Text Mining: Automatic Retrieval, Annotation and Visualisation of Clinical Trials Text using Ontology

A dissertation submitted to The University of Manchester for the degree of Master of Science in BioHealth Informatics in the Faculty of Engineering and Physical Sciences

2010

PETER MARK RODDA
SCHOOL OF COMPUTER SCIENCE
Table of Contents

List of Figures ................................................................. 3
List of Tables ................................................................. 4
Declaration ........................................................................ 5
Copyright Statement .......................................................... 6
Abstract ............................................................................. 7
Introduction ........................................................................ 8
Methodology ...................................................................... 11
  Retrieval ........................................................................... 11
    PubMed .......................................................................... 11
    Pubget ........................................................................ 18
    ClinicalTrials.gov .......................................................... 20
  Annotation ......................................................................... 24
    Ontology ......................................................................... 24
    Ontology of Biomedical Investigations ............................... 25
    National Drug File .......................................................... 26
    Foundational Model of Anatomy ....................................... 26
    Gaps in the Ontology - other Clinical Trial specific terms .... 27
  Annotation Services ............................................................ 28
    Terminizer ....................................................................... 28
    NCBO Annotator .............................................................. 30
  Other Services ..................................................................... 32
    Converting PDF to Text .................................................... 34
    Text Cleaning ................................................................... 34
    Sentence Splitting ........................................................... 34

Visualisation ....................................................................... 34
  Ontogrator ......................................................................... 35

Software Design .................................................................. 39
  Overview ............................................................................ 39
  Additional Requirements ..................................................... 41
  Database ............................................................................ 43
  Performance of Views and Locking Contention ...................... 55
  The Box ............................................................................ 55
  Ontology Parser .................................................................. 59
  Ontogrator ......................................................................... 62

Evaluation and Results ......................................................... 68
  Methodology ....................................................................... 68
  Current Approaches .......................................................... 69
    Internal Approach ............................................................ 69
    Outsourced Approach ....................................................... 71
  Ontogrator Approach ........................................................ 72

Discussion .......................................................................... 74

Conclusion and Recommendations ........................................ 77

References ........................................................................... 79

Word Count: 12583
List of Figures

Figure 1: NCBI eSearch result object................................................................. 13
Figure 2: NCBI eFetch result object................................................................. 15
Figure 3: NCBI eFetch result object................................................................. 16
Figure 4: NCBI eFetch result object................................................................. 17
Figure 5: Pubget result object........................................................................... 19
Figure 6: ClinicalTrials.gov initial search query result object......................... 22
Figure 7: ClinicalTrials.gov study object......................................................... 23
Figure 8: Terminizer results object................................................................. 29
Figure 9: NCBO Annotator results object....................................................... 31
Figure 10: Original Ontogrator Database and Table structures........................ 36
Figure 11: Original Implementation of Ontogrator .......................................... 38
Figure 12: Ontogrator and The Box implementation....................................... 41
Figure 13: New Ontogrator database and table structure ................................ 44
Figure 14: Demonstration of a nested set concept.......................................... 47
Figure 15: Views expressing document entries .............................................. 49
Figure 16: Views expressing document "hits" .................................................. 50
Figure 17: Views expressing ontology term data............................................. 51
Figure 18: Views expressing direct parent of ontology terms............................ 52
Figure 19: Views expressing entire hierarchy above ontology terms................ 53
Figure 20: Views expressing ontology term synonym data............................. 54
Figure 21: The Box configuration..................................................................... 57
Figure 22: The Box......................................................................................... 58
Figure 23: Ontology Parser application........................................................... 61
Figure 24: Ontogrator process......................................................................... 62
Figure 25: Ontogrator screenshot................................................................. 63
Figure 26: Searching Ontogrator and the FMA ontology for "heart" ................ 64
Figure 27: Selecting a search ontology term result from within Ontogrator ...... 64
Figure 28: "Phase II" trials matching the term "Cardiac ventricle".................. 65
Figure 29: Double-clicking a specific document in the tabbed list in Ontogrator 66
Figure 30: Summary of Internal Approach.................................................... 70
Figure 31: Summary of Outsourced Approach............................................. 71
Figure 32: Summary of Ontogrator Approach.............................................. 72
List of Tables

Table 1: Custom Clinical Trials Ontology.................................................................27
Table 2: An example of resulting calls to the database when retrieving the hierarchy of an ontology term.................................................................37
Table 3: Advantages and disadvantages of Internal Approach ..................................70
Table 4: Advantages and disadvantages of Outsourced Approach .............................71
Table 5: Advantages and disadvantages of Ontogrator Approach ................................73
Declaration

No portion of the work referred to in this dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.
Copyright Statement

i. Copyright in text of this dissertation rests with the author. Copies (by any process) either in full, or of extracts, may be made only in accordance with instructions given by the author. Details may be obtained from the appropriate Graduate Office. This page must form part of any such copies made. Further copies (by any process) of copies made in accordance with such instructions may not be made without the permission (in writing) of the author.

ii. The ownership of any intellectual property rights which may be described in this dissertation is vested in the University of Manchester, subject to any prior agreement to the contrary, and may not be made available for use by third parties without the written permission of the University, which will prescribe the terms and conditions of any such agreement.

iii. Further information on the conditions under which disclosures and exploitation may take place is available from the Head of the School of Computer Science.
Abstract

The annotation of huge resources of freely available clinical trials data held by PubMed and ClinicalTrials.gov, combined with ontology developed by a number of organisations, provide for an indexing of such data, a process otherwise known as text-mining, in a form that can be easily readable and retrievable by users with various levels of technical ability. This research combines a number of freely available resources of varying functionality to provide a simple, low-cost and effective solution to the presentation of annotated clinical trials data to all levels of user. A pre-existing demonstration web-based user-interface named Ontogrator, featuring faceted browsing functionality, has been adapted and developed further according to user requirements to provide real-world functionality for the purposes of retrieving, annotating and presenting clinical trials documents in a dynamic, indexed manner, thereby enabling quick and effective text-mining and retrieval without the prerequisite for high-end technical ability or in-depth training. Whilst very complex querying of the underlying database are not as accessible with this category of faceted-browser user-interface, this in turn leads to an interface that accentuates ease-of-use to the user by adopting familiar forms-based user-interface design, while at the same time hiding the complexities of the search-engine beneath. The results of the study show that this faceted approach to indexed text can act as an effective preliminary search tool of unstructured data, leading to a more streamlined and focussed complex search using more advanced tools, simplifying the process of defining and refining queries of free text.
Introduction

There is a huge resource of unstructured clinical trials data freely available in the form of free text, so huge to be deemed unmanageable and prohibitively time-consuming by conventional means; that is, the method of reading with the human eye. Retrieving and annotating documents automatically may enable a person to quickly and effectively find single or multiple documents of interest within the huge resource of clinical trial documents held by services such as PubMed and ClinicalTrials.gov.

There are a number of challenges to be overcome in order to develop such an automatic process, such as the automatic retrieval of desired documents, accuracy of prospective indices on which to base annotation of the text within, and a useful and effective means of presentation of the results to end-users possessing differing technical abilities as well as having varying requirements.

Currently, there are a number of text-mining solutions though they are largely proprietary and expensive, and therefore inaccessible. An effective and accessible means to view freely available clinical trials data in an annotated form is the challenge, as well as to present this data using widely familiar methods in order to reduce technical challenges of those interested in searching such data. The hypothesis is that clinical trials data can be accurately and dynamically annotated on-demand using pre-existing, freely-available ontology and presented to multiple end-users simultaneously in a user-interface that emphasises ease-of-use patterns and allows hierarchical browsing and query-building using familiar graphical user-interface design.
The problem arises out of a request from AstraZeneca for a method to quickly surface individual documents located within the abundance of unstructured clinical trials text held in services such as PubMed and ClinicalTrials.gov. AstraZeneca are keen to gain a competitive edge over their rivals by making their own clinical trials cost-effective and efficient as well as quickly highlight and view their competitor’s trials in a specified area of interest. Although many commercial text-mining tools exist, licence terms may be prohibitive and thus lead to a certain bottle-necking of accessibility. Additionally, such tools are complex to use, and require a high level of computing skill to be at their most effective, as well as requiring a significant investment of time to achieve meaningful results. It is hoped that this study shows an alternative and/or complementary means of searching that is effective and highly accessible with relative ease-of-use.

