Aberystwyth University

Progress Report

Partridge: An Intelligent Literature Analysis and Recommendation Suite.

Author:
James Ravenscroft
jrr9@aber.ac.uk
090407039

Supervisors
Amanda Clare
Maria Liakata

Status: Draft
Revision: 5b2fed3
November 16, 2012

Contents

1 Project Summary

This document describes the creation of Partridge, a new web-based tool designed to assist in information processing and knowledge acquisition within the domain of scientific research.

Since the advent of the 'Digital Age' and the ability of computers to copy and reproduce information for a negligible cost, the amount of information available to researchers has been increasing drastically. B-C Björk (2009) estimates that approximately 1.4 Million papers were published in the year 2006 alone[6]. Moreover, the growing popularity of Open Access publishing (making papers available to readers for free online[27]) across most scientific disciplines[6][12] is providing scientists and researchers with an even larger volume of information to be processed. As available information increases, relevant material becomes progressively more difficult to find and the need for an automated information retrieval tool more apparent. The problem is even more vital for General Practitioners. Goldacre (2008:97) points out that "there have been an estimated 15 million medical academic articles published so far, and 5000 journals published every month... picking out what's relevant is a gargantuan task."[11]

Partridge's main objective is to provide reading recommendations and information retrieval for scientists and researchers. This should reduce the amount of extraneous information that users have to read for themselves by helping them find information specific to their interests more quickly. Partridge will achieve this through the use of several existing techniques in Natural Language Processing which are discussed below.

From the point of view of it's users, Partridge will assist researchers in two ways. The system will provide filtering of papers based upon their specific domain (i.e. is the paper primarily concerned with methodology within an experiment in chemistry or is it about ethics in psychological studies?) and their result, whether the paper yielded positive, negative or inconclusive evidence for a hypothesis. Depending upon the time constraints of the project, it is hoped that Partridge will also offer a user 'profiling' system that provides recommendations for researchers based on their reading history. This feature should help users find relevant papers more quickly or find research that they may have otherwise overlooked.

Search engines such as Google (`http://www.google.com/`), and social citation management tools such as CiteULike (`http://www.citeulike.org`), do offer some assistance in tracking down relevant information. However, these tools are often too general or rely upon the user knowing exactly what keywords to use before carrying out the search. These drawbacks are further discussed in Section 2.2 below.

To overcome the drawbacks of these existing systems, Partridge will make use of several cutting edge Artificial Intelligence (AI) techniques in order to analyse and process the papers in a more in depth way. AI is a very complex and field and implementing the above features will be incredibly challenging. To help with this, Partridge will build upon the system implemented by Liakata et al. for recognising hypotheses, methods, results, conclusions and a number of other scientific concepts in scientific articles [16]. The Natural Language Toolkit for Python will also be used to build models for recognising different fields and topics, classifying results as positive or negative and recognising different types of papers, from case studies to methodology papers[5].

2 Current Progress

2.1 Background

2.1.1 Artificial Intelligence

In science fiction literature and films, Artificial Intelligence (AI) and the ability of machines to automatically process and understand human language is almost always taken for granted. The current state of AI is a long way behind these fantastic visions. Dale(2000) comments that "Even the most ardent exponent of artificial intelligence research would have to admit that the likes of HAL in Kubrick's 2001: A Space Odyssey remain firmly in the realms of science fiction[10]".

Despite lacking behind the imagination of authors and script writers, over the last 60 years there has been a huge amount of progress in AI techniques. The phrase 'Artificial Intelligence' was coined in 1956[24] and can be used as an umbrella term, describing many subfields from "learning and perception to... diagnosing diseases*(Ibid)*."

Turing(1950) proposed a test for determining whether a machine could be considered intelligent or not[29]. Turing's test is based upon whether a computer can communicate in a natural language proficiently enough to deceive a human into thinking that the machine is also human. A machine able to pass such a test would need to possess the ability to represent and learn from knowledge, to be able to reason about what it knows and to be able to process natural language[24].

2.1.2 Natural Language Processing

Natural Language Processing (NLP) enables the automated extraction of meaningful information from texts written in human languages such as French or English. Liddy(2001) defines Natural Language Processing as:

> A theoretically motivated range of computational techniques for analyzing and representing naturally occurring texts at one or more levels of linguistic analysis for the purpose of achieving human-like language processing for a range of tasks or applications [18].

Over the last 60 years, NLP has been used in a wide variety of applications such as automated translation between languages[13], the engineering of systems for querying databases using natural languages [21] and for building 'chatterbot' systems designed to communicate with their users in a human-like way[1]. More recently, NLP has been used for text classification purposes such as classifying emotions within phrases and sentences [30] and even within suicide notes [15].

2.1.3 CISP, CoreSC and SAPIENTA

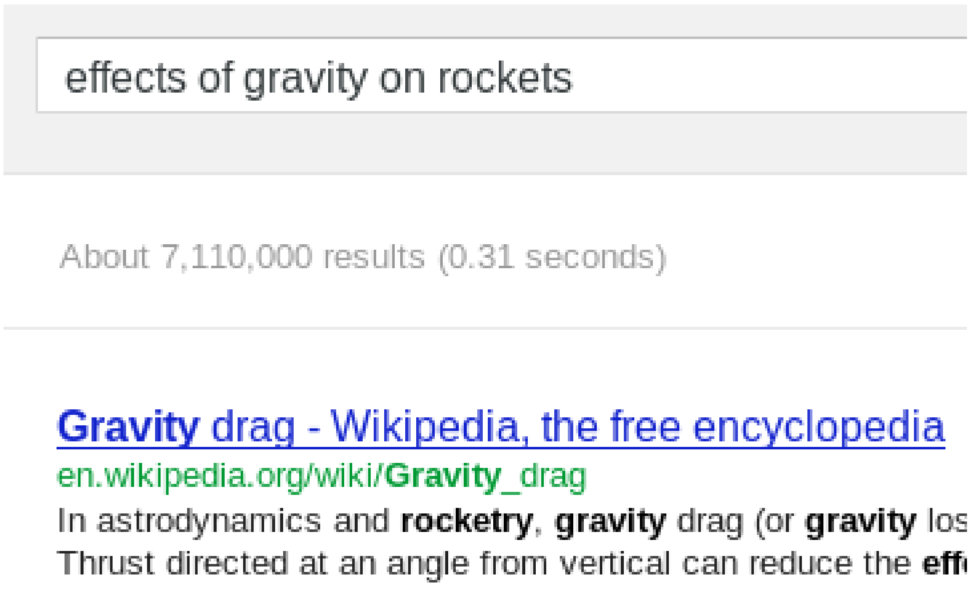
Liakata et al. (2012) describe a system for automatically processing and classifying sentences in a research paper according to the scientific concept they describe (e.g. a sentence can be categorised as a hypothesis, background information, a method or similar)[16]. SAPIENTA (`http://www.sapientaproject.com`) is a machine learning application, trained using a corpus of physical chemistry and biochemistry research papers that were pre-processed and annotated using Core Information about Scientific Papers (CISP)[17].

CISP, Soldatova and Liakata(2007), is a way to formally represent core scientific concepts (CoreSC), e.g. background, hypothesis, method etc., that should be present in the articles in a logical ontology[26]. Subsequent work implements this set of concepts as a three layered sentence based annotation scheme (CoreSC scheme) where annotations are encoded in Extended Markup Language (XML)[17].

