Software Infrastructure for Language Engineering

Hamish Cunningham
Yorick Wilks
Robert J. Gaizauskas

Institute for Language, Speech and Hearing (ILASH), and
Department of Computer Science
University of Sheffield, UK

h.cunningham@dcs.shef.ac.uk
http://www.dcs.shef.ac.uk/research/groups/nlp/gate.html
http://www.dcs.shef.ac.uk/~hamish

April 2nd 1996
AISB LEDAR Workshop
Sussex University

This work was supported by
the EPSRC (grant GR/K25267)

1 Introduction

An increasing number of research and development efforts have recently positioned themselves under the banner *Language Engineering* (LE). This signals a shift away from well-established labels such as *Natural Language Processing* (NLP) and *Computational Linguistics*. Examples include the renaming of UMIST's[1] *Department of Language and Linguistics* (location of the *Centre for Computational Linguistics*) as the *Department of Language Engineering*, and the naming of the European Commission's current relevant funding programme *Language Engineering* (the previous programme was called *Linguistic Research and Engineering*). The new journal of *Natural Language Engineering* is another example[2], as is the name of this workshop.

Engineering language-based systems is a qualitatively different process from scientific investigation of the nature of language or of the properties of linguistic formalisms or algorithms. Engineered systems are evaluated by criteria such as costs vs. benefits, and deploy technology on the grounds of expected conformance to pre-specified performance targets. The increasing focus on engineering in our field reflects improved prospects for viable applications of language processing technology. In this context the software infrastructure used to deliver NLP technology becomes central to the efficiency of the development process. Several research programmes have recognised this, and several standard software architectures have been proposed.

This paper describes a freely-available system called GATE [Cunningham, Gaizauskas, Wilks 1995] – a General Architecture for Text Engineering – which integrates much of this work. GATE is an *architecture* in the sense that it provides a common infrastructure for building LE systems (rather like the frame of a building or the interface specifications for the bus and peripherals of a computer). 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).

GATE is distributed with an Information Extraction (IE) system as used in the Sixth Message Understanding Conference (MUC-6 [ARPA, 1996]) and this is described. We also detail arrangements for collaborative work with GATE.

GATE profits extensively from related projects, and we review two existing systems:

- MULTEXT [Thompson 1995a; Ballim 1995; Finch, Thompson, McKelvie 1995], a different but largely complementary approach to some of the problems addressed by GATE, which is particularly strong on SGML support and elements of which we intend to integrate with GATE;
- TIPSTER [ARPA 1993a], whose architecture [Grishman 1995] has been adopted as the storage substructure of GATE, and which has been a primary influence in the design and implementation of our system.

2 GATE

The increasing trend towards engineering in NLP leads to a number of requirements for support software that aims to address the infrastructural needs of the field. A general architecture for LE R&D should:

[1] The University of Manchester Institute of Science and Technology, Manchester, UK

[2] The editorial of the first issue also discusses the new name [Boguraev, Garigliano, Tait 1995].

- support collaborative research;
- support hybrid systems, 'plug-and-play' module interchangeability, and easy upgrading;
- support the reuse of existing and future algorithmic components and data resources, whether they be the results of PhD projects or multinational strategic initiatives;
- contribute to software-level robustness, quality and efficiency;
- contribute to portability across problem domains and application areas;
- support comparative evaluation, preferably at lower cost than the US MUC and TREC (Text REtrieval Conference – the information retrieval equivalent of MUC) programmes and without stifling innovation;
- contribute to software portability across languages and across operating systems and programming languages.

We believe that these requirements may best be met by the provision of dedicated support software for researchers and applications developers, in the form of an *architecture* and *development environment*, and we have developed an initial version of such a system, called GATE - a General Architecture for Text Engineering.