The study aims to address a number of technical challenges. Firstly, it is important to demonstrate the ability to retrieve data, along with meta-data, regarding clinical trial documents of a dynamically specified subject of interest. Secondly, demonstrate the effectiveness and accuracy by which this retrieved data may be annotated or indexed. Thirdly, the annotated or indexed data must be easily and accessibly searched, queried and/or browsed in a hierarchical manner, enabling a user to perform a multiple-pronged “attack” on the data according to their own preferences. Finally, the process by which the data came to be presented to the user must be easily accessible in order for the user to make a final determination of the accuracy of the presented data.

The efficacy of the solution can be measured alongside the current processes of searching at AstraZeneca. An interview with the users of such solutions will enable a direct comparison, whilst also allowing a measurement of ease of use and the relevance of the
results obtained with regards to the search performed. The overall usefulness of the software as a whole, where the solution fits into the current process and also target user or users may then be determined based upon the outcome.
Methodology

The technical aspects of the solution as a whole are divided into three distinct parts; retrieval, annotation and visualisation.

Retrieval

The solution has a requirement to facilitate an automatic and dynamic means of retrieving clinical trials text in order to annotate and present the results. PubMed, Pubget and ClinicalTrials.gov have been identified as rich and free sources of clinical trials text that are routinely updated with new texts. Each of these data sources provides methods of querying their respective databases to return a computer-readable semi-structured result that can be described as an “object” which has attributes of well defined types and are part of a defined structure in which they are contained by or a container of other objects, or both. In order to retrieve the objects and effectively use them it is vital to know their definitions and construction. It should be noted that the objects described in this section are devised and designed by the data source providers themselves, and reports of idiosyncrasies, quirks, errors, unnecessary duplications and other evidence of poor design should be attributed to those that are responsible for their design and definition.

PubMed

PubMed contains abstracts and links to the full-text, and is free to use for all. However, as access to the full-text requires redirection and independent logging in or payment to certain publishers or corporations, this process can not be automated. The realistic
expectation then is that abstracts and certain meta-data may only be obtained from this source.

There are two objects returned from PubMed in XML form. There is an initial search web service named eSearch which essentially returns PubMed identifiers from a specific search pattern (documented at the NCBI eUtils website). As an example, examine the following HTTP post:

```
```

The above will return the first twenty PubMed identifiers for documents where the terms “breast” and “cancer” appear in the text and where the document is marked as having a Publication Type or [pt] of “Clinical Trial or Clinical Trial Phases I-IV”. The resulting object is described in Figure 1. Adding &retmax=n to the post string, we can extend the number of PubMed identifiers returned by the search, where n is the number of identifiers to retrieve.
The idList attribute in Figure 1 is the main object of interest; it is a string array of PubMed identifiers that match the search parameters. However, there are other attributes of interest, notably an item of type Count, RetMax and RetStart; these represent the number of items that match the search, the number of PubMed identifiers contained in this eSearch object and starting point at which to start searching. For example, assuming there are 11,296 results returned from the aforementioned HTTP post, Count will contain 11296, RetMax will contain 20 (the default if not specified), and RetStart will contain 0 (zero, also a default). Upon adding &retmax=35&retstart=21 to the post string, the result will be that Count will still contain 11296, RetMax will contain 35, RetStart will contain 21, and the idList will contain the 35 PubMed identifiers that reside at positions 21-55 in the list of 11,296. In this way it is possible to iterate through each PubMed identifier that is returned by the search, avoiding requesting a prohibitively long list of identifiers from the NCBI servers. TranslationSet and TranslationStack break the query...
into statistical parts, and do not provide any information of particular interest with regards to the context of this document.

The results returned by eSearch, excluding the PubMed identifier, do not contain any document contents or metadata. However, using the result set, it is possible to retrieve individually or as a batch with a complementary NCBI web service, eFetch. This service accepts a collection of PubMed identifiers and returns all stored information for each identifier, including the abstract if one exists. For example, the following HTTP post will return, in XML form, the articles identified by the PubMed identifiers 11850928 and 11482001;


Figures 2, 3 and 4 describe the resulting object from eFetch.
Figure 2: NCBI eFetch result object

Figure 3
Text Mining: Automatic Retrieval, Annotation and Visualisation of Clinical Trials Text using Ontology

Figure 3: NCBI eFetch result object
As is clear from the description of the eFetch result object, there is an abundance of meta-data that accompanies each document, any or all of which is available for further analysis. For the purpose of text-mining, we are mostly interested in the `articleTitle` and `Abstract` objects (containing the title and abstract of the document respectively) within
the Article object. However, certain meta-data is useful for providing further information to the user regarding each article, such as ArticleDate and AuthorList.

It is not the author’s intention to list all the possible parameters for eSearch and eFetch, these are detailed at http://eutils.ncbi.nlm.nih.gov/entrez/query/static/eutils_help.html.

Pubget

Pubget is a commercial complementary web-based product that provides access to both abstracts and full-text of PubMed articles by automating the logging-in process by network location, thus possessing the ability to assess whether your login credentials have the viewing rights to the full-text of documents. In this way, an automated process can be developed that downloads meta-data and abstract as well as the full-text of each article, provided the network location is listed as having access. However, access to the semi-structured form of the data, required for automation, is limited to paying customers costing 2¢ (USD) per search (at the time of writing), with a $1,000 (USD) ceiling per month, amounting to $20 (USD) per 1,000 documents searched, in addition to any and all costs of access to the desired publications. However, due to the inaccessibility of full-text documents for text-mining, this solution may be worth further study and exploration.

Much simpler than PubMed, a single HTTP post is sent to the Pubget server and a collection of objects describing matching articles is returned. A static maximum of ten articles are returned per search, however one can specify a page in the search by adding &page=n where n is the desired page number. For example, the following search will return, in XML form, the first ten articles matching the term “breast cancer”;

http://pubget.com/developer/search?q=breast+cancer
The next search will return the articles ordered 21 to 30 in the list of articles matching the term “breast cancer”;


The returned object contains some meta-data and the abstract, as well as a link to the full article if the network location specifies rights of access. Figure 5 describes the object returned from the Pubget servers. As well as this object, a count is returned detailing the total number of matching article for the specified query. This may be used to calculate the number of pages (of ten) to iterate. More parameters are available for querying the Pubget database and are detailed at http://pubget.com/help/use.

![Figure 5: Pubget result object](image)

Although much less meta-data is provided with the Pubget object, the unique id inside the articlesArticle object is the PubMed identifier, and so it is possible to perform an additional query to PubMed to acquire further information. If the Pubget servers have determined the network location has access to the full article, the attribute pdf contains
the http link to the full article, otherwise it contains the text “no access at your institution”.

**ClinicalTrials.gov**

ClinicalTrials.gov differs greatly from the previous two data sources in two very significant ways; it contains only clinical trials, and stores them in a structured form. In the latter respect, a clinical trial held in ClinicalTrials.gov is not an article *per se* but a record. However, not all data related to a clinical trial can be accessed by an automatic process at the time of writing, as the features of the website are still under development. For instance, study results may be accessed via the HTML version of the website but are currently inaccessible via XML. Nevertheless, ClinicalTrials.gov is free to use, US-government funded and contains an abundance of clinical trials data and meta-data.

Companies and corporations holding clinical trials, as responsible parties, are required in the US to register their clinical trials with ClinicalTrials.gov as of 2007 [1]. Consequently, ClinicalTrials.gov may be rich in terms of more recent clinical trials data, but may not be as comprehensive before this date.

