A Framework for Text Mining Services

A Framework for Text Mining Services

Claire Grover Harry Halpin Ewan Klein[†] Jochen L. Leidner
Stephen Potter Sebastian Riedel Sally Scrutchin
Richard Tobin

School of Informatics, University of Edinburgh
[†]*and National e-Science Centre, Edinburgh*

Abstract

The growth of online scientific literature, coupled with the growing maturity of text processing technology, has boosted the importance of text mining as a potentially crucial tool. However, there are several challenges to be addressed before sophisticated text mining services can be deployed within emerging workflow environments. Our work contributes at two levels. At the invocation level, we have developed a flexible XML-based pipeline architecture which allows non-XML processors to be readily integrated. At the description/discovery level, we have developed a broker for service composition, and an accompanying domain ontology, that leverage the OWL-S approach to service profiles.

1 Introduction

There is increasing interest in deploying text mining tools to assist researchers in various tasks. For example, a social historian may wish to query a gazetteer with location names retrieved from a collection of transcribed interviews in order to obtain parish codes; these can then be used in analysing aggregate regional statistics for the period. An astronomer who detects an X-ray signal in some region of the sky may want to investigate the online literature to see if any interesting infra-red signals were detected in the same region. And a biologist who has been given a list of 100 genes detected in a microarray experiment may wish to trawl through the existing published research for papers which throw light on the function of these genes.

In such scenarios, we envisage the researcher using a workflow editor to incorporate one or more text mining services within a larger application. The service might be essentially a document classification tool which retrieves documents in response to a string of keywords. However, it might be more elaborate, such as an information extraction system which populates a database that is subsequently queried by some further service.

Typically, a text mining system that is deployed as a service will have been assembled out of modules, each of which carries out a specific natural language processing (NLP) task. For example, an information extraction system might include modules for tokenization, part-of-speech tagging, named entity recognition, and relation detection. The *scientist end-user* will not care about the details of these modules, and we might assume therefore that the service is presented as a black box. Nevertheless, there are a number of reasons why we might care about the internal architecture.

First, the external surface of a service might change according to the application: in one case, the output of the relation detector would be directed to a database service or a presentation layer; in a second case, the relation detector would be discarded, and the output of the named entity recognizer would be exposed as the external interface. For the sake of generality, therefore, we do not want to impose too rigid a grain-size on the components that can be offered as services.

Second, a *provider* of text mining services would want the ability to easily reconfigure systems for both testing and deployment purposes. In general, new application domains will require statistically based tools to be retrained, and rule-based tools to be provided with new rule sets; these will both potentially affect the overall system configuration. More importantly, ongoing rapid progress in the development of data-intensive text processing tools has the consequence that the top-performing systems will usually be heterogeneous, in the sense of incorpo-

rating processing modules developed by a variety of research groups. This argues strongly in favour of an underlying architecture which supports loosely-coupled, interchangeable processors. This in turn raises the possibility that a text mining system which is presented as a black box to the scientist end-user might in fact be decomposable for the computational linguist as a workflow of independent, reusable services—in other words, we might want to offer the computational linguist a workflow environment that allows her to rapidly prototype and test applications built from tools exposed as web services. In such a case, we could interoperate with the end user by providing a pre-packaged workflow that could be included within a larger application.

In order to explain our perspective on text mining services, we will start (Section 2) by describing our underlying XML-oriented approach to shallow text processing. In order to illustrate the kind of utility that our text processing tools support, we will focus on a task that arose in from collaborative work with EDINA[1] on the the JISC-funded **geoXwalk** project,[2] namely identifying placenames in text documents. We will then briefly describe how we have put together a series of text processing steps into a service-oriented workflow that contributes to the overall application. In Section 3, we turn to the question of how to use semantic characterizations of text processing services in order to help the user discover and compose the services into useful applications.

