A RESTful interface to Annotations on the Web

Steve Cassidy

Centre for Language Technology
Department of Computing
Macquarie University
Sydney
steve.cassidy@mq.edu.au

Abstract

Annotation data is stored and manipulated in various formats and there have been a number of efforts to build generalised models of annotation to support sharing of data between tools. This work has shown that it is possible to store annotations from many different tools in a single canonical format and allow transformation into other formats as needed. However, moving data between formats is often a matter of importing or exporting from one tool to another. This paper describes a web-based interface to annotation data that makes use of an abstract model of annotation in its internal store but is able to deliver a variety of annotation formats to clients over the web.

1. Introduction

There has been considerable work in recent years on building generalised models of annotation and defining interchange file formats such that data can be moved between tools. This work offers the hope that annotation data can be released from the project or discipline specific dungeons it is often locked in due only to the difficulty in understanding data from foreign tools. However, while data sits in files on a researcher's disk it remains hard to discover it and get access, let alone collaborate on the development of a corpus.

A second problem is that annotations, even in well known and widely distributed corpora, can't be cited in the same way that we might cite a result in a research paper. Exceptions to this are cases where the authors of a corpus have taken care to define reference codes for segments of the corpus (e.g. the line numbers of the Brown corpus).

We propose that both of these problems can be addressed by defining a well structured interface to corpora and annotations over the web. Such an interface would have the advantage of defining a public URI for every corpus and annotation within the corpus that could be cited in a research paper. It could also allow widespread access to data from remote locations to facilitate collaboration and sharing of annotations. Using the infrastructure of the web allows technologies such as caching and access control to be layered on top of the basic interface.

This paper describes the core of a web based interface to corpora. At present this interface only supports reading of annotations from a central annotation store. However, the design has been built with a view to enabling read/write access to data over the web.

2. Background

A number of proposals have been made in recent years for generalised data models for Linguistic annotation. These models provide an abstract representation of annotation data that subsumes practices in the majority of research areas where language data is annotated or marked up in some way. While there are some differences in the proposals they are largely compatible with each other; this is perhaps not surprising since they are designed to support transformation to and from a similar set of end-user formats.

Two examples whose design is particularly focussed on interchange of annotations between formats are Annotation Graphs (Bird and Liberman, 2001) and the Linguistic Annotation Framework (Ide and Romary, 2007). Both are structured as directed graph structures with annotations as nodes in the graph; annotations are distinct objects carrying arbitrary feature structures (attribute-value pairs) and may be related to each other by many kinds of relations. Both formats make use of so called *stand-off* markup where the annotations are stored separately to the primary data itself. Locations in the primary data are indicated by pointers; for audio and video data these are time values, for textual data they can be character offsets or XPointer references.

The use of annotations that point into primary data instead of being embedded in it was motivated in part by the need to be able to represent overlapping hierarchies. Since XML, a common format used for annotation, can only directly represent a single hierarchy, a solution that separated the different hierarchies into different XML files was used. A side effect of this change is that annotations can be managed separately to the primary data, paving the way for an annotation architecture that uses an abstract interface rather than an application specific file format.

The work described here develops on this idea of an abstract interface to an annotation store as an alternative to reading and writing annotation files. Instead of thinking of annotations as elements in files and corpora being collections of these files we abstract these ideas to make all of these things *resources* within an annotation store. Internally in our system, we store annotations as assertions in an RDF triple store and provide an abstract interface for creation, deletion and query of annotation data. The proposal in this paper though, does not make any assumptions about the kind of store that is used; only that it supports the idea of annotations as separate entities. This is true of the Annotation Graph system for example and will generally be true of any tool that displays and manipulates annotation data.

This work has been implemented in a development system that is being used as part of a larger project to support collaborative annotation on language resources. A demonstration of the service may be available at the URI `http://dada1.ics.mq.edu.au/` depending on the

current status of the software.

This paper first highlights the capabilities of the HTTP transport layer, then develops the design of an interface to annotation data over HTTP and finally describes some extensions to this interface that we are currently exploring.

