 4). Most of linguistic processing is performed in external executable modules, invoked by the CGI script.

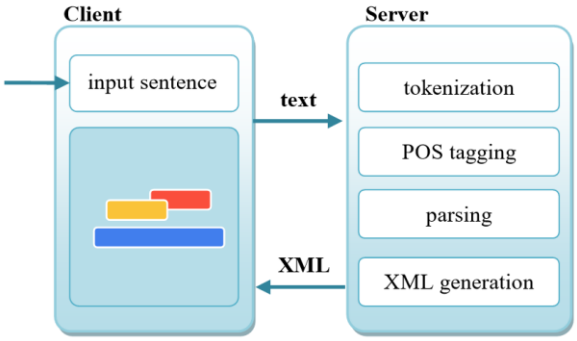

Figure 4. Architecture of the system.

The backend has to perform a number of operations, transforming the input sentence into a tree structure. They are invoked in the following order.

1. Tokenization. The input sentence is divided into a number of tokens, corresponding to individual words and punctuation marks of the sentence. Tokenization treats contracted constructions as separate tokens. For example, the sequences *you'll* and *don't* are tokenized as *you* | *'ll* and *do* | *n't* respectively.

2. Part of speech (POS) tagging. The tokenized sentence is marked with Penn Treebank-style part-of-speech tags [14]. The tagger relies on the *maxent* toolkit [15] that implements maximum entropy modeling technique [16]. The tagger was trained on the manually annotated part of Open American National Corpus [17]. In our cross-validation experiments the tagger exhibited 96.40% accuracy, comparable to the state of the art.

3. Syntactic parsing. The sequence of tagged tokens is passed to the dependency parser that converts the input into a collection of dependency trees, providing results as a CoNLL-U-formatted text document [18]. Our parser is based on the source code of Layer-Based Dependency Parser LDPar [19]. The parser was trained on the WSJ section of Penn Treebank [20]. The resulting accuracy can be considered as acceptable: 84.54% for unlabeled, and 83.28% for labeled parsing.

4. XML generation. The final processing stage involves conversion of the CoNLL-U data into the XML document supported by WordBricks. Since CoNLL-U format contains all required information (word boundaries, part-of-speech tags, and labeled syntactic dependencies), this operation is relatively straightforward, and is performed by the main CGI script.

IV. Conclusion

The task of visualizing the output of NLP tools, and a natural language parser in particular, is a subject of numerous research projects. Our approach to display parse trees is powered with a custom visualization module, based on WordBricks system. The proposed solution possesses a number of attractive features. It works on a mobile platform, and thus can serve as a module of a mobile language learning software system. It generates diagrams that are easy to understand for beginner-level language learners. The diagrams are very compact, and can be shown on a relatively small smartphone screen. All computationally expensive operations are performed on a server, so the application remains lightweight, and does not require hi-end devices to operate. We hope the system will be useful both to the specialists in natural language processing and foreign language learners.

References

[1] M. Munezero, C. S. Montero, M. Mozgovoy, and E. Sutinen, "EmoTwitter – A Fine-Grained Visualization System for Identifying Enduring Sentiments in Tweets," in *International Conference on Intelligent Text Processing and Computational Linguistics*, 2015, pp. 78–91.

[2] C. Collins, S. Carpendale, and G. Penn, "Docuburst: Visualizing document content using language structure," in *Computer graphics forum*, 2009, pp. 1039–1046.

[3] C. Culy, V. Lyding, and H. Dittmann, "Visualizing Dependency Structures," in *Conference of the German Society for Computational Linguistics and Language Technology (GSCL)*, 2011, pp. 81–86.

[4] I. Ortiz, M. Ballesteros, and Y. Zhang, "ViZPar: A GUI for ZPar with Manual Feature Selection," *Procesamiento del Lenguaje Natural*, vol. 53, pp. 181–184, 2014.

[5] M. Ballesteros and R. Carlini, "MaltDiver: A Transition-Based Parser Visualizer," in *6th International Joint Conference on Natural Language Processing*, 2013, pp. 25–28.

[6] A. Radford, *An introduction to English sentence structure*: Cambridge University Press, 2009.

[7] M. Akhlaghi and G. Zareian, “The Effect of PowerPoint Presentation on Grammar and Vocabulary Learning of Iranian Pre-University EFL Learners,” *Academic Research International*, vol. 6, no. 1, pp. 160–165, 2015.

[8] I. Çakir, “Instructional Materials Commonly Employed by Foreign Language Teachers at Elementary Schools,” *International Electronic Journal of Elementary Education*, vol. 8, no. 1, pp. 69–82, 2015.

[9] P. Krčelić and A. S. Matijević, “A Picture and a Thousand Words: Visual Tools in ELT,” in *8th International Language Conference on the Importance of Learning Professional Foreign Languages for Communication*, 2015.

[10] M. Park, M. Purgina, and M. Mozgovoy, “Learning English Grammar with WordBricks: Classroom Experience,” in *Proceecings of the 2016 IEEE International Conference on Teaching and Learning in Education*, 2016.

[11] M. Mozgovoy, “Grammar checking with dependency parsing: a possible extension for LanguageTool,” *Informatica*, vol. 35, no. 4, 2011.

[12] R. McDonald, F. Pereira, K. Ribarov, and J. Hajič, “Non-projective dependency parsing using spanning tree algorithms,” in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, 2005, pp. 523–530.

[13] J. Havelka, “Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures,” in *45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 608–615.

[14] B. Santorini, “Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision),” Technical Reports (CIS), Paper 570, University of Pennsylvania, Department of Computer and Information Science, 1990.

[15] Z. Le, “Maximum entropy modeling toolkit for Python and C++,” *Natural Language Processing Lab, Northeastern University, China*, 2004.

[16] A. L. Berger, Pietra, Vincent J Della, and Pietra, Stephen A Della, “A maximum entropy approach to natural language processing,” *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[17] N. Ide, C. Baker, C. Fellbaum, and C. Fillmore, “MASC: The manually annotated sub-corpus of American English,” in *In Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, 2008.

[18] J. Nivre, *CoNLL-U Format.* Available: http://universaldependencies.org/format.html.

[19] P. Jian and C. Zong, “Layer-Based Dependency Parsing,” in *PACLIC*, 2009, pp. 230–239.

[20] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: The Penn Treebank,” *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.

[21] V. Lyding, C. Culy, "Visualizing linguistic data: from principles to toolkits for doing it yourself", *AVML Conference*, 2012.