2 LT-XML Language Processing

2.1 Overview

The LT-XML [12] architecture for text processing was designed to support loosely coupled processors of the kind just alluded to, but using command line programs communicating via Unix pipes. In order to be pipelined in this way, all processors must share a common data format, and the choice of XML allows us to accommodate a variety of models of linguistic annotation. Following standard Unix philosophy, individual processors each carry out very specific tasks. However, the fact that they all stream their input and output in XML allows them to be freely combined, and different combinations of the same suite of processors can carry out a range of complex text processing tasks. Example applications developed over recent years include identifying gene and protein names in Medline abstracts [7, 8, 6], tokenizing technical terms in Medline abstracts as a prelude to wide-coverage parsing [11, 10], extracting information about laptop computer specifications from web pages [13], identifying geographical locations in Scottish historical documents [22], and high-compression summarization of legal documents [9, 15].

To date, all NLP applications built within LT-XML have involved adding annotation to plain or HTML-formatted textual input. With relatively minimal assumptions about the appropriate data model for linguistic annotation, it is possible to adopt a simple processing model: each processor in a pipeline (except the very first) inspects the markup produced by the preceding processor, and modifies this or adds new markup of its own. Although the markup process need not be strictly monotonic,[3] in practice it almost always is. Figure 1 illustrates this point: in the topmost tree, the text has been annotated to show word (‘<w>’) and sentence (‘<s>’) units, while the bottom tree has additional annotation which marks part-of-speech information on words, and adds a named entity element (‘<ne>’) of type *location*.

A base set of LT-XML text processors has been built using `lxtransduce`, a general purpose deterministic transducer which rewrites a stream of characters or XML elements according to a set of rules. The input is matched with regular expressions or XPath patterns. Character input can be arbitrarily modified and marked up with XML elements. XML input can be re-ordered and marked up with additional elements and attributes. Input can be matched against lexicons and the lexical information incorporated in the output. With suitable rule sets, `lxtransduce` allows us to construct tokenizers, named entity recognizers and partial parsers.

However, as argued earlier, a high priority is to allow heterogeneous systems, incorporating third-party— typically non-XML—processors into an XML-based pipeline. This requires us to use format converters (or ‘shims’ following [24]) to map out of, and back into, XML. Most of our legacy code addresses this need with *ad hoc* scripts written in `xmlperl`,[4] a powerful and flexible tool for rapid prototyping. However, we are also experimenting with XSLT stylesheets for conversion out of XML, and with Yacc/Bison for the converssion into XML.

The creation of reliable shims is important since the performance of most text processing applications is usually limited by the performance of the weakest component. However, most NLP frameworks (e.g., GATE [3], SDL [18] and to a certain extent LT-XML)

[1] `http://edina.ac.uk/`

[2] `http://www.geoxwalk.ac.uk/`

[3] By ‘monotonic’, we mean that annotations can be added but not removed.

[4] `http://www.cogsci.ed.ac.uk/~dmck/xmlstuff/`

limit the use of processors available in an application to those that conform to their internal data interchange format. Since state-of-the-art tools often use non-XML formats, shims must be created for those tools so that they can be integrated and thus obtain the optimal performance for the system as a whole.

However, when modularizing a text processor and creating its shim we do not want to limit its use to the XML data interchange format of a particular framework. By publishing the XML format used by the output and input of a processor (e.g. making its XML Schema or DTD publicly available), it should often be possible to create an XSLT stylesheet to translate between the XML data interchange formats used by the various frameworks and processors. If processors are wrapped using one of the popular XML data interchange formats, then only a few standard transformations from one common data interchange format to another may need to be created. By using the method of schema inclusion, only the relevant parts of the XML data interchange formats need to translated. This approach lets any NLP processor be used in any text processing workflow, provided that mappings can be created between the relevant parts of the data formats. For our particular pipeline we used the LT-XML data interchange format, and mappings can be created from that format to parts of other formats such as NITE [2].