It may be a pertinent question to ask; why text-mine structured data? Although much of the data in a clinical trial record in ClinicalTrials.gov is, in theory, already annotated, much of the values contained within are of entire blocks of free text, and as a result the annotation becomes similar to a headed section of a document, i.e. a grouping. Although the context of the free text is more easily derived, it is still a challenge to annotate and identify the interesting properties of these blocks of free text for quick and effective retrieval.
In terms of the technical aspects of ClinicalTrials.gov, there is some similarity to PubMed, in that there are two requests required to obtain a specific clinical trial from a general query. Firstly, one must post a request to return basic meta-data of trials that contain a specific term or condition, for example, the following HTTP post will return, in XML form, the first twenty Phase II and Phase III trials that contain the term “breast cancer” and are closed to recruitment;


Phase numbers are indexed at zero, and so “phase=0” is synonymous to Phase I, “phase=1” to Phase II, and so on. One can also specify &start=n, where n is the position to start retrieval, allowing the process to iterate through all matches. Additionally one can specify &count=n, where n is the maximum number of trials to retrieve. For example, the following HTTP post will retrieve a maximum of fifteen trials at positions 40-54 of Phase II and Phase III trials that contain the term “breast cancer” and are closed to recruitment;


Again, this is not an exhaustive list of parameters, and these are detailed at http://www.clinicaltrials.org. Figure 6 describes the search query object returned from ClinicalTrials.gov.
The `nct_id` attribute contains the unique identifier to ClinicalTrials.gov of the trial. To retrieve the trial’s comprehensive details in XML form one must post another search using the syntax http://www.clinicaltrials.gov/show/nct_id?displayxml=true, where `nct_id` is the unique ClinicalTrials.gov identifier. This post must be performed for each single trial returned by the original query search. For example, the following HTTP post retrieves the trial with the unique identifier NCT00229723;

http://www.clinicaltrials.gov/show/NCT00229723?displayxml=true

Figure 7 describes the object returned from such a search.
Figure 7: ClinicalTrials.gov study object
There is some duplication in the design of the ClinicalTrials.gov *clinical_study* object, for example *contact* and *contact_backup* are identical in structure, as are *lead_sponsor* and *collaborator*, *primary_outcome* and *secondary_outcome*, and *completion_date* and *primary_completion_date*. These are essentially the same class of objects but have been isolated from each other unnecessarily, and would benefit from an enumeration detailing the type of date, sponsor or contact the object describes, so as to make proper use of class relationships and encapsulation. Additionally, there are a number of *textblock* or *description* attributes that invite free-text to be input, thus preventing a normalisation and subsequent effective analysis of the potentially important data held within. In summary, the object as an entirety does not represent a properly formed class of encapsulated objects, and is more based upon the structure of a freely written article, thus becoming a target for text-mining, rather than direct querying.

**Annotation**

**Ontology**

It is obviously vital to the process that a determination is made as to how best to annotate the text once it has been retrieved. Two unrelated web-based services, named *Terminizer* and *NCBO Annotator* respectively, have been identified as accurately annotating text according to a pre-defined collection of contextual terms, controlled vocabulary, dictionary and thesauri or “ontology”. Both of these services have a simple function; to accept text and search for terms in the text that match a number of specified ontology. To perform this task of annotation, the service must already have knowledge of the specified ontology, and each service supports a variety of ontology. Some ontology
have more time and money invested than others and as a result are richer in terminology, and additionally only some ontology are relevant in the area of clinical trials data. In order to properly demonstrate and maximise the efficacy of text annotation only ontology rich in context and terminology, as well as being contextually appropriate for clinical trials text, have been identified for use in this area. In order to identify appropriate ontologies, the following sentence was input to each the Terminizer and NCBO Annotator, with instructions to annotate against all available ontologies;

“A phase II trial of neoadjuvant gemcitabine, epirubicin, and docetaxel as primary treatment of patients with locally advanced or inflammatory breast cancer”

Any ontology that returned a hit from this sentence was analysed for availability, age, relevance and relative quality and size when compared to other ontology returning hits on the same term. This reduced the list of ontology from just over two-hundred available ontology to three.

**Ontology of Biomedical Investigations**

The Ontology for Biomedical Investigations, otherwise known as the *OBI*, describes biological and clinical investigations, including terms applicable across various biological and technological domains, as well as domain-specific terms. This ontology aims to “support the consistent annotation of biomedical investigations, regardless of the particular field of study” and “represent the design of an investigation, the protocols and instrumentation used, the material used, the data generated and the type analysis performed on it” [2].
National Drug File

The National Drug File Reference Terminology or NDF contains a novel reference hierarchy to describe physiologic effects of drugs [3]. It is a rich resource of information detailing not only drugs and their effects but also associated synonymous terms. According to Rosenbloom et al, this ontology “categorises the physiologic processes [cellular, tissue or organ processes or functions altered by drugs] that drugs induce in bringing about both clinical effects and unintended consequences”. The NDF is the result of an ongoing collaboration between several organisations including The Department of Veterans Affairs (USA), National Library of Medicine (USA), Food and Drug Administration (USA), National Institute of General Medical Sciences (USA), and the National Cancer Institute (USA).

Foundational Model of Anatomy

The Foundational Model of Anatomy or FMA is an ontology of “biomedical informatics ... concerned with the representation of classes or types and relationships necessary for the symbolic representation of the phenotypic structure of the human body” [4]. According to the University of Washington School of Medicine (USA) [4], “the FMA is a domain ontology that represents a coherent body of explicit declarative knowledge about human anatomy” of which the “ontological framework can be applied and extended to all other species”. Essentially, it is a rich source of contextual, hierarchical data detailing human anatomy, along with synonymous terms.
Gaps in the Ontology - other Clinical Trial specific terms

Due to the lack of a specific clinical trials ontology that is term-rich, a simple custom ontology was created in consultation with AstraZeneca. See Table 1 for a full description of the resulting, admittedly very simple, ad hoc ontology.

<table>
<thead>
<tr>
<th>Parent</th>
<th>Term</th>
<th>Synonyms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phase</td>
<td>Phase I</td>
<td>Phase 1</td>
</tr>
</tbody>
</table>
|          | Phase II  | Phase 2
|          |           | Phase IIa
|          |           | Phase IIb               |
|          | Phase II/III | Phase 2-3
|          |           | Phase 2/3
|          |           | Phase 2/Phase 3       |
|          |           | Phase II-III           |
|          |           | Phase II/Phase III     |
|          | Phase III | Phase 3                |
|          | Phase IV  | Phase 4                |
| Blinding | Open      | Open label
|          |           | Open-label             |
|          |           | Blinded                |
|          | Blind     | Single-blind           |
|          |           | Single-blinded         |
|          |           | Single blinded         |
|          | Double-blind | Double-blinded       |
|          |           | Double blind           |
|          |           | Double blinded         |
|          | Triple-blind | Triple-blinded        |
|          |           | Triple blind           |
|          |           | Triple blinded         |
| None     | Randomization | Random               |
|          |           | Randomized            |
|          |           | Randomised            |
|          |           | Randomisation         |

This allows the entry of terms that are fundamental markers for the indexing of clinical trial documents, but which are absent in the available ontology at the time of writing. The absence of these fundamental clinical trial terms would undoubtedly negatively affect the
evaluation of the final product, and may have caused an experienced user to lose confidence and subsequently cause any evaluation to fail unnecessarily.

**Annotation Services**

In order to evaluate the major text-mining solutions whilst also allowing the final solution to remain flexible with regards to its dependent services, it was decided to use both Terminizer and NCBO Annotator services. The Terminizer currently does not support annotation using the National Drug File, and therefore all text was annotated against the NDF using NCBO Annotator. The Terminizer was used to annotate text against the OBI and FMA. An *ad hoc* in-process development was identified as being necessary to annotate the text according to the Custom Clinical Trials ontology.

