A Practical Introduction to ATLAS

Christophe Laprun, Jonathan G. Fiscus, John Garofolo, Sylvain Pajot

National Institute of Standards and Technology
100 Bureau Drive Mail Stop 8940
Gaithersburg, MD 20899-8940
{claprun, jfiscus, jgarofolo, pajot}@nist.gov

Abstract

In this paper we introduce the ATLAS [Architecture and Tools for Linguistic Analysis Systems] architecture by describing how a proposed annotation task would be modeled using ATLAS concepts. We first present a brief motivation for this work and then move on to describe an example annotation task that will serve as the basis for this paper. Next we model the example task using ATLAS. We describe how the introduction of the new Meta-Annotation construct enables us to move forward by allowing unambiguous description of the annotation task and modeling of relations between the different annotation elements. We conclude by describing the current state of the ATLAS framework.

1. Introduction

Annotated corpora are a central component of research in human language technology. As corpora have proliferated across languages, disciplines, and technologies, the lack of common exchange and storage formats has become a critical problem. This profusion of formats has made reusing annotated data or adapting existing tools for new annotation tasks significantly more difficult.

Standardization of tag sets – an approach we tried with our Universal Transcription Format (NIST, 1998) – is of moderate usefulness since language research is usually an open-ended task, subject to constant revision as the research domains change and the theories evolve.

A solution to this "bazaar of tools and formats" (Bird et al., 2000) is to interpose a generic annotation model via which annotation data is manipulated. This level of indirection[1] has several benefits. In particular, physical storage and application logic are clearly separated. This separation did not exist when most tools were written to directly read and write data using a specific format, hindering their reusability. Interposing an intermediate data model permits tools to be written in terms of the generic abstractions instead of having to deal with the particulars of specialized data formats. Tools can therefore work with any data that can be represented using the abstractions of the data model regardless of their actual physical storage.

The recently updated ATLAS [Architecture and Tools for Linguistic Analysis Systems] framework, which we introduce in this paper, makes use of such a generic data model. ATLAS provides an architecture targeted at facilitating the development of linguistic annotation applications[2] and is comprised of four main components: a data model, an Application Programming Interface (API), the ATLAS Interchange Format (AIF, 1999) and the Meta-Annotation Infrastructure for ATLAS (MAIA[3], 2002). The data model at its core provides the abstractions on which the rest of the framework is built. These abstractions can be implemented using any full-featured programming languages. NIST has created a Java instantiation of the data model. This implementation provides a set of objects that can be used to quickly develop linguistic applications. These objects each publish operations via which their data can be manipulated and behavior controlled. The ensemble of these operations defines the ATLAS API. ATLAS annotations can be serialized to XML using AIF to facilitate their exchange and reuse. Finally, MAIA, an important new component, was added to permit constraining of ATLAS' generic constructs for specific needs.

ATLAS began as a collaboration between the LDC, MITRE and NIST in 1999 following Bird and Liberman's seminal work on Annotation Graphs (Bird and Liberman, 1999) that demonstrated commonality across a diverse range of annotation practices and defined a formalism based on labeled, directed acyclic graphs. ATLAS was formally introduced at LREC 2000 in Athens, Greece and was the subject of (Bird et al., 2000).

After LREC, the LDC moved on to implement Annotation Graphs – also called *AGs* or *ATLAS level 0* (AG, 1999) – to address immediate needs in annotation infrastructure for linear signals. NIST, on the other hand, decided to defer immediate implementations to pursue the development of the generalized version of ATLAS. NIST's version encompasses signals of arbitrary dimensions because, as well-suited to linear signals as it was, the logical model provided by AGs did not scale well to higher-dimensional cases. A first implementation of the generalized model was made available in April 2001. This first release allowed us to gather valuable feedback, from which we decided to evolve the framework towards better expressiveness while fixing existing problems. The data model was modified to accommodate new ideas and significant parts of the implementation were rewritten. A Beta release of this redesigned implementation was released at the end of January 2002.