2.2 Related Works

2.2.1 Search Engines

There are already many existing systems for finding and filtering information on the World Wide Web. Search engines are very useful for information retrieval in the very large and generalised search domain of the Internet. Most people have heard of Google (`http://wwww.google.com`), Yahoo (`http://www.yahoo.co.uk`), Bing (`http://www.bing.com`) and Ask (`http://www.ask.com`).


effects of gravity on rockets

About 7,110,000 results (0.31 seconds)

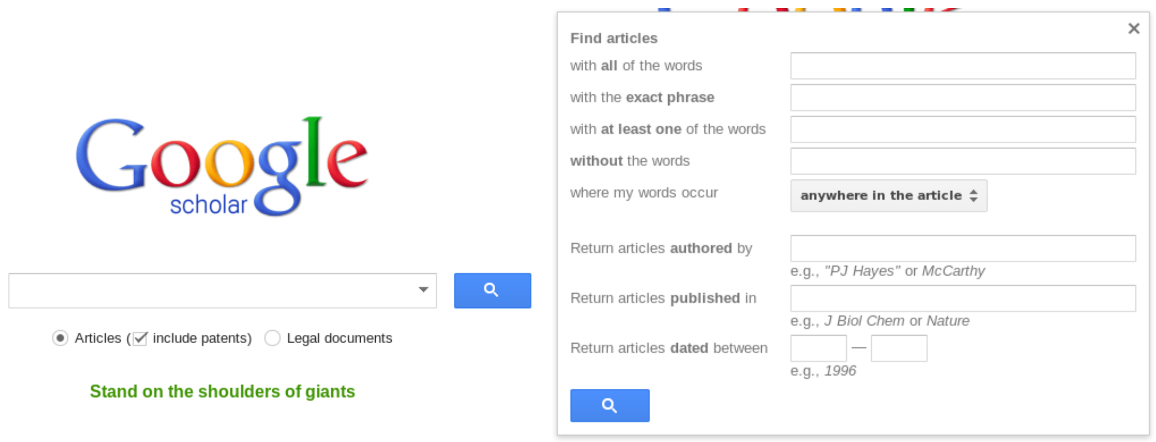
Gravity drag - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Gravity**_drag
In astrodynamics and **rocketry**, **gravity** drag (or **gravity** los
Thrust directed at an angle from vertical can reduce the **eff**

Figure 1: Google showing over 7M results for "effects of gravity on rockets"

Despite their use as general search tools on the World Wide Web, the problem space that search engines deal with is usually too big for them to find scientific papers and journals given a set of keywords. Internet search engines index a huge proportion of irrelevant information compared to useful information[4], and as a result, even relatively specific queries such as "effects of gravity on rockets" yield millions of results (as shown in Figure 1). This shortcoming has already led to the development of search engines designed specifically to find scientific papers.

2.2.2 Scientific Paper Search Engines

There are also a number of search and indexing systems that specifically look for scientific papers as opposed to web pages. One of the most publicised and well known paper search systems is Google Scholar (`http://scholar.google.com`). As can be seen in Figure 2a, This is an adaptation of Google's general search engine (discussed above) to specifically index and search scientific papers. Google also offers advanced query options specific to Scholar that allow searching by author, year and for words that occur only in the document title as shown in figure 2b. Whilst this does deal with the problem of 'information overload' and provides even more fine control over the information returned from searches, the user is still required to have a very good idea of what they are looking for in terms of keywords and/or specific authors. It is possible that a user would not know what they are looking for until they've seen it. Even if the user has a set of keywords to search for, they can only search the title of the paper or the content as a whole. This means that users who want to find a particular phrase within a CoreSC part of the paper (e.g. only look for this phrase in the 'Result' section of the paper) are unable to get results at their desired level of detail.


Google scholar
Articles (✓ include patents)
Legal documents
Stand on the shoulders of giants

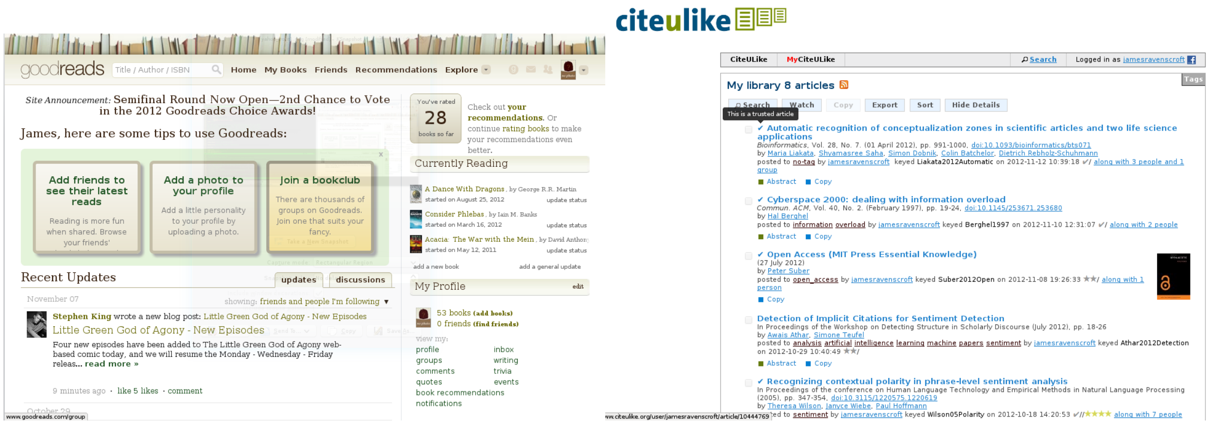
Find articles
with **all** of the words
with the **exact phrase**
with **at least one** of the words
without the words
where my words occur: **anywhere in the article**
Return articles **authored** by
e.g., *"PJ Hayes"* or *McCarthy*
Return articles **published** in
e.g., *J Biol Chem* or *Nature*
Return articles **dated** between —
e.g., *1996*

(a) Google Scholar's General front page (b) Advanced search features

Figure 2: Google Scholar's user interface

2.2.3 Social Citation and Recommendation Engines

Social citation and recommendation engines also provide a partial solution to the 'information overload' problem. Services like Goodreads (`http://www.goodreads.com/`) and CiteUlike (`http://www.citeulike.org/`) allow you to register your interest in specific authors and subjects. This allows the sites to build up a profile of the sorts of materials that you might be interested in and provide lists of recommendations as in Figure 3.



Site Announcement: Semifinal Round Now Open—2nd Chance to Vote in the 2012 Goodreads Choice Awards!

James, here are some tips to use Goodreads:

Add friends to see their latest reads
Reading is more fun when shared. Browse your friends'

Add a photo to your profile
Add a little personality to your profile by uploading a photo.

Join a bookclub
There are thousands of groups on Goodreads. Join one that suits your fancy.

Recent Updates — updates — discussions

November 07 — showing: friends and people I'm following ▾

Stephen King wrote a new blog post: Little Green God of Agony - New Episodes
Little Green God of Agony - New Episodes
Four new episodes have been added to The Little Green God of Agony web-based comic today, and we will resume the Monday - Wednesday - Friday releas... read more »

9 minutes ago · like 5 likes · comment

October 29

www.goodreads.com/group

You've rated 28 books so far

Check out your recommendations. Or continue rating books to make your recommendations even better.