2.1. HTTP and the Web

The Hypertext Transfer Protocol (HTTP) is the base protocol of the World Wide Web and defines the conversation that takes place between a web server and a client such as a web browser. The original web was conceived as a read/write medium and the design of HTTP reflects this in the provision of actions for creating, updating and deleting resources as well as retrieving them. Until recently, the two-way nature of HTTP was not widely exploited but the development of web services following the REST (Representational State Transfer) architecture (Fielding, 2000) has highlighted the power of the original design.

The REST view of the web is as a means to provide access to *resources* that are identified by unique addresses (the Uniform Resource Identifier or *URI*). Resources are accessed through a constrained set of operations for transferring state information between client and server; be it a GET request to retrieve the current state of a resource or a POST request to update it. State information can range from the content of an HTML web page to the contents of a shopping cart or a value in a data store. It is also common to differentiate the internal form of the resource from the surface form that is transferred over the network. Hence, the current temperature on a web accessible device could be transferred as a simple text file, an XML document or an HTML web page. The form of the response is determined by the request that is sent from client to server.

The most common request in HTTP is *GET* which retrieves the current state of a resource. A *POST* request is often used to submit form data to a web service but in general is intended to submit data to a resource and can be interpreted as creating a subsidiary resource (e.g.. a file within a folder) or updating an existing resource. Less commonly used are *PUT* and *DELETE* which create new resources and delete them; since these generally imply creating and deleting files on a server they are not generally implemented for security reasons. HTTP supports a few other kinds of request and there are a number of extensions to the protocol to support additional applications (for example WebDAV to support remote file stores).

While HTTP is an inherently open protocol, it is able to support secure and authenticated access to resources. Encrypted connections using the Secure Sockets Layer (SSL) mean that traffic over the network cannot be intercepted. Authentication can be layered on top of the basic HTTP protocol using cookies - additional headers exchanged with every transaction. In combination, these can provide secure access to resources mediated via appropriate authentication and authorisation controls. This is an important feature for working with language resources which often need to be protected from general access; some work relating to this will be outlined later in the paper.

3. Annotations on the Web

3.1. What gets a URI?

The first question in designing an interface to annotations over the web is that of designing the *URI space* – the logical structure of URIs used to retrieve and modify annotations. Closely tied to this is the question of what should have a URI of its own. Our proposal is for a three-level abstraction of resources from the annotation store: *corpora*, *annotation sets* and *annotations*. We also include an explicit representation of an annotation end point (start or end time or pointer to a document location) called an *anchor*.

Each of these kind of resource is identified by a unique URI. This is both a canonical name for the resource and a means of accessing a description of it over the HTTP interface.

Corpora represent collections of documents whose annotations are stored on the server. A corpus might be a traditional curated collection such as the TIMIT or BNC corpora, or an ad-hoc collection by a single researcher. A corpus has a URI of the form `http://example.org/corpora/NAME` where *NAME* is a symbolic name for the corpus[1] A collection of corpora housed on a given server will also have a URI (`http://example.org/corpora/` here) that could be used to discover what data is available on this server.

Annotation Sets are containers for the annotations on a single document or media file. It is common to have this level of abstraction when using a tool such as ELAN (Wittenburg et al., 2006) or Transcriber (Barras et al., 1998) that stores all annotations on a media file in a single XML file. Annotation sets might correspond to more than one of these XML files in the case when multiple kinds of annotation are stored in different files. An annotation set is always part of a corpus and has the corpus URI as a prefix of its URI which is of the form `http://example.org/corpora/NAME/ASID`; ASID here is a unique identifier for the annotation set.

Annotations are the individual annotations that make up an annotation set. A single annotation might store the part of speech of a word or a phonetic label for a segment of a speech signal. The URI of an annotation has an annotation set URI as a prefix: `http://example.org/corpora/NAME/ASID/ANNID` where ANNID is an annotation identifier.

