Supporting Text Mining for e-Science: the challenges for Grid-enabled Natural Language Processing

John Carroll
University of Sussex

Roger Evans
University of Brighton

Ewan Klein
University of Edinburgh

Abstract

Over the last few years, language technology has moved rapidly from 'applied research' to 'engineering', and from small-scale to large-scale engineering. Applications such as advanced text mining systems are feasible, but very resource-intensive, while research seeking to address the underlying language processing questions faces very real practical and methodological limitations. The e-Science vision, and the creation of the e-Science Grid, promises the level of integrated large-scale technological support required to sustain this important and successful new technology area. In this paper, we discuss the foundations for the deployment of text mining and other language technology on the Grid — the protocols and tools required to build distributed large-scale language technology systems, meeting the needs of users, application builders and researchers.

1 Introduction

The advent of large quantities of text in electronic form has transformed the field of Natural Language Processing (NLP) over the last fifteen years. Most notably, statistical modelling approaches have become very popular and significant progress has been made in areas such as part-of-speech tagging, word sense disambiguation and shallow parsing, with results which are being deployed in real applications such as information retrieval and text mining. Statistical NLP techniques require lots of data and processing power. Harder problems, and real-world applications, require even more data and more processing power. We are now at a point where text databases (corpora) of a billion words are routinely used for modelling, and the web, estimated to be in the region of 50–100 billion words [12] is being viewed as itself a single corpus. Managing and processing such corpora is a difficult and specialised task [18].

A similar situation faces the NLP application developer. It is now possible to apply results from large-scale NLP research in real applications (search engines, spam filters, mobile phones, statistical machine translation ...), and more advanced applications are coming within reach, such as sophisticated text mining, alerting services and question answering systems. But these often rely on huge data sets and complex system configurations beyond the scope of typical local computing resources. Effective delivery of such services therefore depends on giving potential users easy access to large-scale shared computing resources.

We can identify three largely distinct groups with an interest in well-founded NLP on the Grid. First are the e-Research 'end-users': scientists and social scientists who would like to invoke a language processing service as a subsidiary part of some larger task. A second group is the increasingly large body of engineers and service providers (such as the National Centre for Text Mining) who wish to deploy mature NLP technologies in high throughput / high availability applications, and seek to benefit from the redundancy and failover properties of distributed architectures. Finally, there are NLP researchers themselves who wish to experiment with different techniques and algorithms, perhaps on very large data sets, without the overhead of having to worry about inter-operability of different third-party components or how to frame an experiment so that it completes in hours rather than days or weeks.

It is clear that many of the core aspects of Grid computing directly benefit all these NLP client groups, providing management of and access to large data sets, access to high performance computing power, security and data transfer protocols, and practical Grid middleware. But there are also issues which we argue are more specific to the domain of NLP, and therefore require more focused specific attention by the NLP community. In this paper we discuss some of the key challenges which need to be addressed in order to fully support a vision of 'Grid-enabled' NLP. We distinguish between issues relating to the underlying technological requirements of NLP on the Grid, focusing on data representation, system configuration and deployment on the Grid, and the delivery of this technology to different user groups: end-users, service providers and NLP researchers.

2 Technology

2.1 Data representation

In common with many other fields, NLP has more or less adopted XML as a *lingua franca* for external data representation and communication. Thus we would expect to see XML as the interchange format on the Grid, employing tools such as those described by [7] for wrapping Unix executables as web services, and [16] for data conversion between components.

However, unlike many other fields, NLP employs XML not just for *data markup* but also for *document markup*. Technically the difference is small, but it is highly significant: in data markup, XML is used to represent just one thing, a data structure; in document markup, XML represents two things, a document plus data *annotations* over the document — the basic textual content has a status and existence independently of the annotations.

Additional considerations arise when XML is used in this way: a single document can be subject to successive layers of annotation, or to non tree-structured annotations, or to parallel, possibly conflicting annotations for different purposes. Within the XML/NLP community, techniques such as *stand-off* annotation [19, 11] have been developed to support distributed, compact, multiple annotations of documents. In stand-off annotation, annotation labels are associated with (sequences of) pointers to the original text, rather than separate copies of the text. Thus rather than marking a sentence by directly inserting, for example, `<sentence>`, `</sentence>` elements into the text, stand-off annotation might include a `<sentence start='302' end='387'/>` element in a separate file, indicating the presence of a sentence between characters 302 and 387 of the underlying text file.

