

Rochester Institute of Technology

RIT Digital Institutional Repository

Theses

5-1-2009

BioMeRSA: The Biology media repository with semantic augmentation

Adam Cornwell

Follow this and additional works at: <https://repository.rit.edu/theses>

Recommended Citation

Cornwell, Adam, "BioMeRSA: The Biology media repository with semantic augmentation" (2009). Thesis. Rochester Institute of Technology. Accessed from

This Thesis is brought to you for free and open access by the RIT Libraries. For more information, please contact repository@rit.edu.

**BioMeRSA:
The Biology Media Repository with
Semantic Augmentation**

Approved: _____
Director of Bioinformatics

Head, Department of Biological Sciences

Submitted in partial fulfillment of the requirements for the Master of Science
degree in Bioinformatics at the Rochester Institute of Technology.

Adam Cornwell
May 2009

Abstract

With computers now capable of easily handling all kinds of multimedia files in vast quantity, and with the Internet now well-suited to exchange these files, we are faced with the challenge of organizing this data in such a way so as to make the information most useful and accessible. This holds true as well for media pertaining to the field of biology, where multimedia is particularly useful in education, as well as in research. To help address this, a software system with a Web-based interface has been developed for improving the accuracy and specificity of multimedia searching and browsing by integrating semantic data pertaining to the field of biology from the Unified Medical Language System (UMLS). Using the Biology Media Repository with Semantic Augmentation (BioMeRSA) system, users who are considered to be 'experts' can associate concepts from UMLS with multimedia files submitted by other users to provide semantic context for the files. These annotations are used to retrieve relevant files in the searching and browsing interfaces. A wide variety of image files are currently supported, with some limited support for video and audio files.

Thesis Committee Members:

Advisor:

Dr. Michael Osier

Assistant Professor,
Department of Biological Sciences,
Rochester Institute of Technology

Dr. Gary Skuse

Professor,
Department of Biological Sciences,
Rochester Institute of Technology

Dr. David Lawlor

Associate Professor,
Department of Biological Sciences,
Rochester Institute of Technology

Acknowledgements

I would like to thank my parents for their support of this academic journey, Jessie for her understanding and encouragement, and friends for withstanding my talk about the project and progress I would also like to thank Dr. Osier for his role as thesis advisor, and the other members of the committee for being able to provide their time to help achieve this goal.

Table of Contents

Abstract.....	ii
Acknowledgements.....	iv
1. Introduction	1
1.1 Overview	1
1.2 The Problem.....	2
1.3 Background	2
1.4 Prior Related Work.....	9
1.5 Lessons from Prior Related Work	14
2. Materials and Methods.....	15
2.1 Overview	15
2.2 Server Environment	15
2.3 Semantic Data- The Unified Medical Language System (UMLS).....	16
2.4 The Web Application.....	22
2.5 Testing.....	31
3. Results.....	33
3.1 User Administration, Authentication, Access Control	33
3.2 Session Management for BioMeRSA	39
3.3 Semantic Data – UMLS.....	41
3.4 Content Submission and Management	53
3.5 Expert Annotation System	61
3.6 Searching and Browsing for Media Files	69
3.7 Program Code and System Environment.....	73
4. Discussion.....	75
4.1 The Implemented Product- BioMeRSA	75
4.2 Issues During Development	76
4.3 Advantages of the BioMeRSA Design.....	79
4.4 Possible Future Work	81
5. Conclusion.....	93
6. Sources Cited	94
6. Appendix A – Perl Modules Utilized.....	101

1. Introduction

1.1 Overview

As our ability to quickly and automatically generate multimedia content increases, so must our methods of storing, sorting, categorizing, and searching that data advance to make that content optimally accessible and useful. In addition to research and laboratory procedures which may generate images or video, such as anything from microscopy to animal studies, educators in the biological sciences are constantly creating new multimedia content. Some of this content may be presented in classrooms, while other times may be provided as an additional resource to supplement classroom learning (as could be the case in a blended learning environment), or may be part of a class which is offered entirely in an online format. Creating or finding this content could be very time-consuming, especially if the subject matter is new, or the desired format is uncommon. Animations, video, or interactive applications (such as in Flash) may be particularly intensive to develop. Much of this effort may be 're-inventing the wheel'- as another individual may have already created some suitable media and may be willing to share their work, but perhaps it is on a page not indexed by search engines, or uses differing terminology, and thus making it difficult for others to utilize.

This project provides an implementation of a database and associated web-based user interface for storing and browsing multimedia files. A key component of this system, which differentiates it from other similar databases, is the integration of the Unified Medical Language System (UMLS) Metathesaurus, which provides 'concepts' that can be linked to stored media items. Linking to the Metathesaurus provides multiple benefits- reduced ambiguity of search terms, and easy discovery of related items, among others. This project was inspired by a collection of 35mm slides, which we hoped to make available to educators in the biological sciences, both inside and outside of the RIT community. This concept was then expanded to

include other multimedia which might be utilized in a similar manner, primarily audio files, video files, and image files derived from other sources. Such a resource may also be useful beyond the classroom, as media can be important to research in some fields.

1.2 The Problem

Since many institutions do not have a central data repository tailored for the storage of multimedia, such media files are likely to stay on individual machines, which may especially be the case if they were generated by an instrument which is directly attached to a single computer. Although often times now such data may eventually be moved to a server accessible by others, unless some sort of organizational standard is enforced, it may be difficult to find particular files of interest. Files are likely to be organized into a few folders, maybe by experiment or specimen and date, with possibly a document in a directory that might contain more detail about that set of data. Especially as science is increasingly multi-disciplinary, it becomes desirable to create systems which may be able to make collaboration as easy as possible. Along with that, it is beneficial to include descriptive data about the media files- metadata- because most multimedia such as images and video, are not self-describing, and may require experts in the relevant field for analysis. Including relevant metadata may provide the information that collaborators would need, without having to seek further assistance, and could also aid the original authors in working with their own data again later.

1.3 Background

1.3.1 Metadata

Metadata is popularly described as 'data about data', and does not strictly refer to data in a computing sense alone. For example, a note in the margin of a book, or a date printed on a developed photo could both be taken as metadata. There may also be multiple metadata levels (in both the physical and computational worlds), where a metadata item itself has metadata. In

the scope of this project, we seek to combine the metadata of the media items with data from UMLS, and store it in a computer-readable format. Examples of such metadata could be the name of a procedure used to generate the subject of a photo, or the names of organisms that might be seen in a video file. This allows the system to present more descriptive information about each item than would otherwise be possible.

Associating metadata with a file is by no means a new concept; for example, the Exif standard specifies a file format used by imaging devices for JPEG and TIFF images, which allows metadata about the camera and imaging settings to be recorded along with the image file (Japan Electronics and Information Technology Industries Association, 2002). This format is now utilized by almost all major digital camera manufacturers, and can be useful to assist with the automation of some image processing tasks, such as image rotation based on the orientation information recorded by the camera at the time a shot was taken. Digital audio file formats like MP3 and WMA can also include metadata (such as the ID3 tag format for MP3), often used for adding artist name, album name, song name, etc. when storing music. This trend is not limited to media files either, and many other documents such as files created by Microsoft Word or PowerPoint include metadata, which may be collected automatically from the content of the document, manually entered by the user, or retrieved from the operating system (such as date/time, or the name of the logged-in user who is creating the file). These metadata formats just mentioned are designed to be both human- and computer-readable, allowing users to interpret the data themselves, and enabling programs capable of parsing the metadata to implement additional functionality- such as enhanced searching and sorting of files, or specific manipulations on the file data.

Many public sites for storage and access of multimedia have evolved on the Internet, many of which offer search capabilities, and may be enhanced by systems utilizing metadata

entered by the users who submitted the media. A popular method of doing this is by ‘tagging’ files with user-entered text keywords, which provide some semantic context for the content, usually along with the option to enter a more detailed full text description if desired. A good example of this kind of system would be the popular photo sharing site, Flickr, which allows up to 75 tags to be added for each submitted photo (Flickr/Yahoo inc.). Searching for a term that matches a tag will result in all of the photos that have been tagged with that keyword, and viewing a photo will also bring up the other keywords the photo has been tagged with, which can be used to find other related photos. Some sites, Flickr included, offer access to some of this metadata more directly through APIs (Application Programming Interfaces) which have enabled interested researchers to access a large amount of tag data, which has been used in attempts to create structures from it, such as taxonomies (Marlow *et al.*, 2006). Although these public sites are often fast and reliable (because of their commercial nature), they are usually dedicated to one type of media (e.g. Flickr for photos, Youtube for video) and offer limited fields for descriptive data, and would thus not be suitable hosts for the array of multimedia content that would need to be stored for educational or experimental fields. Additionally, sites such as Youtube and Flickr are optimized for quick web browser-based viewing, and as a result the original uploaded content is recompressed. The original quality files may not be available, or download functionality may not be present on the host site. The ability to acquire media for local use is certainly a requisite factor for being able to effectively reuse it.

1.3.2 Learning Objects

In discussing metadata for a media system intended for deployment in an educational environment, mention should be made of ‘learning objects’. Learning Objects (LOs) are the basis of a developing approach for managing and assembling content used in education. The core methodology of this approach involves breaking down educational material (which could range

from lesson plans to exams and textbooks) into smaller, reusable, self-contained chunks with a specific focus. These smaller pieces may be then more efficiently reused and repurposed. To help facilitate this, both the IEEE Learning Technology Standards Committee and the IMS Global Learning Consortium have developed standards and specifications related to the utilization of learning objects, with a focus on metadata, most notably the IEEE 1484.12.1-2002 standard (Learning Technology Standards Committee of the IEEE, 2002). Annotating LOs with appropriate human- *and* computer-readable metadata enables systems which can search for objects pertaining to a specific topic, and organize them in a logical manner, which may even be done automatically. Lesson plans and presentations can be re-purposed or updated easily, along with quizzes, tests, and even informational course websites, for example. This would be particularly advantageous if more of the current work on adaptive learning technology makes it out of research, and into widespread practical use. Already, Blackboard, a leading learning management system software provider, has integrated support for learning objects, and participates in the IMS Global Learning Consortium (Blackboard Inc., 2005). As more content is adapted into learning objects, more educators may begin to utilize course creation software with support for LOs. It is particularly foreseeable that learning objects and adaptive learning technologies will have significant impact on online learning curricula in the future, where course material may be able to be automatically tailored to a student's level of knowledge and learning style, by dynamically rearranging appropriate learning objects. If the LO approach is eventually adopted for this resource, it will likely make it easier to utilize the content in adaptive learning systems in the future.