Currently Reading

A Dance With Dragons, by George R.R. Martin
started on August 25, 2012 — update status

Consider Phlebas, by Iain M. Banks
started on March 16, 2012 — update status

Acacia: The War with the Mein, by David Anthony
started on May 12, 2011 — update status

add a new book — add a general update

My Profile — edit

53 books (add books)
0 friends (find friends)

view my:
profile — inbox
groups — writing
comments — trivia
quotes — events
book recommendations
notifications

My library 8 articles

Search — Watch — Copy — Export — Sort — Hide Details

This is a trusted article

✓ Automatic recognition of conceptualization zones in scientific articles and two life science applications
Bioinformatics, Vol. 28, No. 7. (01 April 2012), pp. 991-1000, doi:10.1093/bioinformatics/bts071
by Maria Liakata, Shyamasree Saha, Simon Dobnik, Colin Batchelor, Dietrich Rebholz-Schuhmann
posted to no-tag by jamesravenscroft keyed Liakata2012Automatic on 2012-11-12 10:39:18 ✓/ along with 3 people and 1 group
Abstract Copy

✓ Cyberspace 2000: dealing with information overload
Commun. ACM, Vol. 40, No. 2. (February 1997), pp. 19-24, doi:10.1145/253671.253680
by Hal Berghel
posted to information overload by jamesravenscroft keyed Berghel1997 on 2012-11-10 12:31:07 ✓/ along with 2 people
Abstract Copy

✓ Open Access (MIT Press Essential Knowledge)
(27 July 2012)
by Peter Suber
posted to open_access by jamesravenscroft keyed Suber2012Open on 2012-11-08 19:26:33 ★★/ along with 1 person
Copy

Detection of Implicit Citations for Sentiment Detection
In Proceedings of the Workshop on Detecting Structure in Scholarly Discourse (July 2012), pp. 18-26
by Awais Athar, Simone Teufel
posted to analysis artificial intelligence learning machine papers sentiment by jamesravenscroft keyed Athar2012Detection on 2012-10-29 10:40:49 ★★/
Abstract Copy

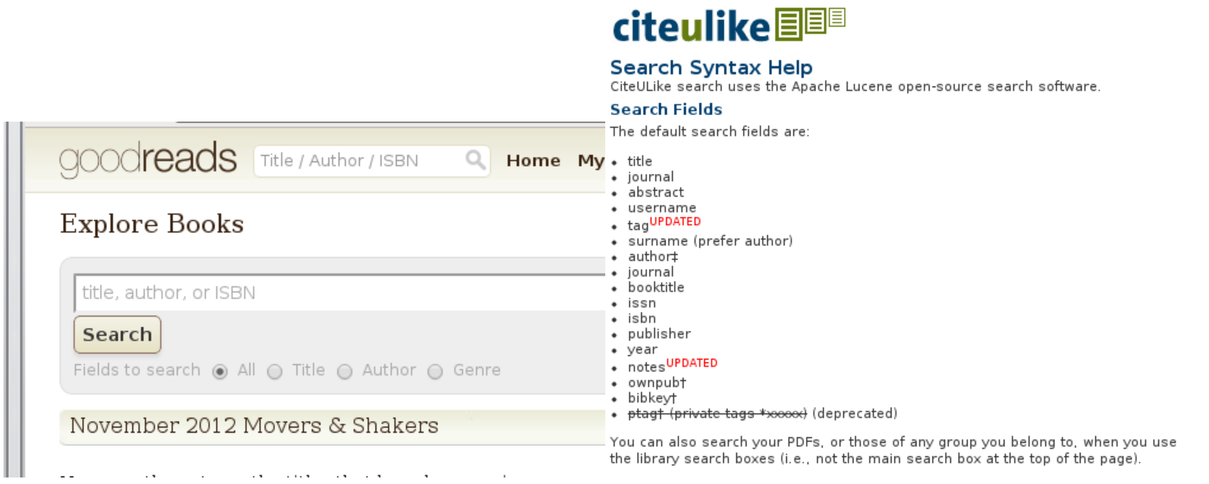
✓ Recognizing contextual polarity in phrase-level sentiment analysis
In Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing (2005), pp. 347-354, doi:10.3115/1220575.1220619
by Theresa Wilson, Janyce Wiebe, Paul Hoffmann
ted to sentiment by jamesravenscroft keyed Wilson05Polarity on 2012-10-18 14:20:53 ✓//★★★★ along with 7 people

ww.citeulike.org/user/jamesravenscroft/article/10444769

(a) Goodreads user profile page (b) a CiteULike user profile page

Figure 3: Goodreads and citeulike social recommendation systems

These systems have the ability to make recommendations to the user without requiring specific keywords or search terms. They do this by learning the user's profile and taking into account the preferences of their 'friends' and their browsing history. However, the above-named systems do not take into account the content of the paper or book. They only deal with metadata as can be seen in Figure 4. This means that important discriminatory information that could be contained within the actual document content is overlooked completely.



Explore Books

title, author, or ISBN

Search

Fields to search ● All ○ Title ○ Author ○ Genre

November 2012 Movers & Shakers

Search Syntax Help

CiteULike search uses the Apache Lucene open-source search software.

Search Fields

The default search fields are:

- title
- journal
- abstract
- username
- tag UPDATED
- surname (prefer author)
- author‡
- journal
- booktitle
- issn
- isbn
- publisher
- year
- notes UPDATED
- ownpub†
- bibkey†
- ~~ptag† (private tags *xxxxx)~~ (deprecated)

You can also search your PDFs, or those of any group you belong to, when you use the library search boxes (i.e., not the main search box at the top of the page).

(a) Goodreads advanced search page (b) CiteULike advanced search options

Figure 4: Goodreads and citeulike search only deal with metadata.

2.3 Methodology

The language that Partridge will be written in is one of the most fundamental and important choices to be made. The language choice effects not only the end performance of the application, but the speed at which the project is developed and the portability of the end code [8]. It was decided that Python would be the best language for developing Partridge.

Python is a programming language with a famously shallow learning curve, readable syntax and dynamic inspection that facilitates rapid prototyping [5]. The language was invented by Guido Van Rossum as a multi-use, flexible, interpreted scripting language, extensible via compiled C modules[2]. Developing Partridge in python would provide a flexible prototyping environment and readable syntax allowing for investigative work to be carried out quickly and efficiently with minimal overheads for writing boiler-plate code and debugging. The ability to extend the language using native code would give Partridge an even greater advantage. Since NLP techniques can be quite processor intensive and Python is an interpreted language, and in its very nature, slower than a compiled language, inefficient sections of Python code could be re-written as C extensions and compiled into the application.

Other languages that were considered were Java and C. However, it was decided that Python's clean syntax and ease of use for prototyping were preferable to the C and Java style syntax. As stated above, C may still be used in part for developing Python extensions to improve the overall performance of the application.

2.3.1 Web Presentation Frameworks

Partridge's web frontend will be used by researchers to wish to query the system for information on scientific papers. As it is a web tool, Partridge will be presented as a set of HTML web pages.

For this purpose, the Flask (`http://flask.pocoo.org`) library was chosen. Flask is a python 'microframework' for web development. Ronacher (2012) writes that "Flask aims to keep the core simple but extensible...[and] won't make any decisions for you[22]." It uses a simple templating system that can optionally be swapped out for an equivalent system and does not impose any limitations or requirements on the layout of the program or the type of database that is used. Flask also integrates very simply into existing software using Python annotations*(Ibid)*.

An alternative approach that was considered was using a separate software stack such as the Apache web server (`http://httpd.apache.org/`) and PHP (`http://www.php.net`). However, this would involve adding more software to the project and building an interface between the PHP scripts and Partridge's python backend. The Django framework (`http://www.djangoproject.com/`) for Python was also considered but it was determined to be overly complicated in terms of Partridge's requirements.

2.3.2 Natural Language Libraries

.

It was decided that a pre-existing NLP library should be used to provide implementations of existing algorithms and techniques for Partridge. For this purpose the The Natural Language Toolkit (NLTK) was selected. The NLTK is a simple and intuitive library written for Python. Bird(2009) states that NLTK was designed "to provide an intuitive framework along with substantial building blocks, giving users a practical knowledge of NLP[5]". NLTK provides implementations for sentence splitting, tagging (the process of analysing sentences and classifying words as verbs, nouns etc.), and information retrieval from text *(Ibid)*. NLTK also provides implementations of classifiers and a framework for extracting features to be used for classification from text. There is also a free book that accompanies the project available at `http://nltk.org/book/` which provides a huge amount of information on how to implement many popular NLP techniques using the library.

