Annotation Tools for Large-Scale Corpus Development: Using AGTK at the Linguistic Data Consortium

Kazuaki Maeda and Stephanie Strassel

Linguistic Data Consortium
University of Pennsylvania
3600 Market Street, Suite 810
Philadelphia, PA, USA
{maeda, strassel}@ldc.upenn.edu

Abstract

Large-scale corpus development demands substantial infrastructure. As part of this infrastructure, the Linguistic Data Consortium (LDC) has adopted the Annotation Graph Toolkit (AGTK) as a primary resource for annotation tool development. This paper reports on LDC's experiences using AGTK to develop and implement highly customized annotation tools for a variety of large-scale corpus creation efforts. We describe two primary tools that are currently in active use at LDC, one speech- and one text-based, as well as other new AGTK-based annotation tools. We also describe the use of AGTK to develop tools for comparing and adjudicating divergent annotations in order to produce gold standard evaluation data and to measure inter-annotator consistency. Finally, we discuss various issues in creating AGTK-based tools across a wide range of annotation tasks and divergent research areas.

1. Introduction

The Linguistic Data Consortium (LDC) at the University of Pennsylvania supports language-related education, research and technology development by creating and sharing linguistic resources: data, tools and standards. In many cases these resources include richly annotated training, development and evaluation data to support sponsored common task technology evaluations. Producing large volumes of richly annotated data under strict deadlines demands substantial infrastructure. As the needs of research communities expand to include more sophisticated annotations in a growing variety of languages, the Annotation Graph Toolkit (AGTK) has emerged as a critical component of this infrastructure.

2. The Annotation Graph Toolkit

The Annotation Graph Toolkit (AGTK) is based on the annotation graph model Bird and Liberman have developed for expressing the logical structure of linguistic annotation (Bird and Liberman, 2001). An annotation graph is a directed acyclic graph where edges are labeled with fielded records, and nodes are (optionally) labeled with time offsets. Bird and Liberman demonstrated that it can encode a great variety of existing annotation types.

AGTK is a collection of software for the development of annotation tools, instantiating the annotation graph model (Maeda et al., 2002). AGTK includes application programming interfaces (APIs) for manipulating annotation graph data and importing data from other formats, wrappers for scripting languages, graphical user interface (GUI) components specialized for annotation tasks, and demonstration applications.

2.1. The AG Library

The AG library, which provides APIs for creating and manipulating AG data, is the main component of AGTK. Examples of basic functions provided by the API include the following:

- CreateAnnotation, CreateAnchor, CreateTimeline
- SetStartOffset, SetEndOffset, SetFeature, GetFeature
- GetAnnotationSet, Load, Store

The core AG library is implemented in C++; however, AGTK provides wrappers for using the AG library in various programming languages, including Python, Tcl, Perl and Java. All annotation tools described in this paper are written in Python.

Another important component of the AG library is the plug-in file I/O architecture. Various file I/O modules that import and export data between AG and other file formats have been written. Due to its unique plug-in architecture, new modules can be written and installed without recompiling the C++ library.

2.2. Applications

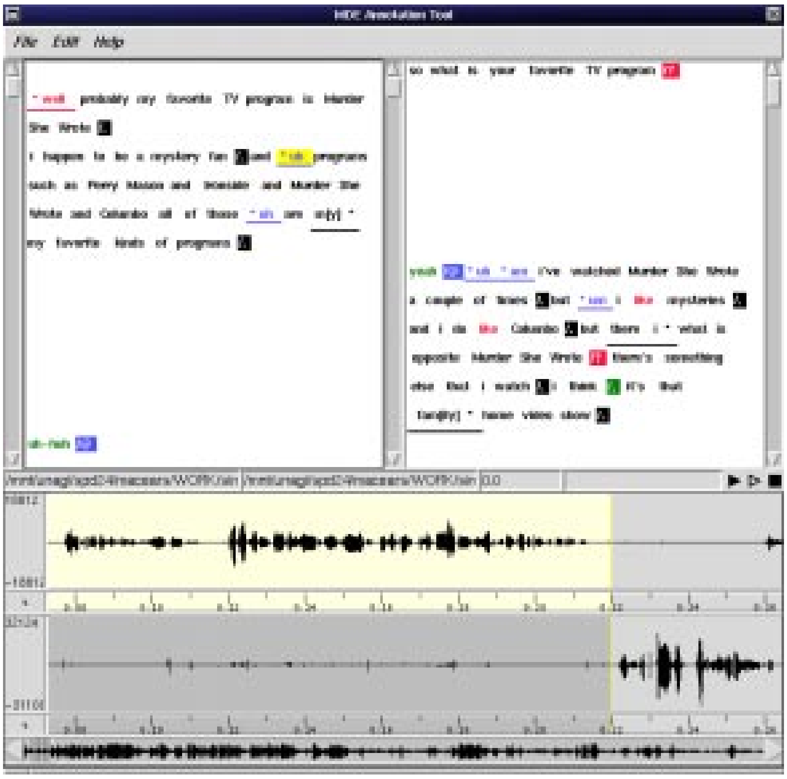
Various tools that use the AG library have been developed and used in annotation projects. Bird, et. al. (2002) describes some of the tools developed as part of AGTK, including TableTrans, a spreadsheet-style audio annotation tool and MultiTrans, a multi-channel transcribing tool. As new tools are developed, they are added to the CVS repository of the AGTK web site (`http://agtk.sf.net/`). All software components of AGTK are open-source.

