

Language engineering tools for collaborative corpus annotation

H. Cunningham*, V. Tablan*, K. Bontcheva*, M. Dimitrov**

* Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP
{H.Cunningham, V.Tablan, K.Bontcheva}@dcs.shef.ac.uk

**Ontotext Lab, Sirma AI Ltd.

38 A Hristo Botev Blvd. Sofia 1000, Bulgaria

marin.dimitrov@sirma.bg

1. Introduction

A vital step towards the creation of distributed corpora is the provision of tools for collaborative corpus annotation, in order to enable researchers to collaborate on annotating corpora regardless of their physical location. This problem can be decomposed in two major tasks: (i) provide users with access to distributed corpora; and (ii) provide visualisation and editing tools that require no installation effort and are easy to use. In this paper we will present the new collaborative corpus annotation facilities, recently developed as part of the GATE language engineering tools and infrastructure. These facilities have been used to build OLLIE – a client-server application that allows users to use the collaborative corpus annotation facilities in their own Web browser.

This paper is structured as follows. An overview of GATE is provided in Section 2. Section 3 describes the support for distributed language resources implemented using relational database servers. The client-server architecture of OLLIE is introduced in Section 4 and details of its collaborative corpus annotation facilities are provided in Section 5. Section 7 concludes the paper and outlines future work.

2. GATE overview

GATE (Cunningham et al 2002) is an open-source language engineering infrastructure¹ which, among other things, is being used for the creation and annotation of a number of corpora in many languages, e.g., the American National Corpus and EMILLE - a 63 million word corpus of Indic languages (Baker et al 02). It is written in Java and exploits component-based software development, object orientation and mobile code. One advantage of GATE from a corpus processing perspective is that it uses Unicode throughout (Tablan et al. 2002), and has been tested on a variety of Slavic, Germanic, Romance, and Indic languages (Gambäck and Olsson 2000), (Pastra et al 2002).

Another advantage is its support for a wide range of input and output document formats – XML, HTML, SGML, email, RTF, and plain text – and GATE can also be extended easily to handle other formats, in a manner transparent to the rest of the system. When a document is loaded in GATE, its format is analysed and converted into a GATE document, which consists of content and one or more layers of annotation. The annotation format, a modified form of the TIPSTER format (Grishman 1997), is largely isomorphic with the Atlas format (Bird and Liberman 1999) and successfully supports I/O to/from XCES and TEI (Ide et al. 2000). An annotation has a type, a pair of nodes pointing to positions inside the document content, and a set of attribute-values, encoding further linguistic information. Attributes are strings; values can be any Java object. An annotation layer is organised as a Directed Acyclic Graph on which the nodes are particular locations in the document content and the arcs are made out of annotations. All mark-up contained in the text used to create the document content is

¹ GATE is available for download from <http://gate.ac.uk> and is distributed freely as an open-source system under the GNU library license.

automatically extracted into a special annotation layer and can be used for processing or for exporting the document back to its original format.

In addition to corpus annotation facilities, GATE provides a set of language processing components, e.g., tokeniser, part-of-speech tagger, named entity recogniser; services for persistent storage of language resources; and extensive visualisation and evaluation tools aimed at facilitating the development and deployment of language processing applications.

The new distributed corpus annotation facilities discussed here were built to resemble GATE's easy-to-use and extendable facilities for text annotation of locally stored documents. These facilities allow both manual and semi-automatic annotation of corpora, based on GATE's language processing components. The text is annotated in a point-and-click fashion, where the user first highlights the text to be annotated and then clicks on the desired annotation type (e.g., ENAMEX, POS).

The rest of the paper will discuss how GATE has been extended to support distributed language resources and how these facilities were used to build OLLIE – a client-server application for collaborative corpus annotation.

3. Distributed language resources

Big corpora, such as the British National Corpus (BNC) are best stored on server machines with large amounts of disk space, so they are installed once and shared among many researchers. GATE offers support for such distributed language resources by storing them in relational databases, which run on the servers and accessed by the GATE visual environment that acts as a client. Currently Oracle and PostgreSQL are supported, of which the latter is freely available.