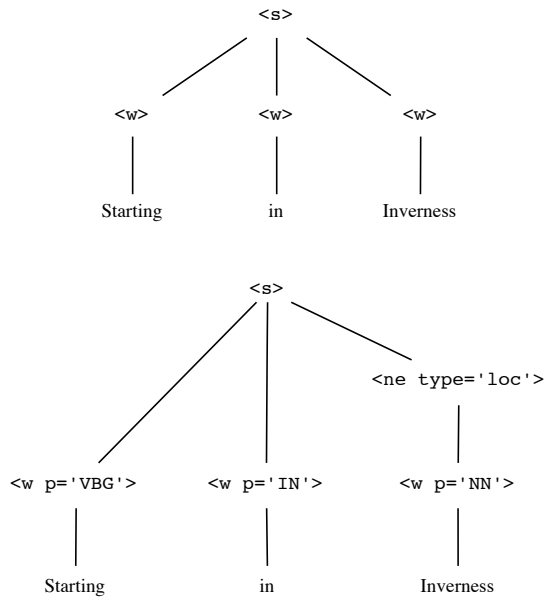

Figure 1: Incremental XML Annotation

2.2 GeoParsing

We will illustrate a typical usage of our tools by focussing on *geoparsing*—the task of automatically identifying location names in documents. This needs to be seen within the broader context of working with spatial data.

A wide range of socio-economic data, for example on population size, health, crime, and unemployment. is regularly collected and published by organizations and governments. Such datasets relate their primary variables to spatial information in various ways [20]; in particular, they use different *geographies*, ranging from explicit grid references or parish codes to implicit references such as postal addresses or association with a placename. In order to support the process of relating data to a representation of its spatial scope (sometimes called *geo-referencing*, services such as *geoXwalk* [5] have been developed to translate between these different geographies.

Mapping across different systems of spatial reference allows for the fusion of different datasets, which in turn is a precondition for sophisticated mapping. For example, a social scientist studying the relationship between crime and deprivation in Scotland might want to draw a *thematic map* showing the strength of the spatial correlation between crimes occuring in a given place and standardized scores of deprivation in the same area [21]; the former might be based on address geo-referenced police incidence records and the latter on UK Census boundaries.

The particular geoparsing system we describe here is intended to take HTML documents as input, and to output documents with the same HTML markup, but supplemented with `<span>` tags which visually highlight the placenames in the document. Figure 2 gives a browser view of a document after location names have been marked up. In recent experiments that we have carried out, running the geoparser (which incorporates an off-the-shelf maximum entropy classifier [4]) over a test set drawn from the Statistical Accounts of Scotland has achieved excellent results, with an f-score (combined precision and recall) over 94%; see [22] for more details.

In addition, the system uses the location names extracted from the source text to query an online gazetteer service.[5] For example, a placename query for *Sudbury* will retrieve an XML record which contains an element `standard-reports` that is made up of a sequence of `gazetteer-standard-report` elements, one for each gazetteer entry. Each of these in turn contains a `bounding-box` element that specifies the location's bounding coordinates. There is also an element to describe the location sub-type:

[5]The service is hosted by EDINA at `http://dearg.ucs.ed.ac.uk:9101/adlcrosswalk/`.

Walking Pages - County Pages

Suffolk

There is something very English about Suffolk. Here you will find the quintessential English village. Along the river Stour south of Lavenham is 'Constable Country', home of England's most famous landscape painter and the subject of his glorious paintings which so poignantly captured rural English life. The Stour flows peacefully through a chalky valley with willow and poplar lining lush water-meadows, and walks between Sudbury and Flatford capture the essential 'Englishness' of the scenery. Flatford is, of course, the location for one of Constable's most famous paintings and is now a major tourist attraction. Upstream at Sudbury is the birthplace of another great English painter, Thomas Gainsborough. The long distance route The Stour Valley Path follows the river all the way from from Newmarket to Manningtree.