In the following sections, we report on new annotation tools we have developed and used extensively in large-scale annotation projects at LDC.

3. Speech Annotation Tools

3.1. Metadata Extraction

In the speech domain, LDC creates a variety of linguistic resources for the DARPA EARS program. In one


MDE Annotation Tool
File
Edit
Help
/mnt/unagi/spd2/macsars/WORK/sin
/mnt/unagi/spd2/macsars/WORK/sin
0.0
10812
-10812

^well probably my favorite TV program is Murder She Wrote

i happen to be a mystery fan and ^uh programs such as Perry Mason and Ironside and Murder She Wrote and Columbo all of those ^uh are m[y] ^ my favorite kinds of programs

uh-huh

so what is your favorite TV program

yeah ^uh ^um i've watched Murder She Wrote a couple of times but ^um i like mysteries and i do like Columbo but there i ^ what is opposite Murder She Wrote there's something else that i watch i think it's that fam[ily] ^ home video show

Figure 1: MDE Annotation Tool

EARS track, technology providers must create Speech-to-Text systems whose outputs are substantially richer and more accurate than is currently possible. A secondary EARS track, Metadata Extraction (MDE), targets systems that can refine the raw STT output into forms that are more useful to humans and downstream processes. In 2003, LDC defined an annotation task to support MDE research and created approximately 75 hours of English training, development and evaluation data.

Because the first year of MDE research saw an ever-evolving task definition and a compressed timeline for data production, having task-specific, highly customized and easily modifiable annotation tools was essential. Using AGTK, LDC was able to create an effective annotation tool in a short time, and the tool proved to be highly modifiable in response to the evolving task definition and increasing demands for annotation speed and accuracy. The MDE annotation task requires annotators to flag non-content words like filled pauses and discourse markers, identify and characterize sections of disfluent speech, and create boundaries between natural breakpoints in speech (called SUs). Users can easily highlight relevant spans of text, play the corresponding segments, and then record annotation decisions with a few mouse clicks or keystrokes. Annotators can further verify their decisions by viewing the resulting ”cleaned-up” transcript that removes fillers and disfluencies and displays each SU as a separate line of text.

Figure 1 shows a screenshot of the MDE annotation tool.[1]

3.2. Multi-speaker Transcription Tool

In response to increasing demands for carefully transcribed meeting data, LDC has been developing a new AGTK multi-speaker transcription tool called XTrans. Many good transcription tools already exist, but they are not typically optimized for transcribing recordings of multiple speakers on a single channel. While the existing AGTK MultiTrans tool provides basic functionalities of this kind, the new XTrans tool incorporates the best features of existing AGTK transcription tools while solving the problem of multi-speaker single channel transcription.