**Terminizer**

The Terminizer is a web-based service made freely available by the NERC Environmental Bioinformatics Centre. The service accepts a number of parameters which direct it in what text to annotate, and using which ontology or ontologies. For example, the following HTTP post directs the Terminizer to annotate the text “Phase II trial studying patients with breast cancer” against the OBI and FMA ontologies;

```
```

The full instructions and list of available parameters are documented by NERC at [http://www.terminizer.org/help/TerminizerWebServiceDetails.html](http://www.terminizer.org/help/TerminizerWebServiceDetails.html). On sending a request to the Terminizer, an object is returned serialised in XML form. Figure 8 describes this object.
The *Token* object contains each word of the text passed, and assigns a numerical index to the word; attribute *index*. This is referenced in the *MatchedTermListMatchedTerm* object by the attribute *tokenIndices*, in order to relay information regarding which entry in the ontology matched against which word. Where more than one word matched the ontology entry, *tokenIndices* contains each token index separated by a comma. The attribute *accession* in the object *MatchedTermListMatchedTerm* contains the identifier for the matched ontology entry.
**NCBO Annotator**

The NCBO Annotator is also a freely available web-based service that accepts a number of parameters that control what text is annotated against which ontology and in what manner of term-matching. The following HTTP post directs the NCBO Annotator to annotate the text “Phase II trial studying patients with breast cancer” against the NDF and OBI ontologies:

```
http://rest.bioontology.org/obs/annotator?longestOnly=true&wholeWordOnly=true&scored=true&ontologiesToExpand=40402,40832&ontologiesToKeepInResult=40402,40832&textToAnnotate=Phase+II+trial+studying+patients+with+breast+cancer&withDefaultStopWords=true&format=xml&levelMax=0&mappingTypes=null
```

The numbers 40402 and 40832 indicate the proprietary NCBO identifiers of NDF and OBI ontologies respectively, and other parameters control how the text is analysed. For instance, the parameter `&wholeWordOnly` specifies that the word must be listed as a whole word to match, e.g. the “Ph” in “Phase” may return a match for the Ph measure of acidity if `&wholeWordOnly` is set to false, and so false positive matches are increased.

Documentation for all parameters for the NCBO Annotator web service can be found at [http://www.bioontology.org/wiki/index.php/Annotator_User_Guide](http://www.bioontology.org/wiki/index.php/Annotator_User_Guide). Including the parameter `&format=xml` instructs the NCBO Annotator to return the result as a XML-serialised object, which is described in Figure 9.
Offering a similar object to the Terminizer results object, the NCBO Annotator reflects all the parameters back at the caller in the `successDataAnnotatorResultBeanParameters` object. Each match is returned within the `successDataAnnotatorResultsBeanAnnotationBean` object, which provides a concept...
(successDataAnnotatorResultsBeanAnnotationBeanConcept) and a context
(successDataAnnotatorResultsBeanAnnotationBeanContext). The concept object
describes the ontology entry that has been matched against, and the context object
describes the text that was matched. The localConceptId and localOntologyId
attributes together in the concept object describe the matched ontology entry and the
containing ontology respectively. The from and to attributes in the context object identify
the indices of the text in textToAnnotate attribute (contained within
successDataAnnotatorResultBeanParameters) that resulted in the match.

Other Services
Due to the raw computing power required to index occurrences of terms in text, as well
as the idiosyncrasies and rules of using HTTP requests and processing XML, it is necessary
to perform certain operations on the text before it is sent to Terminizer or NCBO
Annotator. This includes;

- Converting binary files such as PDF files that contain text into arrays of characters
  or strings so that string-related functions can be executed upon them
- “Un-escaping” and “escaping” certain characters in text that are used as
  syntactical characters
  - For example, ampersand & which must be converted \& and vice versa
- Removing diacritics from characters
  - For example, converting é to e, á to a, or ú to u, causing the full word to
    remain intact whilst removing certain forms that would confuse or hinder
certain computed operations.
- Converting superscript numbers to normal character set
  - As an illustration, merely stripping out a superscript character such as the $^2$ in $CO^2$ because it is illegal in XML or an HTTP post would result in a match of “CO” to “Carbon Monoxide”. However, converting the $^2$ to 2 retains the full term CO2 and results in an accurate match of “CO2” to “Carbon Dioxide”.

- Stripping out all otherwise unimportant, illegal characters

- Reducing very large strings to “bite-sized” portions so as not to overload services, in turn reducing bottlenecks in data throughput and request/response timeouts

There are a number of freely available web services that allow the above points to be resolved simply by calling the relevant web service for the desired effect. To reduce the timescale in which to develop a solution the following services were proposed as useful, though the use of such services increase the total time taken to process a document due to an increase in communications over relatively low-speed internet connections. It is recommended that steps are taken to rewrite these processes as in-process to decrease unnecessary communications and thereby decrease overall processing time, as well as decreasing dependency on the uptime of services over which one has no control.
Converting PDF to Text

The PdfToTextService [5] web service provides a method to return a byte representation of text from a byte representation of a PDF file;

```
public abstract byte[] PdfToText.pdfToText(byte[])
```

Text Cleaning

The TextCleanerService [6] web service provides two methods to remove all XML-invalid and/or non-ASCII characters from a provided string;

```
public abstract String TextCleaner.cleanText(byte[])
public abstract String TextCleaner.cleanTextASCII(byte[])  
```

Sentence Splitting

The SentenceSplitterService [7] web service provides a method to split a string into sentences (or “bite-sized” chunks);

```
public abstract List SentenceSplitter.splitIntoSentences(String)
```

Visualisation

It is proposed the results of the annotation be presented using faceted browsing – a technique for presenting multi-dimensional data and used widely in web-based eCommerce such as Amazon and eBay. Navigation is performed via facets, each of which segments the data in one dimension, for example genre to filter music, or phase to distinguish between clinical trials. Facets can range in complexity from lists of words and ranges of numbers to spatial regions and hierarchical structures such as taxonomies, concept maps and ontologies. Harnessing a technique that is already widely used and known increases the likelihood of an interface being intuitive to a variety of categories of users without the requirement for in-depth training.
Ontogrator

The Ontogrator is a web-based faceted browser developed and funded as a proof-of-concept by the NERC Environmental Bioinformatics Centre. According to the unpublished Ontogrator literature it intended to "to support semantically enhanced data exploration over federated databases utilising faceted browsing", and is mainly concerned with biological data and ontology, along with geographical data. However, due to the Ontogrator being a "dumb" interface that simply extracts and presents data from a MySQL database, with some development it may be unbound from this purpose and bound to clinical trials data, as well as, in theory, any other purpose that may be required. This enables a certain "tweaking" of the Ontogrator design in order to minimise the effort and time required in visualising clinical trials textual data based on text-mining via the chosen ontology.

Ontogrator is written using a combination of JavaScript, PHP and MySQL-flavoured SQL. The source code and overall design was created specifically with biological, geographical and environment data in mind from specific data sources. The result is a design where variables and tables are designed for a specific type of data. For instance, document-ontology hits (or successful annotations) for the Silva RNA database against the Environment Ontology (or EnvO) were expected to be stored in the `SILVA_ENVO_hit` table in the MySQL database. Unfortunately this type of non-generic design makes it very difficult to maintain or alter, e.g. replace a current ontology or introduce a new data source. Furthermore, no views or stored procedures were defined, meaning all SQL code was contained within the PHP source code; the PHP being coupled with the database logic makes the task of defining behaviour of the different processes extremely difficult,
leading to a design that is virtually impossible to build upon or improve. Figure 10 describes the structure of the original Ontogrator database.

Figure 10: Original Ontogrator Database and Table structures
It is clear from the database model that the rigid structure in place was very much explicit as to the data sources, ontologies and types of document that were expected to be mined and visualised. It was therefore imperative that this rigid structure be made more flexible in order to mine other categories of data using ontologies of our own choosing. It should also be noted that the ontology is stored on “name” and “parent” basis, meaning that when the full hierarchy of a term is required, multiple calls to the database are required, e.g. to retrieve the “parent” of a “parent” of a “parent” of a “name”, one must make a call for each “parent”. Table 2 describes this process using the example “retrieve from the database the full hierarchy of the ontology term heart in the FMA ontology”, resulting in a total of 8 calls to the database.