Some other libraries were considered[19][9]. However, these were written for Java and Partridge is a python project. They also require a steeper learning curve than NLTK.

2.4 Prototyping/Current Work

2.4.1 Sourcing Scientific Papers

For Partridge to be a success, it is important to include a large variety of different scientific papers in its corpus. Therefore, several sources of scientific papers were explored and considered.

The most convenient and accessible source is the ART corpus that SAPIENTA was trained with[15]. This corpus is already stored using the CoreSC schema and has been pre-annotated. This means that Partridge would not need to do any conversion or pre-processing on the papers. These scientific papers are all chemistry papers. Therefore, more papers from other scientific domains are needed to get a wider variety of topics and make Partridge useful to as wide an audience as possible.

The 'mega-journals' PLOS ONE `http://www.plosone.org/` and arXiv `http://www.arxiv.org/` were suggested as sources for more open access articles that could be added to Partridge. Both of these sites were found to contain large volumes of open access papers. However, most of these papers were published as PDF files which meant that some conversion would be required to make them compatible with Partridge.

2.4.2 PDF Conversion

Most scientific papers available on the internet are formatted as PDF documents. However, Partridge uses and stores documents that use the CoreSC schema by Soldatova and Liakata[14]. Therefore some spike work was carried out to determine the feasibility of converting papers published as PDF documents into XML documents. Townsendi et al(2009) liken converting PDF to XML to "converting hamburgers into cows," they go on to explain that PDF documents do not contain any semantic data and documents lose much of their explicit structure when they are formatted in this way [28]. Therefore, to convert PDF documents into an NLP-friendly format, some heuristics must be used to detect the document's structure[25].

This was the first big challenge in the project. A prototype script was written using a Python PDF extraction library called PDFMiner (`http://www.unixuser.org/~euske/python/pdfminer/index.html`). This toolkit already contains some heuristics about how to extract text from PDF documents, grouping together characters that appear very close to each other, and separating paragraphs and headings when a larger area of whitespace is detected[25]. Despite these rules, the library still produced some extraneous whitespace and newline characters as part of the output. A subroutine to trim whitespace and newlines was added to the script to resolve this problem.

The next stage was to split the text into sentences in preparation for processing with SAPIENTA. With the assistance of the NLTK library, a sentence splitting subroutine was implemented. This used a machine learning algorithm that had been trained to recognise sentence boundaries to split the text. Each sentence was then added to a CoreSC compatible XML document for processing by SAPIENTA.

Initially, the PDF conversion subroutine had a very high error rate due to the variation in the formatting of scientific papers. It was suggested that PDFX (`http://pdfx.cs.man.ac.uk/`), a free service hosted by the University of Manchester could be used instead of PDFMiner for the initial PDF data extraction. The main advantage of PDFX over the PDFMiner library is that it is a trained machine learning system that has been trained using a large full-text selection of scientific articles; PDFMiner uses more general heuristics designed to process a large selection of different types of PDF document.

PDFX also provides output that already has some metadata, such as title, author, and abstract, associated with it. PDFMiner did not provide any metadata, and it was necessary for the script to guess which passage of text was the abstract after the initial text extraction stage.

With the new PDF extraction method in place, the script ran without the need to modify either of the whitespace sanitiser or sentence splitter routines. The process was much more successful and able to produce SAPIENTA-compatible documents from most of the PDF input files that were provided.

2.4.3 Pre-processing with SAPIENTA

With a successful PDF conversion script, the next step was to try and run SAPIENTA over the converted papers and annotate them, ready for inclusion in the Partridge corpus.

By default, SAPIENTA is packaged as a web-based tool, written in Java, that can be downloaded (from `http://www.sapientaproject.com/software`) and used to annotate one paper at a time. Dr Liakata was able to provide information on two alternative ways of using the system. One method was to submit a remote procedure call (RPC) to a server running SAPIENTA with a batch of papers and retrieve the output. The other method was to use an alternative version of the code that runs locally in a Python environment and could be modified to process papers as a batch.

A script was written to send un-annotated XML documents to the remote SAPIENTA server and retrieve a list of annotations. This worked well until the server stopped replying to requests. This meant that no further conversions could be carried out until the server was repaired and raised concerns about how the remote servers might cope with a large number of automated requests from a full version of Partridge.

The Python version of SAPIENTA was then downloaded and a test executed. Unfortunately there were several data files missing from the package that had to be acquired from Dr Liakata.

SAPIENTA for Python also relies upon a package called CRFSuite which implements Conditional Random Fields, a method for segmenting and labelling sequence data[20]. This library did not compile properly on the test environment and its creator had to be contacted via a mailing list (See Appendix C. After a few days, the owner responded and the library was compiled successfully.

Once all the data files and libraries were successfully in place, the Python version of SAPIENTA was used to process some of the papers converted from PDF. This was a great success and the annotations provided by SAPIENTA seemed to be accurate.

2.4.4 Interface Design

To help in visualising how the final web frontend will appear, some diagrams of the user interface have been drawn. You can see these diagrams in Appendix A.

2.4.5 System Processes

To visualise how some of the system processes will work and how they should be programmed, some flow diagrams have been produced and attached in Appendix B

3 Planning

3.1 Development Methodology

3.1.1 Existing Methodologies

Selecting a suitable development methodology for building Partridge is another very important choice for the project.

Under the traditional ‘Waterfall’ Software Development model, Requirements Gathering, Analysis, Program Design, Coding, Testing and Operations were all defined as formal phases in the development cycle. There is little flexibility other than moving back up the waterfall to rectify mistakes after testing[23]. This model was very focused on paperwork and bureaucracy, trying to maintain a paper trail and manage risk through accountability

(Ibed). This approach to software development is very heavyweight and slow and often produced software that did not match the users' needs as a result [7].

As an alternative to the heavyweight Waterfall approach, Beck et al came up with the principle of the Agile Manifesto, favouring a lightweight, responsive development model over the heavyweight slow waterfall system[3]. Many of Beck's ideas focus around working in a team of developers and prioriting communication between team members *(Ibid)*. This is most prominent in the Extreme Programming (XP) method of software development. Since Partridge is an individual project, XP is not really applicable. However, some concepts like rapid prototyping/spike work and iterative release cycles will be used as part of the Partridge development methodology.

3.1.2 Partridge's Development Methodology

Partridge is an individual project but does involve discussions with supervisors. The customers have been identified as the end-users of the system. Therefore, a customised methodology has been adopted. Firstly, all design and planning documentation have been written up and placed on a wiki which is accessible and modifiable by the author and both supervisors. This creates a paper trail for all tasks and also allows collaboration between involved parties through the Internet. A full printout of the wiki is available in Appendix E.

Weekly meetings are held with both supervisors. The notes from the preceeding week are analysed and each task discussed in depth. New tasks are then noted down along with any observations that should be documented. These new notes are uploaded to the wiki the following day or earlier. Each party present at the meeting adds their own observations to the notes page. This page is then reviewed at the next meeting. As seen in Appendix E, this practice has already been running for several weeks and has so far proven to be highly effective.

Partridge will adopt an Agile approach to release cycles, producing a working software package at iterations of one month. Each iteration, the software will include more of the desired functionality discussed above and in the wiki. GitHub's issue manager program is being used to track tasks and bugfixes and plan which tasks will be carried out in which iteration. Tasks that are created in a full iteration (where no development time is left) will be added to a backlog and integrated in the next iteration with enough development time to contain it. Tasks are also assigned a priority, higher priority issues being tackled before low priority ones.

Partridge's testing strategy consists of multiple unit tests that are run at integration of new code into the codebase. As soon as the first release is built, Partridge will be made available for use by the public and users encouraged to test the system and submit any bugs via the GitHub issue manager. It is hoped that colleagues at Aberystwyth University and Dr. Liakata's colleages at the EBI will try to use the system one it becomes available.