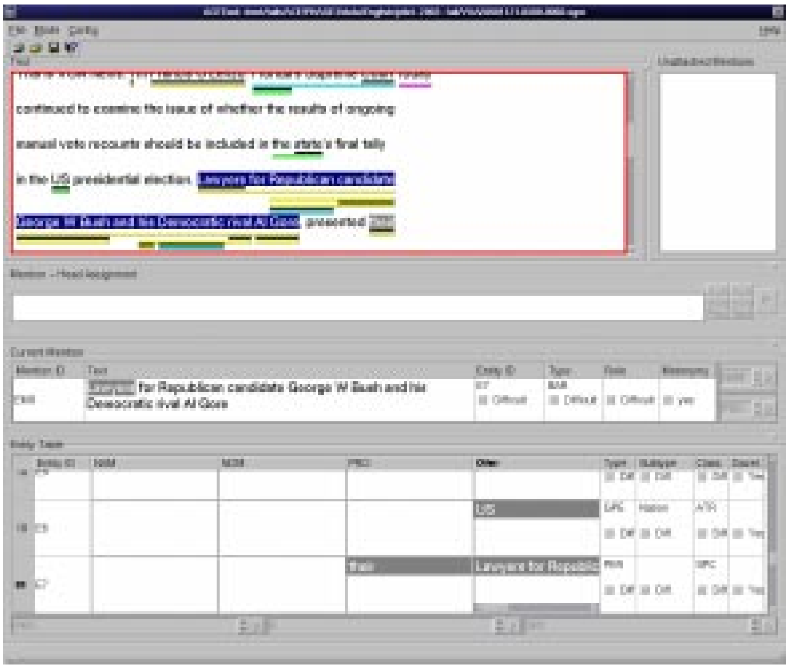
4. Text Annotation Tools

4.1. ACE Entity, Relation and Event Annotation Tool

Among the many text-based annotation efforts at LDC, the Automatic Content Extraction (ACE) Program presents perhaps the biggest set of challenges. Currently operating under the DARPA TIDES umbrella, ACE targets development of information extraction technology to support automatic processing of source language data. This includes technologies that automatically detect and characterize entities, relations between entities and atomic events. In 2003, LDC provided hundreds of thousands of words of annotated data as well as annotation task definitions to support technology evaluations in English, Chinese and Arabic.

As data requirements for ACE have increased in scope, volume and complexity over the past several years, we have relied crucially on AGTK to enable rapid creation of high-quality resources. Designed with input from the annotation team, the ACE tool relies on color-coded underlining to display layers of annotation across spans of text. This is

[1] This tool uses WaveSurfer (Sjölander and Beskow, 2000) as the waveform display module.



continued to examine the issue of whether the results of ongoing manual vote recounts should be included in the state's final tally in the US presidential election. Lawyers for Republican candidate George W Bush and his Democratic rival Al Gore presented

for Republican candidate George W Bush and his Democratic rival Al Gore

Figure 2: ACE Annotation Tool

particularly useful in ACE, where multiple annotations are often embedded or overlapping. The tool is highly specialized for ACE annotation: it requires users to record valid annotation decisions for one task before moving on to the next decision point, and includes customized modules for entity, relation and event tagging. As with all AGTK tools, the ACE tool is platform-independent, and supports multilingual annotation, including bidirectional text display (as in Arabic). Figure 2 shows a screenshot of the ACE annotation tool.

4.2. Simple Entity Annotation Tool for TIDES Surprise Language Exercise

In 2003, LDC participated in TIDES sponsored project called the Surprise Language exercise as one of the main coordinators as well as a primary data resource provider. This was a project in which participating sites assembled linguistic resources within one month's time frame after a target language – one for which resources have not been extensively distributed – was announced. The announced language for the 2003 Surprise Language exercise was Hindi.

During the exercise, LDC identified native speakers of Hindi in the local area, trained them to perform a named-entity annotation task, and produced a training corpus and an evaluation corpus for automatic named-entity extraction systems that the participating sites created. In order to facilitate quick learning of the annotation task and tool, we developed a very simple named-entity annotation tool with a point-and-click interface using AGTK. This tool is capable of displaying Hindi text correctly, and the annotators found it very easy to learn.

5. Annotation Adjudication Tools

A concern for many large-scale annotation projects is demonstrating and measuring inter-annotator consistency through dual annotation and discrepancy resolution. Using AGTK we have developed a number of solutions that allow project managers to easily incorporate dual annotation into the regular data production pipeline. This permits managers to review the output of dually annotated data, compare results and adjudicate differences in order to create "gold standard" files. Currently, there is an adjudication tool for the MDE project and another for the ACE project. These tools take two annotation files for the same source data, highlight the differences and ask for the adjudicator's judgment on each discrepancy.