Architecture overview

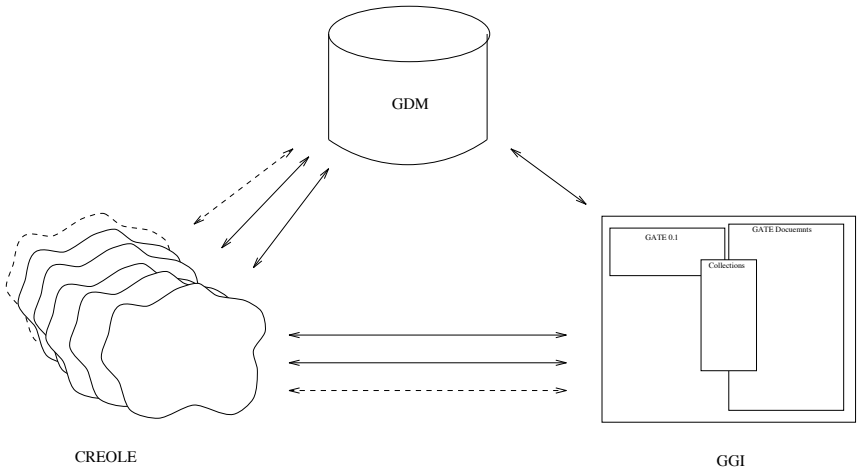

Figure 1: The three elements of GATE

GATE comprises three principal elements (figure 1):

- 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. It is planned to enhance the SGML capabilities of this model, possibly by exploiting the results of the MULTEXT project (we thank colleagues from ISSCO and Edinburgh for making available documentation and advice on this subject).

GDM provides a central repository or server that stores all 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, Seligman 1994; 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 stringing objects into different system configurations.

All the real work of analysing texts (and maybe producing summaries of them, or translations, or SQL statements...) 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 (to begin with, more of a pidgin than a creole!). Some of 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, Humphreys, Wakao, Cunningham 1995; Gaizauskas, Humphreys, Wakao, Cunningham, Wilks 1996]. This set is called VIE – a Vanilla IE system. See section 3 for an overview. CREOLE will expand quite rapidly during 1996, to cover a wide range of LE R&D components (such as those currently available at the ACL-sponsored Natural Language Software Registry at DFKI[5]), but for the rest of this section we'll use IE as an example of the intended operation of GATE.

The recent MUC competition, the sixth, defined four IE tasks to be carried out on Wall Street Journal articles. Sheffield's system did well, scoring in the middle of the pack in general and doing as well as the best systems in some areas. Developing this system took 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

[3] Where very large data sets need passing between modules other external databases can be employed if necessary.
[4] Large-Scale IE.
[5] URL: http://cl-www.dfki.uni-sb.de/cl/registry/draft.html

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 MUC technology from other sites (including ours) 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 will 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 programme (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 tasks of the current MUC, we should also be able to offset the tendency of evaluation programmes to dampen innovation.

Similarly we aim to make collaboration between research groups much easier. Sites specialising on different LE subtasks can combine their efforts into bigger application-oriented systems with minimal overhead. We hope that we can help the community squeeze a little more research time out of industrially-oriented projects by cutting down on the time spent integrating research work into demonstrator systems.

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 also endorse object orientation (OO) in this context, as an enabling technology for reuse [Booch 1994], and hope to move towards sub-component level reuse at some future point, possibly providing C++ libraries as part of an OO LE framework [Cunningham, Freeman, Black 1994].

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 gazetteer 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]. Because GATE places no constraints on the linguistic formalisms or information content used by CREOLE objects, 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 *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 [Beale, Nirenburg, Mahesh 1995], which seeks to integrate many types of knowledge source in a useable whole, as well as the LexiCadCam experience at New Mexico [Wilks, Guthrie, Slator 1996] which sought to provide core lexical information as needed in a range of user-specified formats.