Stand-off annotation is flexible, but not particularly efficient, especially for low-level mark-up such as part-of-speech tags. It presupposes a lowest level representation of 'the actual corpus' which may not be uncontroversial (character or word based for text, or digitised signal or segment based for speech), stable (character-based representations of the 'same' corpus may acquire different end-of-line markers) or robust (single missing or spurious characters can throw out the entire annotation). It also depends on an agreed namespace for relating corpus data and annotations. For effective use as a Grid interchange representation, these concerns and issues need to addressed.

Key issues

- Is stand-off annotation a suitable basis for representing language data annotation on the Grid?
- Can we make a specific proposal for an annotation model for Grid-enabled NLP applications, do we need more than one (eg for text vs speech), or do we need to allow for application-specific annotation models?
- How visible can/should the annotation model be to the user (or to different classes of user)?

2.2 Configuration and Composition of Components

A typical NLP system involves processing by multiple components, and the task of building such applications in a modular, easily configurable manner is a central challenge within the field [4]. In the context of the Grid the ability to specify, discover and configure components which are potentially widely distributed is key, and a suitably rich and flexible approach is clearly required. One NLP-oriented approach is the document-centred view proposed by [13], building on the framework of OWL-S[15]. This offers a semantically oriented perspective, allowing metadata associated with a service to be potentially much richer than the low-level syntactic interface expressed in WSDL, and supporting service matching which involves subsumption between service descriptions [17];

In parallel, numerous research groups have been developing so-called workflow languages and tools for e-Science applications (see [1] for a recent overview). Although much of this work is applicable to NLP scenarios [6, 7], there is as yet no emerging common framework for e-Science workflow within the community, nor a shared view on using rich metadata for service description.

The ability to save and reuse application configurations in the form of composed service workflows is already supported by tools such as Taverna.[1] For many NLP applications, the ability to

[1]`http://taverna.sourceforge.net`

build and save *application templates* for common use cases may also be important, and is as yet little supported. This would allow the creation of abstract workflows that only specify classes of components rather than specific instances (for example, a part-of-speech tagger or a word-sense disambiguator, without identifying a particular method), and the specification of global properties of such workflows, such as the use of a particular DTD as the schema for all data interchange.

At the invocation level, although the modularity and abstraction enforced by a service-oriented architecture is an essential part of any proposal, it is far from clear *what* should be encapsulated as a service. For example, the application for toponym detection described by [7] suggests that the factorization of complex pipelines into re-usable subsegments does not line up neatly with traditional component architectures. Moreover, adopting too fine a granularity of services is likely to degrade performance and hinder rapid prototyping.

Key issues

- What is an appropriate framework for describing the configuration and composition of NLP applications?
- What is an appropriate interface between NLP components and other parts of a larger system?
- How important is the ability to abstract over services and workflows, and work with saved templates, and how should it be achieved?
- What actual service components can we identify as being potentially useful, and what are the trade-offs between different decompositions?

2.3 Distribution and Optimisation

Application configuration addresses issues of task definition and resource and data set identification. However, beneath this level of abstract specification, the actual deployment of resources to implement an NLP application raises further issues. Many advanced NLP architectures are not pure pipelines. Thus recent generic proposals such as the GATE information extraction framework [5], the question answering architectures described by [8, 14] and the RAGS generation architecture [2, 3] all recognise the need to support non-pipelined configurations. However deploying such configurations on a Grid scale may lead to severe dataflow bottlenecks.

Much large scale NLP iterates a single NLP task many times over independent parts of a large data set. Throughput can be increased if it is possible to run multiple instances of the NLP task in parallel, and such a system can be further tuned by balancing parallelism between more and less CPU-intensive modules so that the processing capacity throughout the execution stream remains constant. In some cases, there is clear potential to view a data set as a collection of independent components, (for example, sentence-parsing over a collection of separate documents), and in such cases a data-centric approach might be optimal [9, 10]. In other cases, it is really the collection as a whole that is being processed, and the overhead of merging results computed on components independently may exceed the benefits of parallism.

These differences indicate a need for a wide range of system configurations, from single-processor systems with vast main memory capacity, sufficient to load a large text corpus into memory in order to process intensively across the whole structure, through tightly-coupled multi-processor systems for high-bandwith parallelism, for example blackboard architectures, to loosely-coupled systems for lower-bandwith and streamed pipeline systems. Deploying systems on the Grid without sensitivity to both the application and the Grid topology could easily result in disastrous inefficiency.