This paper focuses on this release of the generalized version of ATLAS (subsequently referred to as "ATLAS"). We present an introduction to ATLAS from a

[1] "Computer Science is the discipline that believes that all problems can be solved with one more layer of indirection." – Dennis DeBruler

[2] Atlas was (in the Greek mythology) the Titan condemned to bear heavens upon his shoulders. ATLAS is designed to bear the complexity of annotation management for the benefit of linguistic applications!

[3] Maia was one of Atlas' daughters in the Greek mythology.

practical point of view in the context of our new Rich Transcription (RT) evaluation project. We first describe part of the annotation task for this project. Next, we model these annotations using ATLAS concepts that we introduce along the way. We then briefly introduce the Meta-Annotation concept before concluding with a discussion of the benefits of using ATLAS and a summary of the current status of the framework. Note that this paper focuses on providing an overview of the core ATLAS concepts and how they can be used. Later papers will describe specific aspects of the architecture in greater detail.

2. Example Annotation application: rich transcription

In order to present a practical approach to using ATLAS, we will work through an ATLAS example based on an emerging NIST language technology evaluation.

For over fifteen years, NIST has been conducting common evaluations of the performance of automatic speech recognition technology. Traditionally, these evaluations have focused on the accuracy of automatically-generated orthographic word transcriptions. The technology is now evolving and future speech recognition systems will be required to output a variety of integrated metadata as well as orthography to produce data which will be more useful for downstream processing and human readability. We refer to these enriched systems as "Rich Transcription" (RT) systems. To support research and evaluation in automatic RT, NIST is conducting a pilot evaluation, Rich Transcription 2002 (RT-02, Garofolo et al., 2002) that, in addition to the evaluation of orthographic transcription, will also address the automatic production of metadata. Given time and resource constraints, the metadata annotation to be explored in the pilot evaluation will involve only the segmentation of an audio excerpt by speaker and then clustering the speaker segments according to speaker identity within the excerpt. However, a variety of other metadata annotations can be envisioned for future RT evaluations including, but certainly not limited to:

- Speaker segmentation and identification
- Sentence or phrasal unit segmentation and classification
- Acronym detection and expansion
- Verbal edit detection, identifying regions of disfluency
- Named entity detection/classification
- Numeric expression detection/classification
- Temporal expression detection/classification

Note that some of these detection tasks involve the generation of sub-recognition or classification information. This diversity of output requires a new flexible non-monolithic approach to the development of evaluation software.

The goal of the speaker segmentation and identification task is to divide the audio signal excerpt into homogeneous units spoken by the same person. The task involves two concepts, identifying a set of speakers in the recording and maintaining a list of speaker segments where that speaker was talking. The speaker concept encapsulates global attributes about the speaker, for instance the gender, dialect, and etc., while a speaker segment is used to record the extent of a particular speaker utterance. Once speaker segments are identified, the next step is to transcribe what was said. To represent the orthography, we are using the generic concept of ‘orthographic units’ (OUnits).

OUnits are a way to generalize the transcription of spoken units of language. There are many types of units that need to be transcribed including the continuum from non-speech verbalizations to spoken words. In English, these are represented with printed words, in Mandarin, printed characters. As such, OUnits can have a variety of attributes that will further define their content.

While it is conceivable to attribute metadata directly to the time stream, it is often expedient to attribute these phenomena to the transcribed OUnits. Therefore, once OUnits are established, other forms of metadata can then be identified via the set of OUnits that constitute them. For example, verbal edit detection can be accomplished by identifying the set of OUnits involved in a verbal edit. Detection of acronyms, Named Entities (NEs), numeric expressions and temporal expressions can be identified in the same way. Since the OUnits are related to a speaker through a speaker segment annotation, these metadata can be traced back to the speaker they are attributed to.

We envision that over time, RT systems will become very complex, generating metadata covering a variety of linguistic and non-linguistic phenomena. To support current and future rich transcription evaluation needs, we are developing evaluation software leveraging the ATLAS framework. Using ATLAS, we are able to design a relatively generic evaluation software engine that can elegantly accommodate a vast variety of different metadata types.

For the purpose of this paper, we will create a hypothetical integrated RT annotation from an audio excerpt from a speaker containing a word transcript and a variety of metadata annotations. Our hypothetical excerpt will be a recording of the spoken sentence: "Joe f- fell off the log", containing a verbal edit ("f-") and a NE ("Joe").