<table>
<thead>
<tr>
<th>Database call</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Retrieve parent of “Heart”</td>
<td>Organ with cavitated organ parts</td>
</tr>
<tr>
<td>2 Retrieve parent of “Organ with cavitated organ parts”</td>
<td>Cavitated organ</td>
</tr>
<tr>
<td>3 Retrieve parent of “Cavitated organ”</td>
<td>Organ</td>
</tr>
<tr>
<td>4 Retrieve parent of “Organ”</td>
<td>Anatomical structure</td>
</tr>
<tr>
<td>5 Retrieve parent of “Anatomical structure”</td>
<td>Material anatomical entity</td>
</tr>
<tr>
<td>6 Retrieve parent of “Material anatomical entity”</td>
<td>Physical anatomical entity</td>
</tr>
<tr>
<td>7 Retrieve parent of “Physical anatomical entity”</td>
<td>Anatomical entity</td>
</tr>
<tr>
<td>8 Retrieve parent of “Anatomical entity”</td>
<td>None</td>
</tr>
</tbody>
</table>

Additionally, as the Ontogrator was designed for demonstration purposes rather than as part of a practical requirement, the data loading scripts simply loaded static data into the database, rather than dynamically querying data sources such as PubMed for specific documents, e.g. recently added documents, or documents with certain meta-data. To perform this data-loading task, Ontogrator used a proprietary web-based server named
Omixed, and it was decided that due to the subsequent learning-curve as well as the unsuitable design of the loading process that Ontogrator would be decoupled from Omixed and a separate process designed to load and annotate data from the data sources (PubMed, Pubget and ClinicalTrials.gov), with the intention to store the results directly into the MySQL database. This would have the added benefit of being able to utilise the NCBO Annotator, as the original implementation was specific and exclusive to Terminizer. Figure 11 describes the original implementation of Ontogrator.

In summary, it was decided that the software and underlying database be redesigned to allow a more generic nature, for effective further development, maintenance and configuration.

![Figure 11: Original Implementation of Ontogrator](image-url)
Software Design

Overview

The design of the software is divided into four distinct portions;

- **Database logic**
  - independence from source code for flexibility and proper delegation of processes

- **“The Box”**
  - Direct and process user queries to the data sources
  - Download and prepare documents for mining
  - Annotate text using the Terminizer and NCBO Annotator services
  - Persist the results to the database

- **“Ontology Parser”**
  - Parse given ontology and persist to database for use by Ontogrator and The Box
  - Description of full ontology vital as neither of the annotating services provide hierarchy for coded annotations
  - Acts as part of the initial configuration for the solution i.e. must be part of the pre-use setup

- **“Ontogrator”**
  - Visualise the annotated results in an intuitive graphical user interface,
    giving easy access to required information regarding all annotated documents
In essence this is a tiered structure, where there are three layers, the *database* tier at the bottom, the *application* tier in the middle and the *client* tier at the top;

- The *database tier* consists of the database management system (DBMS), which allows definition, construction and manipulation of databases and in this case is MySQL, and the physical database, which is simply a collection of relational data. The responsibility lies here for the management, storage and retrieval of data as well as the ability to simultaneous access data securely, whilst ensuring integrity.

- The *application tier* is essentially a means of communication between the database tier and the client tier, in that it interacts with the database tier according to requests from the client tier. Accordingly, it consists of most of the application logic, is generally the most complex part of the solution as an entirety, and typically accounts for the bulk of developmental effort.

- The *client tier* is the part of the application that typically interacts with a human being. This is also sometimes referred to as a “thin” client as, ideally, very little application logic resides in the client tier, and thus this tier is intentionally limited in scope and nature. The client tier should be very much manipulated by the application tier, through the use of public interfaces, thus giving the flexibility of maintaining or replacing an application tier with no indication of change to the end-user.

The overall summary design of the intended solution is described in Figure 12.
Additional Requirements

Certain additional requirements became apparent from interviews with AstraZeneca. These requirements originated with some use of the original Ontogrator and also their “wish-list” from gaps in their current solutions. As a result some slight changes to the original database tables were required, in order to surface and visualise the data on-
demand from the client tier, i.e. Ontogrator, as well as modifications to the Ontogrator user interface itself.

- Identify links between different sources
  - Investigate whether associated alternative ID’s are available in the meta-data or free text of a document, in effect centralising the document and providing links to different sources that hold related information regarding that document.

- Identify and visualise the reasons why a document has been annotated
  - The keywords that have been matched against a term, the context of those keywords within the document, as well as information regarding the ontology and its place in the hierarchy.

- The ability to select and filter through more than one term in the same ontology
  - For instance, “Phase II” and “Double-blind” may exist in the same ontology; it is imperative that one is allowed to filter documents that hit against both of these terms, i.e. “give me all Phase II, double-blind trials”.

- Make use of synonyms in an ontology
  - Some ontology specify multiple synonyms for terms, e.g. a search for trials involving “paracetamol” should also return trials involving “acetaminophen”, if the ontology makes it possible to do so.

- A means should be available to display all documents from all sources
  - Where it is not necessary to filter via data source, documents from all data sources should be displayed cumulatively
• Text searching ontology terms should be displayed as part of the hierarchy to give full context
  ○ When searching an ontology with synonyms or otherwise, once a resulting term is selected, the entire hierarchy should be displayed to allow the user to derive a context, e.g. the term “ventricle” may refer to the brain or heart – the context is required for a full understanding.

Database

The database is redesigned to be more normalised, more generic in its terms, and less complex in its design. Non-generic identifiers of tables are changed to more descriptive and generic identifiers. However, because the client tier directly addresses specific table identifiers and columns, within the database logic a layer of views was developed to sit on top of the underlying tables and columns, essentially hiding them from and allowing for less mandatory changes in the client tier. For instance all of the \textit{xxx\_hit} tables, which contain details of a match against ontology and are each identical to the other, are merged into one table named \textit{Hits}. Previously, there is one \textit{xxx\_hit} table for each ontology, multiplied by the number of available data sources. To properly allow one to retrieve a “hit” for a given ontology and data source, the new \textit{Hits} table must then contain additional columns describing the ontology and data source associated with each individual “hit”. Several views can then be placed on top of the \textit{Hits} table in order to hide this change from the client tier whilst retaining the integrity of the database structure. In order to make identifiers more generic, the names attributed to these views indicate the identifier, from left to right incrementally, of the “Pane” and “Tab” in which the ontology or data source appears in the client tier. For example, as ontology sit in “panes” and data
sources reside in “tabs”, pane “1” may logically identify the left-most pane and tab “1” the left-most tab. So, instead of identifying ontologies by their name, a generic positional index is attributed. Figure 13 describes the resulting database and table structure.

![Database and Table Structure Diagram]

Figure 13: New Ontogrator database and table structure
The DocumentEntries table replaces the xxx_entry tables and the Hits table replaces the xxx_hit tables. As all the documents are assumed to be clinical trials, the same meta-data will be displayed for each, and so the Links, Title and Authors columns were added to the DocumentEntries table in order to accommodate this data. The Links column was envisaged to hold any link to the document from another data source, such as a link to a ClinicalTrials.gov trial in PubMed, or vice versa. In order to retain the functionality of having different meta-data for each “tab” or data source, the Ontogrator was altered to enable the display of any given column for any data source, with any given title, e.g. display the values of column Authors with the title Officials for the data source ClinicalTrials.gov, or even hide the column completely from view.

The TextMinerSearches table allows the user or administrator to input searches (the column SearchString) for The Box to start processing against any specified data source, as indicated by the column ProcessType. The StartAt and EndAt columns instruct The Box to start and finish mining at explicit positions in the search results of the data source, and Active having a value of 1 instructs The Box to start the search immediately (which will set Active immediately to zero on beginning the query).

