GATE – a General Architecture for Text Engineering

Hamish Cunningham
Institute for Language,
Speech and Hearing /
Dept. Computer Science
Univ. Sheffield, UK
hamish@dcs.shef.ac.uk

Yorick Wilks
Institute for Language,
Speech and Hearing /
Dept. Computer Science
Univ. Sheffield, UK
yorick@dcs.shef.ac.uk

Robert J. Gaizauskas
Institute for Language,
Speech and Hearing /
Dept. Computer Science
Univ. Sheffield, UK
robertg@dcs.shef.ac.uk

http://www.dcs.shef.ac.uk/research/groups/nlp/nlp.html

Abstract

Much progress has been made in the provision of reusable data resources for Natural Language Engineering, such as grammars, lexicons, thesauruses. Although a number of projects have addressed the provision of reusable algorithmic resources (or 'tools'), takeup of these resources has been relatively slow. This paper describes GATE, a General Architecture for Text Engineering, which is a freely-available system designed to help alleviate the problem.

1 Resource Reuse and Natural Language Engineering

Car designers don't reinvent the wheel each time they plan a new model, but software engineers often find themselves repetitively producing roughly the same piece of software in slightly different form. The reasons for this ineffcency have been extensively studied, and a number of solutions are now available (Prieto-Diaz and Freeman, 1987; Prieto-Diaz, 1993). Similarly, the Natural Language Engineering (NLE[1]) community has identified the potential benefits of reducing repetition, and work has been funded to promote reuse. This work concerns either reusable resources which are primarily *data* or those which are primarily *algorithmic* (i.e. processing 'tools', or programs, or code libraries).

Successful examples of reuse of data resources include: the WordNet thesaurus (Miller et al., 1993); the Penn Tree Bank (Marcus et al., 1993); the Longmans Dictionary of Contemporary English (Summers, 1995). A large number of papers report results relative to these and other resources, and these successes have spawned a number of projects with similar directions, one of the latest examples of which being ELRA, the European Language Resources Association.

The reuse of algorithmic resources remains more limited (Cunningham et al., 1994). There are a number of reasons for this, including:

1. cultural resistance to reuse, e.g. mistrust of 'foreign' code;
2. integration overheads.

In some respects these problems are insoluble without general changes in the way NLE research is done – researchers will always be reluctant to use poorly-documented or unreliable systems as part of their work, for example. In other respects, solutions are possible. They include:

1. increasing the granularity of the units of reuse, i.e. providing sets of small building-blocks instead of large, monolithic systems;
2. increasing the confidence of researchers in available algorithmic resources by increasing their reuse and the amount of testing and evaluation they are subjected to;
3. separating out the integration problems that are independent of the type of information being processed and reducing the overhead caused by these problems by providing a software architecture for NLE systems.

Our view is that succesful algorithmic reuse in NLE will require the provision of support software for NLE in the form of a general architecture and development environment which is specifically designed for text processing systems. Under EPSRC[2] grant GR/K25267 the NLP group at the University of Sheffield are developing a system that aims to implement this new approach. The system is called GATE – the General Architecture for Text Engineering.

[1]See (Boguraev et al., 1995) or (Cunningham et al., 1995) for discussion of the significance of this label.

[2]The Engineering and Physical Science Research Council, UK funding body.

GATE is an *architecture* in the sense that it provides a common infrastructure for building language engineering (LE) systems. It is also a *development environment* that provides aids for the construction, testing and evaluation of LE systems (and particularly for the reuse of existing components in new systems). Section 2 describes GATE.

A substantial amount of work has already been done on architecture for NLE systems (and GATE reuses this work wherever possible). Three existing systems are of particular note:

- ALEP (Simpkins, Groenendijk 1994), which turns out to be a rather different enterprise from ours;
- MULTEXT (Thompson, 1995), a different but largely complimentary approach to some of the problems addressed by GATE, particularly strong on SGML support;
- TIPSTER (ARPA, 1993a) whose architecture (TIPSTER, 1994; Grishman, 1994) has been adopted as the storage substructure of GATE, and which has been a primary influence in the design and implementation of the system.

See (Cunningham et al., 1995) for details of the relation between GATE and these projects.

2 GATE

Architecture overview

GATE presents LE researchers or developers with an environment where they can use tools and linguistic databases easily and in combination, launch different processes, say taggers or parsers, on the same text and compare the results, or, conversely, run the same module on different text collections and analyse the differences, all in a user-friendly interface. Alternatively module sets can be assembled to make e.g. IE, IR or MT systems. Modules and systems can be evaluated (using e.g. the Parseval tools), reconfigured and reevaluated – a kind of edit/compile/test cycle for LE components.

GATE comprises three principal elements:

- a database for storing information about texts and a database schema based on an object-oriented model of information about texts (the GATE Document Manager – GDM);
- a graphical interface for launching processing tools on data and viewing and evaluating the results (the GATE Graphical Interface – GGI);
- a collection of wrappers for algorithmic and data resources that interoperate with the database and interface and constitute a Collection of REusable Objects for Language Engineering – CREOLE.