Currently the database servers support storage of corpora and documents and their associated annotations. These GATE objects are stored in about 20 relational tables, grouped in the following categories²:

- 1) Language Resource-related tables: contain information common for all language resources, e.g., their name and type.
- 2) Corpus-related tables: contain information about the stored corpora, such as which documents belong to them.
- 3) Document-related tables: contain information about the stored documents, e.g., their encoding, content, and which annotation sets belong to them.
- 4) Annotation-related tables: contain information about the annotations, the sets they belong to, and the feature-value pairs associated with them.
- 5) Security related tables: contain information about users and groups.

The GATE users are completely isolated from the concrete ways in which LRs are stored in the relational tables. When choosing to store an LR in a distributed database, users just have to choose which of the available servers they want to use and provide their user name and password. Then GATE transparently converts all LRs in the corresponding database objects and stores them via JDBC. In this way, all GATE users and applications are completely isolated from the technicalities of using a database server for distributed LR storage - remote RDBMS datastores are accessed in the same way as local file system based datastores.

An important aspect of supporting distributed LRs is security and access control. In GATE we have implemented role-based access control and every language resource is associated with security properties specifying the actions that certain users and groups may perform with this resource. When users create LRs into the database datastore, they specify access rights for users, the group of the user and other users. For example, LRs created by one user/group can be made read-only to others, so they can use the data, but not modify it. The access modes supported by GATE are detailed in Table 1.

If needed, ownership can be transferred from one user to another. Users, groups and LR permissions are administered in a special administration tool, by a privileged user, i.e., the database administrator.

² For a detailed description of GATE's database storage mechanism and the relational tables used see <http://gate.ac.uk/gate/doc/persistence.pdf> and GATE's User's Guide (<http://gate.ac.uk/sale/tao/>).

As already mentioned, the RDBMS persistence is fully JDBC compliant but in order to improve the performance, specific features of the two databases were used when possible. While this approach induces certain development overhead, since small parts of the implementation are database specific (such as properties of the database objects, stored procedures, SQL optimizer hints, etc.) it ensures that the optimal performance is achieved for each database.

Mode	Owner (Read/Write)	Owner's group (Read/Write)	Other users (Read/Write)
World Read / Group Write	+/+	+/+	+/-
Group Read / Group Write	+/+	+/+	-/-
Group Read / Owner Write	+/+	+/-	-/-
Owner Read / Owner Write	+/+	-/-	-/-

Table 1 Database access rights.

Another performance related feature of the database-based persistence is the load-on-demand implementation of certain GATE resources. Since it is inefficient to load huge amounts of data from a remote datastore, all language resources in GATE employ a load-on-demand behaviour when loaded from database datastore. In such cases the corpus documents, document content and annotation sets will be loaded from the datastore only when first accessed from the user (or the application). In a similar manner a complex tracking system ensures that only the parts of a resource that were changed are propagated back to the database for updates. This approach significantly reduces the amount of data transferred and the round trips to the server.

Read-only resources allow shared use of existing corpora such as the British National Corpus (BNC), which only need to be saved in the database once and then become available through GATE to all users, who do not need to install and store them locally. In the future we will allow users to extend such resources by creating a new 'composite' LR, which references inside the original read-only LR. New data is added to the modifiable part of the composite LR, while users can access data from both parts. In this way, different groups can share a read-only copy of e.g. the BNC or WordNet, which is referenced by their separate customised LRs which store the new information, and made accessible to other groups/users as required.

Key benefits of the approach include:

- 1) The database infrastructure takes care of multi-user access, locking and so on without any intervention by the user.
- 2) LRs can in this way be easily distributed across networks, and need not be installed locally on each machine used by their developers and users.
- 3) GATE's graphical display tools can be used to visualise and edit the data in the resources.
- 4) GATE's framework can be used for embedding the resources in diverse applications.