Figure 2: Web page marked up with location name

e.g., there are *Sudbury* entries variously annotated as 'parish admin', 'ward', and 'village or hamlet'. The module which sends queries to the gazetteer uses the query results to create a new XML document containing a list of `virtual_locations` found in the input document: 'virtual' because the same name is often shared by more than one place; for example, there exist around 30 places called 'Aberdeen' on earth, and England alone has two 'Leeds'. We are currently working on methods to resolve these ambiguities automatically [19]. The results return by the query module are illustrated in Figure 3.

```
<virtual_location>
  <occurrence>
   <text>Sudbury</text>
   <index>49</index>
  </occurrence>
  <occurrence>
   <text>Sudbury</text>
   <index>65</index>
  </occurrence>
  <standard-reports>
    ...
  </standard-reports>
</virtual_location>
```

Figure 3: A list of virtual locations

That is, a `virtual_location` lists the tokens which have been marked as placenames in the text, together with the token identifiers, and then gives the `standard-reports` element from the output of the gazetteer query. The information in this XML file can be presented in an interactive window which allows the user to disambiguate the intended referent of the placename, to select places of particular interest, ask for more information about them from other sources, and request a map which displays the selected places.

2.3 Service Architecture

The initial implementation of the geoparser described above ran as a single Unix script on a single system. However, by making its major modules available as web services, the entire system can become distributed across multiple platforms, and components can be reused and interchanged.

Nevertheless, the foundation of many text processing components as pipelines has ramifications for their development as web services. Many web services are currently based on well-defined typed arguments, typically with the service itself built over a strongly-typed programming language such as Java. In contrast, text processing web services are almost always document-centred, exchanging and passing linguistic information as XML input and output documents embedded within SOAP messages. While these documents may be validated by a schema, the crucial information needed by the web service is embedded within a single XML input document itself, not as multiple strongly-typed parameters. Also, a single XML document is usually returned as the output. To facilitate this document-centric approach to web service workflows, we created our own web service wrappers and servers.

The first stage in transforming the geoparser into a service oriented application involves identifying the major modules within the geoparser, so that each module can be turned into an independent web service and composed into a workflow. The level of granularity is necessarily open to discussion, but we have been guided by what we take to be the 'minimal units of reusability'. The resulting workflow involves four major steps:

Tokenization: Identifying and marking up words and sentences in the input text.

Location Tagging with a classifier: Using a trained maximum entropy classifier to mark up location names.

Location Tagging by Lexicon: Using a lexicon of location names to mark up additional locations not identifier by the tagger.

Gazetteer Query: Sending location names extracted from the text to a gazetteer resource, and presenting the query results in an application-appropriate form.

In addition, the worklow is initiated by a step that converts HTML markup into XML, and terminated by a step that re-converts the XML back into HTML.

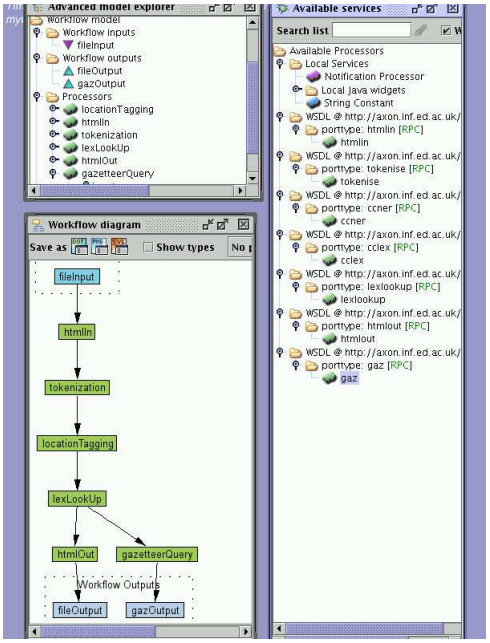

Figure 4: View of Location Tagging Workflow

Each of these steps is implemented as shell script on Linux, and consists of a pipeline of command-line programs. The scripts are then wrapped as web service using a utility called `lxsoap`. A simple configuration file acts as the input to an XSLT transformation that creates a WSDL file describing the service; it also allows `lxsoap` to map between SOAP messages and the program's input, output and command-line arguments. We have successfully reconstructed the overall tool pipeline as a workflow by calling these services from the Taverna Workbench.[6]

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

[7] For more details, see [17].