3.1.3 Work Timeline

The tasks involved in Partridge have been carefully calculated and prioritised. They were then added to the GitHub issue management system and a report generated listing them in the order that they are expected to be accomplished. This report can be seen in Appendix D.

3.1.4 Mid-Project Demonstration Plan

The mid-project demonstration is scheduled for after iteration two of the Partridge project. If everything runs to schedule, then at this point it will be possible to demonstrate keyword search within the project's database and filter based upon the polarity of the paper's results. For redundancy purposes, Partridge will be configured to run as a server on multiple computers. Both this and the final project demonstration will require a room with Internet access. However, should this be unavailable, then Partridge could be run locally on the author's laptop.

3.1.5 Final Project Demonstration Plan

The final demonstration of Partridge will be fairly similar to the Mid-Project demonstration. However, it should include all of the planned classifiers and if there is extra time on the project, the profiling/recommendation engine will also be demonstrated. This demonstration will use the same redundancy precautions as the Mid-project demonstration above, and will also need a room with the internet if available.

A User Interface Designs

A.1 Partridge Query Interface

Figure 5 shows the user interface that will be utilised to search and query Partridge's corpus.

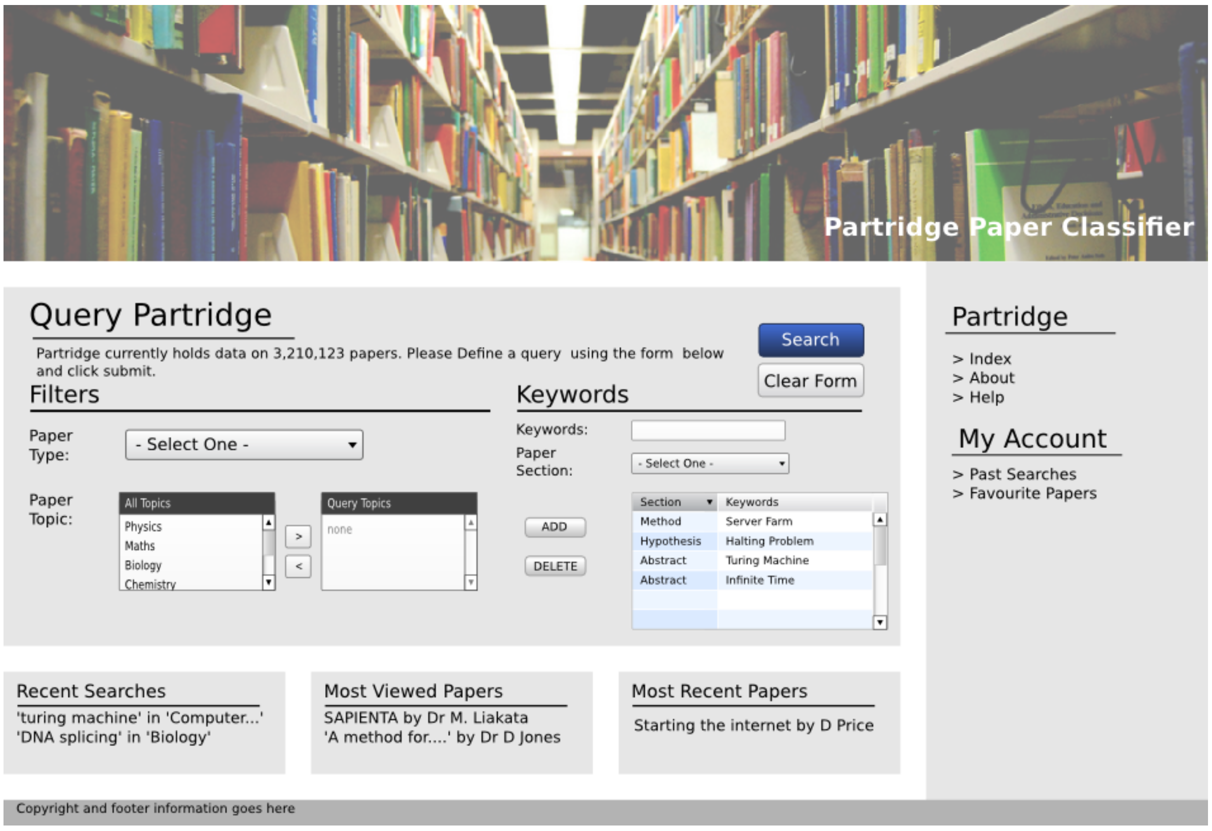
The query form shows two main sections: Filters and Keywords.

Filters can be used to show a large set of papers where the user is unsure of what to search for. Users will be able to look for papers by type and topic as discussed above. The paper result filter has not been included in this diagram. However, it is expected that a drop down menu for result type would be included in the filters section of the form.

Keywords allows the user to look for a set of keywords that is specifically within a CoreSC concept for a paper. For example, the reader may want to find experiments that use a server farm to do lots of calculations. Therefore, they would enter "Server Farm" as their keywords and chose "Method" from the paper section. The user then adds this to the set of keyword queries in the table below the text field with the add button.

Once a user has configured both filters and keywords to their preferences, they click the search form to run the search itself.

Other notable features on this mockup include a list of most recent searches. This helps the user to understand how to use Partridge and gives them some suggestions for what they might search for. Similar listings are provided for the most popular papers in the Partridge database and the most recent papers added to the system.



Partridge currently holds data on 3,210,123 papers. Please Define a query using the form below and click submit.

Section	Keywords
Method	Server Farm
Hypothesis	Halting Problem
Abstract	Turing Machine
Abstract	Infinite Time

'turing machine' in 'Computer...'
'DNA splicing' in 'Biology'

SAPIENTA by Dr M. Liakata
'A method for....' by Dr D Jones

Starting the internet by D Price

Copyright and footer information goes here

Figure 5: User interface design for querying Partridge

B System Process Diagrams

This section documents the system process diagrams that have been produced. The first completed diagram shows how a paper will be added to Partridge and the actions that will be carried out upon it.

B.1 Adding a Paper to Partridge

The system starts off by converting the document to XML from PDF if necessary through the use of the PDFX tool provided by Manchester University. A sentence splitting tool

