# Distributed Natural Language Processing
## *A Web Services Framework for Grid-scale NLP Applications*

*Harry Halpin*

h.halpin@ed.ac.uk

## Introduction

NLP applications currently exist that have a high enough performance that they can be used in everyday applications. The World Wide Web is the largest body of digital text in existence. Little work has been done on integrating existing NLP applications into a Web-based framework for use over and with the Web. We present a a way forward using a combination of Web Services, workflows, and Semantic Web technologies.

## Problems with NLP

While the field of Natural Language Processing (NLP) has existed for many years, the resultant applications are difficult to deploy and even harder to compare. NLP tools use a wide diversity of programming languages over many platforms, and so the average natural language engineer is a "desperate Perl hacker," spending excessive time on data format conversions and installing applications. To combat this, there are loosely-coupled pipeline-based toolkits such as **LT TTT** and integrated frameworks such as **GATE**. However, pipelines require restriction to a common format, limiting the amount of easily usable components, and lack control structures. Frameworks provide control structures but severely restrict the amount of components, leading to a high learning curve and often sub-optimal performance. Both require a local installation and put the burden of updating the components on the user (Leidner 2003).

## Web Services

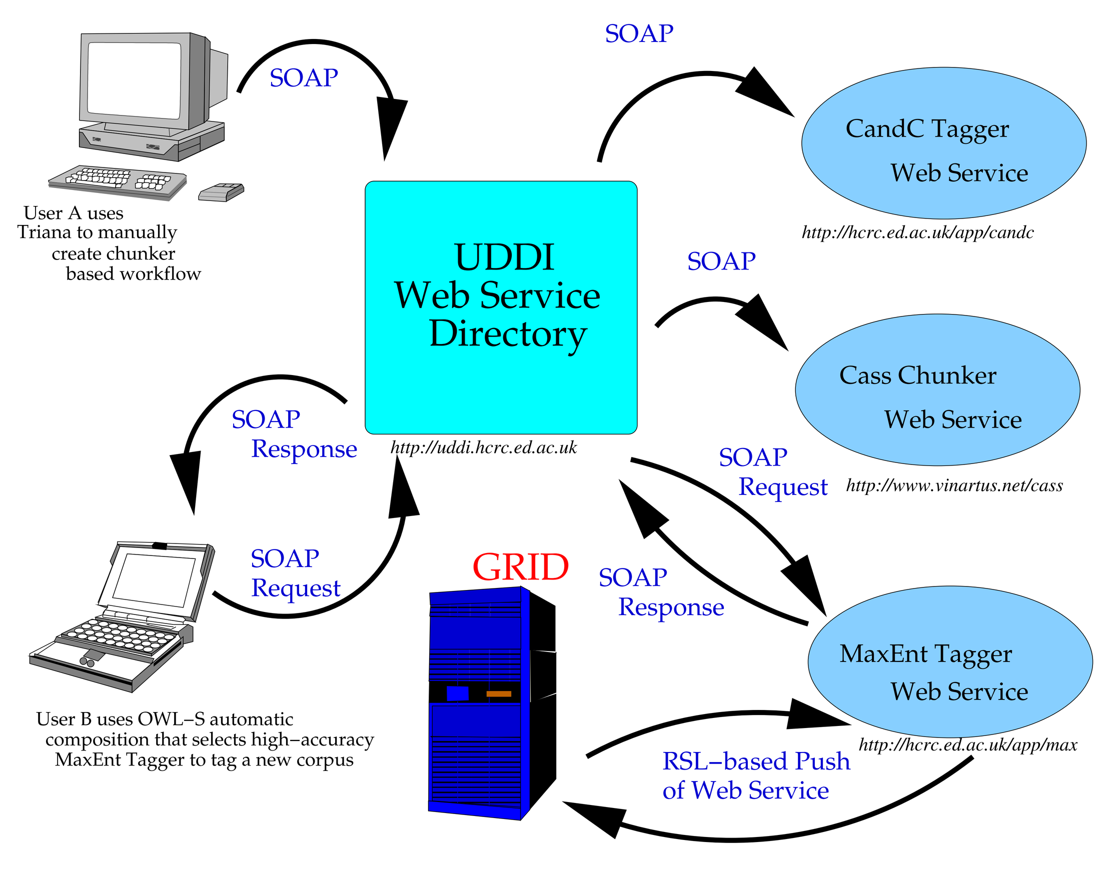
Web Services provide a solution to these problems by leveraging the power of the Web to provide applications as services, in contrast to the traditional use of the Web to provide information. Web Services are self-describing and self-contained modules that can be located and discovered on the Web. They use HTTP as a protocol to send data over the Web and through firewalls. For example, in a standard NLP application one might have a tokenizer, the tagger, and the chunker; these would all be separate Web Service components. Web Services communicate via the XML-based **Simple Object Access Protocol (SOAP)**. This allows any XML document, such as an XHTML web-page or a standard XML-formatted corpus, to be used encapsulated along with metadata.