Anchors are the endpoints of annotations and are represented as explicit resources to allow them to be shared between annotations. For example, one anchor may be the end point of one annotation and the start point of a second. Anchors appear in some form in many annotation formats including Annotation Graphs (Bird and Liberman, 2001) and ELAN (Wittenburg et al., 2006) which calls them *time slots*. Since anchors are also contained within annotation sets, they also have a URI that has an annotation set URI as a prefix: `http://example.org/corpora/NAME/ASID/ANCHID` where ANCHID is an anchor identifier.

[1] In these examples we use a common prefix of `http://example.org/corpora/` in all URIs, this is arbitrary and will depend on the server used to store the corpora.

Each of these kind of resources can be described by a feature structure (in the TEI or ISO 24610-1 sense (M. Laurent Romary and TC 37/SC 4/WG 2, 2006)) containing information about the resource. This structure supports attaching feature sets to any level of detail from the corpus to the annotation itself. Feature values can include relations between resources; these are easily expressed since each resource has a unique URI that can appear as the value of a feature. The vocabulary used in defining features is of course important; we note that the Linguistic Annotation Framework (Ide and Romary, 2007) is directly addressing this need in setting up standards for a Data Category Registry that would allow mapping of feature names between resources.

3.2. Responses to URIs

Having said that these resources have unique URIs that can be published and accessed to allow sharing of annotations, we still need to define what exactly will be returned if someone enters one of these URIs into a web browser.

By default, the response to a request for a URI from the annotation server will be an HTML representation of the resource being referenced. This means that someone can access one of these published URIs in a web browser and see a human readable representation of the corpus, annotation set or annotation. The actual representation that is returned is the concern of the implementer of the server and need not be uniquely defined; for example, a server that holds annotations of video data might be able to serve a representation of an annotation set as a page with the video embedded alongside a browseable version of the annotations similar to that developed by the EOPAS project (Thieberger and Schroeter, 2006) for ethnographic data.

Our current implementation includes links to all of the subordinate resources in the HTML representation. So, the page generated for a corpus links to all of the annotation sets in the corpus while the annotation set links to all of the annotations. The page for an annotation includes all of the properties associated with the annotation and links to any other associated annotations (e.g.. parents, dependancies, etc.).

3.2.1. Content Negotiation

A little used option in HTTP is the ability to have the web browser request certain types of content when requesting a resource. For example, I can ask for `http://example.org/data` while saying that I will accept plain text or PDF. The server can then respond with whichever of these it is able to produce. This process is called *content negotiation* and is not widespread partially because of the lack of support for it in all browsers.

The web service described here makes use of content negotiation to serve different kinds of content to different clients. If the client is a conventional web browser, the server will generate HTML descriptions of resources; on the other hand if the client is an annotation tool, it can request data, for example, in ELAN eaf format.

Content negotiation will allow us to serve different representations of each of the resources to different kinds of client. We can, for example, return a version of an annotation set in the format required by an annotation tool such as ELAN or Transcriber. In this way, the interface can realise the format conversion functionality that is at the core of standards such as LAF (Ide and Romary, 2007) or AG (Bird and Liberman, 2001) transparently. The same annotation could then be accessed by an ELAN user and a Transcriber user without having to distribute two distinct versions of the annotation or go through any explicit conversion process.

In some situations, content negotiation is not possible - for example when including links in a web page or when dealing with older HTTP client software. In these cases it is possible to achieve the same end by augmenting the URI of a resource with a query string indicating the type of representation required. So, to retrieve an ELAN format representation of an annotation set one could retrieve `http://example.org/corpora/andosl/foobar?format=application/xml+eaf` (the exact keyword and format indicator needs standardisation, this example is included to illustrate this capability).

3.2.2. Low Level Access

There is a third possibility though that offers to realise the full potential of the web based annotation store. That is to return a form of each resource that can form the basis of a read/write interface to the store. The idea here is that instead of reading and writing annotation files in an XML format, a tool could query the server directly for information about the annotations on a document or media file. To support this, the response to a request for an annotation set could be a simple XML list of the URIs of the annotations, perhaps with a small amount of data from each such as a label or start and end times. Using this, an annotation tool could determine which annotations are of interest and query the server for more information about each. The response to a request for an annotation could be a simple XML representation of the annotation as a feature structure.