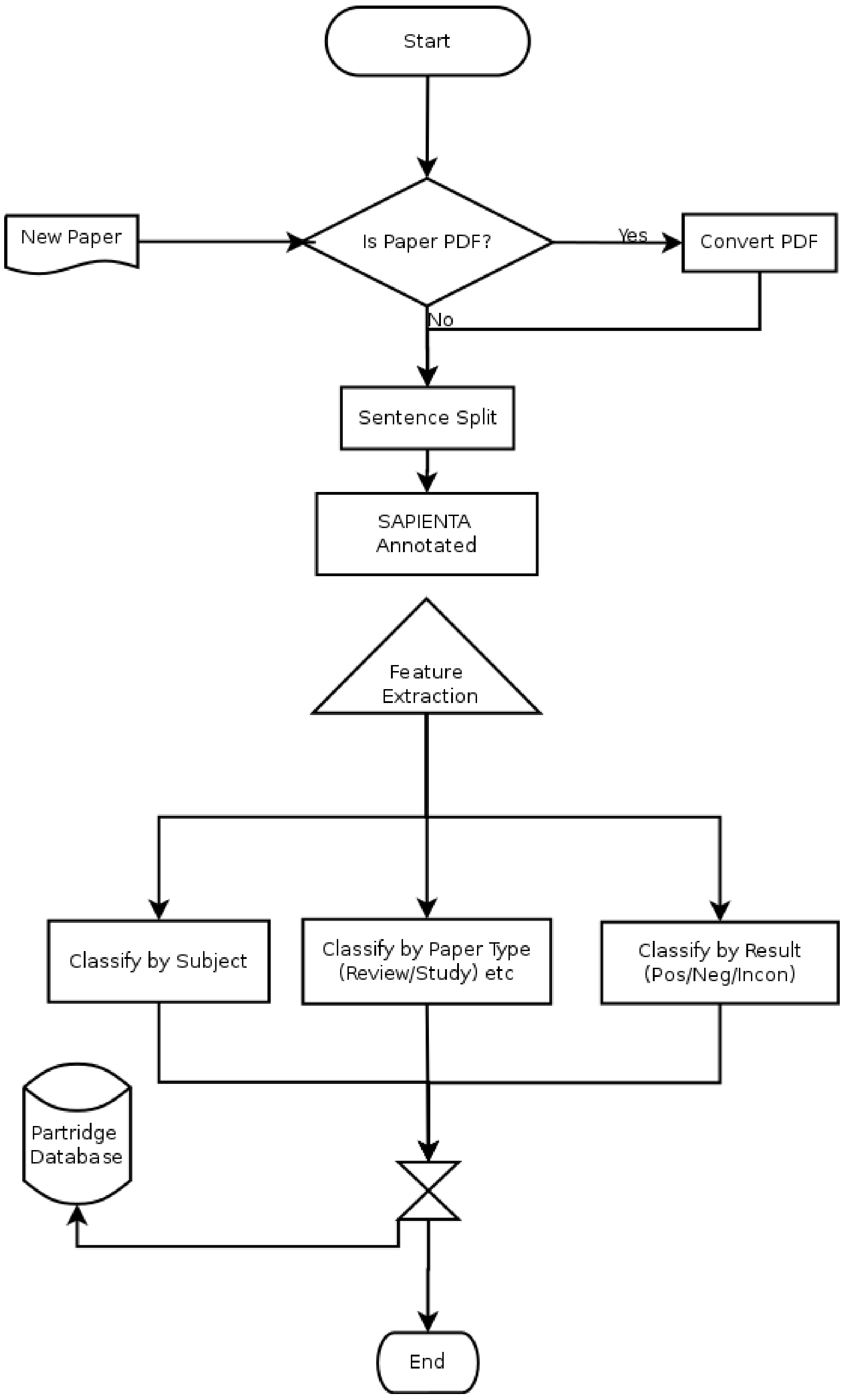


Figure 6: The process of adding a new paper to Partridge

is then used to parse the resulting document and separate the sentences.

Once the sentences have been split, Maria's SAPIENTA tool is used to annotate each sentence to determine which of the core scientific concepts it covers. This information is stored back in the XML file with the document.

After this, feature extraction is carried out upon the document to find useful features for the following classification tasks:

1. Paper topic/subject -i.e. is the paper a biology paper or a computer science paper?
2. Type of paper - i.e. is the paper a case study, an experiment or a literature review?
3. Paper result - i.e. did the paper have a positive, negative or inconclusive result?

These classifiers are then run (they could potentially be run in parallel if processing power is available) to determine the paper's class for each classifier respectively. The data gathered along with any paper metadata captured such as title, author, date, institute etc are then stored in the database.

C Email for CRFSuite

```
Hi All,

I'm trying to build CRFSuite's python extension
(http://www.chokkan.org/software/crfsuite/) on an Arch Linux 64-bit
environment using Swig 2.0.8 (PCRE enabled).

When I run SWIG, I get the following output:

<<OUTPUT OMITTED FOR BREVITY>>

Full output available at http://sourceforge.net/mailarchive/message.php?msg_id=300242

As far as I can see, all of these problems stem from this declaration in
the SWIG input file (http://pastebin.com/9JipJ1C1):
%template(ItemSequence) std::vector<CRFSuite::Item>;

I've had a look around on google and on this mailing list and the only
other "cannot copy typemap" errors I've encountered have been where
people have excluded the std namespace in favour of a 'using' statement.
As you can see, this example uses absolute class names so that isn't the
issue here. I haven't had any luck contacting the software maintainer
for CRFSuite yet either.

As you might expect, if I continue to compile the extension without
heeding these errors, when I try to make use of the ItemSequence object
I get the following python error:
TypeError: in method 'ItemSequence_append', argument 2 of type
```

```
'std::vector< CRFSuite::Item >::value_type const &'

I'm going to keep trying to sort this out and if I find the solution
independantly, I'll make sure to post a follow-up email. However, if
anyone else has any ideas about what might be happening here, please let
me know.

Thanks,

James Ravenscroft
AI & Robotics Undergraduate
Aberystwyth University
```

D Project Timeline

This table shows the projected timeline for the Partridge project. These calculations were carried out under the assumption of 2 full days of work per week on the project and one iteration equating to one month (28 days). There will be 7 iterations in total and after each, a new version of Partridge will be made available (except for iteration zero which is a research only iteration).

			Total Time (Days)	Filled Time (Days)	
	Iteration 0	October 2012 – November 2012	8	7	
Task ID	**Task**	**Description**	**Estimated Timeframe (Days)**	**Status**	**Dependencies**
1	PDF Conversion	To enable a wider variety of papers to be processed, a conversion script from PDF to XML must be developed.	1	Complete	-
2	Web Framework Evaluation	I need to determine which web framework for python would be the best	1	Complete	-
3	Feature Evaluation	I need to determine which features the system will use to classify the papers	2	Late	-
4	Classifier Evaluation	Which classifiers can be used to help the user find suitable papers?	1	Complete	-
5	Data Storage Evaluation	I need to determine how the data in the system will be stored? Are conventional RDBMSs suitable for this sort of task?	2	Late	-

			Total Time (Days)	Filled Time (Days)	
	Iteration 1	November 2012 – December 2012	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**
6	Web Interface Design	Interface mockups for all forms in the project should be generated using a graphical mockup tool	3	Ongoing	-
7	Basic web framework module	Use the chosen web framework to generate a base web server that will be used to access partridge's web frontend	1	Pending	2
8	Web Page Implementation	For each of the pages designed, create an HTML layout that can be served through the partridge web server	1	Pending	6,7
9	Data Storage Engine	The chosen database system needs to be implemented and an interface to the web server generated	1	Pending	5
10	Data-interface interaction	Simple query behaviour should be implemented to allow users to search papers by keyword-in-CoreSC using SAPIENTA classified papers	2	Pending	7,8,9

			Total Time (Days)	Filled Time (Days)	
	Iteration 2	December 2012 – January 2013	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**

11	Prototype Result classifier	This will mainly be investigative work into sentiment analysis on a paper's result to classify based on outcome: positive,negative etc	5	Pending	-
12	Integrate Result classifier	Integrate the prototype result classifier into partridge's backend system	2	Pending	11,10
13	Paper Upload Form	Create a form to allow users to upload papers to Partridge. The option to remove copyright infringing papers must be available	1	Pending	10

			Total Time (Days)	Filled Time (Days)	
	Iteration 3	January 2013 – February 2013	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**
14	Investigate Subject Classifier	More information is required before a classify-by-subject classifier can be implemented.	2	Pending	
15	Implement Subject Classifier	Once the investigation is completed, build a prototype subject classifier	4	Pending	14
16	Integrate Subject Classifier	Integrate the prototype subject classifier into the partridge backend	2	Pending	15,10

			Total Time (Days)	Filled Time (Days)	
	Iteration 4	February 2013 – March 2013	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**
17	Investigate Paper Type Classifier	Carry out some investigation work into how a paper type classifier could be built (type being experiment, case study etc)	2	Pending	
18	Implement Paper Type Classifier	Build prototype paper type classifier module	4	Pending	17
19	Integrate Paper Type Classifier	Integrate prototype paper type classifier into partridge's backend system.	2	Pending	18,10

			Total Time (Days)	Filled Time (Days)	
	Iteration 5	March 2013 – April 2013	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**
20	Acceptance Testing	Acceptance tests and bug fixing for Partridge and it's backend.	6	Pending	12,13,16,19
21	Final Report – Draft	Start drafting the final report for the project	2	Pending	20

			Total Time (Days)	Filled Time (Days)	
	Iteration 6	April 2013 – May 2013	8	8	
Task ID	**Task**	**Description**	**Estimated Timeframe**	**Status**	**Dependencies**
21	Final Report – Release	Compl;ete the final report for the project for hand in at beginning of May.	8	Pending	21

E Project Wiki

All pages from the Wiki are shown below with the exceptions of those pages which have already been mostly featured in the document (UI Mockups and Progress Diagrams and the project timetable are not shown in this section).