1.3.3 The Semantic Web

In addition to supplementing the capabilities of a standalone system, metadata is increasingly being used to provide some meaningful context for individual systems or

components tied together in a network. This was most famously proposed by Tim Berners-Lee as the basis for the 'Semantic Web' (Berners-Lee, Hendler, & Lassila, 2001). The 'Semantic Web' movement aims to place web content into a framework which gives it computer-parsable semantic context, 'meaning' which programs are able to 'understand' based on the structures in the framework, while maintaining the decentralized nature of the Internet as we are familiar with it. This would essentially enable a more 'intelligent' web, where relevant data is accessible efficiently, made possible by this background semantic processing and communication between systems. This has obvious advantages for education and science- reduced ambiguity will lead to finding useful resources faster, automated methods of knowledge mining and aggregation/integration may lead to increased collaboration and sharing of data, also increasing the reusability of existing content, when applicable. Progress has already been made toward this vision of ubiquitous semantics- as mentioned before, use of metadata has already been implemented in many applications, ranging from desktop software (media players, image viewers, etc.) to large-scale commercial web sites. Simply having the semantics available however does not make it a web- there needs to be ways of sharing that data. On top of that, these systems need to be able to speak the same language (on a computational level), or be able to reliably transform their internal language with some external standard for communicating this semantic data, just like the other protocols which make up the human-readable web of today.

To achieve the requisite interoperability between semantic systems, the World Wide Web Consortium (W3C) has created three specifications they categorize as 'recommendations' (essentially standards) to deal with semantic data (World Wide Web Consortium, 2008). At the base, the Resource Document Framework (RDF) provides a way of representing metadata objects in a 'subject-predicate-object' triplet form, structured by implementation in

XML(Berners-Lee, Hendler, & Lassila, 2001). An example of data represented in an RDF 'triplet' might be "Rochester : is a city in : New York". When the expressiveness of RDF is inadequate, the OWL Web Ontology Language can be used to represent the information in an ontology, built on top of (implemented in) the base of RDF + XML; the OWL standard evolved from DARPA's DAML+OIL (DARPA Agent Markup Language + Ontology Interface Layer), which had also been implemented using RDF(World Wide Web Consortium, 2004). There is also a specification for a query language to be used with RDF - SPARQL, similar to SQL as used with relational databases (World Wide Web Consortium, 2008).

1.3.4 Ontologies

Ontologies will be fundamental to the effective representation of the data in the current vision of the semantic web, a function of their organization. Although all ontologies have the same basic structure, it is sometimes difficult to elucidate a clear picture from the definitions in much of the literature. Gathering many such definitions enables identification of the essential elements:

- An ontology is used to represent knowledge from a specific domain or subject area.
- Individual concepts from the domain of interest are represented, which may take the form of a vocabulary, with definitions of the terms. These concepts are akin to classes or objects in the Object Oriented model.
- Any relationships or interconnections between these terms must also be defined (Gruber, 2007)(Uschold *et al.*, 1999)(The Gene Ontology Consortium, 2000)(Berners-Lee *et al.*, 2001).

So in this case, we are referring to the use of ontologies to represent the concepts, terminology, and relationships essential to a specific domain of interest (Berners-Lee, Hendler, & Lassila, 2001). The relationships in an ontology lend a hierarchical structure, similar to a

taxonomy. However, ontologies differ from taxonomies (and thesauri) in that they have definitions of concepts, and include rules and restrictions which may be required to adequately represent the target knowledge domain logically. One of the main reasons ontologies are useful is their ability to unambiguously represent knowledge (if properly constructed). Relationships allow a program which might be utilizing an ontology to determine that not only is a 'truck' a type of 'vehicle' but also that a 'lorry' is a type of vehicle and is equivalent to a truck. Or perhaps a more appropriate example, if a teacher is using a multimedia resource system which is semantically enhanced via the use of ontologies to look for scanning electron micrographs of the cyanobacterium *Microcystis aeruginosa*, if there are no results for the initial query, the system might be able to return SEM micrographs of other *Microcystis* species, or perhaps transmission electron micrographs of *M. aeruginosa*, based on the relationships it found for these concepts in the ontology. Although these other results might not match the teacher's original query, they might still be of some interest.

Some concerns have arisen as to whether RDF and OWL can provide the expressiveness that may be required for complex semantically aware applications(Denny, 2004), but regardless both are already well supported, and migration toward a standard will help progress the Semantic Web, by allowing individual implementations to begin to interact. This field is still being very actively researched and thus remains a moving target, and these standards may need to be revised in the future. It may be perfectly feasible to modify OWL as needed to increase expressivity for expanded applications, while maintaining compatibility with the current standard. In the case that an ontology is not suited to representing all the information about a concept that needs to be stored, other metadata standards may also be utilized. In addition to the metadata specifications already discussed, another which has come into frequent use is the Dublin Core Metadata Initiative's standard set of metadata elements (often known simply as

'Dublin Core'), which can be used along with ontologies is often implemented in RDF(Hillman, 2005). Dublin Core has fifteen base elements, such as 'Title', 'Format', 'Rights', etc. and is utilized similarly to Learning Objects metadata- to add a standard minimal set of descriptors to digital content.

Ontologies are represented computationally in formal ontology languages- such as the OWL Web Ontology Language previously mentioned, but there are others as well. There is no longer a need to implement an ontology by working directly in an ontology language, as there is now a staggering array of ontology authoring tools which have been developed to aid with the task (Denny, 2004). Thus, any individual with expert knowledge of a field can build an ontology without extensive training in an ontology language- since we desire to eventually be able to represent all knowledge domains in such structures, this is a key factor. Artificial Intelligence methods may eventually be able to provide more aid in this task, but more development in that field is required. Some AI technologies are already being utilized. However, large projects such as WikiProteins/WikiProfessional are using text and data mining methods to garner information from the literature to augment existing semantic structures in a semi-automated fashion (Barend, *et al.*, 2008).

1.4 Prior Related Work

1.4.1 The Gene Ontology

There have arisen many community-based projects for the creation of ontologies covering knowledge in biological fields, some with wider scope than others. Some projects, such as the Gene Ontology (GO) over time have expanded their area of coverage. The GO project initially set out to create an ontology for concepts related to genes and gene products in Eukaryotes, to help automate the process of annotating newly sequenced genes and genomes. Now, at least partially due to an explosion in the availability of whole sequences, the GO is open

to any type of organism (The Gene Ontology Consortium, 2000). The GO is actually a set of three ontologies: molecular function (of a gene product), biological process (that gene products are involved in), and cellular component (where gene products are localized in cells). Due to the number of ontology projects available, online directories have been created to assist interested individuals find an appropriate resource, these include Swoogle (UMBC Ebiquty Group, 2008), SchemaWeb (Lindesay), and OntoSelect (Buitelaar, Eigner, & Declerck, 2004), to name a few.

1.4.2 The Unified Medical Language System (UMLS)

The National Library of Medicine's (NLM) Unified Medical Language System (UMLS) project is much more expansive, aiming to cover data from the whole of medical science. UMLS integrates the ontologies and other structures produced by other projects- for example, it now includes the GO, as well as an organism taxonomy provided by the National Center for Biotechnology Information (NCBI) with over 100,000 organisms(McCray, 2003)(NLM, 2008). The UMLS system also contains multiple components; the 'Metathesaurus' is an ontology-like structure of all of the merged resources, in which each concept is linked to a 'Semantic Network' providing broad classifications of terms i.e. 'organism' or 'chemical', etc. The UMLS Metathesaurus may not strictly be an ontology when considered along with the Semantic Network, and is not typically described as such. This may have been due to the concept of ontologies in an information science sense not having been yet popularized at the time the UMLS project was started in 1986(Kleinsorge, 2005)(Gruber, *Ontology (Computer Science)*, 2008). The Metathesaurus has come to be structured very much like an ontology- it can be represented in a hierarchical fashion, utilizes 'concepts', and defines relationships between items- however NLM states that UMLS does not provide a comprehensive enough representation of the biomedical domain to be considered an ontology(National Library of

Medicine, 2006). However, this difference in terminology does not make it any less useful in the context of semantic systems.

1.4.3 MEDIANOVO

A multimedia repository, MEDIANOVO, targeted at medical education and research was developed in Germany, and has much of the desired functionality (Wunderlich, Ott, & Bernauer, 2003). MEDIANOVO is described as a system with a web-based front-end, capable of storing multiple media formats, allowing users to upload and annotate content, and includes a system for authentication of users and management of copyrighted media. The stored media could be searched and browsed, and multiple options for downloading were presented to users. The ability to annotate content is of particular note, as they were able to utilize the NCBI MeSH vocabulary (NLM, 2008) and the UMLS ontology (NLM, 2008) for assigning keywords to media, and searching. Unfortunately this resource appears to no longer be available. A similar resource sharing a similar fate, the life sciences Global Image Database was a project of GlaxoWellcome, which was discontinued upon the merger with Smith Kline. GID provided a Java applet-based web interface for annotating, submitting, and browsing scientific images. A minimum set of annotation properties was identified and required, and submitted with the media by the authors of the content. These annotations included sample information, authors, experimental protocols utilized, etc. It was originally envisioned that GID would eventually be a large and diverse enough resource to be useful in the development of novel image processing, classification, and visualization techniques (Gonzalez-Cuoto, Hayes, & Danckaert, 2001).

1.4.4 Online Library of Images for Veterinary Education and Research (O.L.I.V.E.R.)

A more recent project featuring the development of a structured image repository for the purpose of maximizing use of teaching/learning resources is the Online Library of Images for Veterinary Education and Research, or O.L.I.V.E.R.(McGreevy *et al.*, 2007). An interesting note

about this resource is that the content is available only to those at the University of Sydney, and no content is made publically available. Each item in O.L.I.V.E.R. is treated as a 'learning object', an approach to utilizing educational material which has been gaining popularity in the field of interactive learning. A 'learning object' may be defined as "any entity, digital or non-digital, that may be used for learning, education, or training" (Gasevic, Jovanovic, & Devedzic, 2007). O.L.I.V.E.R. uses its own metadata system for learning objects, and each file is indexed manually, and tagged with appropriate terms from MeSH (mentioned earlier) and SNOMED. This manual interaction is required even when staff submits images, and although it takes more time before submitted content goes live, it ensures a standard for quality of material is maintained.