The database infrastructure offered by GATE is employed in a number of projects and production systems. Some real world statistics from OntoText Lab for a project that uses GATE to annotate news articles look like:

- Database size: 3.5 GB (growing by 20MB per day)
- Number of documents: 170,000 (1000 new per day)
- Number of annotations: 3,500,000
- Number of annotation features: 10,500,000
- Total number of rows in the database: 23,500,00



Figure 1 The OLLIE corpus and document browser

4. The OLLIE architecture for collaborative annotation

OLLIE is a client-server application providing support for collaborative corpus annotation by using any Java-enabled Web browser. The three main functions of the OLLIE client are:

- 1) support user authentication and profiles;
- 2) support collaborative corpus annotation;
- 3) provide language processing tools to (partially) automate this task.

OLLIE also supports training and running different Machine Learning methods but these aspects will not be discussed here due to space limitations.

The OLLIE server on one hand communicates with the OLLIE client and on the other uses GATE and its distributed database facilities (described in Section 3) to store the user data and the language resources being edited (see Figure 3 for an illustration). Every user has a username and a password, used to retrieve their profiles and determine which documents they can access (according to the database access policies discussed above). The profiles contain information provided by the user, specifying the types of information that they are annotating in the corpora. For example, the profile for basic named entity annotation will specify Person, Organization, and Location. These tags will then be provided in the document annotation pages in the client (see Figure 2).

The OLLIE server is implemented as a set of Java Server Pages (JSPs) along with some supporting Java classes. The entire application is then packaged for deployment into a servlet container such as

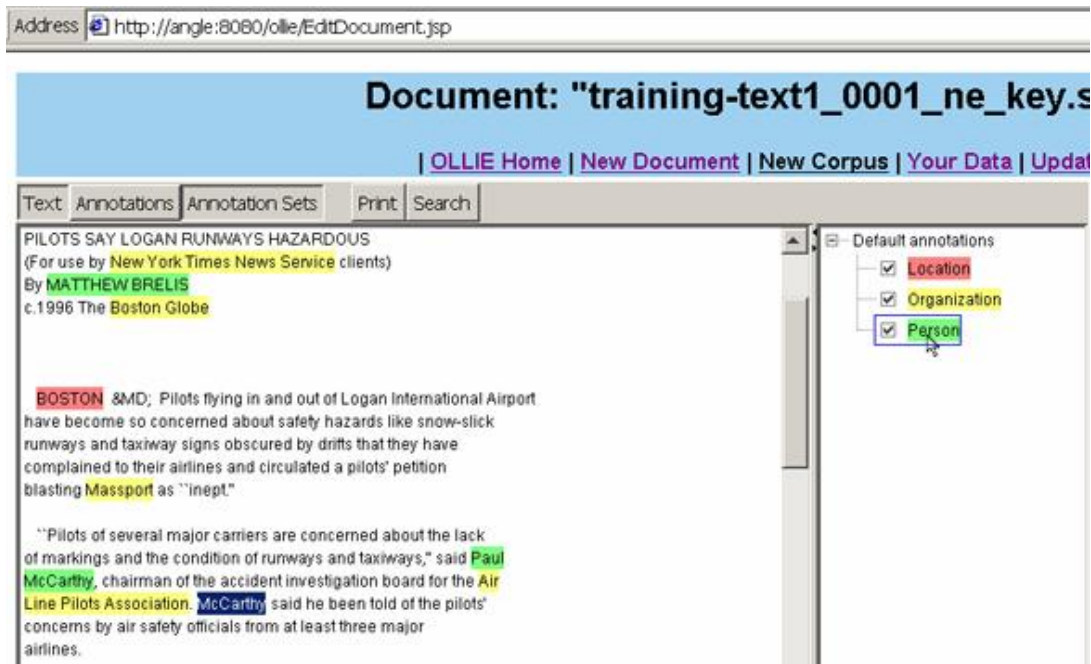


Figure 2 Remote document editing in OLLIE

Apache Tomcat. All the configuration data (e.g. URLs to the databases holding the security data or the GATE distributed database) is handled through the configuration of the servlet container (which in the case of Tomcat consists of a set of XML files). The server's efficiency is improved by using connection pooling for efficient database access.