3 Ontology-based Service Discovery

Although web services can be composed into a workflow by simply selecting them from a list, as allowed by the Taverna Workbench, it would be desirable to use a more semantically-oriented approach to composing services. To move towards a semantic service environment, over the last couple of years the OWL-S upper ontology for describing web services has been developed. The intention of this effort is to provide an XML-based ontology which stipulates the basic information that services should expose to the environment in order to facilitate their automatic discovery, invocation, composition and monitoring [23]. This ontology is specified in the OWL Web Ontology Language, which provides a language for specifying Description Logic constructs in the syntax of XML and is built on top of the RDF data model. For the purposes of discovery, the most important aspect of the OWL-S ontology is the **Service Profile**. This describes the essential capability of the service by characterizing it in functional terms: the inputs a service expects, the outputs it produces, the preconditions that are placed on the service and the effects that the service has. By 'typing' data in this fashion, the ability to define and instantiate 'semantic pipelines' of data through a workflow consisting of a number of distinct services also becomes possible.

The OWL-S ontology is domain-independent: to express concepts of particular domains it is necessary to extend the OWL-S ontology into these domains through the use of additional ontologies. We have developed such an ontology in the domain of NLP (specifically, by extending the OWL-S notion of a **Profile Hierarchy**).[7] Our approach rests on the following assumptions:

1. NLP processors have documents as both input and output.
2. Documents have properties which impose preconditions on the processors and which also record the effects of processing.
3. A specification of the properties of documents, as input/output parameters, induces a classification of NLP processors.

Figure 5 illustrates the case of a part-of-speech tagger, defined so that its input document is marked-up for word and sentence tokens, while its output is a document which is in addition marked-up for parts of speech.

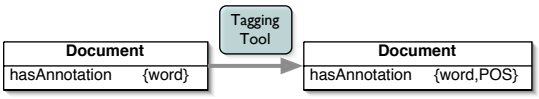

Figure 5: The Tagger class as a function

We can subclass Document class in various ways in order to further refine different classes of processors, as illustrated in in Figure 6.

In order to be able to reason about NLP services, we have developed a broker service, built on top of the RACER [14] Description Logic engine. The broker maintains a description of the NLP ontology; when it receives service advertisements, which have been described using OWL-S and the domain ontology, it classifies and stores them as instances of the appropriate class in the hierarchy. On receiving an OWL-S query, it composes a class description from this and then returns, as potential solutions, any service instances of classes subsumed by this description.

On the client side, we have developed a prototype composition tool for composing sequences of services and querying the broker. The user is able to specify either the type of processing resource that is needed, or the constraints on the data inputs and outputs to some abstract service (or a combination of both); and the tool constructs the appropriate OWL-S, sends this to the broker and then presents the alternative services to the user. The selection menu offered to the user is shown in Figure 7. Once a user selects one of these, the tool fetches the URL of the service to extract more detailed information about the service, and the composition view is updated accordingly.

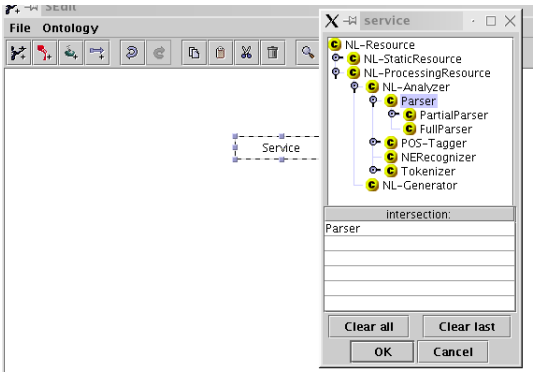

Figure 7: Selecting a Text Mining Tool from the class hierarchy

The composition tool is able to output its configuration file in Scufl format [1], thus enabling the workflow to be saved, and to be enacted within Taverna.

The text processing web services that we described in Section 2 can be classified within the framework of our NLP Profile Hierarchy, and should allow services such as the geoparser to be composed by semantically-based brokering, although the exact details of such a composition are still under investigation.

4 Conclusion

In this paper, we have addressed two potential user populations: the scientist who is a potential consumer of text mining applications but is not interested in the internal make-up of the applications, and the computational linguist who wishes to construct text mining applications from a combination of in-house and third-party tools. We believe that both groups can benefit from 'virtualizing' language processing tools as web services.