1.4.5 Health Education Assets Library (HEAL)

HEAL (Health Education Assets Library) is another ambitious and large-scale educational material resource, utilizing the 'learning object' concept and descriptive metadata (Candler, Uijdehaage, & Dennis, 2003). The HEAL project has existed since 2000, and now contains 22,358 resources, contributed by individuals and institutions (Health Education Assets Library, 2008). The types of content indexed include images, videos, animation, and text (such as questions for quizzes or tests). One of the primary concerns in creating HEAL was to ensure the system of quality control was sustainable- this was accomplished by requiring manual screening and indexing by project staff and librarians, and later with the addition of a system of peer review (Health Education Assets Library). An interesting feature of HEAL is the ability to index resources hosted by external institutions- software is provided to these institutions to aid in annotating and linking their content to HEAL. In this way, it might be possible for an institution which does not already have their collection of educational multimedia organized to actually gain easier access to their own material by participating. Although mentioned by other projects, HEAL also gives specific attention to issues surrounding copyright and material ownership. To address this,

usage limitations are displayed for each resource, as well as any logos for the institution which had submitted the resource. It was also recognized that patient identity must remain confidential when resources are submitted that may be derived from medical data; the aforementioned manual curation process required before resources are publicly viewable helps to ensure the intellectual property and confidentiality standards are adequately met.

1.4.6 Other Related Projects

Other resources with a similar purpose appear to have media covering either a very wide range or rather narrow range of subject material. The National Science Foundation's (NSF) National Science Digital Library (NSDL) project is intended to provide multimedia material to support education spanning science, technology, engineering, and mathematical fields (NSF). The NSDL itself does not actually host the material however, and although it is possible to limit searches to a specific media type, i.e. images or video, the search results are links to external websites which have the resources. This lack of control over the content means that it is more difficult to ensure consistent quality, regular updates, or even that the providers will continue to host the material. It may also lead to a more confusing user experience, because the presentation is not uniform, and each site will need to be navigated differently. It is however, in its current form, a reasonable directory from which to start a search for science information where a broader context is desired. On the other hand, the BioImage project from the Image Bioinformatics Research Group at the University of Oxford (also no longer available) was a more specialized resource, utilizing an object-relational database to store and retrieve images and video from microscopy, along with relevant metadata and keywords, and related links to external databases (PDB, SwissProt, etc.) (Carazo & Stelzer, *The Bioimage Database Project: Organizing Multidimensional Biological Images in an Object-Relational Database*, 1999)(Carazo J. *et al.*, 1999). There are many other image resources provided publicly by individuals and

research groups to share their work, which may not be open to submissions; examples would include Biodic (Vos, 2007) hosted by the Université libre de Bruxelles for biology-related electron micrographs, and BIODIDAC (BIODIDAC Group) at the University of Ottawa, which is a searchable resource with drawings of organisms, microscopy photos, and some videos/animations.

1.5 Lessons from Prior Related Work

There is something of a trend which can be seen amongst the previous work that has been mentioned in that the longevity of some of these multimedia repository projects has not been particularly exceptional. In most of these cases, the reasons for the discontinuation of the projects are not known, but it can be postulated that a lack of technical, financial, or personnel resources played a large role. Knowing this history, it will be essential to strike a balance between quality of the service provided, and the resources required to manage and administer the system, to ensure availability in the long term. Another important factor in manageability is the scale of the system; there needs to be enough interested users to make it worth the effort imparted into development/maintenance, but not too ambitious to become unfeasible for the given resources. Intellectual property issues must also be addressed as part of this, especially as more concerns are raised over copyright and ownership of media on the Internet. Finally, there must be an interface for users presented with usability in mind, especially when dealing with the management of a large amount of information. Despite the importance of the user interface, very few of the papers published about the cited resources have direct mention of this, MEDIANOVO having given it the most attention.

2. Materials and Methods

2.1 Overview

The components of BioMeRSA development can be viewed as three parts: The semantic and content data, the database and business logic, and the application logic and user interface. The business logic includes the portion of the application which fetches data from and stores data to the database upon request from the application logic. The application logic handles actions performed by users in the user interface, and processes data so that it can be displayed in the user interface. The semantic and content data is what's stored in the database, and is what the application as a whole is designed to handle. Development was not limited to a single part at a time, since the design of components of one section altered that of others, and in some cases this caused the implementation to deviate from the original plan.

2.2 Server Environment

2.2.1 Hardware

The server environment available for development did have an influence on some implementation decisions. The development server will also be utilized as the production server until it is found that more resources are needed to maintain the system. The server is a Dell PowerEdge 1850 1U rack machine with two Hyper-Threaded Intel Xeon CPUs at 2.8 GHz, 4GB system RAM, and a RAID 1 mirrored storage array with a volume capacity of 540 GB, formatted with the EXT3 filesystem. The Operating System is Fedora Core Linux 4, originally released in June of 2005, utilizing a 64-bit version of the Linux 2.6 kernel. Only one of the two CPUs is useable, as the installed kernel is not Symmetric Multiprocessing (SMP) capable. However, there still appears to be two CPUs because of Hyperthreading. Other installed software essential to this system includes the Apache HTTP Server 2.0.54, Perl 5.8.6, and MySQL 4.1.2. Note that these were not the latest versions of the respective software packages at the time of development, however it was determined that upgrades would not be performed unless found

to be necessary, as the server is not dedicated to BioMeRSA and downtime could affect other users.

2.2.2 Software

A database management system was needed to store, retrieve, and assist in managing the content metadata, data about users, and the semantic data. While it was originally planned that MySQL 5 would be used as the DBMS for its support of advanced features such as stored procedures and triggers, which are lacking in MySQL 4 but present in competing DBMS packages such as PostgreSQL (Conrad, 2004), it was found that only MySQL 4 was installed and was thusly used. While MySQL 4 provides a fast and stable database platform, the additional features of version 5 may be found to be desirable enough in the future to warrant an upgrade. MySQL was chosen over PostgreSQL and other competing RDBMS solutions for its reputation for speed (Windows Skills, 2007); however as development continues on other systems, the advantage that MySQL holds has shrunk in some performance metrics, and it may be reasonable to eventually re-evaluate that decision as necessary. In particular, it appears that PostgreSQL may scale better to higher loads on servers with multiple processors than MySQL (Brent, 2007). However, because database configuration, the data being stored and its organization, and the server hardware specifications can have such a high impact on database benchmark results, only real-world testing for a particular application can determine the best DBMS to use.

2.3 Semantic Data- The Unified Medical Language System (UMLS)

2.3.1 UMLS Subset Generation

Semantic data is provided by the Unified Medical Language System (UMLS) for annotation of user-submitted content. UMLS was chosen for its scope and depth of included data, frequency of updates, and quality of documentation and tools. Although there is not much third-party documentation or support available, the documentation provided by UMLS on their

website is mature enough to make this a non-issue (National Library of Medicine, 2008).

However, UMLS is provided in such a way as to maximize the range of suitable applications- not unexpected given the scope of the data and as such, some customization is necessary to utilize it in a specific system. UMLS release 2008AA (April 2008) was obtained via the UMLS Knowledge Source Server (UMLSKS) and the included MetamorphoSys utility was used to generate a reasonable subset consisting of data from sources expected to be most useful. This is a necessary step, as MetamorphoSys generates formatted data flat files, and SQL scripts which can be used to load the data from the data files into a database. MetamorphoSys also enables exclusion of data on a per-source basis- in this case, it had been determined that only English-language sources would be needed, so non-English sources were not selected to be part of the subset. The full list of sources selected for the generated subset can be found in the results section. The 'default subset' selected before source selection was 'Level 0' (sources which do not require any license agreements over the default UMLS license) + SNOMEDCT, which requires compliance to the additional, separate license for SNOMEDCT. Precedence and suppressibility options were left at the defaults for the subset. Generation of database load scripts was enabled, and Rich Release Format (RRF) output data files were selected over the legacy Original Release Format (ORF). Although ORF is still supported to an extent, RRF is a newer format designed to be easier to work with, and includes some data which is not provided when ORF files are used(National Library of Medicine, 2009). Subset generation using MetamorphoSys was first attempted under Windows XP Pro, but due to disk space issues, was later completed on the development server in Linux.

2.3.2 UMLS Database Tables

The data from the generated RRF subset files was loaded into MySQL using SQL scripts generated for table creation and data import for both the Metathesaurus data and the Semantic

Network (which are treated as separate datasets). These scripts take care of executing the statements required to make the tables (one table per RRF file) and for table population, using the MySQL 'load data local infile' statement. For example: "LOAD DATA LOCAL INFILE 'ambiglui_new.rrf' INTO TABLE AMBIGLUI FIELDS TERMINATED BY '|' LINES TERMINATED BY '\n';". In this manner, the data in the flat files is parsed into appropriate fields in the database. However, it was found that the data load statements were actually designed for MySQL 5 and were not compatible with MySQL 4. This issue was solved by importing the data into a different system running MySQL 5, exporting it using the 'SELECT INTO OUTFILE' syntax, and then re-importing it in MySQL 4 on the development server.

To ensure that no other artifacts were introduced in the process of the script incompatibility workaround, the number of records for each table was recorded after importation in MySQL 5 and then again after export and importation in MySQL 4. One table, MRSAT, was found to have gained 418 rows during the process. Due to the large size of the MRSAT source data file (around 1.5GB) it was not possible to run Diff or other similar programs which are commonly used to find differences between file data, so a small program was constructed in Visual Basic .Net to compare the data file before importation and the data file after exportation into/from MySQL 5. After determining that the issue stemmed from unescaped backslash characters, a second Visual Basic program was written to count the instances of '\n', which was found to be 418, matching the previously found row discrepancy. The solution was to replace every instance of '\' with '\\' in the original data file, which properly escaped the backslashes so that they would not be considered special characters.

The same importation process was performed for importing the Semantic Network data into the database, except that there were no issues with backslashes.

It was determined that some of the UMLS tables that were created would not be needed for use in the application. Ignoring some of the tables also simplifies the process of normalization and customizing the structure to better suit the needs of this system- the UMLS distribution is not designed for use in any specific implementation or type of system, and is intended to be as malleable as possible for ease of utilizing it in any of a very wide range of possible applications. As such, while there is a script to add indices and keys to some of the Metathesaurus tables, no foreign keys are created by default and many tables lack primary keys. If any specific level of normalization is required, it must be done by the developers implementing UMLS in their applications. For BioMeRSA, the tables have not been fully normalized to a specific level of normalization.