The client-server communication during on-line document annotation is carried over Java *Remote Method Invocation* (RMI), which supports data sending and exchange of messages between the client and the server. The continuous communication comes with the added benefit of data security in case of network failure. The data on the server always reflects the latest version of the document, so no data loss can occur.

5. The Web-based collaborative corpus annotation client

OLLIE supports collaborative annotation of documents and corpora by allowing their shared, remote use and by making updates made by one client immediately available on the OLLIE server. In this way users can share the annotation task with other users. For example, one user can annotate a text with organisations, and then another annotates it with locations. The documents reside on the shared server which means that one user can see errors or questionable mark-up introduced by another user and initiate a discussion.

The OLLIE client provides facilities for loading documents/corpora for annotation from a URL, uploaded from a file, or created from text pasted in a form. A variety of formats including XML, HTML, email and plain text are supported. As part of this process, the user also specifies the access rights to the document/corpus, which determine whether it can be shared for viewing and collaborative annotation (see Section 3). Figure 1 shows the OLLIE client window listing all corpora and documents accessible to the user and available for annotation through the Edit button. The Admin button is for specifying the access rights and there is also a button for deleting the LR from the server.

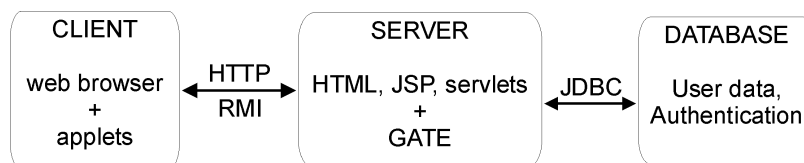


Figure 3 The OLLIE client-server architecture

The document editor is used to annotate the text (see Figure 2). The right-hand side shows the classes of annotations (as specified in the user profile) and the user selects the text to be annotated (e.g., "McCarthy") and clicks on the desired class (e.g., Person). The new annotation is added to the document and the server is updated immediately. The client also provides facilities for deleting wrong annotations, which are then propagated to the server, in a similar way.

The annotation facilities in the OLLIE client are designed to work exactly in the same way as those in GATE's visual environment, so that users familiar with one can use the other without a major overhead. The advantages of using the OLLIE client is that it does not require any installation and runs in any Java-enabled Web browser, which makes it ideal for collaborative corpus annotation, especially if users tend to change their locations and machines. On the other hand, the GATE visual environment provides more extensive facilities, e.g., specialised viewers/editors for complex structures such as syntax trees, while still supporting distributed LRs via the Oracle/PostgreSQL databases, but needs to be installed locally on the machine. In addition, due to its more generic nature, as a development environment for language processing applications, it offers many additional features which make its user interface too generic, whereas OLLIE's client is specifically designed for collaborative corpus annotation.

The users do not need to manually annotate all data, instead they can use GATE's language processing tools to bootstrap the annotations by running them on the corpus. The actual processing occurs on the OLLIE server where the corpus resides and can also be scheduled to take place overnight. While some other infrastructures, e.g., AGTK (Ma et al 2002) offer corpus sharing in a manner similar to GATE, GATE has the advantage of offering the bootstrapping support discussed here.

OLLIE supports bootstrapping with a wide range of linguistic information - part-of-speech, sentence boundaries, gazetteer lists, and named entity class. This information, together with tokenisation information (kind, orthography, and token length) is provided by the language processing components available within GATE, as part of the ANNIE system (Cunningham et al 2002). The user chooses which of this information is needed and the respective language processing components are run on the OLLIE server, the document is updated with new annotations, and they are propagated immediately to the OLLIE client and shown to the user. The user can then correct these annotations by deleting the wrong ones and adding new ones. However, such bootstrapping would help only if the language processing tools have acceptable performance. Therefore, if the automatic results for some annotation types (e.g., Organizations) have proven unreliable, the OLLIE client also offers the facility to delete all annotations of a given type by selecting this type in the right-hand side pane and pressing the Delete key.