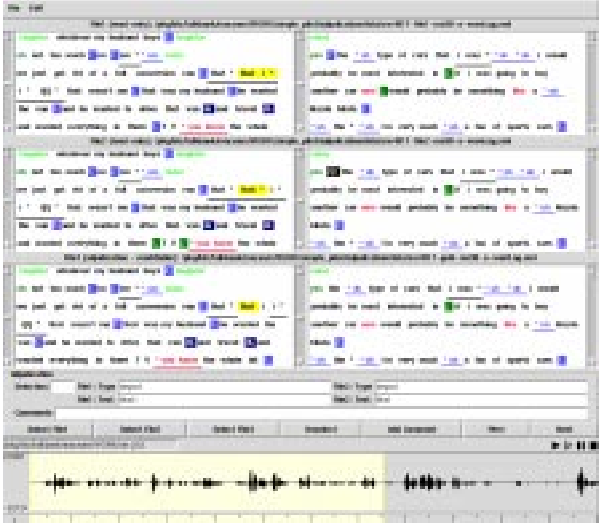


Figure 3: MDE Adjudication Tool

6. Discussion

6.1. AG Library

The AG model is a simple, yet powerful and flexible model to represent linguistic annotations. This model has been able to represent all of the kinds of linguistic annotations that we have dealt with. At the same time, the flexible nature of this model allows data to be represented in many ways. For example, annotations that always cover the same extents of speech or text can be represented with either multiple annotations sharing the same start and end anchors or one annotation with multiple features. How to express annotation data with AG is governed by the applications.

In order to address this issue, we have used an additional layer of APIs outside of the AG library in the projects reported in this paper. This layer ensures that the AG data structures created by the applications, such as annotation tools, adjudication tools, and quality control tools, are consistent and compatible with one another.

For the MDE and ACE projects, we used an object-oriented approach to create an API layer that wraps the AG API. Examples of classes defined for the ACE project include *Entity*, *EntityMention*, *Relation*, *RelationMention*, *Event* and *EventMention*. Each of these have methods, such as *SetType*, *SetSubType*, *SetStartOffset*, *SetEntity*, and so on. These methods, in turn, use functions provided by the AG API to create the AG data. All applications for annotation, adjudication and quality control use these APIs to create or access AG data for the projects.

6.2. File Format Issues

The AG library provides two file formats that can store AG data: the xml-based *AG* format and the plain-text-based *CAG* format. *CAG* stands for *Compact AG*, and this format can store the same information as the AG format. Using these file formats, any data created by the AG API can be stored without losing information.

File formats are often determined by many factors. Sometimes there are legacy file formats we are required to use. Other times the project sponsors or the research communities request data to be delivered in specific file formats. When files formats are neither of the AG native formats, we have used one of the following two approaches:

- Use the AG model for the internal representation of the annotation tools only; have the tools read and write the project specific file format directly. (Example: Surprise Language tools, transcription tools)

- Use the AG or CAG format for in-house file storage; use a converter to convert files in these formats to the final file format. (Example: ACE tools, MDE tools)

Both methods use a file I/O module we have developed for each project. While the first approach simplifies the data production pipeline, the second approach has the advantageous capacity to store additional in-house annotation, such as comments and *difficult decision* tags. Also with this approach, it is easy to adjust to changing file format specifications.

6.3. User Interface Issues

The user interface design is a substantial factor that determines the efficiency of an annotation process. Even though we have released GUI components as part of AGTK tools, the design and implementation of good user interfaces for complicated annotation tasks is still a significant challenge.

Being a data creator and having large scale in-house annotation projects have helped the software developers at LDC to identify issues in good user interface design. Often, requirements for good user interfaces are common among projects, even when the tasks are different. Our on-going plan is to reflect our experiences in creating better reusable GUI components and release them as part of AGTK.

7. Conclusion

The annotation graph data model has proven adequate for handling the full range of annotation tasks currently required, and it provides a very flexible model to represent evolving data specifications. The AG XML file I/O module can store data in a stand-off annotation format. With the file I/O plug-in architecture provided by AGTK, converting between the native AG data representation and legacy data formats required by particular research communities is straightforward. In addition, the modular design of graphical user interface components in AGTK helped us create task-specific tools in a timely fashion. Even though the creation of new tools for complex annotation is not a simple task, our adoption of AGTK as the primary resource for annotation tool development has helped LDC meet the sizeable data requirements of multiple, concurrent, evolving technology programs, despite limited programmer time and tightly constrained data production schedules.

8. References

Bird, Steven and Mark Liberman, 2001. A formal framework for linguistic annotation. *Speech Communication*, 33:23–60.

Maeda, Kazuaki, Steven Bird, Xiaoyi Ma, and Haejoong Lee, 2002. Creating annotation tools with the annotation graph toolkit. In *Proceedings of the Third International Conference on Language Resources and Evaluation*.

Sjölander, Kåre and Jonas Beskow, 2000. WaveSurfer – an open source speech tool. In *Proceedings of the 6th International Conference on Spoken Language Processing*. `http://www.speech.kth.se/wavesurfer/`.