The Ontology group of tables hold the full ontology as well as a subset of the ontology. The subset contains only those ontology entries that have resulted in a hit against a document. It is this OntologyEntriesSubset table, the structure of which is identical to the OntologyEntries table, which controls what is displayed in the Ontogrator; it would be unnecessarily time-consuming as well as counter-intuitive to list ontology terms that do not have any hits against them. The OntologySubsetQueue table is meant to assist in the maintenance of the OntologyEntriesSubset table by indicating which ontologies
(and subsequently, relations) require population into the `OntologyEntriesSubset` table. A frequently and automatically executed stored procedure will lift these entries from the queue and populate the subset as required. The `OntologySynonyms` table lists all and any synonyms for each entry in `OntologyEntries`. The `OntologyRelations` table describes the hierarchy of the ontology or relationships between ontology entries, in order to quickly retrieve all relations of a specific entry, e.g. its parents, grandparents and so on all the way to the root, or its children, grandchildren and so on all the way to the last leaf. The limitations of the current method were discussed earlier, and so an alternative method is used, and that is the “nested set” method. This method employs the indexing of the positions of each node in the tree using `left` and `right` values, as well as the `depth` of the node. Each index is assigned by naturally and recursively navigating through the ontology hierarchy, incrementing the index with each assignment. The `left` value is assigned the index value on entering the node, and the `right` value is assigned upon leaving the node to pass up to the parent or sideways to a sibling. The `depth` value is incremented with each recursive step into a recursive call, and decremented with every step upwards out of a recursive call. Figure 14 describes the results of this process for a very small subset of an ontology describing the context of carbon monoxide.
The yellow box values are the depth and the black boxes contain the left and right values respectively. This method allows all parents or children nodes to be retrieved in a single call to the database. For instance, referencing the figure, retrieving all nodes with a left value between 17 and 26 will return all nodes beneath “Electrolytes”, regardless of
their position. Referencing the figure again, one might retrieve all nodes where the left value 19 is between their own left and right values; this would return all parents, grandparents, and so on, of “Anions”.

As can also be seen from the Figure 14, the ontology entry “Carbon Monoxide” has multiple parents, and therefore more than one position in the hierarchy, so while “Carbon Monoxide” will only exist as an ontology entry once in the table OntologyEntries, there will be several entries cross referencing this entry in OntologyRelations detailing each position in the hierarchy.

By completely redesigning the database tier in this way, the client tier will now be dysfunctional. In order to achieve a minimal effort of development in the limited time available, whilst still allowing the Ontogrator-specific application tier (the process responsible for serving data to the client tier) to behave appropriately as before, a layer of database views can be placed on top of the table structures to create the illusion of the table structures being the same as before. A database view is essentially a stored SQL query that functions as a virtual table, and so is queried exactly as if it were a table. This collection of views act as an interface enabling the Ontogrator-specific application tier to be effectively “plugged into” the database tier. These views are described in Figures 15-20.
Figure 15 illustrates how the client tier receives data from the application tier regarding document entries. The views split out and form the data for specific data sources (“tabs”) in the structure that the client tier is expecting.

The next figure, Figure 16, illustrates how the client tier receives data from the application tier regarding ontology term matches or “hits” against documents. The views split out and form the data for the specific ontology (“panes”) and data sources (“tabs”) in the structure that the client tier is expecting.
Annotations of Document against Ontology

Views expressing document "hits" on specific Ontology vs Data Source

Figure 16: Views expressing document "hits"
Figure 17 illustrates how the client tier receives data from the application tier regarding specific ontology terms. The views split out and form the data for the specific ontology (“panes”) in the structure that the client tier is expecting.

![Diagram of ontology entries subset and views expressing ontology term data](image-url)
Figure 18 illustrates how the client tier receives data from the application tier regarding the direct parent of specific ontology terms. The views split out and form the data for the specific ontology ("panes") in the structure that the client tier is expecting.
Figure 19 illustrates how the client tier receives data from the application tier regarding the entire hierarchy above specific ontology terms. The views split out and form the data for the specific ontology (“panes”) in the structure that the client tier is expecting.

![Ontology subset and relations diagram]

**Figure 19**: Views expressing entire hierarchy above ontology terms
Finally, Figure 20 illustrates how the client tier receives data from the application tier regarding the synonyms of specific ontology terms. The views split out and form the data for the specific ontology (“panes”) in the structure that the client tier is expecting.
Performance of Views and Locking Contention

There is a possibility that by using views there will be a performance hit due to the added complexity in compiling an efficient query evaluation plan (QEP) inside the database tier. One of the aims of any future development should study the benefits of changing the source code of the Ontogrator-specific application tier to identify the tables directly rather than accessing through views. Another option is to “materialise” the views, that is, to transform the views into physical tables, much like a data warehousing process. This can be done periodically at a chosen time, for instance every 24 hours at 0200 hours. The “data warehousing” method, whilst eliminating any performance hit (should one be proved to exist), would mean the visualised data would only be updated once a day (using the aforementioned example). An added benefit in this method is that it would also eliminate any locking contention issues in the database tier whilst perusing data using Ontogrator and simultaneously performing text-mining processes in The Box. Locking contention issues occur when rows, pages or even whole tables of data are locked by a process that is reading and/or writing data, whilst another process may be sitting “in limbo” waiting for that process to release the lock so it can continue with its work. These issues can be dealt with at the application tier by configuring the database to only lock at a fine granular level (expensive in terms of performance) or to not lock at all (expensive in terms of integrity) in different circumstances, resulting in a certain amount of potentially time-consuming fine-tuning and eventual compromise.

The Box

The Box is designed to accept queries from the user and then gather data from those queries, storing the results from annotation services. A typical query will specify which
data source to query (PubMed, Pubget or ClinicalTrials.gov), and the syntax of the query. The syntax of the query, as discussed previously, differs for each data source due to the differing designers as well as the different meta-data available from each source. The Box is required to perform the user-specified query and iterate through each resulting document, storing each annotation returned from the annotation services (Terminizer and NCBO Annotator) in a way that is later retrievable by Ontogrator for presentation to the user. The Box is configured via an XML file, which specifies the characteristics and behaviour of process, including:

- Database connection settings
- Number of threads to instantiate
- The type of process of the thread
  - PubMed
  - Pubget
  - ClinicalTrials.gov
- Text-mining behaviour
  - Types of text-mining process
    - Terminizer
    - NCBO Annotator
  - Ontologies attributed to each text-mining process
    - OBI
    - NDF
    - FMA

Figure 21 describes the configuration file object.
Any number of threads of any valid type may be instantiated, with the exception of OntologySubsetWorker which may only have one thread. For instance, 10 threads of type PubMed may be simultaneously instantiated, executed and configured to perform the same search query returning 10,000 results with one thread starting at positional index zero, one thread at position 1,000, another at position 2,000, and so on. As the...
threads are running in parallel, this reduces the amount of time required to search through the 10,000 documents to the time it takes the thread taking the longest time to iterate through 1,000 documents, as opposed to the time it takes one thread to iterate through 10,000 documents. As mentioned previously, the search queries are set in the TextMinerSearches database table and are picked up by each thread on a first-come-first-served basis. The single OntologySubsetWorker thread is responsible for maintaining the OntologyEntriesSubset table. Any ontology entries that have had a “hit” against them are populated into the ontology subset so they can be available for selection in Ontogrator. The process of The Box is described in Figure 22.
Each instantiated data source thread iterates through documents from a data source determined by its process type according to a user specified search query. Once a document has been retrieved, the text is cleaned, split into sentences and then sent to the Terminizer and NCBO Annotator text mining services. Any ontology matches or “hits” returned are then persisted into the MySQL database. Each individual process returns to a polling state (looking for unprocessed search queries) when there are no more documents to retrieve from the search query or when the current positional index of the last document retrieved is equal to or exceeds the specified positional index. The Box executes as a Microsoft Windows service and so is login-independent. It was developed using Visual Studio 2008 in the C# language with .NET framework 3.5.

**Ontology Parser**

The text mining services available by design specify only the identifier for each ontology term matched, for example, if the word “heart” is matched against the FMA ontology, the returned identifier is 7088, not the actual term “heart”. Additionally, this design does not supply the hierarchy and therefore context of an ontology term. In order to have full knowledge of each ontology term and its place or places in the hierarchy, and allow serving of this data to the client tier, access to the entire ontology is required by The Box. With this in mind a very simple Windows Form application (developed in C# .NET Framework 3.5 using Visual Studio 2008) has been designed to parse the full ontology into the database. Currently this application supports OWL and OBO files, which are current standards for storing ontology, but can easily be altered to parse any text file, proprietary or otherwise, into the database. For instance, SNOMED CT [8] contains
procedures and medical terms, in essence a clinical and medical ontology, and is currently distributed in a tab-delimited, relational format but very little developmental effort is required to support these files. However, the cooperation would be needed by those responsible for the maintenance of Terminizer or NCBO Annotator, as one of these services would also be required to parse these files into their own databases in order to perform ontology matches against text using ontology that is currently not supported.

