The FAST Annotation Service

Nicola Ferro

University of Padua, Italy
ferro@dei.unipd.it

Abstract. This paper presents the FAST annotation service, which is able to add annotation functionalities to both digital libraries and the Web by supporting annotations that range from content to metadata. It discusses the annotation model adopted by FAST and its XML representation; the architecture of the annotation service; and how annotations can be exploited to develop search and retrieval algorithms.

1 Introduction

Almost everybody is familiar with annotations and has his own intuitive idea about what they are, drawn from personal experience and the habit of dealing with some kind of annotation in every day life, which ranges from jottings for the shopping to taking notes during a lecture or even adding a commentary to a text. Therefore, annotations have been adopted in a variety of different contexts, such as content enrichment, data curation, collaborative and learning applications, and social networks, as well as in various information management systems, such as the Web (semantic and not), digital libraries, and databases.

The Flexible Annotation Service Tool (FAST) covers many of the uses and applications of annotations discussed above, since it is able to represent and manage annotations which range from metadata to full content; its flexible and modular architecture makes it suitable for annotating general Web resources as well as digital objects managed by different digital library systems; the annotation themselves can be complex multimedia compound objects, with varying degree of visibility which ranges from private to shared and public annotations and different access rights. The FAST annotation service has proven its flexibility and adaptability to different applicative contexts in many different ways. It has been integrated into the DelosDLMS [3], the prototype of next generation digital library system developed by DELOS, the European network of excellence on digital libraries. It has been used as architectural framework for the DiLAS project [1]. Finally, a completely new and re-engineerd version of it has been recently integrated into The European Library (TEL)[1] development portal; The European Library is the portal which offers access to 48 national libraries in more than 20 languages and represents a first step towards the creation of the European Digital Library.

[1] http://www.theeuropeanlibrary.org/

The paper is organised as follows: Section 2 presents the annotation model supported by the FAST annotation service; Section 3 introduces the architecture of FAST; Section 4 discusses our search and retrieval framework; finally, Section 5 draws some conclusions.

2 Annotation Model

FAST adopts and implements the formal model for annotations proposed in [2] which has been also utilised in the reference model for digital libraries[2] developed by DELOS [4] and now carried on by DL.org[3], the European coordination action on digital library interoperability, best practices and modelling foundations. The formal model provided us with a sound basis for developing an eXtensible Markup Language (XML) Schema[4] for FAST, which is available at `http://ims.dei.unipd.it/xml/fast-schema-instance` and is shown in Figure 1.

According to this model, annotations are compound multimedia objects constituted by different *signs of annotation* which materialize the annotation itself. For example, we can have *textual signs*, which contain the textual content of the annotation, *image signs*, if the annotation is made up of images, and so on. In turn, each sign is characterized by one or more *meanings of annotation* which specify the semantics of the sign. Moreover, an annotation is uniquely identified by an handle, has a scope which defines its visibility, and can be shared with different groups of users.

The `annotation` element provides the basis for modelling annotations and has the following attributes: `identifier` is a unique identifier for the annotation, e.g. an Uniform Resource Identifier (URI); `namespace` is the namespace to which the annotation belongs; `mime-type` is the Multipurpose Internet Mail Extensions (MIME) media type of the annotation and specifies the kind of content of the annotation; `scope` specifies wheter the annotation is private, shared, or public; `created` and `modified` represent, respectively, the creation timestamp and the last modification timestamp of the annotation.

The `user` element represents a user, who is the author of the annotation, and it is characterized by the following attributes: `identifier` is the username of the user; `namespace` is the namespace to which the user belongs; `password` is the password of the user; `full-name` is the complete name of the user; `e-mail` is the e-mail address of the user; and `language` and `country` are, respectively, the language and country of the user; the `groups` elements specifies to which groups the user belongs.

The `group` element represents a users group and it is characterized by the following attributes: the `identifier` of the group, the `namespace` to which the group belongs, and a `description` of the group; the `users` elements specifies which users belong to the group.