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). Using GATE for the delivery of a system is illustrated in figure 2. A delivered system

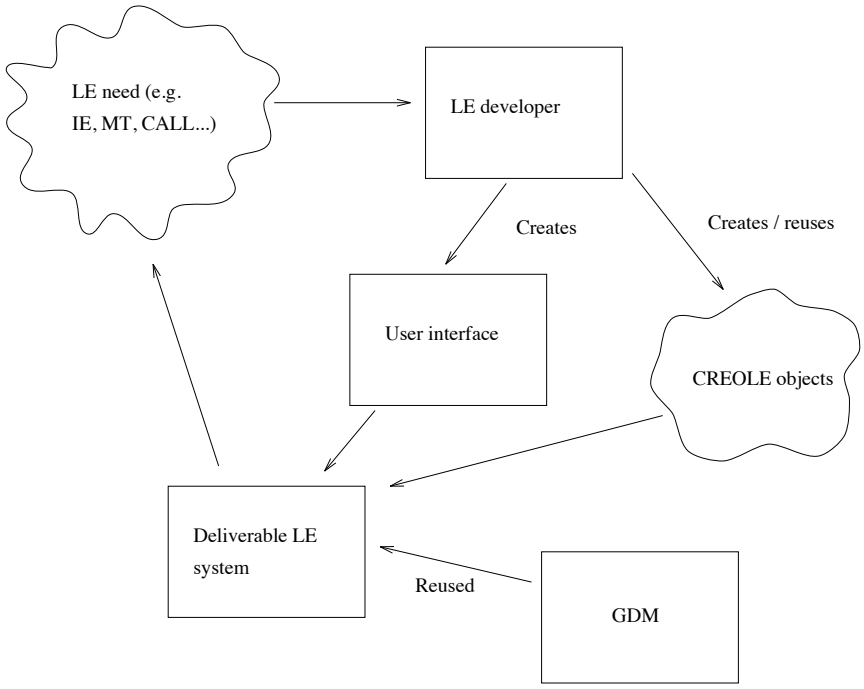

Figure 2: Delivering systems with GATE

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 may upgrade by swapping parts of the CREOLE set if better technology becomes available elsewhere. This model for the commercialisation of LE technology is already begininning to operate in the US, where a number of organisations are preparing TIPSTER-compatible modules for sale or distribution for research. (These organisations include NMSU, SRA, HNC, University of Massachusetts, Paracell, Logicon [Dunning 1995, personal communication].) All TIPSTER-compatibile modules will also work with GATE, as GATE itself is designed to be a TIPSTER-compatible system. Thus the pool of easily reusable LE resources available to researchers and developers using GATE has the potential to become a large, rich set of modules from a good proportion of the LE community world-wide. Also, it may well become the case that organisations purchasing LE software will require TIPSTER compatability (this will be true of US government organisations, for example).

At the software engineering level GATE:

- contributes to robustness and quality by providing a mature infrastructure;
- contributes to efficiency via the design of the TIPSTER text model and by access to fast database technology underlying GDM.

As regards operating system independence, GDM and GGI will initially be available for Linux, SunOS, Solaris 2 and other UNIX platforms as required, but will avoid using UNIX-specific facilities. A Windows version may follow at some point. CREOLE portability is more difficult. GATE places no constraints on the implementation languages and platforms of CREOLE objects, so they may or may not be portable.

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 ECRAN project[6]; the SEAL project [Evans, Kilgariff 1995]). The modularity of GATE-based systems should, however, contribute to cutting the engineering overhead involved.

Collaboration using GATE and VIE

Sheffield will support collaborative work using GATE/VIE for LE research groups (typically academic groups), businesses with IE needs and producers of lexicons and dictionaries. The three groups are *technology*, *data* and *resource providers* respectively, contributing CREOLE modules, test data (e.g. manually extracted information and the relevant source texts) and machine-readable language resources (e.g. dictionaries). The projected benefits for participants include:

- comparative quantitative evaluation of candidate technologies for IE;
- technology providers may specialise on components of the IE task, avoiding the overhead of providing a complete IE system while still working within the framework of a complete NLP application;
- data providers (typically industrial concerns) get access to IE technology applied to their particular textual problem domains;
- resource providers can assess the performance of their products and increase the market for them by encouraging their use in applied LE systems.

