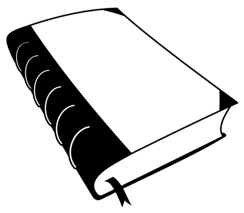


Aberystwyth University

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 15, 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[31]) across most scientific disciplines[6][15] 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."[13]

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 [19]. 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[11]".

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[28] 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[33]. 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[28].

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 [21].

Over the last 60 years, NLP has been used in a wide variety of applications such as automated translation between languages[16], the engineering of systems for querying databases using natural languages [24] 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 [34] and even within suicide notes [18].

2.1.3 CISP, CoreSC and SAPIENTA

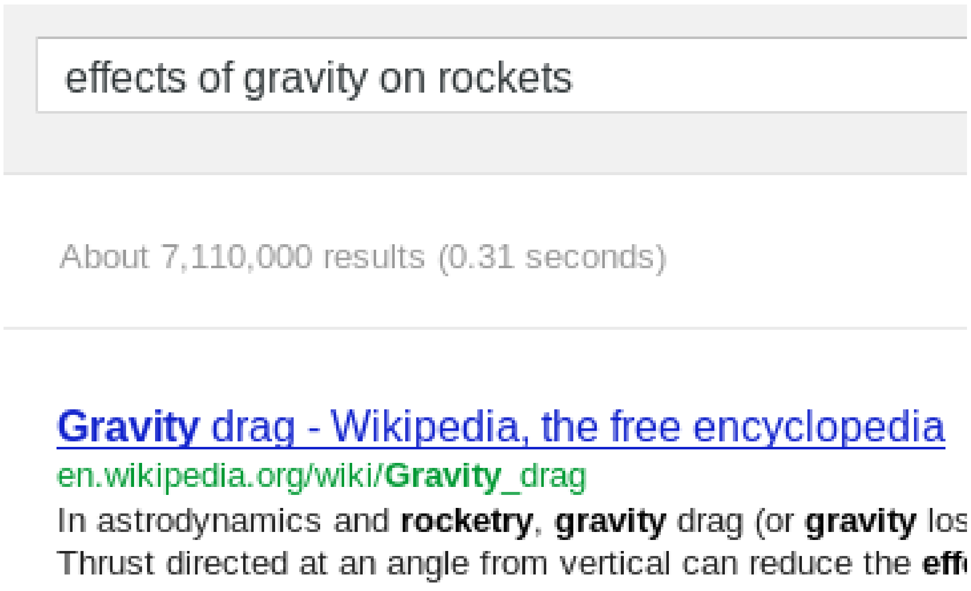
Liakata et al. (2012) describe a system for automatically processing and classifying sentences in a research paper according to the scientific concept they describe (e.g. a sentence can be categorised as a hypothesis, background information, a method or similar)[19]. SAPIENTA (`http://www.sapientaproject.com`) 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)[20].

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[30]. 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)[20].

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)

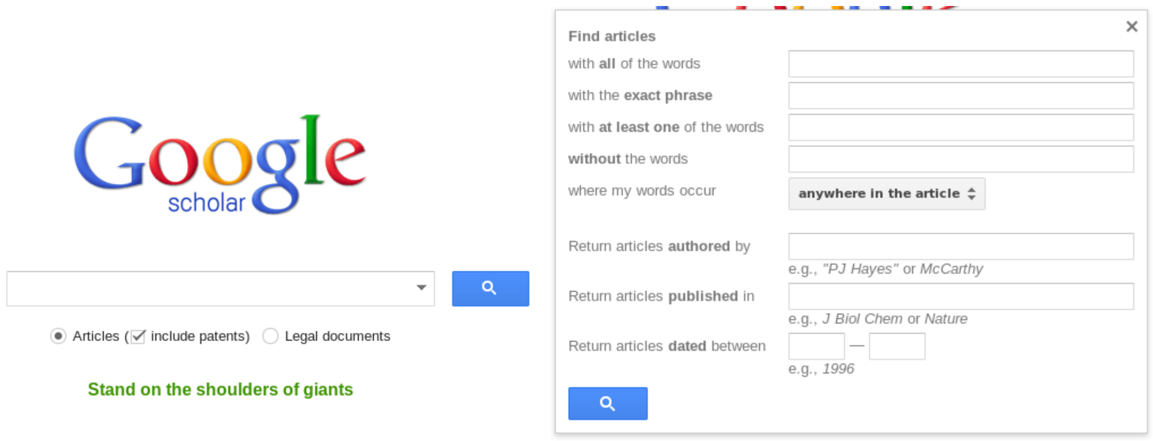
Gravity drag - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Gravity**_drag
In astrodynamics and **rocketry**, **gravity** drag (or **gravity** los
Thrust directed at an angle from vertical can reduce the **eff**

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.