Key issues

- How should we describe the topology of NLP services? How should it reflect the topology of both the algorithm and the dataset?
- What range of provision exists on the Grid, and do we need to enhance what is available with Grid hardware resources specifically tuned to NLP requirements?
- How do we match application requirements to available Grid resources, and how important is the accuracy of the matching process?

3 Delivery

3.1 Supporting NLP application developers and users

NLP applications such as text mining or question answering typically process large numbers of documents and attempt to extract information from them. A typical system would use a number of NLP modules, for example with components for query analysis, selection of relevant documents from an indexed document collection, analysis of these documents to determine which parts of them may be relevant, extraction and selection of the most appropriate answer text, and response generation (see figure 1).

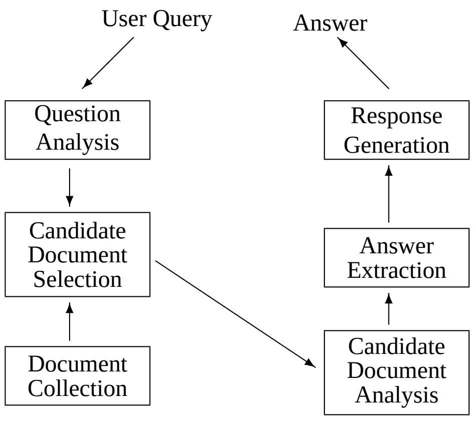

Figure 1: Architecture of a typical question-answering system

Grid-scale computing allows the data sets involved to be very large and the processing to involve applying a set of CPU-intensive modules to a large number of candidate answer documents. But the relatively generic nature and utility of such tasks means they have a potential user-base far beyond specialised scientific communities of many Grid-related applications. This in turn brings into play many more conventional application considerations, such as ease-of-use and real-time response. The challenge here is to deploy advanced Grid and NLP technology in a way that is completely hidden from the end user.

Key issues

- Can complex NLP Grid applications be delivered in a way that offers near Google-level ease of use and response times?
- Are there benefits from doing so, compared with a dedicated Google-like service? (For example, better access and interoperability with other Grid services, or the infeasibility of setting up a dedicated service.)
- If not, what would be acceptable ease of use and response time levels for (at least) typical e-Science users?

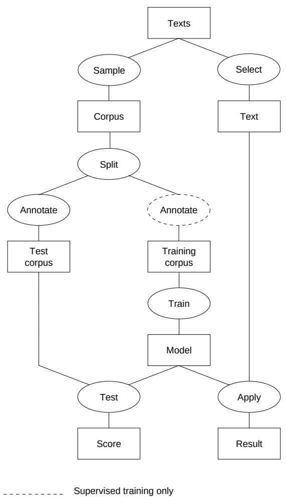

Figure 2: A generic empirical NLP system

3.2 Supporting empirical NLP research

The needs of empirical NLP researchers are somewhat different from users or application developers. While equally involved in large-scale computation, researchers are most interested in the ease of (re-)configuring modules, control over the turnaround of experiments (through automatic distribution and monitoring of computations), management of large quantities of different kinds of data, and replicability of results through precise recording and replay of experimental setups. In fact, large parts of the overall configuration of empirical systems are quite standard in design For example, a typical empirical NLP research system may include many of the following features (see figure 2): the use of several kinds of data resource (lexicons, statistical models, etc.); a clear-cut separation of input text data into training, held-out and test sets and use of these in similar ways but at different times; and when running experiments, the use of n-fold cross-validation and other training and testing regimes, and the need to compare results against a fixed 'gold standard' under various parametrisations of each module. Providing standard ways of defining NLP research system workflows and of expressing and packaging up standard procedures in template form, and providing pre-configured supporting services via the Grid, would result in more efficient and productive empirical studies.

Key issues

- Is there enough agreement about component functionality and experimental design to support the take-up of community-wide workflow templates?
- Can we obtain or develop the persistent stable datasets and resources required to support true replicability?
- Can we promote the take-up of this approach, among ourselves and our students, to achieve a step-change in the methodological rigour of empirical NLP research.

4 Concluding remarks

In this paper we have sought to begin exploration of the question of whether there are particular characteristics of Natural Language Processing which require specific kinds of support for deployment on the Grid. In doing so we are not trying to claim some special status for NLP over other e-Science disciplines — it seems likely to us that many other research fields will engage in similar discussions, and this is just a natural second stage beyond the generic Grid provision now in place. But equally we do conclude that NLP does have requirements of its own, that are not directly addressed by Grid infrastructure, and may not be shared with other discplines. If that is correct, then it is clearly the responsibility of our own community to address these requirements, to ensure that maximum benefit of Grid provision for NLP and the maximum benefit of the application of NLP to the e-Science community and beyond.