Note that there will be no requirement to supply source code for contributed modules, and that intellectual property and other rights will be safeguarded by appropriate legal agreements.

3 VIE, a Vanilla Information Extraction system

As originally envisaged [Wilks, Gaizauskas 1994], GATE will be distributed with a set of CREOLE objects that together implement a complete information extraction system capable of producing results compatible with the MUC-6 task definitions. This CREOLE set is called VIE, a Vanilla IE system, and it is intended that participating sites use VIE as the basis for specialising on sub-tasks in IE. By replacing a particular VIE module – the parser, for example – a participating group will immediately be able to evaluate their specialist technology's potential contribution to full-scale IE applications. Sheffield has access to the MUC-6 scoring tools (and the PARSEVAL software) and will run periodic evaluations of various VIE-based configurations.

[6] http://www.dcs.shef.ac.uk/research/groups/nlp/ecran.html

It is envisaged that LE research groups (typically academic research groups) supply modules to replace parts of VIE. Businesses with IE needs can also participate in the programme by contributing test sets and task definitions. Resource builders like dictionary publishers will be approached to supply research versions of their online texts.

The most recent MUC competition, MUC-6, defined four tasks to be carried out on Wall Street Journal articles: named entity (NE) recognition, the recognition and classification of definite entities such as names, dates, places; coreference (CO) resolution, the identification of identity relations between entities (including anaphoric references to them); template element (TE) construction, a fixed-format, database-like enumeration of organisations and persons and attributes associated with them; scenario template (ST) construction, the detection of specific relations holding between template elements relevant to a particular information need (in this case managers joining and leaving companies) and construction of a fixed-format structure recording the entities and details of the relation.

VIE is an integrated system that builds up a single, rich model of a text which is then used to produce outputs for all four of the MUC-6 tasks. Of course this model may also be used for other purposes aside from MUC-6 results generation, for example we currently generate natural language summaries of the MUC-6 scenario results.

Put most broadly, and superficially, VIE's approach involves compositionally constructing semantic representations of individual sentences in a text according to semantic rules attached to phrase structure constituents which have been obtained by syntactic parsing using a corpus-derived context-free grammar. The semantic representations of successive sentences are then integrated into a 'discourse model' which, once the entire text has been processed, may be viewed as a specialisation of a general domain model with which the system sets out to process each text.

Features which distinguish the system are:

- an integrated approach allowing knowledge at several linguistic levels to be applied to each MUC-6 task (e.g. coreference information is used in named entity recognition);
- the absence of any overt lexicon – lexical information needed for parsing is computed dynamically through part-of-speech-tagging and morphological analysis;
- the use of a grammar derived semi-automatically from the Penn TreeBank corpus;
- the use and dynamic acquisition of a world model, in particular for the coreference and scenario tasks;
- a summarisation module which produces a brief natural language summary of scenario events.

VIE will be available with the initial release of GATE.

4 Related work

We discuss two projects here, MULTEXT and the TIPSTER Architecture project.

MULTEXT

MULTEXT [Thompson 1995, Finch, Thompson, McKelvie 1995, Ballim 1995] is another EC project, whose objective is to produce tools for multilingual corpus annotation and sample corpora marked-up according to the same standards used to drive the tool development. Annotation tools under

development perform text segmentation, POS tagging, morphological analysis and parallel text alignment. The project has defined an architecture centred on a model of the data passed between the various phases of processing implemented by the tools. The MULTEXT architecture is based on a commitment to TEI-style[7] SGML[8] encoding of information about text. The TEI defines standard tag sets for a range of purposes including many relevant to LE systems. Tools in a MULTEXT system communicate via interfaces specified as SGML document type definitions (DTDs – essentially tag set descriptions).

TIPSTER II

The TIPSTER programme in the US, currently in its second phase, has also produced a data-driven architecture for NLP systems [Grishman, Dunning, Callan 1994, TIPSTER 1994]. Like MULTEXT, TIPSTER addresses specific forms of language processing, in this case information extraction and document detection (or information retrieval – IR). As will become clear below, however, TIPSTER's approach is not restricted to particular NL tasks.