6. Web Services based collaboration

Web services are a new breed of web applications that are based on the service-oriented architecture (SOA). Web Services are self-contained, modular applications that can be published, located and invoked across the Internet. They ease the inter-enterprise application integration, collaboration and process automation, reduce complexity associated with development and deployment and make it possible to easily build loosely-coupled systems that allow dynamic binding and just-in-time integration.

At present Web Services functionality is being added to GATE. This will allow clients to use remotely the GATE functionality and use services (processing resources) that are otherwise not available locally. Making GATE Web Services aware will lead to:

- increased collaboration between disparate research sites
- easier integration of the HLT functionality in GATE with other applications
- reduced cost and time of the development of GATE based applications
- easy integration with light-weight and non-java components that require HLT functionality

7. Conclusion

In this paper we described a set of language engineering tools for collaborative corpus annotation, which allow users to share work on language resources distributed over the network and also to automate the annotation process by running relevant language processing tools. These tools form part of GATE, the General Architecture for Text Engineering. Recently we have also provided machine

learning tools as part of GATE, in order to enable users to have trainable NLP tools they can use for corpus annotation.

In future work we plan to enhance the GATE support for collaborative corpus annotation by providing access to the GATE corpora and tools via Web services. As shown by some recent work on using Web services and SOAP for distributed language resources (Dalli 2002), this model combines very well with efficient database storage mechanisms, like the ones used in GATE.

8. Acknowledgements

Work on GATE has been supported by the Engineering and Physical Sciences Research Council (EPSRC) under grants GR/K25267 and GR/M31699, and by several smaller grants. The first author is currently supported by the EPSRC-funded AKT project (<http://www.aktors.org>) grant GR/N15764/01