This kind of server would allow updates to be made to annotations using the same kind of messages sent from the client to the server. To add a new annotation to an annotation set the client would make a POST request to the annotation set URI `http://example.org/corpora/as123/` with a request body containing the feature structure for the new annotation. In response to the POST request, the server creates a new annotation and returns a HTTP response confirming that it was created with the URI of the new annotation. Similarly, a POST request to an existing annotation URI has the effect of updating the annotation. Finally, the DELETE request to an annotation or annotation set URI can be used to remove the corresponding resource. These requests can be used by an annotation tool to directly manipulate the annotations stored on the server rather than working through any kind of file format.

4. Building Upon the Interface

One of the primary advantages of defining a web based interface based on HTTP access to resources is that the existing infrastructure of the web can be leveraged to add new functionality with little extra effort. The web is a very mature family of technologies and many issues around effi-

cient, secure distribution of data have been addressed in general purpose technologies layered on top of HTTP. A few of the possibilities are outlined here.

4.1. Caching and Proxies

A significant problem with providing remote access to resources such as annotations or primary linguistic data is the time lag between a request and the response being delivered over the network. This would be an immediate barrier to adoption of this kind of technology in some applications which require very fast access to data. This is not a problem unique to annotation and since we have layered our interface on top of HTTP we can take advantage of HTTP caches to speed access to frequently accessed data.

An HTTP cache acts as a proxy between the client and server such that most transactions occur just as they would if no proxy were in place. The cache will however, remember the responses to some requests and, if configured appropriately, will return a local copy of the response if it is requested again. A cache can be run on an individual machine or within an organisation where the requests from all users within a research group would be cached together, speeding access to the resources being used by the group.

While a generic HTTP proxy cache such as Squid (`http://www.squid-cache.org/`) can be used in this way there is scope for writing a special purpose proxy cache that knows about usage patterns of annotation data. Such a proxy could pre-fetch annotations that might be used in the near future.

While caching files can be one important function of a proxy server, it can also fulfil another role in this context. A proxy acts as a mediator between the client and one or more servers and as such can federate access to multiple annotation servers. One could imagine a departmental or institutional proxy supporting access to many servers via a common cache while also serving local resources transparently to users. A network of such proxies could effectively provide distributed, redundant, storage of annotation data.

4.2. Authentication and Authorisation

As described so far, all resources are available to anyone on the internet to read and possibly update; this is clearly not what would be required by most researchers and for many language resources which must be restricted in some way. Again, we can make use of existing technology on the web to layer authentication and authorisation on top of the HTTP interface described above.

HTTP provides a simple authorisation scheme as part of the protocol which would allow resources to be password protected. Web servers such as the Apache server allow configuration settings that protect different URIs with different user names and passwords and this could be used to restrict access to distinct groups of users. Similarly, the operations that update an annotation (PUT, POST, DELETE) can be given different levels of password protection using standard server settings.

A more sophisticated solution has been developed for applications that require more complex authorisation rules to be enforced. The XACML (XML Access Control Markup Language, `http://www.oasis-open.org/committees/xacml/`) standard allows complex access control rules to be written which take into account external factors such as the date or file properties such as size or source of data. We are currently investigating the use of XACML in conjunction with our annotation server to provide fine grained access control to both annotations and primary data. For example, one might want to restrict access to part of a recording based on the identity of a speaker in that recording. XACML allows the rules to be written to express this restriction; we are now looking at how the server infrastructure needs to be configured to put this into practice.

Rather than require every server to maintain passwords and user credentials for authorised users, the Shibboleth system `http://shibboleth.internet2.edu/` implements a federation of identity providers such that a user can be authenticated against their home institution. An identity federation such as this would allow groups of researchers to be granted access to resources based on, for example, their host institution or membership of some project. We are currently working with the RAMS project at Macquarie `http://www.melcoe.mq.edu.au/projects/RAMP/` on integrating our server with the Muradora data repository `http://www.muradora.org/`, a version of the popular Fedora server that integrates Shibboleth and provides a web based interface to building XACML policy documents. Our work here aims to illustrate how access to source data, meta-data and annotations can be mediated by appropriate authentication and authorisation.