Next, we outline the steps taken to create the transcript described above and introduce ATLAS’ main concepts.

3. Annotating using ATLAS

The entry point to ATLAS is the core linguistic annotation ontology upon which it is built:

> An annotation is the fundamental act of associating some content to a region in a signal.

These four primary concepts are represented within the ATLAS framework by *Annotation*, *Content*, *Region* and *Signal* constructs. The preceding notation defines a convention that we follow in this paper: ATLAS concepts are formatted using *this font and style* to allow us to distinguish them.

Even though ATLAS’ core ontology is fairly simple, it is nevertheless surprisingly powerful when it comes to expressing complex annotation schemes. The annotation process using ATLAS can be broken down into three major steps:

- Identification of regions of interest in a signal
- Association of content with these regions to form annotations
- Linking related annotations together

Next, we detail each step in sequence.

3.1. Identification of regions of interest in a signal

The annotation process begins by specifying a *Signal*. An ATLAS *Signal* is an immutable, N-dimensional space containing phenomena that might be the target of *Annotations*. Even though typical *Signals* can be equated to physical signal files (speech waveforms, newswire text, video or other more complex data with higher dimensionality), it is not a necessity. In ATLAS, a *Signal* is an entity that identifies a logical (as opposed to a physical file) target for *Annotations* and can thus refer, for example, only to the left track of a stereo recording. ATLAS also does not prescribe to any single format or dimensionality for physical signals, but there must be a way to define an unambiguous coordinate system for the *Signal*.

Once a *Signal* is established, it is possible to begin annotating it. The first step is to identify a region within the *Signal* relative to a phenomenon of interest. In ATLAS, a *Region* is an abstraction for identifying an area of the *Signal* space. *Regions* are delimited by a set of coordinates that mark specific locations delimiting the interesting area. These markers are modeled (in ATLAS) by the *Anchor* construct, thus called because they are used to “anchor” annotations to *Signals*. *Anchors* are the only ties that annotations have to the physical structure of the signal and the only ATLAS concept that is signal-specific. *Regions* use as many *Anchors* as needed to index into *Signals*. The *Region* construct encapsulates the complexity of multidimensionality by abstracting away the specificity of the underlying signal. This is a particularly important aspect of ATLAS since it allows the framework to evolve and scale gracefully, when confronted with new classes of signals, without requiring change to the basic ontology.

For our RT example, the *Signal* is a speech waveform and time is the dimension along which *Anchors* are specified. *Regions* of interest are thus intervals in time and are modeled in ATLAS via the use of two *Anchors* – one marking the beginning of the *Region*’s extent and the other marking the end of it.

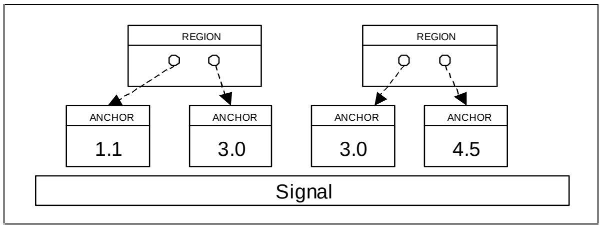

Figure 1: Identifying regions in a signal

Figure 1 shows two Regions for our RT example. ATLAS constructs as described above are represented using boxes. The number in the Anchor boxes records the instant in time that the Anchor points to in the Signal. The first interval of interest in our Signal thus spans from the time recorded by the first Anchor it references to the time recorded by the second Anchor. Note that this information is signal specific: a different kind of signal (a video for example) would have required a different coordinate scheme. Anchors encapsulate the specificity of signals (coordinate scheme, units, etc…) allowing signal-specific information to be strictly localized.

3.2. Association of content with the regions to form annotations

Once an interesting linguistic phenomenon has been located in the Signal via the use of a Region, an Annotation is built by associating some content with it. Content constructs represent (in ATLAS) any information that annotators would like to specify about the linguistic event occurring in the specified Region. The information can be a simple data type like a string or a complex data structure like those used in typical programming languages.