2.3.3 Database Normalization

The effort to impart additional structure to the UMLS tables was started with the Semantic Network, which is what the first phase of UMLS-related application development was concerned with. It was first determined that some of the tables contained redundant information in different but equivalent forms- *SRSTRE1* and *SRSTRE2* both include the same data for fully inherited node relationships, but *SRSTRE2* stores Semantic Network node names, and *SRSTRE1* stores the node IDs. It was also noted that *SRSTR*, which gives the structure of the Semantic Network, also used only names (as strings) for the nodes. To address these inconsistencies in data representation, a new table was created with the Semantic Network node ID as the primary key and fields for information common to both semantic type and relationship nodes: node type, name, tree number, definition, and abbreviation. This table was called *sty_rl_com*.

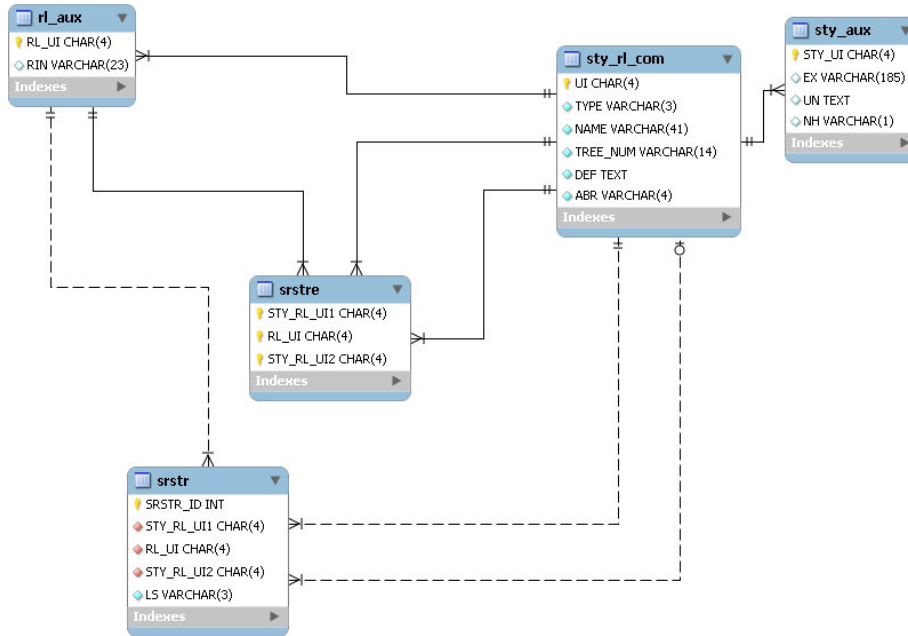


Figure 2.1 - The modified Semantic Network database structure

The only trait common to just relationship nodes was separated into *rl_aux* and the traits common to just semantic type nodes were copied to *sty_aux*, with all three of these tables (*sty_rl_com*, *sty_aux*, and *rl_aux*) utilizing the unique ID (UI) as their primary keys. *sty_rl_com* contains every Semantic Network node ID exactly once (189 Semantic Network nodes in 2008AA), while *rl_aux* and *sty_aux* only reference the IDs of the relationship nodes and semantic type nodes, respectively. SRSTR was transformed to contain the IDs of nodes instead of the names, as it would be expected to be easier to work with the IDs under most conditions, and when needed, *sty_rl_com* can be queried to obtain the name of a node. This new table is *srstr*, and also had to have a new autoincrement ID column added to serve as the primary key, as there might be multiple rows with the same STY_RL_UI1, and STY_RL_UI2 needs to be nullable, ruling out the possibility of a composite primary key. The resulting Semantic Network database table structure is shown in figure 2.1.

For the UMLS tables, there no table with all the concept IDs (CUIs) suitable of being identified as a primary key by default- and thus no table that could be referenced by other

tables containing CUIs as foreign keys. A new table was created containing every unique concept ID (1,221,057 concepts in 2008AA), containing a field for the CUI and a field for the concept name. Although MRCONSO is identified as the primary concept structure table, the primary key is actually the AUI- the atom ID, because each concept may be referenced by more than one term, from multiple sources (illustrated in figure 2.2). Due to this, there is no single name for many concepts, and so the preferred name was taken from MRCONSO for the name field in *conceptindex*. This was done by querying MRCONSO for rows where the term status (TS) was 'P' (preferred term), the string type (STT) was 'PF' (preferred form), and where the record represents the preferred atom for the concept (ISPREF = 'Y'). Each of these fields individually or in any other combination was inadequate to uniquely identify the preferred name for any given concept

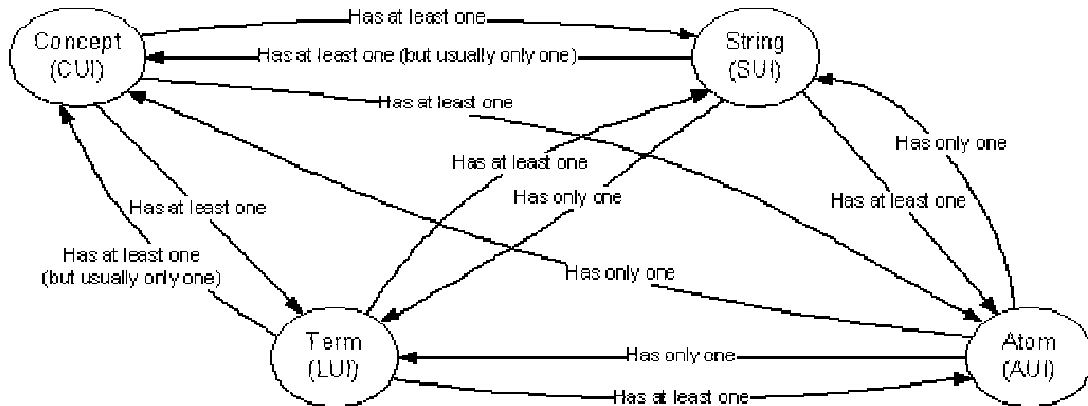


Figure 2.2 - Relationships between the concepts, terms, strings, and atoms in the Metathesaurus.

2.4 The Web Application

The BioMeRSA application that users interact with consists of two parts. The first is an interface for managing users and their accounts for the system (referred to as UserDB). The second is a system for submitting, browsing, as well as an annotation system for the determination of Metathesaurus concept(s) to assign to a content item. Both parts interact with users via a web browser- enabling easy access from nearly any modern workstation without having to install additional software. The user management system, being simpler, was developed first. Code was available which implemented the skeleton of a session and user management interface, which became the basis of UserDB. UserDB and the rest of BioMeRSA are built with a largely overlapping set of tools- both are written in Perl, use the Perl module DBI to provide an interface to the MySQL database, and CGI::Application for enabling the Model-View-Controller (MVC) development approach. The goal of MVC is to separate the business logic (the model, which communicates with the database) from the code for the presentation layer (the view) and the intermediate layer which communicates between the two (the control). Thusly, for any MVC CGI::Application program, three classes (which are usually split into three files) are required. MVC is a common approach for programs involving user interfaces, as it is a sensible method of organizing code, and can make debugging specific sections easier, in addition to encouraging reuse of code (on the model layer in particular). CGI::Application uses a series of 'run modes' which usually correspond each to a page displayed to the user- but since each run mode is actually a subroutine in the control section, it is possible to request data from a specific run mode without having that run mode load a new page. Both parts of the system also utilize HTML::Template, another Perl module which allows the HTML which will be returned to the users' Web Browser to be split out from the Perl code itself and into a 'template file'. This further separates the interface design from the application logic, and can ease maintenance and debugging.

2.4.1 BioMeRSA Component Technologies Overview

As previously mentioned, the rest of the BioMeRSA application is implemented on top of the same base tools and located on the same server as the UserDB. Perl, DBI, MySQL 4, and CGI::Application are all utilized. However, additional components and technologies were needed to provide a good user experience and to implement the desired functionality. First and foremost is the utilization of JavaScript, a client-side Web browser-based scripting language, to provide functions such as client-side form verification, interactive display of complex data structures, and asynchronous content updates (more commonly referred to as AJAX). Due to the rapid evolution of these 'Web 2.0' technologies, different web browsers offer significant variation in their implementations, which must be dealt with during development of applications utilizing JavaScript. In this case, some of the challenges in dealing with legacy versions of common web browsers were reduced by deciding to not support the use of the application with browsers that do not offer JavaScript, or for users without JavaScript enabled. Cross-browser compatibility issues are otherwise simplified by utilizing a JavaScript framework, which takes care of determining what code to execute for a certain task based on the client browser and version. Many JavaScript frameworks are now available, varying in the size and maturity of their codebase, and level of community support. JQuery, for example, is designed primarily to assist with AJAX data transport tasks, and is very lightweight, but yet still extensible by third-party libraries, such as JQuery UI(jQuery, 2009)(jQuery UI). Google and Yahoo have released portions of the tools that they use internally to support development of their own Web 2.0 sites, like the Yahoo User Interface Library (Yahoo!). The Dojo framework was chosen for BioMeRSA because of the wide variety of available functions and UI widgets, apparently good management of HTTP requests, and flexible syntax. Although Dojo is designed to be much more than an AJAX framework, there is a good level of abstraction for asynchronous requests (Schiemann, 2009). 'AJAX' refers to 'Asynchronous JavaScript and XML' (not intended by the

progenitor of the term to be an acronym), a web programming technique for sending requests to a server from a browser while still allowing the user to interact with the application in the browser even if the request data has not yet been returned (Garrett, 2005). This enables a level of interactivity previously only seen in desktop applications, and when combined with the Document Object Model (DOM), it allows the content of a web page to be updated dynamically without having to reload and re-render the entire page. Although XML was originally intended as the data format of choice for transmission between browser and server for AJAX, other content types can also be utilized, most popularly plaintext and JavaScript Object Notation (JSON). JSON utilizes the syntax used to represent data structures in JavaScript objects to format data, and can often be orders of magnitude faster to parse than XML inside JavaScript. JSON is now very often utilized when the structure and formality of XML are not necessary. Additionally, Perl has a very similar syntax to JavaScript for objects, making it easy to create data structures that can quickly be made JSON compatible. The majority of data sent in requests and responses in BioMeRSA is sent as JSON, and the Perl module JSON is utilized to assist in encoding and decoding data being sent to or from the server to the web application (Hannyaharamitu, 2009). Version 1.2.3 of the Dojo Framework was used, as it was the most recent version available at the time that development was started. Some difficulties were encountered in using this release, as the developers were preparing for the major 1.3 release, and in some cases the documentation had yet to be updated from version 1, despite major changes having already been made between version 1 and 1.2. Dojo's community support forums are also not as active at the moment as those of some of the other JavaScript frameworks. SitePen, a company founded by some of the key Dojo developer, provides commercial support for Dojo and may be a factor in Dojo's popularity in large corporate environments.