## Finding Web Services

The Web Services themselves are described via **Web Services Definition Language (WSDL)**, which binds them to a URI and allows components to specify input and output parameters. For example, the Cass Chunker would want to specify its input as POS tagged words and as its output as chunks. WSDLs can be published in a Yellow-Pages directory called **Universal Description Discovery and Integration (UDDI)**. NLP researchers can wrap existing applications as Web Services with a WSDL, allowing access to the them at a UDDI such as a hypothetical *http://uddi.hcrc.ed.ac.uk*, which in turn contains pointers to possible local applications such as the CandC tagger or ones further afield such as the Cass Chunker.

## Using the Grid

Web Services have an *install-once* and use *anywhere with Web access* approach. They do not run on the local computer but usually on the host of the URI of the Web Service. This allows integration with **the Grid**, the next generation Internet that allows distributed computing resources to be used for processing power and storage. As statistical approaches to NLP and large-scale ensembles become increasingly popular and corpora become increasingly large, access to processing and storage far beyond the capability of personal computers is needed. While there is additional overhead caused using Web Services (such as extra XML wrapping and network latency), by interfacing with the Grid the performance of many statistical NLP applications could be faster as Web Services than running them locally. Web Services can rely on toolkits such as **Globus** to interface with the Grid. Globus uses the **Resource Specification Language (RSL)** to "push" resource-intensive NLP applications to the Grid.




*Sample Grid-enabled NLP Web Service Architecture*

## Service Composition

A user can manually build an NLP application by using a graphical Web Services composition toolkit such as **Triana**. This allows a user to create a **workflow** that connects a series of modular Web Services into a complete application. The workflow has the input and output of every component validated. Also, a user may want an NLP application that relies on multiple components, but not care about which particular components are used. The **OWL-S** description logic can abstractly describe NLP Web Services (Klein and Potter 2004). These descriptions can be specified by both the user and the Web Service itself, allowing automatic composition of NLP applications.

## Performance Metadata

A user may care about certain performance issues, such as whether the application can perform above a certain recall. This type of information can be stored as **performance metadata** in the SOAP messages and final processed documents. Since the choice between various compositions of NLP applications is combinatory, performance metadata combined with automatic composition allows rigorous comparison of NLP applications, parameter optimization, and performance-directed NLP application creation.

## Current Work

We are currently enabling common NLP components to be both XML and Web Services based. We are also researching the Web Service and Grid interface. Check *www.gridnlp.org* for updates.

## References

Klein, Ewan and Potter, Stephen (2004). An Ontology for NLP Services. Workshop on a Registry of Linguistic Data Categories within an Integrated Language Resource Repository Area.

Leidner, Jochen (2003). Current Issues in Software Engineering for Natural Language Processing. Workshop on Software Engineering and Architecture of Language Technology Systems held at HLT/NAACL'03.

School of informatics