GDM is based on the TIPSTER document manager. We are planning to enhance the SGML capabilities of this model by exploiting the results of the MULTEXT project.

GDM provides a central repository or server that stores all the information an LE system generates about the texts it processes. All communication between the components of an LE system goes through GDM, insulating parts from each other and providing a uniform API (applications programmer interface) for manipulating the data produced by the system.[3] Benefits of this approach include the ability to exploit the maturity and efficiency of database technology, easy modelling of blackboard-type distributed control regimes (of the type proposed by: (Boitet and Seligman, 1994) and in the section on control in (Black ed., 1991)) and reduced interdependence of components.

GGI is in development at Sheffield. It is a graphical launchpad for LE subsystems, and provides various facilities for viewing and testing results and playing software lego with LE components – interactively assembling objects into different system configurations.

All the real work of analysing texts (and maybe producing summaries of them, or translations, or SQL statements, etc.) in a GATE-based LE system is done by CREOLE modules.

Note that we use the terms *module* and *object* rather loosely to mean interfaces to resources which may be predominantly algorithmic or predominantly data, or a mixture of both. We exploit object-orientation for reasons of modularity, coupling and cohesion, fluency of modelling and ease of reuse (see e.g. (Booch, 1994)).

Typically, a CREOLE object will be a wrapper around a pre-existing LE module or database – a tagger or parser, a lexicon or n-gram index, for example. Alternatively objects may be developed from scratch for the architecture – in either case the object provides a standardised API to the underlying resources which allows access via GGI and I/O via GDM. The CREOLE APIs may also be used for programming new objects.

The initial release of GATE will be delivered with a CREOLE set comprising a complete MUC-compatible IE system (ARPA, 1996). Some of

[3]Where very large data sets need passing between modules other external databases can be employed if necessary.

the objects will be based on freely available software (e.g. the Brill tagger (Brill, 1994)), while others are derived from Sheffield's MUC-6 entrant, LaSIE[4] (Gaizauskas et al., 1996). This set is called VIE – a Vanilla IE system. CREOLE should expand quite rapidly during 1996-7, to cover a wide range of LE R&D components, but for the rest of this section we will use IE as an example of the intended operation of GATE.

The recent MUC competition, the 6th, defined four IE tasks to be carried out on Wall Street Journal articles. Developing the MUC system upon which VIE is based took approximately 24 person-months, one significant element of which was coping with the strict MUC output specifications. What does a research group do which either does not have the resources to build such a large system, or even if it did would not want to spend effort on areas of language processing outside of its particular specialism? The answer until now has been that these groups cannot take part in large-scale system building, thus missing out on the chance to test their technology in an application-oriented environment and, perhaps more seriously, missing out on the extensive quantitative evaluation mechanisms developed in areas such as MUC. In GATE and VIE we hope to provide an environment where groups can mix and match elements of our MUC technology with components of their own, thus allowing the benefits of large-scale systems without the overheads. A parser developer, for example, can replace the parser supplied with VIE.

Licencing restrictions preclude the distribution of MUC scoring tools with GATE, but Sheffield may arrange for evaluation of data produced by other sites. In this way, GATE/VIE will support comparative evaluation of LE components at a lower cost than the ARPA programmes (ARPA, 1993a) (partly by exploiting their work, of course!). Because of the relative informality of these evaluation arrangements, and as the range of evaluation facilities in GATE expands beyond the four IE task of the current MUC we should also be able to offset the tendency of evaluation programmes to dampen innovation. By increasing the set of widely-used and evaluated NLP components GATE aims to increase the confidence of LE researchers in algorithmic reuse.

Working with GATE/VIE, the researcher will from the outset reuse existing components, the overhead for doing so being much lower than is conventionally the case – instead of learning new tricks for each module reused, the common APIs of GDM and CREOLE mean only one integration mechanism must be learnt. And as CREOLE expands, more and more modules and databases will be available at low cost. We hope to move towards sub-component level reuse at some future point, possibly providing C++ libraries as part of an OO LE framework (Cunningham et al., 1994). This addresses the need for increased granularity of the units of reuse as noted in section 1.

As we built our MUC system it was often the case that we were unsure of the implications for system performance of using tagger X instead of tagger Y, or gazeteer A instead of pattern matcher B. In GATE, substitution of components is a point-and-click operation in the GGI interface. (Note that delivered systems, e.g. EC project demonstrators, can use GDM and CREOLE without GGI – see below.) This facility supports hybrid systems, ease of upgrading and open systems-style module interchangeability.

Of course, GATE does not solve all the problems involved in plugging diverse LE modules together. There are two barriers to such integration:

- incompatability of *representation* of information about text and the mechanisms for storage, retrieval and inter-module communication of that information;
- incompatability of *type* of information used and produced by different modules.