[2] `http://www.delos.info/ReferenceModel/`

[3] `http://www.dlorg.eu/`

[4] `http://www.w3.org/XML/Schema`

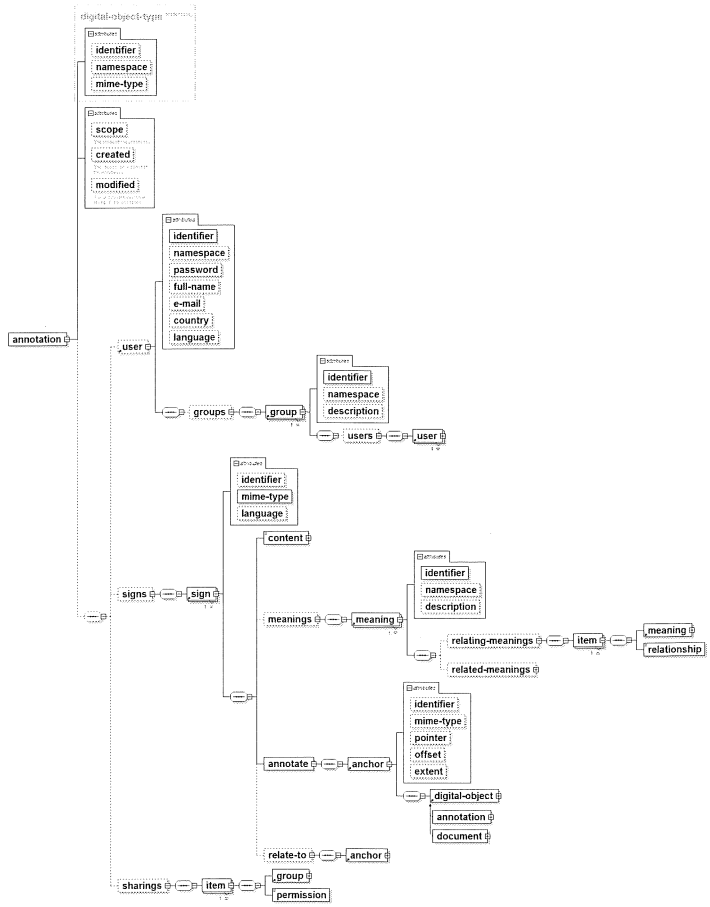

Fig. 1. XML Schema of the FAST annotation model.

The `sign` element has an unique `identifier`, a `mime-type`, and a `language`, if its content is textual. The `content` element represents the actual content of the sign of annotation, e.g. a piece of text or an image, according to its MIME type. If the MIME type of the sign is textual, then `content` contains the textual representation of the sign; otherwise, for binary MIME types, it contains their Base64 encoding. The `meanings` elements contains the meanings of annotations associated with the given sign.

The `meaning` element is characterised by a unique `identifier`, a `namespace` and a `description`. A meaning can be in relationship with other meanings, via the `relating-meanings` and `related-meanings` elements, that allow us to define a graph of meanings where the `relationship` element describes the kind of relationship between two meanings of annotation and organize them into some kind of taxonomy.

The `annotate` element links the sign to the digital object that it annotates. Note that, once we have annotated a digital object, the annotation itself can be considered as a digital object eligible to be annotated to. Users can therefore create not only sets of annotations concerning a digital object, but also threads of annotations, i.e. annotations which reply to one another. The `relate-to` element optionally associates a sign of annotation with the digital object it refers to.

In order to locate a specific part of the annotated or related digital object, both the `annotate` and the `relate-to` elements make use of an `anchor` which allows us to identify the part of the digital object which has to be annotated. It has the following attributes: an unique `identifier` of the anchor; the `pointer` attribute identifies a portion of the digital object, e.g. it could be an XPath[5] expression when an XML document is annotated; `offset` selects a starting offset with respect to the portion identified by `pointer`, e.g. the initial character within an XML element; `extent` specifies the spread of the anchor, e.g. the number of characters that are annotated within the portion identified by `pointer` starting from `offset`; finally, `mime-type` indicates the specific MIME type of the anchored part, since this could be different from the MIME type of the whole digital object, and helps in the interpretation of the `pointer`, `offset`, and `extent` attributes which depends on the actual MIME type of the anchored part.

Finally, the `sharings` element allows an annotation to be shared by one or more groups of users. It contains the `permission` element, which specifies the privileges, e.g. read or modify, granted to a group sharing the annotation.