4.3. Version Management

Annotations are not often static; errors are found and corrected and new versions of corpora are published. Especially in the context of a collaborative annotation tool it must be possible to manage different versions of annotations and integrate version control operations such as roll-back of changes or generating patch sets to send to other users.

As part of our work on the back-end RDF annotation store we have developed a version control system for RDF triple stores that is designed to support these operations on annotation data (Cassidy and Ballantine, 2007).

If the URIs published for annotation sets and annotations are to be useful they must be constant over time. That is, I must be able to publish a reliable URI for the annotation set that I used for a given study, not one which points to the most recent version of that annotation. Hence we must be able to include revision information in the URI.

While we have not yet integrated our version control system with the HTTP interface, there are a number of possible ways in which one could refer to historical versions of data via a URI. One simple option is to prefix the corpus name with a revision identifier: `http://example.org/corpora/101029/andosl/msdjc001/ann0293` - where `101029` uniquely identifies the revision of the annotation that is being referred to. The most recent annotation could still be referenced with out the version identifier but the longer style could be used where longevity of reference is required.

4.4. Mashups of Data and Annotations

One of the defining features of the recent boom of applications on the web has been the growth of *mashups* built from data provided by different sources. A common component of these is Google Maps `http://maps.google.com/` which can be used to visualise geographic data available on the web. The open nature of the web and the fact that data is available in well defined formats using well defined interfaces means that data can be re-purposed into applications that might not have been conceived by the original authors. In the annotation domain there are many possibilities for mashups that might combine annotation data with other widely available data sources such as WordNet, Wikipedia etc. Annotations might also be combined with each other; for example, merging different styles of annotation or augmenting annotations with data from lexical resources. The important point here is that this capability comes for free once we adopt an open, well defined interface using well understood technology.

5. Conclusion

This paper has given a brief overview of the design of a web based interface to an annotation store. The design uses the REST approach to make corpora, annotation sets and annotations available as first class resources on the web.

This approach changes the way that annotation tools work with annotation data. Instead of relying on local storage of data in files, tools can work with an annotation store through an abstract interface. The fact that this interface uses the HTTP protocol of the web means that the store can be remote and shared between users. By layering authentication, authorisation, caching and other standard HTTP technologies on top of the interface we can add additional functionality to the interface.

6. References

Claude Barras, Edouard Geoffrois, Zhibiao Wu, and Mark Liberman. 1998. Transcriber: a Free Tool for Segmenting, Labeling and Transcribing Speech. In *Proceedings of the First International Conference on Language Resources and Evaluation (LREC),*, pages 1373–1376, Granada, Spain, May.

S. Bird and M. Liberman. 2001. A Formal Framework for Linguistics Annotation. *Speech Communication*.

Steve Cassidy and James Ballantine. 2007. Version control for rdf triple stores. In *ICSOFT 2007*, Barcelona, Spain, July.

Roy Thomas Fielding. 2000. *Architectural Styles and the Design of Network-based Software Architectures*. University of California, Irvine,.

N. Ide and L. Romary. 2007. Towards International Standards for Language Resources. In L. Dybkjaer, H. Hemsen, and W. Minker, editors, *Evaluation of Text and Speech Systems*, pages 263–84. Springer.

M. Laurent Romary and TC 37/SC 4/WG 2. 2006. Language resource management - Feature structures - Part 1: Feature structure representation. In ISO 24610–1.

Nicholas Thieberger and Ronald Schroeter. 2006. EOPAS, the EthnoER online representation of interlinear text. In Linda Barwick and Nicholas Thieberger, editors, *Sustainable Data from Digital Fieldwork*, pages 99–124, University of Sydney, December.

Peter Wittenburg, Hennie Brugman, Albert Russel, Alex Klassmann, and Han Sloetjes. 2006. ELAN : a professional framework for multimodality research. In *Proceedings of the 5th International Conference on Language Resources and Evaluation (LREC 2006)*.