2.4.2 User Management

The UserDB has its own tables in the database for the purpose of tracking sessions, storing user data, and keeping track of user permissions. These are named *logins*, *users*, and *userpermissions* respectively. DBI (DataBase Interface) was chosen as the database interface as it is the standard stored data interface for most applications in Perl, and it is quite mature and robust. As a security feature, placeholders are utilized in the definition of the SQL statements to provide adequate protection against most SQL injection attacks. Also for security, passwords are encrypted using SHA1 when stored in the database, and are transmitted between the web browser and the server via POST requests. There is a configuration file in addition to the files already described, which is for quickly modifying certain parameters without having to go through the denser application code to alter their values. The UserDB is intended for use by both normal users and administrators, and different functionality is available depending on the permissions of the user. There are a set of actions defined in the *userpermissions* table which can be granted for each user- for example, there could be a user that can create other users, but not modify the passwords of other users, nor delete users. Session tracking is handled by appending a configurable-length hexadecimal session ID, generated upon login and stored in the *logins* table, to the page as a parameter. The date and time information of the last user activity is recorded, and the session will be marked inactive when the application is called to check the validity of a session and finds that the time since the last activity exceeds the configured timeout period. User accounts may also be locked by those with permissions to do so, if it is desired to prevent a user from logging in without removing them. However, 'removing' a user through the interface will not permanently remove their related database records either, but marks them as inactive, and the records of inactive users will then not be displayed or be modifiable in the normal UserDB interface.

2.4.3 Media File Submission and Management

To store the media content on the server, a set of subroutines was created to create a directory structure based on the IDs of the semantic type nodes of the Semantic Network. The top directory for the content directory tree is set in a configuration file. On the development server it is currently located outside of the `public_html` tree for security purposes. There is a temporary directory immediately under the root content directory where images are stored after they are uploaded and while annotation is ongoing. After going through the expert annotation system, they are moved from the temporary directory to the directory in the tree corresponding to the Semantic Network node related to the 'primary concept' that is assigned. There is a utility page intended for system administrators as part of the application interface which can list the number of files in each content directory, as well as any directories that are missing, and can also re-create the directory structure in the case of the content tree being moved or for re-adding directories that may have been removed. The Perl module `Tree::MultiNode` is used for creating a tree structure in memory from the Semantic Network `srstr` database table; the tree is then used to produce a list of fully qualified directory paths corresponding to every directory under the content root directory (Burton, 2002). The depth of the content folders does not have to represent the full depth of the Semantic Network- the depth to be used is read from the configuration file. By default, this option is set to a depth of four (not including the root). The benefits of structuring content storage in this manner are two-fold: first is an organizational advantage, as it may be easier to find a file manually if needed when stored in a logical structure rather than trying to include more information in filenames to differentiate files while storing them all in a single directory. It would be expected that there would be fewer possible situations where files with the same name are be stored in the same folder. The second motivation for this sort of content structure is to avoid reaching any filesystem limits for the number of files per directory, if such a limit exists. The filesystem of the

current server is EXT3, which has a limit of 31,988 subdirectories per directory, and a maximum number of directories and files (combined) determined by the number of formatted blocks, but does not have a set number of files per directory. However, it has been found that filesystem performance often decreases when a certain number of files per directory is exceeded, and so such a structure may assist in avoiding those performance hits even with EXT3. Some filesystems do have hard limits on the number of files that may exist in a single directory, for example FAT32 has a limit of 65,534. The popular NTFS filesystem, like EXT3, does not have any specific limit on the number of files per directory, but will take longer to access files in directories with larger numbers of files(Microsoft, 2003)(Tom's Hardware, 2004).

As already noted, the media files themselves are not stored in the database. To store such a quantity of large binary data files would involve treating them as Binary Large Objects (BLOBs), which introduces many manageability, speed, and possibly also stability issues. The database is still needed to keep track of files in the system and store associated file metadata, so a set of media-related tables was created (see structure in figure 2.3). One of these tables- *ontolink*- is also responsible for storing the CUIs of the Metathesaurus concepts that a file is linked to. A single file may be linked to multiple concepts, a single concept may be linked to multiple files, and we also wish to keep track of the users who added the annotations so *ontolink* has a composite primary key consisting of *media_id*, *concept_id*, and *user_id*. The semantic type for any concept can then be retrieved from the MRSTY Metathesaurus table, for when the general semantic category of a concept is needed. The *media* table stores details about the media files themselves- including the path and filename, media type (image, audio, video, etc.), file size, format (the compression or container format, such as JPEG), the date and time the file was uploaded, the status of the file in the system (if it's queued for processing in the expert system, or if it's already been fully annotated), and the dimensions of the displayed

image (for video and image files). Other metadata expected to be entered by users at the time of upload is stored in the *usermetadata* table, including the creation or generation date of the media file, the relevant location (if any), the method of generation or creation of the file, the creator (if not the submitting user), any copyright restrictions, and any other comments the user may have (specifically those which might assist in annotation). The *media_data_dic* table serves as a data dictionary for some of the fields in these content-related tables. This data dictionary is a manually populated table containing possible acceptable values for fields like the copyright restrictions and the media data format, along with a definition or description of that term (which is optional). For example, records with attribute_type 'copyright' might include the various Creative Commons licenses along with a description of the associated license terms, as well as 'public domain', 'all rights reserved', etc. This can then be used to create user interface elements, such as pre-populated drop boxes from which a user can select an option, or for including a definition of a term in a hovering dialog box or tooltip.

Users submit files via a form in the web application, where they are able to input any appropriate metadata (just described) as well as select the file to upload. The file will be transferred to the server upon submission of the form. No processing of the file is done on the client side, so if the file is large, the full file will be sent to the server and may take some time to transfer depending on the connection speed. After receiving the file, the Perl module `Image::ExifTool` is used to read data about the file. Although the primary purpose of `ExifTool` is to read the EXIF metadata tags commonly added to image files, it is also able to read tags from a variety of other media files including most audio and video files, and can display the file type of many other types of files (Harvey, 2009). If the uploaded file is found to be a valid file and was able to have the file type extracted by `ExifTool`, the type is checked against a list of acceptable types. For security purposes, files which do not match an accepted type will not be stored, and

an error will be returned to the user. Such an error may also indicate that the upload was incomplete or that the file was corrupt. If the filetype is of an accepted type, the file will be written to the disk under the configured content directory, and the details for the file including the user-submitted metadata will be inserted as records in the database. ImageMagick and the associated Perl module interface ImageMagick::PerlMagick is used to create preview-sized (thumbnail) files if it is determined that the dimensions of the submitted file are too large for normal browser-based display; this file is also written in the same directory as the original file, but with a “_preview” suffix on the filename (ImageMagick). These preview files are always JPEG files, and thus always have the ‘.jpg’ extension. A confirmation page is then displayed to the user, which includes the preview image just generated or the original image if not too large, as well as select EXIF fields read by Image::ExifTool (if the file is of an appropriate type) and the metadata that they entered. The user-entered metadata fields are editable on the confirmation page to allow the user to correct any errors, and add or remove any information before final confirmation. If they find that the wrong file was selected for upload, the confirmation page also offers the ability to delete files that were just uploaded. The preview image shown should help them confirm that the file that was uploaded is the file that was expected. Files are not considered eligible for annotation until the submission is confirmed on this page, which will also commit any metadata changes that were made.



Figure 2.3 - Database tables for media content

2.4.4 Expert Annotation System

The ‘expert annotation system’ component of the BioMeRSA system is essential for the system to adequately serve its purpose. Participation of users deemed as ‘experts’ will be necessary for content to be properly annotated. An ‘expert’ is differentiated by their having permission to utilize the annotation system. Files are not be available for searching, browsing, or downloading by other users until they have had their annotation ‘phase’ completed, as indicated by the file status in the database (the *STATUS* field shown in the media table in figure 2.3). This process involves the ‘experts’ assigning appropriate concepts from the Metathesaurus to the media files. Concepts can be browsed or searched to find the concept(s) that the user wishes to assign. Each file may be associated with multiple concepts, and a concept can be associated with multiple files. For each file, it will be expected that a simple majority (greater

than 50%) of the pool of experts will submit their annotation. All annotations added by the experts will be shown with the file once the annotation phase is completed. The concept with the highest number of 'votes' (annotations) will be assigned as the primary concept for the file, and the semantic type of that concept will be used to determine where the file will be written in the content directory structure, as mentioned earlier.

2.4.5 Searching and Browsing

This system offers a few different modes for end users searching or browsing for content to view or download. On the least specific level, users can browse a tree structure populated with the nodes of the Semantic Network. This tree is rendered using the Dojo framework's Dijit.Tree class. When a node is clicked, a list of the concepts under that node (defined in the Metathesaurus table *MRSTY*) is displayed. There is also the option to view a list of all the media files classified under all of the concepts that are assigned to that semantic type. There is a search box for searching for semantic type nodes in the tree. In another available browsing mode, the concepts themselves can be directly searched, and a list of the available image media assigned to a concept will be displayed when a concept is selected. When a file is selected, a preview will be displayed (if it is an image) along with the stored metadata for the file and the relevant concept names and semantic types. If the logged-in user has the appropriate permissions to download the file, a download link will also be displayed.

2.5 Testing

Due to the compatibility issues already mentioned, testing in different web browsers is important to make sure that the system works as expected under different operating environments. During development, testing was primarily performed using Mozilla Firefox 3 in Windows XP, with the Firebug extension for viewing and diagnosing JavaScript, display, CSS, and data transport errors. However, it is also desirable to ensure that the system works as expected

with Apple's Safari, Opera, Internet Explorer (version 6+) and older versions of Firefox (version 2+) under Linux, OS X and Windows (for all browsers except for Internet Explorer, which is only supported under Windows, and Safari, which is only supported under Windows and OS X) . In situations where pages failed to render as expected and Firebug was not producing error messages, the W3C HTML Validation Service was used to diagnose errors in the HTML (World Wide Web Consortium). To test that the full workflow of the system performs as expected, sample media files were added. These files were derived from the collection of scanned slides which was the original inspiration for the project. These slide scans are particularly good for testing, as they are TIFF-format files of almost 70 megabytes- thus demonstrating the ability of the system to handle large files, and to properly produce and display the preview files for source images that are high resolution and not originally in JPEG format. Other files were used to make sure that only desired file types are accepted, and that video and audio files are consistently detected and appropriately handled.