The Ontology Parser accepts a filename and imports that file to a specified “pane” into the specified database. Figure 23 shows a screenshot of this application.
The “Custom” button imports the Custom Clinical Trials ontology described previously.

The design of this application is based upon there being only four panes. In the future it may be more appropriate to allow the user to specify the pane rather than have a separate button and filename text box for each pane. This application is categorised as being part of the setup and configuration of the entire process and as such it should not be used while The Box is running as the effects would be unpredictable, and would certainly cause a running Ontogrator instance to display spurious results, or even no results at all.
Ontogrator

Ontogrator is a user interface written in JavaScript and PHP running within a standard web browser. The JavaScript side of Ontogrator sends asynchronous commands to the web server, on which runs a PHP server, that in turn sends requests the database. The PHP server forms the results from the database into a simple text structure that can be read by the Ontogrator instance on the workstation, and these results are sent as an event back to the JavaScript running on the user workstation, which visualises the results. Figure 24 describes the processes by which the Ontogrator works.

![Figure 24: Ontogrator process](image-url)
The Ontogrator user interface allows the exploration of documents from federated sources using faceted browsing based upon ontology. Faceted browsing is a technique for presenting views into complex, multi-dimensional data, allowing the user to select filters or indices on which to combine or exclude results. Additionally, Ontogrator allows the hierarchical navigation of each individual ontology, as well as searching based on brief text entry. Figure 25 shows a screenshot of Ontogrator.

As the screenshot shows, Ontogrator has the indices or ontology placed in four separate panes. Documents that match the selected indices or filters are displayed in a separate tabbed list along the bottom of the screen. One can navigate through a hierarchy within a pane or type a search term within the pane’s text box, marked by “Search...”, as shown in Figure 26.
This example shows that upon entering the word “heart” into the FMA ontology pane, a drop down list of terms that match this text will appear. Note that terms beginning with “cardiac” also appear as they are synonymous with the term “heart”.

Upon selecting “Cardiac ventricle” from the list, the term is displayed in its context, i.e. its place in the hierarchy. When a term has more than one place in the hierarchy, only the first place returned is shown.

It is also possible to “drill down” or navigate the hierarchy without the use of the ontology text search.

At this point, the data source tabs will only list documents that match for “cardiac ventricle”, and in this case we can see from the number in brackets after the term there are 7 hits in total. It is now possible to further filter the list by selecting only Phase II trials.
that match “cardiac ventricle”. Moving to the “Miscellaneous” pane, opening the “Phase” node and then selecting “Phase II”, only documents are displayed that match both “cardiac ventricle” and “Phase II”. Figure 28 illustrates this.

It is also possible to select terms from the same ontology – this results in an “AND” search of each term searched, for example, retrieve “Phase II” trials that are also “Open label”, both terms being present in the same ontology pane. Any future development should allow the inclusion of dynamic selection of logical operators such as “AND”, “OR” or even “XOR”, in order to add far more power and flexibility to the querying process of the current implementation.

One can select each tab to list the documents from that data source only. On clicking on the Document ID of a document in the list, the web browser will open a new tab or window and go directly to the underlying URL in order to peruse the document at the
source. Additionally, one can double-click on an individual document in the list to display information regarding why the document was matched to the selected term(s) as well as any other term the document may have matched to. Figure 29 shows the result of double-clicking a document in the list.

![Information for "PubMed" item 7715910](image)

**Figure 29: Double-clicking a specific document in the tabbed list in Ontogrator**

This “information window” displays the full hierarchy of all ontology terms against which the document has matched. The different ontology are separated by tabs within the window. As well as the keyword(s) in the document that matched against the specific ontology term, the full sentence is displayed to allow the user to gain some context in which the keyword was used within the document. The occurrences of the keyword(s) are
underlined in the sentence. Terms highlighted in red are those that are selected in the ontology panes, and are always listed in priority at the top of the window. This window allows the user to make a final reasoned decision as to whether the selected document is of interest.

Due to time constraints, it was not possible to develop an “export” function that would allow the user to export the list of documents and related data in the tabs to an application such as Excel, for further analysis and presentation of the subset of information. For Ontogrator to have a practical application this would be a very important feature.
Evaluation and Results

Methodology

The Ontogrator was made available on the internet at http://ontogrator.org/clinicaltrials in order to allow access to an Ontogrator instance from the AstraZeneca offices. It was agreed to base queries on Phase II and Phase III clinical trials with reference to “breast cancer”, allowing the retrieval of a specific set of data into The Box to use in a comparison with currently implemented methods and processes at AstraZeneca. This data load took around 12 days to achieve, mining and persisting a total of 13,938 documents, made up of 12,611 PubMed documents and 1,327 ClinicalTrials.gov records.

Subsequently, a “typical query” based on this subset of data was identified in discussion with the Biomedical Informatics department at AstraZeneca, i.e. a typical question in a typical form that may be asked by a customer of the department’s services. This “typical query” was a request for;

“Published metastatic breast cancer studies from January 1999 to July 2009, including all treatment types, limited to studies in patients receiving 2+ lines of therapy (i.e. not including first-line studies) & limited to Phase Ila and Phase III studies only”

The intention was to make a direct comparison of the Ontogrator/Box solution with the current approaches to evaluate whether the solution has potential to be included in, or even replace aspects of the current process.
The query was processed using their current approaches, and each approach was documented along with the results. Where an approach was too expensive or time-consuming to perform directly (such as an outsourced approach) the process was described and documented in order to obtain an accurate review of the inner workings of that process and its associated advantages and disadvantages.

Once the current approaches were properly documented and the results described, the Ontogrator approach was performed against the query. An evaluation took place recording the relative merits of the Ontogrator user interface and the accuracy of the data within. This evaluation was achieved entirely through interviews with the Biomedical Informatics department staff and their perception of the efficacy of the solution. Any shortcomings or perceived advantages over current processes, as well as all comments, both positive and negative, were collected and documented.

**Current Approaches**

There are two approaches currently performed at AstraZeneca, internal and outsourced.

**Internal Approach**

The internal approach involves utilisation of commercial text-mining tools and data sources available through paid suppliers, as well as freely available data sources, at AstraZeneca. This approach is described in Figure 30.
The many manual steps involved can be time-consuming and lead to inconsistencies, but with the automated steps there can be a prohibitive number of false positives. As searches become more complex more dictionary or MeSH terms can potentially be used, this may but this can lead to inconsistencies between searches over the different sources.

Table 3: Advantages and disadvantages of Internal Approach

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Use of comprehensive dictionaries and controlled vocabularies</td>
<td>Manual approach is time-consuming</td>
</tr>
<tr>
<td>Powerful text-mining</td>
<td>Automated approach results in many false positives</td>
</tr>
<tr>
<td></td>
<td>Limited by search form of website</td>
</tr>
<tr>
<td></td>
<td>Complex searches are more prone to error as well as less consistent</td>
</tr>
<tr>
<td></td>
<td>Depending on the number of results, can be extremely time-consuming</td>
</tr>
</tbody>
</table>
Outsourced Approach

There is some outsourcing of work to suppliers overseas. The suppliers curate the literature according to a given requirement and deadline and the results are returned typically in the form of an Excel spreadsheet.

This approach requires a very clear definition of the question as well as the data sources that will or will not be searched. Depending on the defined question the external company may process the list of key references, pulling out the required information, saving time for AstraZeneca. Additionally, this approach requires careful consideration of data capture in order to ensure consistency. Generally, this is perceived to be a consistent and thorough approach to literature searching.

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saves time for AstraZeneca staff</td>
<td>Costly</td>
</tr>
<tr>
<td>Thorough searching by trusted supplier</td>
<td>Relies upon</td>
</tr>
<tr>
<td></td>
<td>- Exact definition of question</td>
</tr>
<tr>
<td></td>
<td>- Clear structure of data capture</td>
</tr>
<tr>
<td></td>
<td>- Clear definition of sources to be searched</td>
</tr>
<tr>
<td>Consistency across literature search</td>
<td>Must still wait for a potentially long length of</td>
</tr>
<tr>
<td></td>
<td>time before results are available</td>
</tr>
<tr>
<td>Updates can be done at a later date</td>
<td></td>
</tr>
<tr>
<td>May also be able to get required information curated</td>
<td></td>
</tr>
</tbody>
</table>
Ontogrator Approach

The Ontogrator approach attempts to retrieve as much data as possible derived from the defined question by using Ontogrator and The Box.