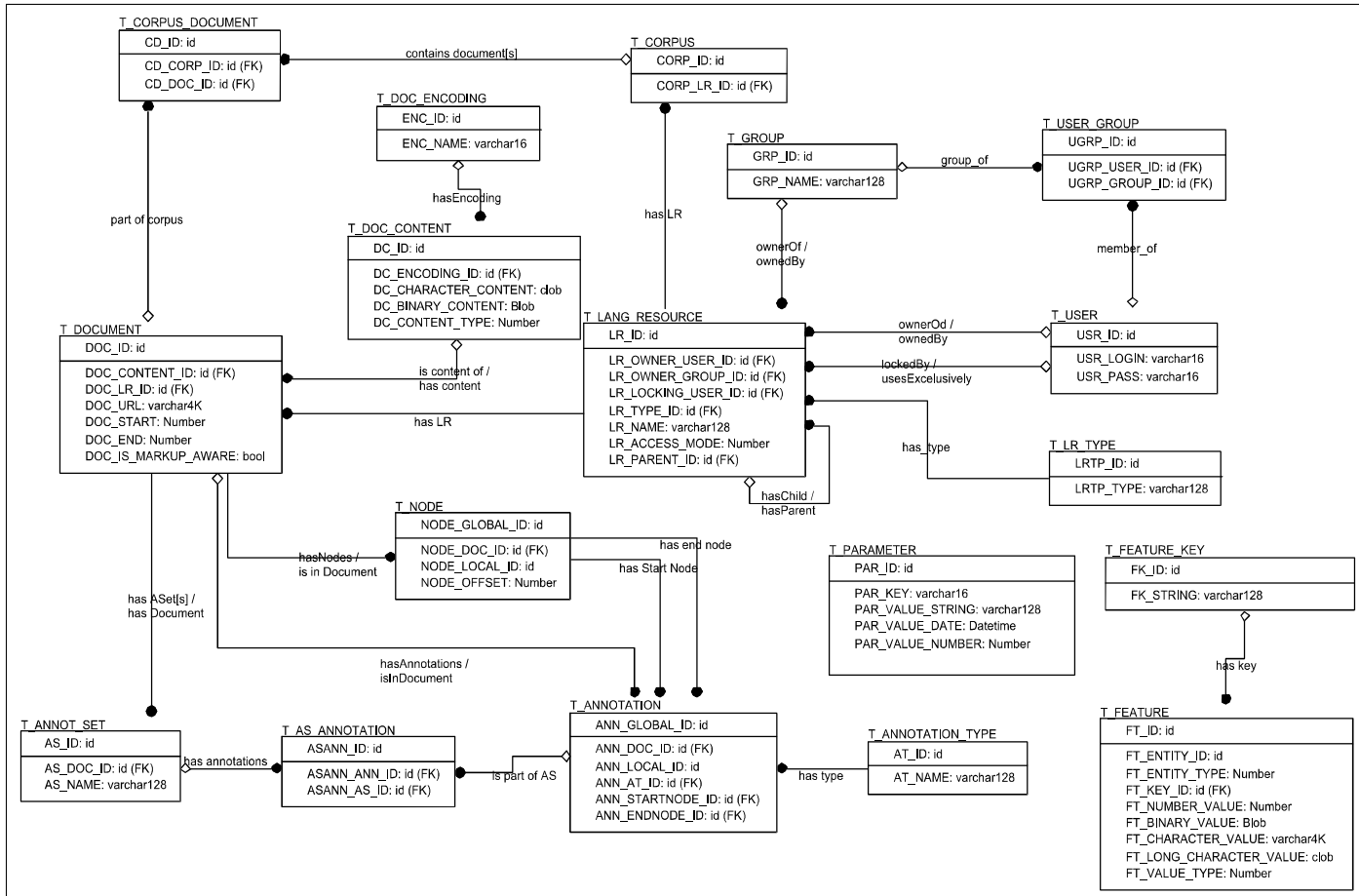


Figure 4 Logical model of the GATE relational schema.

9. References

- (Baker et al.2002) Baker P, Hardie A, McEnery A, Cunningham H, Gaizauskas R. 2002. EMILLE, A 67-Million Word Corpus of Indic Languages: Data Collection, Mark-up and Harmonisation. In *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'2002)*, pages 819--825.
- (Bird and Liberman 1999) Bird S, Liberman M 1999. A Formal Framework for Linguistic Annotation. Technical Report MS-CIS-99-01, Department of Computer and Information Science, University of Pennsylvania. <http://xxx.lanl.gov/~abs/cs.CL/9903003>.
- (Cunningham et al.2002) Cunningham H, Maynard D, Bontcheva K, Tablan V 2002. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics (ACL'02)*. Philadelphia, US.
- (Dalli 2002) A Dalli 2002. Creation and Evaluation of Extensible Language Resources for Maltese. In: *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'2002)*. Gran Canaria, Spain.
- (Gamback and Olsson 2000) Gamback B, Olsson F 2000. Experiences of Language Engineering Algorithm Reuse. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC)*, pages 155--160, Athens, Greece.
- (Grishman 1997) Grishman R 1997. TIPSTER Architecture Design Document Version 2.3. Technical report, DARPA. http://www.itl.nist.gov/div894/894.02/related_projects/tipster/.
- (Ide et al. 2000) Ide N, Bonhomme P, Romary L 2000. XCES: An XML-based Standard for Linguistic Corpora. In *Proceedings of the Second International Language Resources and Evaluation Conference (LREC)*, pages 825--830, Athens, Greece.
- (Ma et al. 2002) Ma X, Lee H, Bird S, Maeda K 2002. Models and tools for collaborative annotation. In *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'2002)*, Gran Canaria, Spain.
- (Pastra et al. 2002) Pastra K, Maynard D, Hamza O, Cunningham H, Wilks Y 2002. How feasible is the reuse of grammars for Named Entity Recognition? In *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'02)*, Gran Canaria, Spain.
- (Tablan et al. 2002) Tablan V, Ursu C, Bontcheva K, Cunningham H, Maynard D, Hamza O, McEnery A, Baker P, Leisher M 2002. A Unicode-based environment for creation and use of language resources. In *Proceedings of 3rd Language Resources and Evaluation Conference (LREC'02)*, Gran Canaria, Spain.
- (Oracle) Oracle database documentation from <http://technet.oracle.com>
- (Postgres) Postgres database documentation from <http://www.postgresql.com/docs>