3 Architecture

Figure 2 shows the architecture of the FAST annotation service. It consists of three layers – data, application and interface logic layers – in order to achieve a better modularity and to properly describe the behaviour of the service by isolating specific functionalities at the proper layer. A set of interfaces defines the behaviour of each component of FAST in abstract terms. Then, a set of abstract classes partially implement the interfaces in order to define the actual behaviour common to all of the implementations of each component. Finally, the actual implementation is left to the concrete classes, inherited from the abstract ones, thus giving FAST the additional possibility of fitting into different architectures.

The FAST annotation service is accessible to client applications by means of a RESTful Web application [5] which offers several APIs build around the following main resources:

- *namespace*: manages all the operations related to namespaces;
- *annotation*: manages all the operations related to annotations;
- *meaning*: manages all the operations related to meanings of annotation;
- *group*: manages all the operations related to groups of users;
- *user*: manages all the operations related to users;

[5] `http://www.w3.org/TR/xpath20/`

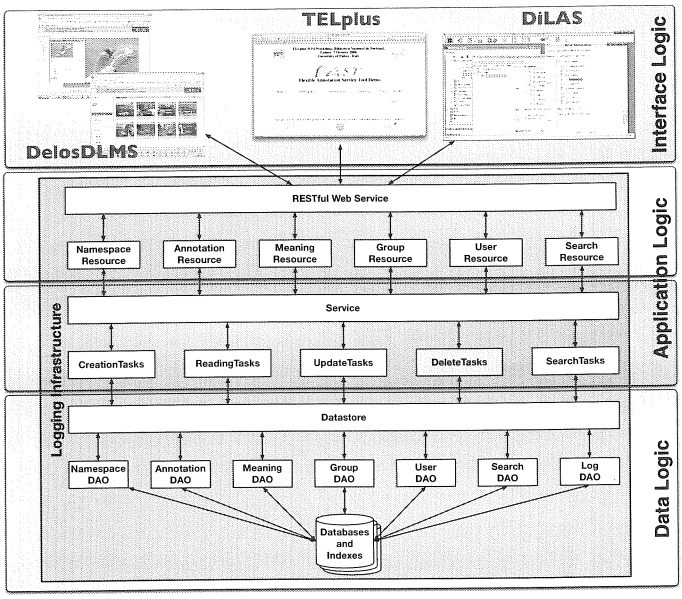

Fig. 2. Architecture of the FAST annotation service.

– *search*: manages the search and retrieval of documents and annotations according search framework described in the following section.

Some resources are publicly available, some others require authentication before being accessed, according to the basic HyperText Transfer Protocol (HTTP) authentication scheme.

The logging infrastructure, which lays behind all the components of the FAST annotation service, captures information such as the user name, the Internet Protocol (IP) address of the connecting host, the action that has been invoked by the user, the messages exchanged among the components of the system in order to carry out the requested action, any error condition, and so on. Thus, besides offering us a log of the system and user activities, it allows us to fine trace the provenance of each piece of data from its entrance in the system to every further processing on it. Moreover, as far as the FAST RESTful Web Application is concerned, it captures also the HTTP logs and represents them according to the W3C Extended Log File Format[6] .

The FAST annotation service has been developed by using the Java[7] programming language, which ensures good portability of the system across different platforms. We used the PostgreSQL[8] database management system for the actual storage of the annotations and its full text extension for indexing and

[6] `http://www.w3.org/TR/WD-logfile.html`
[7] `http://java.sun.com/`
[8] `http://www.postgresql.org/`

searching the full text components of an annotation. The Apache Tomcat[9] Web container and the Restlet[10] framework have been used for developing the FAST RESTful Web Application.

4 The FAST Search Framework

The problem of information access and retrieval by exploiting annotations comprises two different issues: the first concerns the search and retrieval of the annotations themselves; the second regards the search and retrieval of annotated documents. The first case requires the design and development of algorithms able to express complex queries which take into account both the different features of the annotations and the context to which annotations belong. The second case calls for algorithms able to estimate the relevance of annotated documents with respect to a user information need on the basis of the annotations on them.