3. Results

3.1 User Administration, Authentication, Access Control

The user and account management interface of BioMeRSA was the first component to be developed, from a small basic codebase that existed from a previous project. This base served to define the direction of development for UserDB, and also influenced the rest of BioMeRSA. Both parts utilize tables in a MySQL database, are written in Perl using the CGI::Application framework, and have a web front-end as the primary user interfaces. All of these traits are shared by the code on which UserDB was based. UserDB is designed to be a simple interface for displaying and modifying the data in the underlying database, which is then used by the rest of the BioMeRSA application.

To persistently store the required data for each user, such as their login name and access permissions, as well as for keeping track of user logins for security and logging purposes, tables in the MySQL database were created. The tables directly related to users and authentications are *users*, *userpermissions*, and *logins*. The *user* table stores the information for identification of the user and their status, *userpermissions* keeps track of the access each user has to features in the system (both in UserDB and BioMeRSA), and *logins* acts as a log file where each user session and its status is recorded. The columns and structure of the tables can be seen in figure 3.1. Originally, only the user ID, username, password, locked status, and full name (one field for first and last name combined) were stored in the *users* table. This was determined to be inadequate, and the field for the user's name was split into two fields (first and last name), adding a field for their e-mail, RIT status (i.e. student, faculty, not associated, etc.), and active status. A single-field 'level' column for specifying a number corresponding to the access privileges of users in the original codebase was removed in favor of a full table for discrete per-action permissions. Note that the 'active' field is different from 'locked'- users whose accounts are locked (have a '1' stored in the 'locked' field) are considered current users, but do not have

access to the system until the lock status is set back to '0'. However, if a user has a '0' stored for 'active', they are considered to have been effectively deleted. Having a field to specify active users is more flexible than actually deleting records- it can then be possible to re-activate an account later on should it be necessary.

While it is intended that UserDB will be modified to utilize CGI::Application::Plugin::Session for session management (see section 3.2), at present all sessions are tracked manually. This is done by creating an entry in the *logins* table every time a user logs in successfully, which includes the date and time information for the login. Every time the user performs an action, the date/time is checked to see if the time out period has elapsed (set in the configuration file). If the session has not yet timed out, the date and time fields are updated with the current time, and if the session has expired, it is marked as inactive in the database (note that it is not deleted) and the user will be redirected to the login page. Sessions are tracked using a random hexadecimal 'session ID' value of configurable length (currently set at 64 characters) which is appended to the end of the query string, and is thus passed from the server to the client and then back from the client to the server so that the session can be maintained. More information on sessions and the necessity of session management is found in section 3.2.

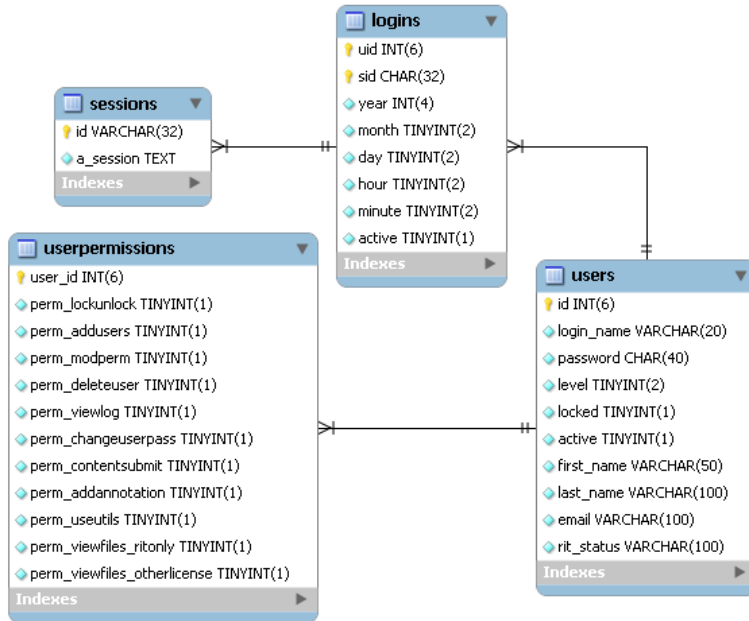


Figure 3.1 – ER diagram for user management database tables

Logging in to UserDB is done by validating the credentials- a username and password- against the information already stored in the database. For security, the passwords are stored in the database after being encrypted with the SHA-1 cryptographic algorithm, so passwords cannot be simply read from the database. SHA-1 is still considered to be relatively secure- while it has been shown that attacks against it are theoretically possible, collisions against the full algorithm have yet to be demonstrated(Henning, 2006)(Schneier, 2008). Users log in via a simple Web form (see figure 3.2) which sends the information to the server, which performs the validation, and returns a message informing them that the login attempt failed if the provided information did not match any users in the database. For the validation, alphabetic case is ignored when looking for usernames in the database, but passwords are case-sensitive.

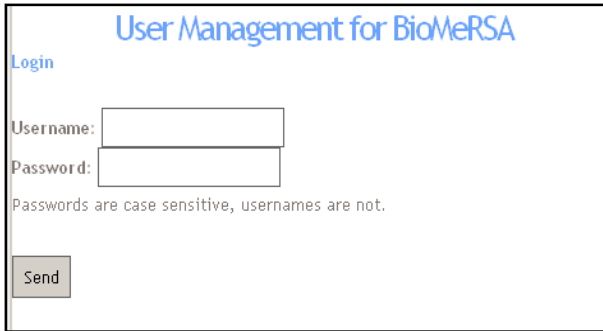


Figure 3.2- Screenshot of the UserDB login screen

Once a user has logged in, they are redirected to a menu which displays the actions they can perform, which is dependent on the access permissions set on their account. The menu with all of the available options displayed is shown in figure 3.3. Permissions for a user are set in the *userpermissions* table upon the creation of a user account, which is done by those with 'perm_addusers' privilege (in the *userpermissions* table) through the 'Add a User' page (figure 3.4). Every time the user loads a page, performs an action which retrieves data from the server, or alters data in the database, a check is done to make sure they have the access rights to do so. This is necessary for data security and integrity.

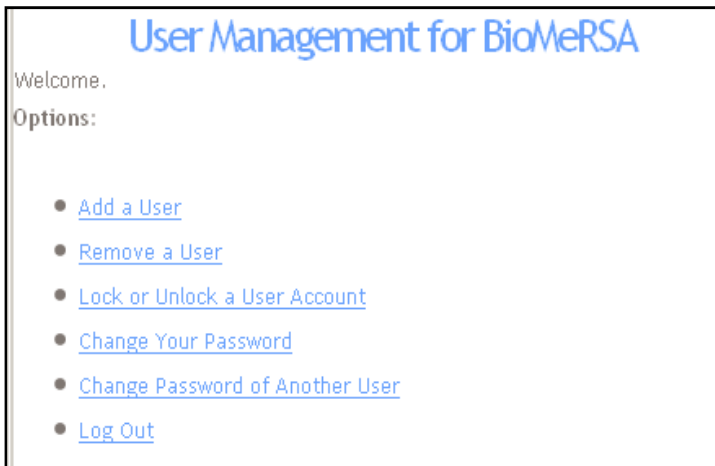


Figure 3.3- Screenshot of the UserDB main menu with all options displayed

User Management for BioMeRSA

Add new users here

Add User

User name:

First name:

Last name:

Email address:

Password:

Please repeat password:

Select permissions, if any, to grant this user:

Select	Permission Name
<input type="checkbox"/>	Lock/Unlock User Accounts
<input type="checkbox"/>	Add User Accounts
<input type="checkbox"/>	Modify User Permissions
<input type="checkbox"/>	Remove User Accounts
<input type="checkbox"/>	View Activity Logs
<input type="checkbox"/>	Change Password of Other Users

Figure 3.4- Screenshot of the UserDB interface to add new users

Users can also be removed (figure 3.5), locked (figure 3.6), change their own password (figure 3.7), or change passwords of other users (figure 3.8). Of all of the possible actions described for UserDB, the only one accessible to any user with an active account regardless of permission settings is that to change their own password. All other actions require that the proper permissions be set. In lieu of a more complex system to allow any user to reset their own password, for simplicity it was determined to allow administrators to set the passwords of other users manually, in the case that they are contacted by a user who may have forgotten theirs, for example. The user can then set their password again themselves, after logging in with the

password given by the admin. Logged-in users can also log out, which will remove their session from the *sessions* table, and mark their session as inactive in the *logins* table.

User Management for BioMeRSA
Remove A User Account

Select a user to remove:

Select	User ID	User Name	First Name	Last Name
<input type="radio"/>	10	billybob	Billy	Bob
<input type="radio"/>	8	abc5565	Adam	Cornwell
<input type="radio"/>	9	johndoe	John	Doe
<input type="radio"/>	5	testuser	Test	User

[Main Menu](#)

[Log Out](#)

Figure 3.5-Screenshot of the UserDB interface to remove users

User Management for BioMeRSA
Lock/Unlock User Accounts

Select User(s) to lock or unlock:

Select	User ID	User Name	First Name	Last Name
<input type="checkbox"/>	10	billybob	Billy	Bob
<input type="checkbox"/>	8	abc5565	Adam	Cornwell
<input type="checkbox"/>	9	johndoe	John	Doe
<input type="checkbox"/>	5	testuser	Test	User

[Main Menu](#)

[Log Out](#)

Figure 3.6- Screenshot of the UserDB interface to lock users

User Management for BioMeRSA

Change Password

For user Adam Cornwell

Current Password:

New password:

Enter new password again:

[Main Menu](#)

[Log Out](#)

Figure 3.7- Screenshot of the UserDB interface for users to change their own passwords

User Management for BioMeRSA

Change User Passwords

Select a user whose password to modify:

Select	User ID	User Name	First Name	Last Name
<input type="radio"/>	10	billybob	Billy	Bob
<input type="radio"/>	9	johndoe	John	Doe
<input type="radio"/>	5	testuser	Test	User

New password for user:

Enter new password again:

[Main Menu](#)

[Log Out](#)

Figure 3.8- Screenshot of the UserDB interface for changing passwords of other users

3.2 Session Management for BioMeRSA

CGI::Application::Plugin::Session, an interface for CGI::Session that is more tightly integrated into CGI::Application, was chosen to assist with managing user sessions for BioMeRSA (Stosberg, CGI::Session - persistent session data in CGI applications, 2009)(Hek, 2006). UserDB will be altered to also utilize this system of session management in the near future, but this effort has not been completed at the time of writing. Utilizing CGI::Session enables more advanced functionality while involving less code per run mode subroutine than the approach

currently utilized in UserDB. Sessions are necessary because the Web is otherwise stateless, and no information is retained between transactions unless there is a system on the client or the server specifically built to store and retrieve it(Adiscon GmbH, 2001)(Seebach, 2008).