E.1 Notes 03/10/2012

E.2 Meeting Notes from 03/10/2012

E.2.1 General topics

- Talked about using research papers instead of books :-
- copyright issues - mainly resolved because of Gutenburg project
- processing issues - too much information to process quickly and realistically
- Maria has some software for identifying sections in papers - could be used to help locate specific features
- There are quite a few sources for papers that could be used for the project:
- http://arxiv.org/
- http://plosone.org/
- Maria's papers are in XML format
- Formatting may or may not be an issue - PDFs are a nightmare to parse

E.2.2 Things to consider

- What sort of recommendations for different sources might there be and how would these recommendations be extracted?
- Feature engineering - what sorts of features might there be to pick out in the papers?
- I need to have a read through the sentiment analysis paper that Maria has been working on
- I need to have a play with the software for identifying sections in papers.

E.2.3 Misc concerns

- Decided that our weekly meeting should be Wednesdays at 3pm
- Investigate setting up source control and a wiki (that's been done as you can tell)

E.3 Notes 10/10/2012

E.4 Meeting Summary 10/10/2012

E.4.1 From Maria's notes

- James has gone through papers and thinks would be more doable to work with papers than Books
- James has found the XML format of PlosOne and ART/CoreSC corpus
- Domain needs to be determined
- Maria to send paper on sentiment analysis of citations

E.4.2 Current features and recommendations

- Analysis of results section - positive or negative result
- Potentially look at writing styles of authors using syntactic analysis
- Isolating terms within sections of papers (i.e. find me papers where methodology x was used)
- finer grain analysis of CoreSC categories
- nltk python toolkit. What other toolkits? Would jerboa be any good?
- common features for sentiment analysis

E.4.3 Basic system design

I have started a basic system design although it is very basic at the moment. [Image Omitted]

E.4.4 Background - Similar Systems and how they work

- Google Scholar
- Mendeley
- Citeulike
- Tweeting or 'liking' on facebook.

E.4.5 Reading

- TF-IDF
- Take a brief look at journal of negative results
- 'Bigrams + trigrams' and syntactic pattern finding
- Common features used in NLP
- Sentiment analysis of citations in papers
- 'Bag of words'

E.4.6 Coding/practical research

- Play some more with SAPIENTA
- Implement a (La)TeX to SciXML or pubmed command-line tool.

E.5 Notes 19/10/2012

E.6 Notes from Meeting 18/10/2012

E.6.1 General Comments

- I need to decide on a domain for the project. It may be that papers on a varied selection of topics would be a good start
- Decided on an agile development cycle with a basic program that is improved in 'iterations'.
- As part of testing the software, it may be possible to get other final year students to trial the engine and use it to make recommendations for their reading.
- I need to investigate ways of analysing papers for syntactic patterns. This may be indicative of:
  - Different authors' writing styles
  - Different types of paper -i.e. psychology vs physics

E.6.2 TODO

- Revisit PDF to XML conversion as conversion from PDF would make life a lot easier
- Maria mentioned that she may have access to an HTML to XML program which would be good to look at.
- Check out python Jerboa to see if it would be useful for the project.

E.7 Notes 24/10/2012

E.8 Notes from meeting 24/10/2012

Amanda was not present for this meeting.

E.9 Discussed

- We discussed the PDF conversion routine and decided that whilst important, it is crucial not to get too caught up in this.
- Maria is going to send me a link to an existing PDF to XML converter that might provide the functionality with some tweaking - or at least help
- We discussed the Python version of Sapienta which currently supports SciXML and simplified versions thereof.
- Maria needs to send me a couple of data files for this to work properly.
- Confirmed that Jerboa is a Java library (I was confused because I was expecting a python toolkit) and need to look at it properly now.
- Discussed the actual functionality of the system and came up with a brief system outline, around which I can plan.

E.10 TODO

- Now that we know what the system is going to do (See System Outline), I need to look at features that will allow us to:
- Pre-classify documents based on their topic, type, result etc
- Determine document similarity based on user preference from their tagging etc. (I can make use of the “Features in sentiment analysis” paper for this)
- Run more tests on the PDF parser and potentially look at another solution if this takes up too much time.
- Have a go with Python SAPIENTA once the data models are present

E.11 Notes 31/10/2012

E.12 James' Meeting Notes

E.12.1 To Think about

- Discussed imminent deadline for project progress report - 19/11/2012
- Decided that having a report finished by 14/11/2012 would be a good report to give Amanda and Maria a chance to check over the report before handin.
- Maria explained that it is possible to send a batch of papers to Sapienta rather than one at a time. She is going to provide information on how this can be achieved at some point during the week.
- I should describe my working processes and why they are helpful.

E.12.2 Todo

- We discussed the merits of using the pdfx converter instead of my script - might have plug-and-play compatibility with Sapienta for a start - need to verify this.
- I need to send my error message the CRFSuite compilation to Maria so that she can contact the maintainer.
- I need to expand on the system outline providing more specific goals and timeframes - this will become my development plan.
- I need to look at textpresso and see if it will help me with my comparison/evaluation

E.13 Notes 07/11/2012

E.14 Notes from 7/11/2012

E.14.1 Discussed

- What I'll need to have for my demo:
  - Simple web interface
  - Few basic classifiers - probably result of paper positive/negative and maybe paper type - proportions of coreSC concepts.
  - Could show off PDF to XML converter script

E.14.2 To Consider

- Is the disparity between bio paper CoreSC classification and other sorts of paper to do with Sapienta's model or does it demonstrate a feature - different subjects may have different proportions of each concept.
- If I'm using distribution of concepts as an aggregate feature, I need to make sure the system does not fall over if a concept is missing.
- Front end interface:
- Need to find a new word for 'section' Scientific Concept may be a bit too 'scary', is there a better term for this?
- If I allow users to upload papers, they must sign a digital disclaimer to say that they have the right to do so
- There should be a "report copyrighted material" system

E.14.3 To Do

- Biggest task this week: Progress Report.
  - Draft due 14/11/2012 for discussion on 15/11/2012
  - Final deadline 19/11/2012
  - Could provide wiki content as an appendix to the report
  - Come up with a project timeline - actually consider dates
  - Start with the web interface and work upwards.
- Send Maria the missing header file from crfsuite
- Link system outline from wiki index
- Maria said she'd look for a topic detection paper for me to peruse
- I need to have another look at relevant features - struggling with this
- Make a wiki list of features that could be used for classification

F References

[1] B. Alfonsi, "“Sassy” Chatbot Wins with Wit," pp. 6–7, January 2006.

Description of a chatbot that used NLP to win a bronze prize for being human-like.

[2] S. U. An and G. V. Rossum, "Python for unix/c programmers copyright 1993 guido van rossum 1," in *Proc. of the NLUUG najaarsconferentie. Dutch UNIX users group*, 1993.

Paper by the creator of python discussing differences between C and Python

[3] K. Beck, M. Beedle, A. van Bennekum, A. Cockburn, W. Cunningham, M. Fowler, J. Grenning, J. Highsmith, A. Hunt, R. Jeffries, *et al.*, "The agile manifesto," *http://www. agilemanifesto.org/principles.html*, vol. 7, no. 08, p. 2009, 2001.