One of the key challenges in building on this paradigm is to present the processors in a way that supports robust interoperability. This can only be achieved if the input and output types of processors can be described using a rich vocabulary. We have argued here for an ontology that leverages properties of the class of documents that can be exchanged between the processors; although this does not exhaust all the constraints on interoperability, it does provide a promising basis for capturing the functional core, especially in terms of the data transformations carried out by NLP processors.

Given the huge amount of linguistic information available online, from natural language annotations in databases to the text available on the Web itself, robust and scalable NLP applications are urgently needed for text-mining. By wrapping NLP components as web services, cutting-edge technology can be made readily available for use for real-world problems. In this paper, as a proof-of concept, we showed how a real-world application could be constructed within a standard web service workflow environment.

There are a number of issues for future research. For very large documents, the network latency involved in web services could lead to possibly slower results than a single central installation. Furthermore, many text processing components have a heavy statistical component which can make their processing time over large documents slow as well. With the further development of the Grid and its integration with web services, the problems of limited bandwidth and slow processing time should be compensated for. For example, a statistical tagger that invokes high computational costs could have its workload shortened by sending itself as a job to the

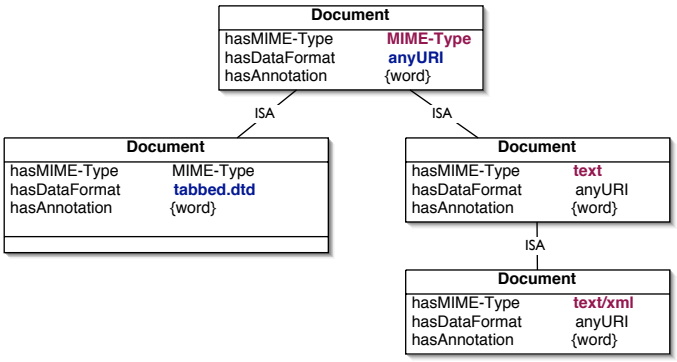

Figure 6: Subclasses of Document

Grid, and so play a role in a workflow without serving as a bottleneck on performance time. There is already evidence that for some NLP tasks the use of the Grid as opposed to a single system can improve performance time over very large data sets, and the results improve as the data set gets larger.[16] In this regard, the future development of web services and e-science is likely to play an increasingly important role in the development of large-scale, rapid, high-performance text-mining.

References

[1] XScufl language reference, 2004. URL `http://taverna.sourceforge.net/docs/xscuflspecificion.html`.

[2] J. Carletta, S. Evert, U. Heid, J. Kilgour, J. Robertson, and H. Voorman. The NITE XML Toolkit: flexible annotation for multimodal language data. *Behavior Research Methods, Instruments, and Computers*, 35: 353–363, 2003.

[3] Hamish Cunningham. GATE, a general architecture for text engineering. *Computers and the Humanities*, 36:223–254, 2002.

[4] James Curran and Stephen Clark. Language independent NER using a maximum entropy tagger. In Walter Daelemans and Miles Osborne, editors, *Seventh Conference on Natural Language Learning (CoNLL-03)*, pages 164–167, Edmonton, Alberta, Canada, 2003. Association for Computational Linguistics. In association with HLT-NAACL 2003.

[5] Ian Densham and James Reid. System demo: A geo-coding service encompassing a geo-parsing tool and integrated digital gazetteer service. In András Kornai and Beth Sundheim, editors, *HLT-NAACL 2003 Workshop: Analysis of Geographic References*, pages 79–80, Edmonton, Alberta, Canada, 2003. Association for Computational Linguistics.

[6] Shipra Dingare, Jenny Finkel, Malvina Nissim, Christopher Manning, and Claire Grover. A system for identifying named entities in biomedical text: How results from two evaluations reflect on both the system and the evaluations. In *The 2004 BioLink meeting: Linking Literature, Information and Knowledge for Biology at ISMB 2004*, Glasgow, 2004.