CGI::Session generates a random 32-character sequence from the MD5 cryptographic algorithm known as the 'session ID' which is stored on the server side in a database (the *sessions* table) and on the client side in a browser cookie. This session ID is generated every time a client visits the site for the first time, is found to have an expired session, or after their login has been validated. When the session ID is stored on the server, other information such as the client IP address and the time of the users' last action are also recorded. The cookie is passed to the server from the client with every server request, and passed from the server to the client with each server response. The 'a_session' field in the *sessions* table is used to store all other data to be associated with the session- CGI::Session by default uses Data::Dumper to store this data in name-value pairs. When the session is accessed and it is found that the duration between the current time and the time of the last action taken by the user has exceeded a timeout value (set in a configuration file) the session is considered to be expired and it is deleted and recreated with a new session ID. Due to this mechanism of refreshing sessions, *sessions* does not serve as a permanent record of user logins. To achieve such a record for the sake of statistics and security, there is the *logins* table, in which there is an entry for each successful login. Thus, as can be seen in figure 3.1, it can be considered that the session ID field of *sessions* is a foreign key of the session ID field of *logins*, because *logins* will contain all the valid session IDs. As a safety measure against situations where an attacker could obtain a valid logged-in session ID by getting a session ID, and then tricking a user into logging in with that session ID, sessions are recreated with a new ID upon successful login.

3.3 Semantic Data – UMLS

The Unified Medial Language System (UMLS) was chosen as the source of the semantic data with which media files can be associated. Due to its size, and to maximize the number of possible applications of the UMLS data, a utility is provided (MetamorphoSys) to generate a selectable subset, which is written to data files with specific formats. The raw data files downloaded from the UMLS Knowledge Source Server are in a binary format and must be processed by MetamorphoSys before the data can be utilized. The subset which was used during the course of development was generated from UMLS Version 2008AA (April 2008) and included Level 0 sources (those that do not require license agreements beyond that of the default UMLS license) as well as data from SNOMEDCT (which has its own license agreement). Only English sources were selected. The selected source list used for subset generation can be found in table 3.1. Note that some of these sources are listed in the table with zero associated concepts. These sources are not used for generation of terms for concepts, but are instead used for determining co-occurrences (Emrick, 2009).

Table 3.1- UMLS sources selected for subset generation

Source Name	Abbrev.	#Concepts
Authorized Osteopathic Thesaurus	AOT2003	278
Beth Israel Vocabulary	BI98	939
Current Dental Terminology	CDT2007-2008B	582
Medical Entities Dictionary	CPM2003	3078
CRISP Thesaurus	CSP2006	16694
Diseases Database	DDB00	169
DSM-III-R	DSM3R_1987	371
DSM-IV	DSM4_1994	452
Gene Ontology	GO2007_02_01	25537
HCPCS Version of Current Dental Terminology	HCDT2007-2008	582
ICP2E-ICD10 relationships from Dr. Henk Lamberts	HLREL_1998	0
HUGO Gene Nomenclature	HUGO2007_01	24082
ICD10, American English Equivalents	ICD10AE1998	1005
ICD10	ICD10_1998	11528
ICPC2 – ICD10 Thesaurus	ICPC2ICD10ENG_200412	38042
ICPC-2 PLUS	ICPC2P_2005	7183
Online Congenital Multiple Anomaly/Mental Retardation Syndromes	JABL99	746
MEDLINE (1998-2002)	MBD08	0
McMaster University Epidemiology Terms	MCM92	41
Master Drug Data Base	Mddb_2008_01_03	11471
MEDLINE (2003-2008)	MED08	0
MedlinePlus Health Topics	MEDLINEPLUS_20040814	1268
Multum Medisource Lexicon	MMSL_2008_01_01	50973
Micromedex DRUGDEX	MMX_2008_01_02	9405
Medical Subject Headings	MSH2008_02_04	286425
UMLS Metathesaurus	MTH	106140
NCBI Taxonomy	NCBI2007_07_02	308335
National Cancer Institute Thesaurus	NCI2007_05E	62969
National Drug Data File Plus Source Vocabulary	NDDF_2008_01_04	30375
Neuronames Brain Hierarchy	NEU99	814
National Library of Medicine Medline Data	NLM-MED	0
Online Mendelian Inheritance in Man	OMIM2007_12_19	55300
Quick Medical Reference	QMR96	940
QMR clinically related terms from Randolph A Miller	RAM99	210
Clinical Terms Version 3	RCD99	186109
Read Thesaurus, American English Equivalents	RCDAE_1999	11143
Read thesaurus, Americanized Synthesized Terms	RCDSA_1999	821
Read Thesaurus, synthesized terms	RCDSY_1999	9116
SNOMED-2	SNM2	35205
SNOMED International	SNMI98	112702
SNOMED Clinical Terms	SNOMEDCT_2008_01_31	308677
Standard Product Nomenclature	SPN2003	4809
ULTRASTAR	ULT93	84
UMDNS: product category thesaurus	UMD2008	12564
University of Washington Digital Anatomist	UWDA173	61376
WHO Adverse Reaction Terminology	WHO97	3168

MetamorphoSys was initially run locally under Windows XP 32-bit on a quad-core Intel Q9400 machine with 2.5GB usable system RAM. MetamorphoSys is written in Java, and is bundled with the Java JRE version 1.5. Initially, the application was failing to start under Windows, and it was found that the application defaulted to having the JVM attempt to allocate 1000MB of memory on startup. Further research indicated that the JVM attempts to allocate contiguous blocks of memory, and while Windows indicated that more than 1000MB of memory in total was free, it was probably not contiguous (Rajas, 2004). Setting MetamorphoSys to attempt to allocate 500MB of memory allowed the program to start. Initial attempts at subset creation under Windows resulted in running out of disk space after four hours. It was also noted that the process did not appear to be CPU intensive and often utilized less than 25% of the CPU capacity (one core of the four cores available), and thus might not be multithreaded under Windows, or has a performance bottleneck in the disk subsystem. A useful feature of MetamorphoSys is the ability to export subset settings to a file, so that settings do not have to be re-set every time a subset is to be generated. This allowed the source data files and the subset settings file to be copied to the development server, and the subset was successfully generated on there. There is some possibility that the program is optimized to run under Linux, as it was noted that the process appeared to be progressing faster, and Top was showing almost 100% CPU use for the process on a CPU capable of executing two threads in Linux, versus the 25% CPU utilization on a system capable of executing four threads in Windows XP. This difference could also be attributed to optimizations of the Java Virtual Machine itself under Linux- not enough testing was done to determine the cause of the discrepancy. Subset generation was fully completed in nine hours under Linux on the development server.

As was mentioned in Materials and Methods, MetamorphoSys was configured to output the subset in UMLS Rich Release Format (RRF) files, a newer format which is able to more

accurately represent the semantics of the source data when compared with the Original Release Format (ORF), and from which complete change sets between Metathesaurus versions can be generated. RRF is the format currently recommended by UMLS. MetamorphoSys generates multiple RRF-format flat files, and each of these files is designed so that it includes fields which are logically equivalent to the columns of a table in a relational database. Thus, it is not difficult to take it a step further, and parse the data into such a database. The RRF format helps to facilitate this- all fields are separated by pipe ('| ') characters and rows are separated by newlines. MySQL supports reading data into tables directly from text files of a certain format, and allows specification of the field split character and of the row split character. The Semantic Network and Metathesaurus subset data files are written to two different directories by default, named NET and META respectively. Files of both data sets have the same format. Although the Semantic Network data files are not referred to as being RRF-type files specifically, they adhere to the same syntax and are loaded into the database using the same SQL syntax. Regardless of the data chosen for output in the subset, all possible data files are created, even if some do not contain anything. Metathesaurus data files are named starting with 'MR' for 'Metathesaurus Relational' and Semantic Network data files are named starting with 'SR' for 'Semantic Relation'. The rest of the filename is an abbreviated description of the content of the file. For example, 'MRCONSO.RRF' is the Metathesaurus file for CONcept Names and SOurces (National Library of Medicine, 2009)(National Library of Medicine, 2009). The complete list of files created by MetamorphoSys is shown in table 3.2. Files with a file size of zero do not contain any data. Note that many of these were the word index files (MRXW_*) for non-English languages, the sources of which were not selected for inclusion in the subset.

Table 3.2- Complete list of files created by MetamorphoSys as part of UMLS subset generation

FILENAME	FILESIZE (bytes)
Semantic Net- NET	
SRDEF	52123
SRFIL	435
SRFLD	683
SRSTR	30898
SRSTRE1	109824
SRSTRE2	349432
SU (Note: not a relational format file)	85884
Metathesaurus- META	
AMBIGLUI.RRF	1827211
AMBIGSUI.RRF	1177981
MRAUI.RRF	6090541
MRCOC.RRF	977541161
MRCOLS.RRF	20459
MRCONSO.RRF	429744389
MRCUI.RRF	20053303
MRCXT.RRF	0
MRDEF.RRF	28358285
MRDOC.RRF	150297
MRFILES.RRF	3436
MRHIER.RRF	1246427717
MRHIST.RRF	120411993
MRMAP.RRF	945924
MRRANK.RRF	4434
MRREL.RRF	1218112306
MRSAB.RRF	95713
MRSAT.RRF	1635642042
MRSMAP.RRF	730094
MRSTY.RRF	83424166
MRXNS_ENG.RRF	270647926
MRXNW_ENG.RRF	511599306
MRXW_BAQ.RRF	0
MRXW_CZE.RRF	0
MRXW_DAN.RRF	0
MRXW_DUT.RRF	0
MRXW_ENG.RRF	495492655

MRXW_FIN.RRF	0
MRXW_FRE.RRF	0
MRXW_GER.RRF	0
MRXW_HEB.RRF	0
MRXW_HUN.RRF	0
MRXW_ITA.RRF	0
MRXW_JPN.RRF	0
MRXW_NOR.RRF	0
MRXW_POR.RRF	0
MRXW_RUS.RRF	0
MRXW_SWE.RRF	0
Metathesaurus- Change tracking files	
DELETEDCUI.RRF	136281
DELETEDLUI.RRF	880475
DELETEDSUI.RRF	1155335
MERGEDCUI.RRF	23161
MERGEDLUI.RRF	18848

As mentioned in Materials and Methods, MetamorphoSys is able to generate database table creation and data population scripts compatible with MySQL and also Oracle (two separate sets of script files). When these load scripts were tested on the development server, it was discovered that they utilized a feature which was introduced in MySQL 5 and is not found in MySQL version 4, resulting in syntax errors when running the scripts on the server (which is running MySQL 4). MySQL 5 allows statements which can transform the data during import, which was being used by the generated scripts to turn blank fields ('| |') in the RRF files into 'NULL' values for insertion in the database in columns where nulls are allowed. A workaround was to install MySQL 5 on a different machine, use the MetamorphoSys-generated script to import the data, and then write another SQL script to export the data into files of the same format, but now already transformed with NULLs in the appropriate fields. These files were then copied to Bucatini, the development server, and a modified load script was then used to import the data into tables under MySQL 4. Also prior to re-importing the data in MySQL 4, the script

was modified so that tables would not be created that did not have any rows- these were tables corresponding to non-English language data, of which no sources were selected during subset generation. The same process was similarly performed later on the Semantic Network data files for the creation of the Semantic Network database tables. The list of tables created in the database corresponding to the Metathesaurus data and the number of rows inserted to each table can be seen in table 3.3. Table 3.4 shows the same, but for the tables originating from Semantic Network data. In table 3.3, note again that tables without any rows were left out of the database on the development server.