The presence of both structured and unstructured content within an annotation calls for different types of search functionalities, since structured content can be dealt with exact match searches while unstructured content can be dealt with best match searches and they may need to be merged together in a query by using boolean clauses. Nevertheless, boolean clauses are best suited for dealing with exact match searches and they need to be somewhat extended to also deal with best match searches. This is discussed in section 4.1.

The hypertext that connects documents to annotations calls for a search strategy that takes it into consideration and allows us to modify the score of annotations and/or documents according to the paths in the hypertext. For example, we could consider that an annotation, retrieved in response to a user query, is more relevant if it is part of a thread where other annotations have also been retrieved in response to the same query rather than if it is part of a thread where it is the only annotation that matches the query. This is discussed in section 4.2.

4.1 Annotation Extended Boolean Retrieval

In order to be able to express queries that range from pure boolean queries to pure vector-space queries, we make use of the the P-norm extended boolean model proposed by [6], which is capable of dealing with and mixing both exact and best match queries.

Consider a set of terms $t_1, t_2, \ldots, t_n$ and let $\mathrm{sim}(a, t_i) \in [0, 1]$ be the similarity score of term t_i with respect to annotation a; $\mathrm{sim}(a, t_i) = 0$ if the term t_i is not present in the annotation a.

Let $p \geq 1$ be a real number indicating the degree of strictness of the boolean operator. A generalized **or**-query is expressed as $q_{\mathbf{or}(p)} = [t_1 \, \mathbf{or}^p \, t_2 \, \mathbf{or}^p \cdots \mathbf{or}^p \, t_n]$; a generalized **and**-query is expressed as $q_{\mathbf{and}(p)} = [t_1 \, \mathbf{and}^p \, t_2 \, \mathbf{and}^p \cdots \mathbf{and}^p \, t_n]$.

[9] `http://tomcat.apache.org/`

[10] `http://www.restlet.org/`

The **extended boolean similarity scores** between an annotation and a query are defined as:

$$\mathrm{sim}_p^{\mathbf{or}}(a,q) = \left[\frac{\mathrm{sim}(a,t_1)^p + \mathrm{sim}(a,t_2)^p + \cdots + \mathrm{sim}(a,t_n)^p}{n}\right]^{\frac{1}{p}}$$

$$\mathrm{sim}_p^{\mathbf{and}}(a,q) = 1 - \left[\frac{(1-\mathrm{sim}(a,t_1))^p + (1-\mathrm{sim}(a,t_2))^p + \cdots + (1-\mathrm{sim}(a,t_n))^p}{n}\right]^{\frac{1}{p}}$$

where t_i indicates a generic term of the query q. Note that for **not**-queries you have to substitute $1 - \mathrm{sim}(a,t_i)$ to $\mathrm{sim}(a,t_i)$ as term weight.

By varying the value of p between 1 and ∞, it is possible to obtain a query processing intermediate between a pure vector-processing model ($p = 1$) and a traditional boolean processing ($p = \infty$), as discussed in [6] to which the reader can refer for further details.

4.2 Annotation Hypertext-driven Retrieval

Consider the document-annotation hypertext $H_{da} = (DO, E)$ where DO is a set of digital objects (either documents or annotations) and E is a set of edges indicating that an annotation is annotating a digital object, as introduced in [2].

The **hypertext similarity score** between an annotation and a query is defined as:

$$\mathrm{sim}_\alpha^{ht}(a,q) = \frac{1}{\alpha}\mathrm{sim}(a,q) + \frac{\alpha-1}{\alpha} \cdot \frac{1}{|\mathrm{succ}(a)|} \sum_{a_k \in \mathrm{succ}(a)} \frac{\mathrm{sim}(a_k,q) + \mathrm{sim}_\alpha^{ht}(a_k,q)}{2}$$

where $\mathrm{sim}(a,q) \in [0,1]$ is a generic similarity function between an annotation and a query, $\mathrm{succ}(a)$ is a function that returns the set of successors of an annotation a_j and α is a real number called the *annotation thread damping* factor. We consider that $\mathrm{sim}(a_j,q) = 0$ for those annotations that do not match the query.