References

[1] Dave Berry and Savas Parastatidis, editors. *e-Science Workflow Services*, Edinburgh, December 2003. e-Science Institute.

[2] Lynee Cahill, Roger Evans, Chris Mellish, Daniel Paiva, Mike Reape, and Donia Scott. The RAGS Reference Manual. ITRI, University of Brighton, 2001.

[3] Lynne Cahill, John Carroll, Roger Evans, Daniel Paiva, Richard Power, Donia Scott, and Kees van Deemter. From RAGS to RICHES: exploiting the potential of a flexible generation architecture. In *Proceedings of ACL/EACL 2001*, pages 98–105, Toulouse, France, 2001.

[4] Hamish Cunningham, Kalina Bontcheva, Valentin Tablan, and Yorick Wilks. Software Infrastructure for Language Resources: a Taxonomy of Previous Work and a Requirements Analysis. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC-2)*, Athens, 2000.

[5] Hamish Cunningham, Diana Maynard, Kalina Bontcheva, and Valentin Tablan. GATE: A framework and graphical development environment for robust NLP tools and applications. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, 2002.

[6] Rob Gaizauskas, Neil Davis, George Demetriou, Yikun Guod, and Ian Roberts. Integrating biomedical text mining servies into a distributed workflow environment. In *Proceedings of UK e-Science All Hands Meeting*, Nottingham, 2004.

[7] Claire Grover, Harry Halpin, Ewan Klein, Jochen L. Leidner, Stephen Potter, Sebastian Riedel, Sally Scrutchin, and Richard Tobin. A framework for text mining services. In *Proceedings of the Third UK e-Science Programme All Hands Meeting (AHM 2004)*, 2004.

[8] Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *Journal of Natural Language Engineering*, 7(4):275–300, 2001. forthcoming.

[9] Baden Hughes and Steven Bird. Grid-enabling natural language engineerin by stealth. In *Proceedings of the NAACL/HLT Workshop on Software Engineering and Architecture of Language Technology Systems*, Edmonton, Alberta, May 2003.

[10] Baden Hughes, Steven Bird, Kim Haejoong, and Ewan Klein. Experiments with data-intensive NLP on a computational grid. Preprint 503, University of Melbourne, May 2004.

[11] Nancy Ide. The XML framework and its implications for corpus access and use. In *Proceedings of Data Architectures and Software Support for Large Corpora*, pages 28–32, Paris, 2000. European Language Resources Association.

[12] Frank Keller, Maria Lapata, and Olga Ourioupina. Using the web to overcome data sparseness. In Jan Hajič and Yuji Matsumoto, editors, *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 230–237, Philadelphia, 2002.

[13] Ewan Klein and Stephen Potter. An ontology for NLP services. In Thierry Declerck, editor, *Proceedings of Workshop on a Registry of Linguistic Data Categories within an Integrated Language Reso urce Repository Area, LREC 2004*, May 2004.

[14] Jochen L. Leidner, Johan Bos, Tiphaine Dalmas, James R. Curran, Stephen Clark, Colin J. Bannard, Mark Steedman, and Bonnie Webber. The QED open-domain answer retrieval system for TREC 2003. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, NIST Special Publication 500-255, pages 595–599, Gaithersburg, MD, 2004.

[15] OWL Services Coalition. OWL-S: Semantic markup for web services, 2003.

[16] Sally Scrutchin. Data transformations for the NLP grid. MSc Thesis, School of Informatics, University of Edinburgh, September 2004.

[17] Evren Sirin, James Hendler, and Bijan Parsia. Semi-automatic composition of web services using semantic descriptions. In *International Workshop on Web Services: Modeling, Architecture and Infrastructure*, Angers, France, 2003.

[18] Fabio Tamburini. Building distributed language resources by grid computing. In *Proceedings of the 4th International Language Resources and Evaluation Conference (LREC 2004)*, pages 1217–1220, Lisbon, 2004. European Langugage Resources Association.

[19] Henry S. Thompson and David McKelvie. Hyperlink semantics for standoff markup of read-only documents. In *Proceedings of SGML Europe '97*, Barcelona, May 1997.