WordFreak: An Open Tool for Linguistic Annotation

Proceedings of HLT-NAACL 2003
Demonstrations , pp. 17-18
Edmonton, May-June 2003

Thomas Morton and Jeremy LaCivita
Department of Computer and Information Science
University of Pennsylvania
Philadelphia, PA 19104
{tsmorton,lacivita}@cis.upenn.edu

Abstract

WordFreak is a natural language annotation tool that has been designed to be easy to extend to new domains and tasks. Specifically, a plug-in architecture has been developed which allows components to be added to WordFreak for customized visualization, annotation specification, and automatic annotation, without re-compilation. The APIs for these plug-ins provide mechanisms to allow automatic annotators or taggers to guide future annotation to supports active learning. At present WordFreak can be used to annotate a number of different types of annotation in English, Chinese, and Arabic including: constituent parse structure and dependent annotations, and ACE named-entity and coreference annotation. The Java source code for WordFreak is distributed under the Mozilla Public License 1.1 via SourceForge at: http://wordfreak.sourceforge.net. This site also provides screenshots, and a web deployable version of WordFreak.

1 Introduction

The adoption of statistical methods in natural language processing (NLP) has also created a need for rapid annotation of new data. While statistical techniques for a variety of NLP processing tasks exist, applying these techniques to slightly different tasks, or even adapting an existing statistical model to a different domain requires the annotation of new text. Often the amount of data needed is minimal but setting up an interface which makes this annotation easy can be a time consuming task. This is especially the case when trying to use active learning to guide the annotation. Out of this set of needs, we began developing an annotation tool which eventually was named WordFreak. These needs led us to focus on making the software easily extensible and reusable, included the integration of automatic annotators, and developed the tools entirely in Java to facilitate multi-language and multi-platform support.

2 Components

WordFreak has a number of different types of components. These include two types of data visualization components, annotation scheme components which define the type of annotation which is taking place, and automatic annotators or taggers. Each of these components implements a common interface so that adding additional components only requires implementing the same interface. Additionally, WordFreak examine the environment in which it is run and gathers up any components which implement one of these interfaces. This allows components to be added to it without re-compilation of the original source code.

2.1 Visualization

The visualization components are called Viewers and Choosers. Prototypically the Viewer is where the user looks to perform the annotation. WordFreak currently contains four such Viewers which display text, trees, a concordance, and tables respectively. While particular viewers are better suited to certain tasks, multiple viewers can be used simultaneously. The viewer are displayed in a tabbed-pane for easy access but can also be removed if the user wishes to see multiple views of the data simultaneously.

The second type of visualization components are called Choosers. These are typically used to display the choices that an annotator needs to make in a particular annotation scheme. Choosers are specific to an annotation scheme but are constructed via a set of re-usable chooser components. For example, a typical chooser consist of a navigation component which allows the user to move

through annotations, a buttons component parameterized to contain names of the relationships your annotating, and a comment component which allows a user to make a free-form comments about a particular annotation. Currently there are chooser components for the above described tasks as well as tree representations which have been used to display annotation choices for tasks such as coreference and word sense disambiguation.

2.2 Task Definitions

Adapting WordFreak to new annotation tasks is a common task. This has led us to try and minimizes the amount of new code that needs to be written for new task definitions. We have used a two tiered approach to new task definitions.

The first employs the inheritance mechanisms available in Java. To define a new task or annotation scheme one can simply sub-classes an existing AnnotationScheme class, initialize what types of annotations the new task will be based on, define the names of the set of relationships you will be positing over these annotations, and specify what chooser components you want to use to display this set of names. While many options can be customized such as keyboard short-cuts, color assignment to particular relationships, and constraints on valid annotation, the default assignments use the most likely settings so a typical annotation scheme requires under 100 lines of well delimited code. Annotation schemes which involve more complicated interactions such as coreference and word sense disambiguation have taken approximately 300 lines of code specific to that task.

The second mechanism, which is currently being developed, allows a task to be parameterized with an xml file. This can be applied if an existing annotation scheme similar to your task has already been developed. At present we have used this mechanism to customize named-entity and coreference task which are similar to their corresponding MUC or ACE tasks. Likewise this mechanism can be used to customize the tag sets used for different types of tree-banking tasks.

2.3 Automatic Annotators

We have integrated a number of automatic annotators to work with WordFreak. These include sentence detectors, POS taggers, parsers, and coreference revolvers. The APIs these annotators implement allow them to optionally determine the order that annotation choices are displayed to the user as well as provide a confidence measure with each annotation they determine automatically. The first mechanism is quite useful for tasks which have a large number of potential choices such as POS tagging or coreference resolution in that the most likely choices can be displayed first. The confidence measure can be used for active learning or just to assist in the correction of the automatic annotator. We are currently in the process of adapting open source taggers to be used and distributed as plug-ins to WordFreak.

3 Source Code

WordFreak's source code is entirely written in Java. This has allowed us to deploy it on a number of platforms including Windows, Mac OS X, Solaris, and Linux. Java's built-in language support and use of Unicode as the underlying representation of strings has made allowing WordFreak to annotate non-English text relatively simple. Currently we have successfully developed annotation schemes for Chinese and Arabic. Finally we have recently released the source code on SourceForge under the Mozilla Public License 1.1. This should allow WordFreak to be extended by others as well as provide a mechanism for wider contribution to this tool.

4 Future Work

We are currently planning on developing an I/O plug-in interface so that WordFreak can be easily extended to support additional file formats. We also have plans to develop a viewer which would render HTML while allowing annotations to reference the underlying text.

5 Conclusions

We have developed an open linguistic annotation tool which can be easily extended via a large number of reusable components. It supports automatic annotation and active learning for rapid annotation of new text. WordFreak is written entirely in Java and can be used with multiple languages and platforms.