The Ontogrator highlights particular weaknesses in the ontology. If the ontology does not include a specifically required term such as “3rd line” or “metastatic breast cancer”, then no annotation takes place at that term. As many clinical trials data sources are not freely available (e.g. TrialTrove, Pubget, MeSH, conference abstract suppliers), these sources were not able to be retrieved and mined. The existing data sources do send information that their structure implies is available, for instance links between PubMed articles and ClinicalTrials.gov - many ClinicalTrials.gov articles cross-reference PubMed but do not include this information in the XML, and so Ontogrator is unable to display this information. Once hits have been retrieved, there remains a manual approach to going through each document to collect required information. However, Ontogrator helps to define a question – it is not necessary to have a predefined question before using Ontogrator as once a starting point is found, it is possible to drill-down and browse the hierarchy of ontology. This dynamic querying simplifies the “tweaking” of search terms, helping to eliminate false positives and false negatives. The ease-of-use of the user interface may also allow for more involvement with the person asking the question, in the
initial stages of defining the question. The intuitive visualisation of the query may be a useful tool in presenting and further defining and refining the question.

Additionally, Ontogrator is highly configurable, and fields can be added to the key references as required, dependent on the meta-data available from the data source.

Ontogrator unfortunately does not allow the result of the search, i.e. the document list in the tabs, to be exported to another application for further analysis. The results of the time spent querying the data must result in a form of data that is portable.

### Table 5: Advantages and disadvantages of Ontogrator Approach

<table>
<thead>
<tr>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>One stop search</td>
<td>Predominantly limited by dictionaries and controlled vocabularies</td>
</tr>
<tr>
<td></td>
<td>• Detailed disease terms e.g. “metastatic breast cancer”</td>
</tr>
<tr>
<td></td>
<td>• Trial design terms e.g. “3rd line”</td>
</tr>
<tr>
<td></td>
<td>• Unable to search “Phase I OR Phase II”</td>
</tr>
<tr>
<td>Results from different sources grouped into a single result format</td>
<td>Not enough data sources</td>
</tr>
<tr>
<td></td>
<td>• Many data sources are paid suppliers</td>
</tr>
<tr>
<td></td>
<td>• Conference abstracts are also useful as a data source</td>
</tr>
<tr>
<td>Easily alter search terms and look at results – help to eliminate false negatives &amp; false positives</td>
<td>Still manual process for the trials hits, i.e. which have PubMed ID’s</td>
</tr>
<tr>
<td></td>
<td>• Predominantly reflection of ClinicalTrials.gov XML</td>
</tr>
<tr>
<td>Flexibility over search interface (fields added when required)</td>
<td>Still manual approach to find key references and the list is not exportable</td>
</tr>
<tr>
<td>Question need not be defined in advance, can be used to ‘drill-down’</td>
<td></td>
</tr>
</tbody>
</table>
Discussion

A vast amount of work has been performed to make available text documents in a semi-structured manner. There is a comprehensive structure in place to present not only the abstract text of the document, but also important meta-data such as author(s), pertinent dates and links to related documents from other data sources. However, the data we are mostly interested in is in the free text, and not all the meta-data can be relied upon for its associated values being entered. It is very disappointing that very little data exists to enable the linking of documents from different data sources.

There is a problem in accessing the full-text of documents which is being addressed by services such as Pubget, for a price. Unfortunately, due to the costs, this project was unable to text-mine full text documents sourced from Pubget or other such subscription services. It is a very good sign though that these services now exist to automatically retrieve the full-text of documents for text-mining. However, the results of such text-mining have been out-of-scope of this document due to the inaccessibility of the services.

Text-mining provides a useful method of extracting the items of interest in text by indexing or annotating the text, but it has a broad spectrum of efficacy in practice. Not only due to the aforementioned costs, but also the cost of time and effort on a local basis, one must maximise the efficiency and accuracy of text-mining algorithms. Simply searching for certain terms and indexing them against a controlled vocabulary, although demonstrating a certain usefulness and proof-of-concept, does not demonstrate well a commercial value when compared against commercial products that allow for the
analysis of semantics such as negation. Furthermore, the type of text-mining that has been demonstrated with Ontogrator has highlighted the weaknesses of available ontology. As the ontology acts as an index of the textual data, it is imperative that rich and robust ontology is developed to describe the facets of clinical trials for any further development in this area. There is also a question mark over how the data will evolve with the index. For instance, the drug extension files of SNOMED CT are updated once a month – it is important that any new terms or corrections are automatically carried over to the reservoir of mined data, otherwise this data could become quickly out-of-date, consequently losing some of its usefulness. In order to perform this, most likely a best solution would be to bring all the outsourced processes, such as the text-mining process “Terminizer”, together as in-process to maximise communication throughput, and local persistence of documents from data sources. This would allow continual text-mining to occur at maximum speed without the dependence on external services over which there is no local control.

ClinicalTrials.gov are attempting to move away from free text, and therefore also text-mining, by offering the entry and retrieval of clinical trials in a structured form. As mentioned previously, all clinical trials in the US must be registered with them, but enforcement is unclear and the effort by which some companies register their trials may be nominal. As a result, the data is not consistently comprehensive, and may vary in consistency for each company or responsible party. Furthermore, it has been observed that the structure of the clinical trial record has unnecessary duplications, redundant or unused fields and has a propensity to encourage the entry of free text into many fields. Additionally, not all the data recorded is available for automatic retrieval. Although this is
a step forward over the reliance of free-text documents, allowing free-text entry into a
database field is barely an improvement over free-text entry into a word processor, and
merely perpetuates the requirement for text-mining, rather than excluding it as is intended.

It appears Ontogrator has a useful place in the initial stages of the process of retrieving
documents according to a specific question. This is because its ease-of-use and intuitive
interface can appeal to users of differing levels of technical ability, although it seems the
current implementation is only useful for defining and refining a question to give a scope
of what will be returned by more complex and powerful searching methods, i.e. a count
of documents returned from each data source, rather than a means to compile a useful,
final and accurate list of key references.
Conclusion and Recommendations

The Ontogrator is a useful yet very limited solution for visualising the results of text-mining clinical trials documents. Modifying this application for this area of research has mostly highlighted the weaknesses of freely available ontology, data sources, and text-mining techniques, rather than the strength of the same. There is a lot of work to be done in improving the ontology and text-mining algorithms, before testing the operation as a whole from within commercial environment, with access to paid articles, conference abstracts and subscription-based data sources.

The Ontogrator itself requires more powerful filtering options than the current “AND” implementation, which is not powerful or flexible enough. For instance, one may want to perform a search on “Phase II OR Phase II trials for breast cancers that are not metastatic”, and this is currently not possible, yet these types of queries are standard and commonplace with other services such as PubMed.

A fundamental requirement exists that related documents from different sources be visually linked, but unfortunately while the structure of the retrieved data allows for this, much of the time these fields are empty. The accuracy of data from commercial data sources needs to be assessed for its comparative accuracy and comprehensiveness, as this information may well be available elsewhere.

It is recommended that a comprehensive clinical trials ontology be developed that encompasses all things related to clinical trials, as well as the implementation of thorough ontology such as SNOMED CT and the associated drug extensions.
Finally, text-mining techniques need to be enhanced to take into account negation and semantic structure, rather than simple matching of specific text to terms. For instance, the sentence “exclude patients with history of transient ischemic attack or cerebrovascular accident” will return a hit for “Brain Ischemia”. This is a very challenging aspect of text-mining, but it is imperative for a successful solution to be efficient and accurate in reducing the number of “hits” or list of key references to a manageable number as a result of a complex query.

In summary, in terms of a commercial interest, this project has potential but is still in its infancy. More research and investment of both time and money is required to fully realise the potential of using ontology to annotate clinical trials texts from the abundance of data sources available.
References