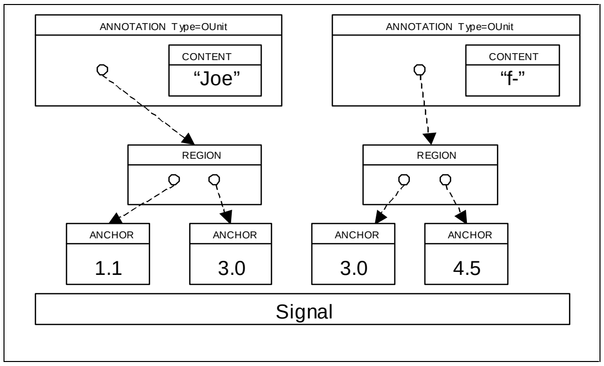

Figure 2: Example of a STT output

Figure 2 shows an example of OUnit Annotations as would be produced by a STT system. We have taken the Regions that we identified in the previous step and associated them with the text transcription associated to the interval identified in the audio signal. This association is recorded in the context of an ATLAS Annotation. We have thus created two OUnit Annotations recording the transcription associated with the intervals that we previously identified.

Embedded boxes represent a composition (tight coupling) relationship. In a composition, parts cannot be decoupled from their parent, meaning that their existence is tied to their parent’s. For example, Content is tightly coupled to its parent *Annotation*: it cannot exist independently of its defining Annotation. Arrows represent references, which are a weaker form of coupling: a reference associates concepts but the participating elements’ existences are not tied to one another. For example, an Annotation has a reference to a Region but the Region exists independently of the Annotation (which can be destroyed without impacting the existence of the Region).

References support another important feature of ATLAS: the reuse of annotation data. Some ATLAS entities (such as Regions, Anchors and Annotations) are reusable via the use of references, thus allowing elements defined in a given context to be reused in different contexts. It is worth noting, though, that reusability is a capability that is judiciously bestowed only to certain ATLAS entities because it comes with added complexity. In particular, in order to be reusable, an ATLAS element must be identifiable by having a unique identifier (Id) explicitly assigned to it. The management of these elements’ identity incurs additional processing. However, the advantages in expressiveness and power far outweigh the inconvenience, especially considering that this complexity is totally handled by the framework (in the Java implementation). Furthermore, reusability of Annotations, which will be discussed later, gives ATLAS most of its flexibility.

Identified constructs are distinguished by their Id regardless of the value they hold (if any) whereas, in the case of composition, two elements holding the same values are interchangeable. Note that different semantics can be expressed by distinguishing between value and identity. In the previous figure, note that two Anchors share the same value but are still duplicated. This is because, for our annotation task, OUnits are not necessarily contiguous. For this reason, the fact that two Anchors share the same offset is purely coincidental. If we wanted to enforce the contiguity of OUnits, we would have made Regions share their Anchors. In this situation, modifying the offset of a shared Anchor allows every Region that references it to access the new value without having to modify them.

3.3. Linking related annotations together

Now that we have an ATLAS representation of a STT system output, RT metadata information can be encoded by extending the base transcript. We are interested, for our particular example, to associate a NE Annotation to the OUnit "Joe" and a verbal edit Annotation to the Ounit "f-". In addition, we want to link together all speaker segments associated with a given speaker.

To do so, we will use the Children construct to model relations between Annotations. The purpose of Children is to maintain a list of Annotations (via Annotation references) that are descendants of a parent Annotation. For example in TIMIT (Garofolo et al., 1986), words are composed of a set of phones, which would be modeled in ATLAS by the fact that word Annotations maintain a list of phone Annotation references via a Children subordinate. For our RT example, we model the relations that a speaker has with its subordinate speaker segments by adding a Children subordinate to the speaker. This Children element will contain a list of the speaker segment Annotation references related to the considered speaker. Each speaker segment is itself linked to a list of OUnit (part of this speaker segment) Annotations via a Children element. A speaker can also be linked to NEs or verbal edits since we want to be able to identify which speaker uttered a given NE or verbal edit. We would therefore model our speaker Annotation as containing three Children elements: one for NEs, one for verbal edits and one for speaker segments.