Original paper/manifesto discussing agile software development

[4] H. Berghel, "Cyberspace 2000: Dealing With Information Overload," *Commun. ACM*, vol. 40, no. 2, pp. 19–24, February 1997. [Online]. Available: http://dx.doi.org/10.1145/253671.253680

Paper presented in the ACM explaining 'information overload' and a summary of the shortfalls of modern search engines in information retrieval.

[5] S. Bird, E. Klein, and E. Loper, *Natural Language Processing with Python*, ser. Oreilly Series. O'Reilly Media, Incorporated, 2009. [Online]. Available: http://books.google.co.uk/books?id=KGIbfiiP1i4C

This book provides an interesting preface on NLP and some reference for the NLTK python library

[6] B.-C. Björk, A. Roos, and M. Lauri, "Scientific journal publishing: yearly volume and open access availability," http://InformationR.net/ir/14-1/paper391.html], 2009.

This paper provided some insight into the growing area of online paper publishing and provided some figures on how many papers are published annually (or were in 2006).

[7] B. W. Boehm, "A spiral model of software development and enhancement," *Computer*, vol. 21, no. 5, pp. 61 –72, may 1988.

Article explaining the spiral software development model and discussing why the waterfall doesn't work very well

[8] C. Britton, "Choosing a programming language," http://msdn.microsoft.com/en-us/library/cc168615.aspx, 2008.

Brief paper discussing how to choose a programming language successfully

[9] H. Cunningham, D. Maynard, and K. Bontcheva, *Text processing with gate.* Gateway Press CA, 2011.

Another NLP toolkit for java. This one seems very complex compared to NLTK

[10] R. Dale, H. Moisl, and H. Somers, *Handbook of Natural Language Processing.* Marcel Dekker, 2000. [Online]. Available: http://books.google.co.uk/books?id=VoOLvxyX0BUC

Provided some good background information on NLP and an interesting preface on modern AI capabilities

[11] B. Goldacre, *Bad Science.* HarperCollins Publishers, 2008. [Online]. Available: http://books.google.co.uk/books?id=Gv1NQubrGNIC

Goldacre explains some bad practices in science and the sheer volume of papers medical professionals are expected to read.

[12] S. Harnad and T. Brody, "Comparing the impact of open access (oa) vs. non-oa articles in the same journals," *D-lib Magazine*, vol. 10, no. 6, 2004.

This paper observed that the popularity of Open Access articles is growing year by year - and so is awareness and visibility of OA.

[13] J. Hutchins, "The first public demonstration of machine translation: the georgetown-ibm system, 7th january 1954," in *AMTA conference*, 2004.

One of the first examples of NLP techniques for automated machine translation.

[14] M. Liakata and L. Soldatova, "Guidelines for the annotation of general scientific concepts," *Aberystwyth University, JISC Project Report http://ie-repository. jisc. ac. uk/88*, 2008.

This paper provides a set of guidelines on how to use CoreSC and CISP to annotate scientific documents.

[15] M. Liakata, J.-H. H. Kim, S. Saha, J. Hastings, and D. Rebholz-Schuhmann, "Three Hybrid Classifiers for the Detection of Emotions in Suicide Notes." *Biomedical informatics insights*, vol. 5, no. Suppl. 1, pp. 175–184, 2012. [Online]. Available: http://dx.doi.org/10.4137/BII.S8967

Research giving an insight into sentiment analysis techniques using multiple output categories rather than just binary classification.

[16] M. Liakata, S. Saha, S. Dobnik, C. Batchelor, and D. Rebholz-Schuhmann, "Automatic recognition of conceptualization zones in scientific articles and two life science applications," *Bioinformatics*, vol. 28, no. 7, pp. 991–1000, Apr. 2012. [Online]. Available: http://dx.doi.org/10.1093/bioinformatics/bts071

This is Maria's key paper on SAPIENTA. It discusses some approaches her and her team took to annotating CoreSC in papers and how the system works

[17] M. Liakata, S. Teufel, A. Siddharthan, and C. Batchelor, "Corpora for the conceptualisation and zoning of scientific papers," in *Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)*, N. C. C. Chair), K. Choukri, B. Maegaard, J. Mariani, J. Odijk, S. Piperidis, M. Rosner, and D. Tapias, Eds. Valletta, Malta: European Language Resources Association (ELRA), may 2010.

Paper that presents the CoreSC annotation schema based on previous CISP work

[18] E. Liddy, "Natural language processing," 2001.

In her encyclopedia entry, Liddy defines natural language processing and the components that make it up. This was used as a basis for some of Partridge's literature review.

[19] A. K. McCallum, "Mallet: A machine learning for language toolkit," http://mallet.cs.umass.edu, 2002.

An NLP toolkit written for Java, deemed to be overcomplicated and underdocumented

[20] N. Okazaki, "Crfsuite: a fast implementation of conditional random fields (crfs)," 2007. [Online]. Available: http://www.chokkan.org/software/crfsuite/

[21] G. Rao, C. Agarwal, S. Chaudhry, N. Kulkarni, and S. Patil, "Natural language query processing using semantic grammar," *International Journal on Computer Science and Engineering*, vol. 2, no. 2, pp. 219–223, 2010.

This paper's background discusses LIFER/LADDER as natural language interface techniques for a database system

[22] A. Ronacher, "Flask documentation," http://flask.pocoo.org/docs/ Retrieved on 13/11/2012, October 2012.

Manual for the Flask microframework with brief forward on the author's motivation

[23] W. W. Royce, "Managing the development of large software systems: concepts and techniques," in *Proceedings of the 9th international conference on Software Engineering*, ser. ICSE '87. Los Alamitos, CA, USA: IEEE Computer Society Press, 1987, pp. 328–338. [Online]. Available: http://dl.acm.org/citation.cfm?id=41765.41801

This is the first paper that introduces a formal waterfall method. It is used show how waterfall can be used to make large software projects more viable

[24] S. Russell and S. Norvig, *Artificial Intelligence: A Modern Approach*, ser. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2010. [Online]. Available: http://books.google.co.uk/books?id=8jZBksh-bUMC

Provided some general background in AI as well as a lot of information about machine learning techniques to be used as a backend for Partridge's NLP system.

[25] Y. Shinyama, "Programming with pdfminer," , 2011.

Brief instruction manual for PDFMiner toolkit, explains how the library extracts text from documents

[26] L. Soldatova and M. Liakata, "An ontology methodology and cisp-the proposed core information about scientific papers," *JISC Project Report*, 2007.

In this paper, the CISP ontology is formalised and suggested as a way of providing better metadata for papers

[27] P. Suber, "Open Access Overview," http://www.earlham.edu/~peters/fos/overview.htm retrieved on 11/11/2012, October 2012.

This article gives a brief overview of Open Access publishing, what its about and how it works.

[28] J. Townsend, J. Downing, and P. Murray-Rust, "CHIC - Converting Hamburgers into Cows," in *2009 Fifth IEEE International Conference on e-Science*. IEEE, Dec. 2009, pp. 337–343. [Online]. Available: http://dx.doi.org/10.1109/e-Science.2009.54

This paper discussed the automatic conversion of PDF papers to the SciXML format. I was able to use it to get some idea of how automated PDF conversion could be carried out and I was able to write a PDF to SciXML converter with the help of the pyPdf library.

[29] A. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

This is one of the first papers to discuss artificial intelligence and provides a link between natural language processing and AI

[30] T. Wilson, J. Wiebe, and P. Hoffmann, "Recognizing contextual polarity in phrase-level sentiment analysis," in *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, ser. HLT '05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 347–354. [Online]. Available: http://acl.ldc.upenn.edu/H/H05/H05-1044.pdf

This paper discusses the best features for classifying the polarity of a phrase within the context of a machine learning NLP system.