Whereas in MULTEXT all information about a text is encoded in SGML, which is added by the tools, in TIPSTER a text remains unchanged while information is stored in a separate database in the form of *annotations*. Annotations associate portions of documents (identified by sets of start/end byte offsets or *spans*) with analysis information (*attributes*), e.g.: POS tags; textual unit type; template element. In this way the information built up about a text by NLP (or IR) modules is kept separate from the texts themselves. In place of an SGML DTD an *annotation type declaration* defines the information present in annotation sets, for example a set of values for MUC-style organisation template elements. Figure 3 shows an example from [Grishman, Dunning, Callan 1994]. SGML I/O is catered for by API calls to import and export SGML-encoded text.

Text

```
Sarah savored the soup.
0...|5...|10..|15..|20
```

Annotations

Id	Type	Span		Attributes
		Start	End	
1	token	0	5	pos=NP
2	token	6	13	pos=VBD
3	token	14	17	pos=DT
4	token	18	22	pos=NN
5	token	22	23	
6	name	0	5	name_type=person
7	sentence	0	23	

Figure 3: TIPSTER annotations example

The definition of annotations in TIPSTER forms part of an object-oriented model that deals with inter-textual information as well as single texts. Documents are grouped into *collections*, each with a database storing annotations and document attributes such as identifiers, headlines etc. Collections are the first-class entities in the architecture. The model also describes elements of IE

[7]The Text Encoding Initiative [Sperberg-McQueen, Burnard 1994].
[8]The Standard Generalised Markup Language [Goldfarb 1990].

and IR systems relating to their use, with classes representing queries and information needs.

Comparison of MULTEXT and TIPSTER

Both projects propose architectures appropriate for LE, but there are a number of significant differences. We discuss six here, then note the possibility of complimentary interoperation of the two.

1. MULTEXT adds new information to documents by augmenting an SGML stream; TIPSTER stores information remotely in a dedicated database. This has several implications. Firstly, TIPSTER can support documents on read-only media (e.g. CD-ROMs). Secondly, TIPSTER avoids the difficulties of representing graph-structured information in SGML. From the point of view of efficiency, the original MULTEXT model of interposing SGML between all modules implies a generation and parsing overhead in each module. Later versions have replaced this model with a compiled representation of SGML to reduce this overhead. This representation will presumably be stored in intermediate files, which implies an overhead from the I/O involved in continually reading and writing all the data associated with a document to file. There would seem no reason why these files should not be replaced by a database implementation, however, with potential performance benefits from the ability to do I/O on subsets of information about documents (and from the high level of optimisation present in modern database technology).
2. A related issue is storage overhead. TIPSTER is minimal in this respect, as there is no inherent need to duplicate the source text (which also means that it works naturally with read-only media like CD-ROMs). MULTEXT potentially has to duplicate the source text at each intermediary stage, although this might be ameliorated by shifting to a database implementation.
3. TIPSTER's data architecture is application-neutral – the objects in the model are generic to all information that is associated with definite ranges of text. (The more concrete aspects of the architecture to do with IE and IR model the objects involved in user interaction with such systems.) MULTEXT's model is tool-specific, as noted above (although the underlying representation language, SGML, is information-neutral).
4. Distributed control is easy to implement in a database-centred system like TIPSTER – the DB can act as a blackboard, and implementations can take advantage of well-understood access control (locking) technology. How to do distributed control in MULTEXT is not obvious.
5. TIPSTER provides no tools or databases (or at least not ones that those outside of the U.S. inner circle can gain access to). MULTEXT is based around a set of tools and reference corpora annotated accordingly.
6. Working implementations of TIPSTER have been available for some months now; MULTEXT will be distributed in 1996.