$\mathrm{sim}_\alpha^{ht}(a,q)$ computes the weighted average between $\mathrm{sim}(a,q)$, the similarity score of an annotation with respect to a query, and the similarity scores which come from the thread to which the annotation belongs. In particular, the thread similarity scores are given by the average between the similarity scores of the successors of a and the hypertext similarity scores of the successors of a; in other words, the hypertext similarity score recursively averages the similarity scores of the annotations that belong to the same thread of the given annotation a. Furthermore, $\mathrm{sim}_\alpha^{ht}(a,q)$ penalizes similarity scores which come from lengthy paths, because for a path $P = a_0 \ldots a_k$ the similarity score $\mathrm{sim}(a_k,q)$ of a_k is weighted $\frac{1}{2^k}$.

By varying the value of α between 0 and ∞, it is possible to obtain a query processing intermediate between a traditional information retrieval model ($\alpha = 1$), when $\mathrm{sim}_1^{ht}(a,q) = \mathrm{sim}(a,q)$ and only the similarity between the annotation and the query is taken into account, and a pure hypertext driven retrieval model ($\alpha = \infty$), when $\mathrm{sim}_\infty^{ht}(a,q) = \frac{1}{|\mathrm{succ}(a)|} \sum_{a_k \in \mathrm{succ}(a)} \frac{\mathrm{sim}(a_k,q) + \mathrm{sim}_\infty^{ht}(a_k,q)}{2}$ and only the thread to which the annotation belongs is taken into account.

Finally, the hypertext-driven retrieval model allows us to compute a similarity score also for the documents that have been annotated, so that it is possible to search and retrieve documents in response to a user query by means of their annotations. The **similarity score by annotation** between the document and a query is defined as:

$$\mathrm{sim}^{a}_{\alpha}(d,q) = \frac{1}{|\mathrm{succ}(d)|} \sum_{a \in \mathrm{succ}(d)} \mathrm{sim}^{ht}_{\alpha}(a,q)$$

Basically, the similarity score by annotation of a document averages the hypertext similarity scores of the annotations that are annotating the document.

5 Conclusions

We have discussed the design and development of the FAST annotation service, by describing its annotation model and architecture. In addition, we have introduced a general framework which allows us to develop annotation retrieval algorithms based on a combination of extended boolean retrieval operators and hypertext driven information retrieval.

Acknowledgements

The work reported has been partially supported by the TELplus Targeted Project for digital libraries, as part of the *eContentplus* Program of the European Commission (Contract ECP-2006-DILI-510003).

References

1. M. Agosti, H. Albrechtsen, N. Ferro, I. Frommholz, P. Hansen, N. Orio, E. Panizzi, A. M. Pejtersen, and U. Thiel. DiLAS: a Digital Library Annotation Service. In J.-F. Boujut, editor, *Proc. International Workshop on Annotation for Collaboration – Methods, Tools, and Practices (IWAC 2005)*, pages 91–101. CNRS - Programme société de l'information, 2005.
2. M. Agosti and N. Ferro. A Formal Model of Annotations of Digital Content. *ACM Transactions on Information Systems (TOIS)*, 26(1):3:1–3:57, 2008.
3. M. Agosti and N. Ferro. Adding Advanced Annotation Functionalities to an Existing Digital Library. In A. D'Atri, M. De Marco, and N. Casalino, editors, *Interdisciplinary Aspects of Information Systems Studies*, pages 279–286. Physica-Verlag, Heidelberg, Germany, 2008.
4. L. Candela, D. Castelli, N. Ferro, Y. Ioannidis, G. Koutrika, C. Meghini, P. Pagano, S. Ross, D. Soergel, M. Agosti, M. Dobreva, V. Katifori, and H. Schuldt. *The DELOS Digital Library Reference Model. Foundations for Digital Libraries.* ISTI-CNR at Gruppo ALI, Pisa, Italy, `http://www.delos.info/files/pdf/ReferenceModel/DELOS_DLReferenceModel_0.98.pdf`, December 2007.
5. R. T. Fielding and R. N. Taylor. Principled Design of the Modern Web Architecture. *ACM Transactions on Internet Technology (TOIT)*, 2(2):115–150, 2002.
6. G. Salton, E. A. Fox, and H. Wu. Extended Boolean Information Retrieval. *Communications of the ACM (CACM)*, 26(11):1022–1036, November 1983.