Since Children elements link to the descendant Annotations via references, it is possible to build overlapping hierarchies reusing the same Annotations without the kinds of problems that occur in other purely hierarchical annotation schemes. For example, our NE Annotations reference the same OUnit as speaker segments. This means that reuse is optimal since already created *Annotations* can be reused in lots of different contexts.

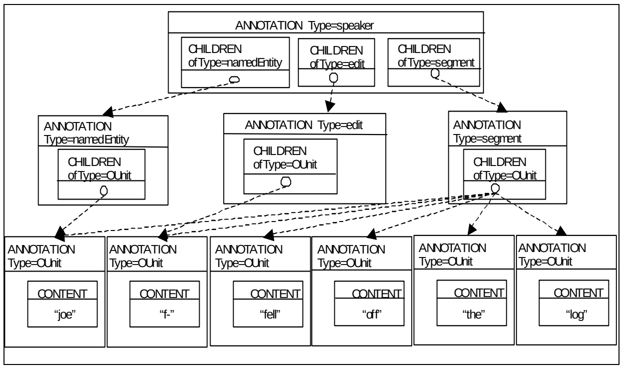

Figure 3: Example metadata annotation

Figure 3 represents the organization of Annotations (with Regions, Anchors and Signal left out). Each Children element can contain a separate hierarchy, but still make use of the OUnits for that speaker. In reality, NEs could extend over speaker segment boundaries but since ATLAS uses the reference mechanism to associate OUnits, speaker segments and NEs independently, this is trivial to represent. It is important to note that annotation schemes that rely purely on inline SGML such as UTF 0 to markup text are not able to handle this type of representation because SGML is strictly hierarchical.

Another interesting aspect of Children is the fact that parent Annotations can use their linked subordinate elements to derive their Content and/or Region. In our example, a speaker segment Annotation derives its Region from the union of the Regions referenced by its subordinate OUnit Annotations. This allows, for example, the modification of sets of words without having to modify every sentence that depend on them.

4. ATLAS Object Type Definitions with MAIA

The ability to generate complex annotations means that an application developer could be forced to devote a significant portion of the development efforts (as was the case in ATLAS' previous implementation) to issues like structural integrity or consistency checking. In order to facilitate application development, ATLAS 2.0 introduced the Meta-Annotation concept to address these needs.

A Meta-Annotation is an unambiguous way to constrain ATLAS' generic constructs for specific applications. In essence, a Meta-Annotation corresponds to a class definition in an object-oriented language. MAIA (Meta-Annotation Infrastructure for ATLAS) implements the Meta-Annotation concept for ATLAS. It provides a scheme language that allows type definitions to be declared using a simple, XML-based syntax. The ATLAS framework can then dynamically and automatically interpret these type definitions. MAIA also provides services (such as the loading and saving of types) that can be utilized by ATLAS implementations. MAIA specifically addresses two issues that we examine next: structural integrity and relations between Annotations.

4.1. Structural integrity

The RT-02 annotation task that we are working with is well defined and makes use of such concepts as speakers, speaker segments, OUnits, etc. More generally, for any given annotation task, users are not interested in working with generic Annotations. They want to work with specific concepts such as OUnits or speaker segments. ATLAS defines a very generic data model that is designed to be able to model a wide range of annotation tasks. Because of this genericity, ATLAS' constructs are minimally constrained (because under-specified by design) and thus not optimally useful: a generic Annotation, for example, encompasses every possible annotation that fits in ATLAS' paradigm. A means of constraining generic constructs is therefore needed in order for the framework and ATLAS applications to handle a particular Annotation as an OUnit, for example.

The realization of the generic Annotation in an OUnit Annotation is done via its type. Types are metadata that are associated with an ATLAS construct to indicate that this particular element models a concept specific to the considered annotation task. Any ATLAS element to which a given type is assigned will behave like any other element of the same type. Moreover, elements with the same type will share the same structure. ATLAS types are thus very similar to classes in object-oriented parlance. However, to be truly useful, ATLAS applications need to be able to automatically interpret the type information without requiring user intervention or developer effort.