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*

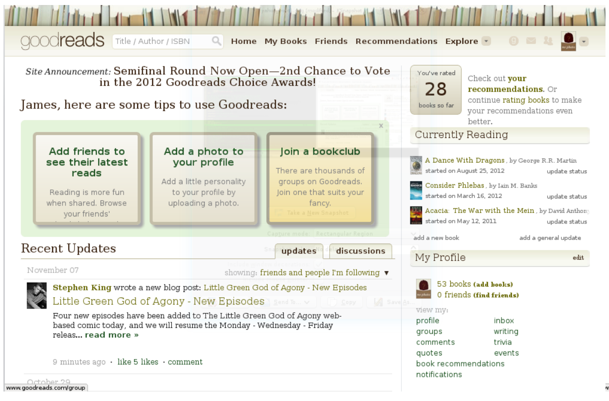
Return articles **dated** between —
e.g., *1996*

(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 ??.


goodreads Title / Author / ISBN Home My Books Friends Recommendations Explore

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

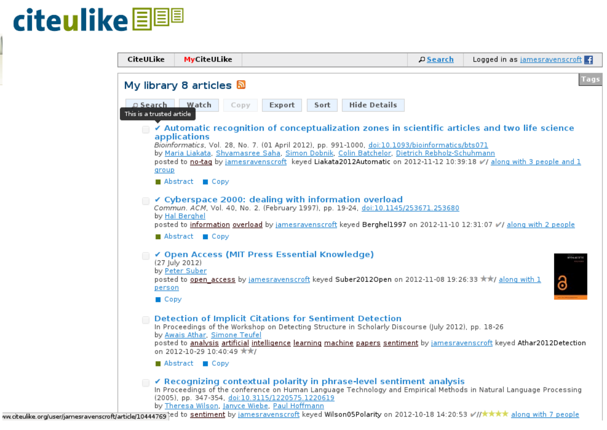
October 29

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

www.goodreads.com/group


CiteULike | MyCiteULike | Search | Logged in as jamesravenscroft

My library 8 articles — Tags
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

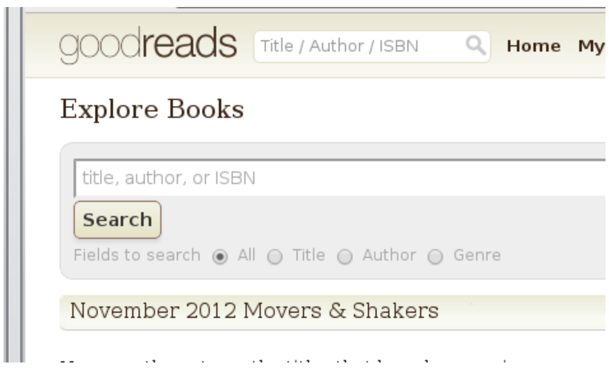
Detection of Implicit Citations for Sentiment Detection
In Proceedings of the Workshop on Detecting Structure in Scholarly Discourse (July 2012), pp. 18-26
by Awais Athar, Simone Teufel
posted to analysis artificial intelligence learning machine papers sentiment by jamesravenscroft keyed Athar2012Detection 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
posted to sentiment by jamesravenscroft keyed Wilson05Polarity on 2012-10-18 14:20:53 ✓//★★★★ along with 7 people

(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.


goodreads Title / Author / ISBN Home My

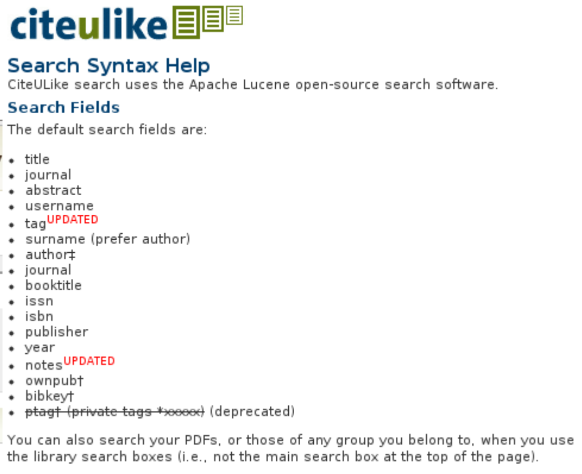
Explore Books

title, author, or ISBN

Search

Fields to search [✓ All] Title Author Genre

November 2012 Movers & Shakers


citeulike

Search Syntax Help

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

Search Fields

The default search fields are:

- title
- journal
- abstract
- username
- tagUPDATED
- surname (prefer author)
- author‡
- journal
- booktitle
- issn
- isbn
- publisher
- year
- notesUPDATED
- 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]. To avoid a learning overhead, it was decided that the programming language should be one that the author is familiar with. This reduced the choice of language down to C, Java and Python.