Interestingly, a TIPSTER system could function as a module in a MULTEXT system, or vice-versa. A TIPSTER storage system could write data in SGML for processing by MULTEXT tools, and convert the SGML results back into native format. Also, the extensive work done on SGML processing in MULTEXT could usefully fill a gap in the current TIPSTER model, in which SGML capability is not fully specified. Integration of the results of both projects would seem to be the best of both worlds, and we hope to achieve this in GATE.

References

[ARPA 1992] Advanced Research Projects Agency. 1992. Proceedings of the Fourth Message Understanding Conference (MUC-4). Morgan Kaufmann.

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

[ARPA 1993b] Advanced Research Projects Agency. 1993. Proceedings of the Fifth Message Understanding Conference (MUC-5). Morgan Kaufmann.

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

[Ballim 1995] Ballim A. 1995. Abstract Data Types for MULTEXT Tool I/O. LRE 62-050 Deliverable 1.2.1.

[Beale, Nirenburg, Mahesh 1995] Beale S., S. Nirenburg, K. Mahesh. 1995. Semantic Analysis in the Mikrokosmos Machine Translation Project. Proceedings of the Second Symposium on Natural Language Processing (SNLP-95), August 2-4, 1995, Bangkok, Thailand.

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

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

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

[Booch 1994] Booch G. 1994. Object-oriented Analysis and Design 2nd. Edtn. Addison Wesley.

[Brill 1994] 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, Freeman, Black 1994] 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, Gaizauskas, Wilks 1995] Cunningham H., R.J. Gaizaukas, 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. Also available as paper 9601009 from the Computation and Language E-Print Archive http://xxx.lanl.gov/cmp-lg.

[Evans, Kilgarriff 1995] Evans R., A. Kilgarriff. 1995. Standards and How to do Lexical Engineering. *Proceedings of the Second Language Engineering Convention*, London 1995.

[Finch, Thompson, McKelvie 1995] Finch S., H. Thompson, D. McKelvie. 1995. Specification of Tool Shell with Discussion of Data and Process Architecture. LRE 62-050 Deliverable 1.2.2.

[Gaizauskas, Humphreys, Wakao, Cunningham 1995] Gaizauskas R.J., K. Humphreys, T. Wakao, H. Cunningham. 1995. LaSIE – a Large-Scale Information Extraction System. Technical report CS - 95 - 27, Department of Computer Science, University of Sheffield.

[Gaizauskas, Humphreys, Wakao, Cunningham, Wilks 1996] Gaizauskas R.J., K. Humphreys, T. Wakao, H. Cunningham, Y. Wilks. 1996. LaSIE – Description of the Sheffield System Used for MUC-6. In [ARPA 1996].

[Goldfarb 1990] Goldfarb C.F. 1990. The SGML Handbook. Clarendon Press.

[Grishman 1995] Grishman R. 1995. TIPSTER architecture 1.50, The Tinman Architecture. Presentation 18/5/95, slides available at http://www.cs.nyu.edu/tipster.

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

[Jacobs ed. 1992] Jacobs P.S., ed. 1992. Intelligent Text-Based Systems, Current Researh and Practice in Information Extraction and Retrieval. Lawrence Erlbaum, Hillsdale NJ.

[Sperberg-McQueen, Burnard 1994] Sperberg-McQueen C.M., L. Burnard. 1994. Guidelines for Electronic Text Encoding and Interchange (TEI P3). ACH, ACL, ALLC.

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

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

[Wilks, Gaizauskas 1994] Wilks Y., R.G. Gaizauskas. 1994. A Research Proposal on Large-Scale Information Extraction. EPSRC proposal October.

[Wilks, Guthrie, Guthrie, Cowie 1992] Wilks Y., L. Guthrie, J. Guthrie, J. Cowie. 1992. Combining Weak Methods in Large-Scale Text Processing. In [Jacobs ed. 1992].

[Wilks, Guthrie, Slator 1996] Wilks Y., L. Guthrie, B. Slator. 1996. Electric Words. MIT Press, Cambridge, MA.