GATE enforces a separation between these two and provides a solution to the former (based on the work of the TIPSTER architecture group (TIPSTER, 1994)). This solution is to adopt a common model for expressing information about text, and a common storage mechanism for managing that information, thereby cutting out significant parts of the integration overheads that often block algorithmic reuse. Because GATE places no constraints on the linguistic formalisms or information content used by CREOLE objects (or, for that matter, the programming language they are implemented in), the latter problem must be solved by dedicated translation functions – e.g. tagset-to-tagset mapping – and, in some cases, by extra processing – e.g. adding a semantic processor to complement a bracketing parser in order to produce logical form to drive a discourse interpreter. As more of this work is done we can expect the overhead involved to fall, as all results will be available as CREOLE objects. In the early stages Sheffield will provide some resources for this work in order to get the ball rolling, i.e. we will provide help with CREOLEising existing systems and with developing interface routines where practical and necessary. We are confident that integration

[4]Large-Scale IE.

is possible (partly because we believe that differences between representation formalisms tend to be exaggerated) – and others share this view, e.g. the MICROKOSMOS project (Onyshkevych et al., 1994).

GATE is also intended to benefit the LE system developer (which may be the LE researcher with a different hat on, or industrialists implementing systems for sale or for their own text processing needs). A delivered system comprises a set of CREOLE objects, the GATE runtime engine (GDM and associated APIs) and a custom-built interface (maybe just character streams, maybe a Visual Basic Windows GUI, ...). The interface might reuse code from GGI, or might be developed from scratch. The LE user has the possibility to upgrade by swapping parts of the CREOLE set if better technology becomes available elsewhere.

GATE cannot eliminate the overheads involved with porting LE systems to different domains (e.g. from financial news to medical reports). Tuning LE system resources to new domains is a current research issue (see also the LRE DELIS and ECRAN projects). The modularity of GATE-based systems should, however, contribute to cutting the engineering overhead involved.

References

Advanced Research Projects Agency. 1993. Proceedings of TIPSTER Text Program (Phase I). Morgan Kaufman.

Advanced Research Projects Agency. 1996. Proceedings of the Sixth Message Understanding Conference (MUC-6). Morgan Kaufmann.

Black W.J. (ed.). 1991. PLUS – a Pragmatics-Based Language Understanding System, Functional Design. ESPRIT P5254 Deliverable D1.2.

Boguraev B., R. Garigliano, J. Tait. 1995. Editorial. *Journal of Natural Language Engineering*, Vol. 1 Part 1, Cambridge University Press.

Boitet C., M. Seligman. 1994. The "Whiteboard" Architecture: A Way to Integrate Heterogeneous Components of NLP Systems. Proceedings of COLING.

Booch G. 1994. Object-oriented Analysis and Design 2nd. Ed. Addison Wesley.

Brill E. 1994. Some Advances in Transformation-Based Part of Speech Tagging. Proceedings of The Twelfth National Conference on Artificial Intelligence (AAAI-94), Seattle, Washington.

Cunningham H., M. Freeman, W.J. Black. 1994. Software Reuse, Object-Orientated Frameworks and Natural Language Processing. Conference on New Methods in Natural Language Processing, Manchester.

Cunningham H., R.J. Gaizauskas, Y. Wilks. 1995. A General Architecture for Text Engineering (GATE) – a new approach to Language Engineering R&D. Technical Report CS – 95 – 21, Department of Computer Science, University of Sheffield.

Gaizauskas R.J., K. Humphreys, T. Wakao, H. Cunningham, Y. Wilks. 1996. LaSIE – a Large-Scale Information Extraction System. In (ARPA, 1996).

Grishman R. 1994. TIPSTER II Architecture Design Document Version 1.52 (Tinman Architecture). TIPSTER working paper 1995, available at http://www.cs.nyu.edu/tipster.

Marcus M., B. Santorini, M. Marcinkiewicz. 1993. Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics* 19(2).

Miller G.A., R. Beckwith, C. Fellbaum, D. Gross, K. Miller. 1993. Introduction to WordNet: an On-line Lexical Database. Distributed with the WordNet software, 1993.

Onyshkevych B., Boyan, S. Nirenburg. 1996. *Machine Translation* 10:1-2, 1996. (Special issue on building lexicons for machine translation.)

Prieto-Diaz R., P. Freeman. 1987. Status Report: Software Reusability. *IEEE Software*, Vol. 4, No. 1.

Prieto-Diaz R. 1993. Status Report: Software Reusability. *IEEE Software*, Vol. 10, No. 3.

Simpkins N., M. Groenendijk. 1994. The ALEP Project. Cray Systems / CEC, Luxemburg.

Summers D. et al. eds. 1995. Longman Dictionary of Contemporary English, 3rd Edition. Longman, Harlow.

Thompson H. 1995. MULTEXT Workpackage 2 Milestone B Deliverable Overview. LRE 62-050 MULTEXT Deliverable 2.

TIPSTER Architecture Committee. 1994. TIPSTER Text Phase II Architecture Concept. TIPSTER working paper 1994, available at http://www.cs.nyu.edu/tipster.