MAIA offers this kind of service to application developers by providing an unambiguous type definition for the annotation task at hand. These type definitions are then used by MAIA to automatically create type constructs that allow the ATLAS framework (and applications depending on it) to perform structural checks by ensuring that elements that are supposed to be of a given type have the correct structure and behave as expected. This is automatically handled by the framework, freeing developers of the burden of having to take care of this. Applications can thus be written in terms of the generic abstractions but still be automatically tailored to custom needs by leveraging type information.

4.2. Relations between Annotations

Another aspect of working with linguistic data, apart from concept modeling, is the difficulty of dealing with relations between these concepts. In our explanation of the Children concept, we specified that speaker Annotations are linked to NE, verbal edit and speaker segment *Annotations* whereas speaker segment Annotations are linked to OUnit Annotations. To make a truly valuable product, an ATLAS implementation should be able to automatically enforce this kind of very specific constraint without hindering the generic data model or requiring code to be written.

MAIA automatically takes care of these details by linking type constructs together and ensuring that, for example, when subordinates are added to an Annotation, they are of the right type. MAIA will eventually support more elaborate typing including support for value and range constraints.

5. Conclusion

We have demonstrated that ATLAS can be used to model both simple and complex annotation tasks. Since its early incarnation, the ATLAS framework evolved to incorporate numerous enhancements allowing improved modeling of annotations with respect to hierarchical relationships. The Java instantiation of ATLAS (jATLAS, 2000) has been re-implemented to take into account these changes and is now currently in Beta version, available for download on the ATLAS web site (ATLAS, 1999).

The introduction of the MAIA concept now also provides a way to model the semantic dimension that was originally left (purposely) out of ATLAS, allowing ATLAS users to define rich associations and constraints between the different aspects of their annotation task. Specific annotation tasks can be modeled with MAIA in an unambiguous way thus allowing ATLAS applications to automatically interpret type information and perform the tedious task of integrity and consistency checking for the application. MAIA is still very much a work in progress and it will be detailed in a forthcoming paper.

At this time, we would like to invite people interested in using ATLAS for their annotation needs to send us descriptions of their annotation tasks to help them get started with MAIA and ATLAS annotation modeling and helps us improve our work by examining how well these different requirements are comprehended by the current ATLAS framework.

6. References

AG, 1999. Annotation Graphs. [http://www.ldc.upenn.edu/AG/]

AIF, 1999. ATLAS Interchange Format. [http://www.nist.gov/speech/atlas/develop/aif.html]

ATLAS, 1999. Architecture and Tools for Linguistic Analysis Systems. [http://www.nist.gov/speech/atlas/]

Bird, S. and Liberman, M., 1999. A formal framework for linguistic annotation. Technical report MS-CIS-99-01, Department of Computer and Information Science, University of Pennsylvania. Revised version appeared in Speech Communications 33 (1,2), pp 23-60.

Bird, S., Day, D., Garofolo, J., Henderson, J., Laprun C. and Liberman, M., 2000. ATLAS: A flexible and extensible architecture for linguistic annotation in Proceedings of LREC 2000 (Athens, Greece, May 2000), pp 1699-1706.

Garofolo, J., Lamel, L., Fisher, W., Fiscus, J., Pallett, D., Dahlgren, N., NIST, 1986. The DARPA TIMIT acoustic-phonetic continuous speech corpus CDROM. [http://www.ldc.upenn.edu/lol/docs/TIMIT.html]

Garofolo, J., Fiscus, J., Martin, A., Pallett, D., Przybocki, M., 2002, NIST Rich Transcription 2002 Evaluation: A Preview, Proceedins of LREC 2002.

jATLAS, 2000. jATLAS, a Java implementation of the ATLAS framework. [http://www.nist.gov/speech/atlas/jatlas/]

MAIA, 2002. Meta-Annotation Infrastructure for ATLAS. [http://www.nist.gov/speech/atlas/develop/maia.html]

NIST, 1998. A Universal Transcription Format (UTF) annotation specification for evaluation of spoken language technology corpora. [http://www.nist.gov/speech/tests/bnr/hub4_98/utf-1.0-v2.ps]

RT-02, 2002. Rich Transcription evaluation for 2002 (RT-02). [http://nist.gov/speech/tests/rt/rt2002/]