[7] Jenny Finkel, Shipra Dingare, Christopher Manning, Malvina Nissim, Beatrice Alex, and Claire Grover. Recognizing genes and proteins in MEDLINE abstracts. In *Proceedings of BioCreAtIvE 2004 (to appear)*, Granada, Spain, March 2004.

[8] Jenny Finkel, Shipra Dingare, Huy Nguyen, Malvina Nissim, and Chris Manning. From syntax to the web. In *Proceedings of the International Joint Workshop on Natural Language Processing in Biomedicine and its Applications at CoLing 2004*, Geneva, Switzerland, August 2004.

[9] Claire Grover, Ben Hachey, Ian Hughson, and Chris Korycinski. Automatic summarisation of legal documents. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law (ICAIL 2003)*, 2003.

[10] Claire Grover, Ewan Klein, Maria Lapata, and Alex Lascarides. NLP tools for processing and interpreting biomedical language. In *The 2nd Workshop on NLP and XML (NLPXML-2002), COLING 2002*, Taipei, September 2002.

[11] Claire Grover and Alex Lascarides. XML-based data preparation for robust deep parsing. In *Proceedings of the Joint EACL-ACL Meeting (ACL-EACL 2001)*, 2001.

[12] Claire Grover, Colin Matheson, Andrei Mikheev, and Marc Moens. LT TTT – A flexible tokenisation tool. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation (LREC 2000)*, 2000.

[13] Claire Grover, Scott McDonald, Donnla Nic Gearailt, Vangelis Karkaletsis, Dimitra Farmakiotou, Georgios Samaritakis, Georgios Petasis, Maria Teresa Pazienza, Michele Vindigni, Frantz Vichot, and Francis Wolinski. Multilingual XML-based named entity recognition for E-retail domains. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC 2002)*, 2002.

[14] V. Haarslev and R. Moller. RACER system description. In *Proceedings of the First International Joint Conference on Automated Reasoning*, pages 701–706. Springer-Verlag, London UK, 2001.

[15] Ben Hachey, Claire Grover, Vangelis Karkaletsis, Alexandros Valarakos, Maria Teresa Pazienza, Michele Vindigni, Emmanuel Cartier, and José Coch. Use of ontologies for cross-lingual information management in the web. In *Proceedings of the International Workshop on Ontologies and Information Extraction*, 2003.

[16] 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. URL `http://eprints.unimelb.edu.au/archive/00000503/`.

[17] 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 Resource Repository Area, LREC 2004*, May 2004.

[18] Hans-Ulrich Krieger. SDL—A description language for building NLP systems. In Hamish Cunningham and Jon Patrick, editors, *HLT-NAACL 2003 Workshop: Software Engineering and Architecture of Language Technology Systems (SEALTS)*, pages 83–90, Edmonton, Alberta, Canada, May 2003. Association for Computational Linguistics.

[19] Jochen L. Leidner, Gail Sinclair, and Bonnie Webber. Grounding spatial named entities for information extraction and question answering. In *Proceedings of the Workshop on the Analysis of Geographic References held at HLT/NAACL'03)*, pages 31–38, Edmonton, Alberta, Canada, May 2003.

[20] D. J. Martin. Spatial representation: the social scientist's perspective. In Paul A. Longley, Michael F. Goodchild, David J. Maguire, and David W. Rhind, editors, *Geographical Information Systems*, volume 1, pages 71–80. Wiley, New York, NY, 2nd edition, 1999.

[21] Philip McLoone. Carstairs Scores for Scottish postcode sectors from the 1991 Census, 2000. data set.

[22] Malvina Nissim, Colin Matheson, and James Reid. Recognising geographical entities in scottish historical documents. In *Proceedings of the Workshop on Geographic Information Retrieval held at the 27th Annual International ACM SIGIR Conference, Sheffield, UK, 29 July 2004*, 2004.

[23] OWL-S. OWL-S: Semantic markup for web services. `http://www.daml.org/services/owl-s/1.0/`, 2003.

[24] Chris Wroe. Semantic mediation in myGrid. In *Workshop on eScience GRID Environments*, Edinburgh, May 2004. e-Science Institute. URL `www.nesc.ac.uk/talks/292/sms/wroe.pp`.