The C programming language was invented by Kernighan and Ritchie and published in 1978[25]. The language is small and optimised[23] and therefore compiled C programs tend to run extremely fast. Unfortunately, C is designed to be general and lack restriction [?] which is often an advantage for programmers who favour optimised code over excessive error checking. However, this means that debugging C programms can be quite long winded and challenging. Since the NLP aspects of Partridge present an amibitious challenge in themselves, having to debug applications without help from a managed programming environment is not desirable.

The Java programming language on the other hand, does provide memory management and error checking[9] at the cost of program performance. Java is a pseudo-compiled language that is translated into bytecode then interpreted at runtime by a Virtual Machine(VM) on the client computer. Gosling (2000) describes java as a "general-purpose... programming language [that allows] developers to write a program once and then be able to run it everywhere on the internet[14]." Java provides a stable programming environment and error checking, both of which would be advantageous in the development of Partridge. Java's syntax is very verbose and a lot of 'boiler-plate' code must be written before a program can be executed. Therefore Java is not suitable as a rapid prototyping language. Since Partridge's development requires a steep learning curve for the author, a language that allows quick prototyping with minimal programming.

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. For these reasons, Python was selected as the preferred programming language for Partridge development.

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.

It is possible to use a separate software stack Such as Apache web server (`http://httpd.apache.org/`) and PHP (`http://www.php.net`) as a web presentation layer. However, this would involve adding more software to the project and building an interface between the PHP scripts and Partridge's python backend. To prevent unnecessary complication, this approach will be avoided and instead, a Python web toolkit will be used and integrated directly with the backend.

Django (`http://www.djangoproject.com/`) is a Python web framework with integrated database management and templating capabilities [12]. The framework provides automatic management of all aspects of your application's data storage systems unless explicitely disabled. The framework also uses a fairly stricty Model-View-Controller pattern and this is enforced in the structure of the application code. These features may be useful in a general web application such as a news site. Partridge requires a much more complex internal structure to take into account the machine learning and NLP aspects of the system. Having to 'shoehorn' Partridge into a specific structure to make it compatible with Django adds extra overhead to the project and is not desirable.

Flask (`http://flask.pocoo.org`) 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[26]." 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 integrates very simply into existing software using Python annotations*(Ibid)*. This approach means that the framework could be adapted to Partridge very quickly, rather than requiring Partridge to be adapted to the framework. Therefore, Flask has been chosen as the preferred web framework for use in the project.

2.3.2 Natural Language Libraries

. There are several existing libraries to facilitate Natural Language Processing. Many are written for Java [22][10] and are very complex or not well documented. The Natural Language Toolkit (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]". The project is relatively mature in comparison to the above named Java libraries. 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. For this reason NLTK was chosen as the primary accompanying library for the project.

2.4 Prototyping/Current Work

2.4.1 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[17]. 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 [32]. Therefore, to convert PDF documents into an NLP-friendly format, some heuristics must be used to detect the document’s structure[29].

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 in a sane way[29]. The script then used the NLTK library to split the text into sentences and save the document in a CoreSC compatible format for processing by SAPIENTA.

The script had limited success due to the high variation of formatting within 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. PDFX’s main advantage is that it uses a machine learning system to guess PDF document structures more accurately than through the use of simple Heuristics. With an adapted backend that sends PDFs for analysis using PDFX and then splits the results using NLTK, the conversion script now has a much higher success rate and will be used as a preprocessor for adding PDF documents to Partridge.

2.4.2 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.

2.4.3 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[27]. 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 D.

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 D, 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 colleagues 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 C.

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

B System Process Diagrams

C Project Timeline

D Project Wiki

E 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] N. Coffey, “Java for c programmers,” http://www.javamex.com/tutorials/how_to/java_for_c_programmers.shtml retrieved on 13/11/2012, 2008.

Provides some notes on the differences between C and Java

[10] 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

[11] 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

[12] Django Software Foundation, "Django documentation," http://media.readthedocs.org/pdf/django/1.4.X/django.pdf Retrieved on 12/11/2012, October 2012.

A brief manual explaining Django web toolkit

[13] 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.

[14] J. Gosling, *The Java language specification.* Prentice Hall, 2000.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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

[20] 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

[21] 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.

[22] 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

[23] P. Prinz and U. Prinz, *C pocket reference.* O'Reilly Media, Incorporated, 2002.

> A pocket reference guide to the C programming language, introduction discusses why C is popular

[24] 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

[25] D. Ritchie, S. Johnson, M. Lesk, and B. Kernighan, "The c programming language," *Bell Sys. Tech. J*, vol. 57, pp. 1991–2019, 1978.

> The original specification for the C Programming language describes the motivation behind C

[26] 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

[27] 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

[28] 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.

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

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

[30] 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

[31] 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.

[32] 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.

[33] 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

[34] 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.