Table 3.3-Statistics for Metathesaurus tables after importing the data into MySQL 5 using the load script generated by MetamorphoSys

DB Table Name	# Cols	# Rows (after importing to MySQL 5 using original generated load script)
AMBIGLUI	2	96169
AMBIGSUI	2	61999
DELETEDCUI	2	2289
DELETEDSUI	3	18231
MERGEDCUI	2	1219
MERGEDLUI	2	992
MRAUI	9	114187
MRCOC	9	17195740
MRCOLS	8	298
MRCONSO	18	3697658
MRCUI	7	687202
MRCXT	15	0
MRDEF	8	105969
MRDOC	4	2354
MRFILES	6	43
MRHIER	9	8637555
MRHIST	9	1855788
MRMAP	26	7381
MRRANK	4	262
MRREL	16	14519056
MRSAB	25	150
MRSAT	13	20201448
MRSMAP	11	7381
MRSTY	6	1432403
MRXNS_ENG	5	4276227
MRXNW_ENG	5	12934853
MRXW_BAQ	5	0
MRXW_CZE	5	0
MRXW_DAN	5	0
MRXW_DUT	5	0
MRXW_ENG	5	12613283
MRXW_FIN	5	0
MRXW_FRE	5	0
MRXW_GER	5	0
MRXW_HEB	5	0
MRXW_HUN	5	0
MRXW_ITA	5	0
MRXW_JPN	5	0
MRXW_NOR	5	0
MRXW_RUS	5	0
MRXW_SPA	5	0
MRXW_SWE	5	0

Table 3.4- Statistics for Semantic Network tables after importing the data into MySQL 5 using the load script generated by MetamorphoSys

DB Table Name	# Cols	# Rows (after importing to MySQL 5 using original generated load script)
SRDEF		189
SRFIL		6
SRFLD		21
SRSTR		612
SRSTRE1		6864
SRSTRE2		6864

After working around the incompatibility between the MetamorphoSys database load scripts and MySQL 4, an inconsistency was found in the number of records for one table- 'MRSAT' had 20,201,448 records in the database in MySQL 5, and then later 20,201,866 records in MySQL 4, an increase of 418 rows. It was found that there were unescaped backslashes ('\') in the original data files. The backslash is considered a special character in MySQL. Whenever the backslash was followed by an 'n', it was treated as a newline character, and since the column these appeared in has type TEXT, a carriage return was inserted. Upon export, this caused records to be split into multiple rows, as the record delimiter was set to be the newline. This was resolved by replacing the single backslashes with escaped backslashes ('\'). It was noted that there were 57,079 backslashes total found in MRSAT.RRF, and when not followed by an 'n' the backslash was removed but did not affect the rest of the row. Most of these were found in the fully qualified Gene Ontology terms. As a precaution, the other source data files were then checked for the presence of backslashes, in which only seven in total were found; five in MRCONSO.RRF and two in MRDEF.RRF. The process that led to the discovery of the inconsistency is shown in figure 3.9, and the procedure used to resolve it in figure 3.10. The Semantic Network source data files were also checked for unescaped backslashes, but none were found. There were no inconsistencies in the Semantic Network tables. In total, including

the Semantic Network, 32 tables were created and populated in the development database, with a total of 98,484,693 records. It was later determined that some of these tables do not hold relevancy to BioMeRSA in its current scope, and these tables, while still in the database, are not accessed in the application.

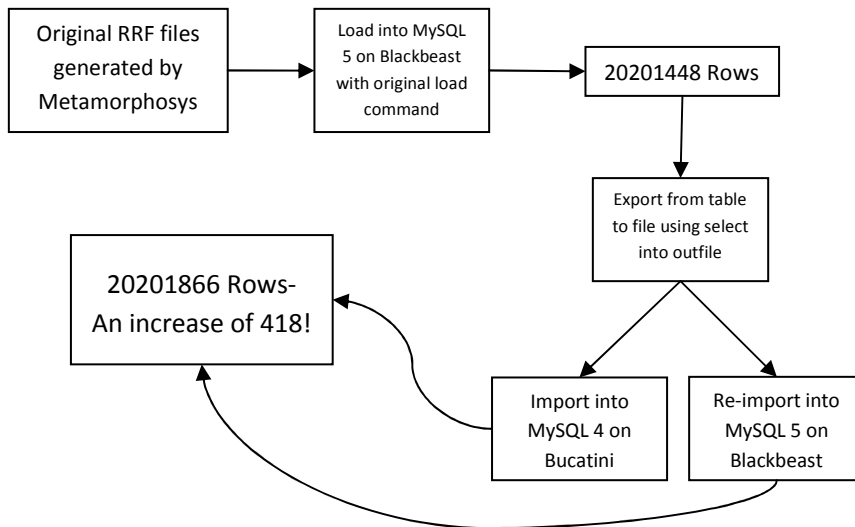


Figure 3.9- Discovering the inconsistency number of records after importing and then exporting the UMLS source data files.

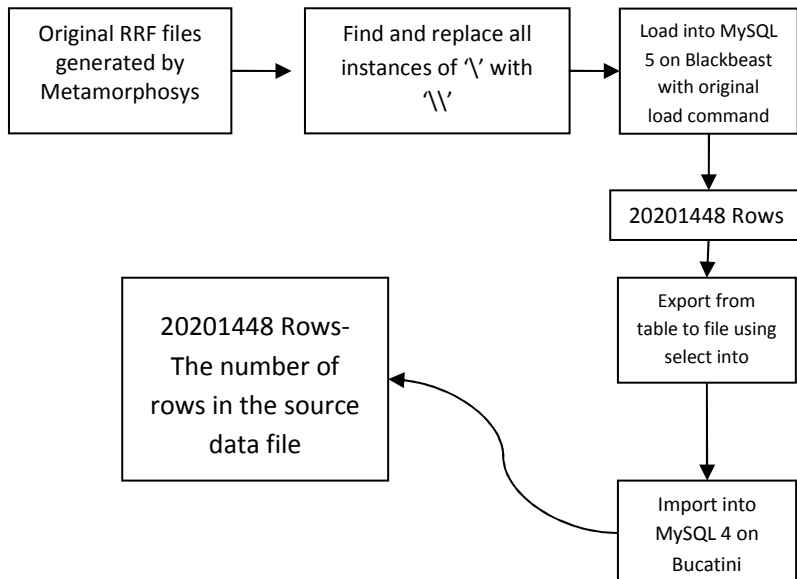


Figure 3.10- The solution to the record inconsistency.

Table 3.5- UMLS Database Tables Considered for use in BioMeRSA. Those highlighted are the ones actually accessed in the current implementation of the system.

Table Name	Description
Metathesaurus Tables- Concepts, Relationships, and Source Data	
MRCONSO	Concept structure
MRCOC	Co-occurrence relationships (not found in MRREL)
MRHIER	Hierarchical relationships (Only the direct parent-child relationships are also found in MRREL)
MRREL	Primary relationship data
AMBIGSUI	Ambiguous strings- might be useful for making suggestions for other searches
AMBIGLUI	Ambiguous terms - might be useful for making suggestions for other searches
MRSTY	Links concepts with semantic types
MRSAB	Source vocabulary information
Metathesaurus Tables- Definitions and Labels	
MRDEF	Definitions
MRDOC	Relationship labels
Metathesaurus Tables- Word Indices	
MRXW_ENG	word index
MRXNW_ENG	Normalized word index
MRXNS_ENG	Normalized string index
Semantic Network Tables	
SRSTR	Semantic Network structure
SRDEF	Semantic type/relation definitions
SRSTRE1	Fully inherited Semantic Network relationships- Identifiers only
SRSTRE2	Fully inherited Semantic Network relationships- names only

Table 3.5 describes the UMLS database tables deemed useful to the BioMeRSA application, and the ones which were found to be needed for the functionality which is currently implemented in the application. Some of the other tables will be likely useful when more features are added in the future (see Section 4, Discussion). Many of the database tables created contain metadata about the Metathesaurus, such as the names of columns in each table, and change records between current and previous Metathesaurus versions, which did not seem to be useful in the current scope. The UMLS data is provided without any normalization, and so the MRSAT table was left out because the contained data was adequately provided by

other tables. Normalization was performed on some tables, in an attempt to impart a more formalized structure to the database, which is easier to visualize and trace any problems through. A specific normalization form was not used. In many cases, data from separate original tables had to be combined into a new table- these new tables are differentiated by their names being in lower-case. No existing UMLS tables were altered except for the addition of primary and foreign key constraints. New tables were created when data needed to be separated or combined. One of the most significant of these changes was the addition of the *conceptindex* table, which has one record per concept in the Metathesaurus- every table which contains a concept identifier (CUI) column has a foreign key to the CUI column of *conceptindex*. It was found that there are 1,221,057 unique concepts in version 2008AA of the Metathesaurus. The Semantic Network tables were more thoroughly normalized than those of the Metathesaurus, as the Semantic Network involves many fewer tables, with fewer columns each, and is thus more manageable. Figure 3.11 shows the results of the normalization effort on the Semantic Network. See Materials and Methods section for more details on the normalization methodology.

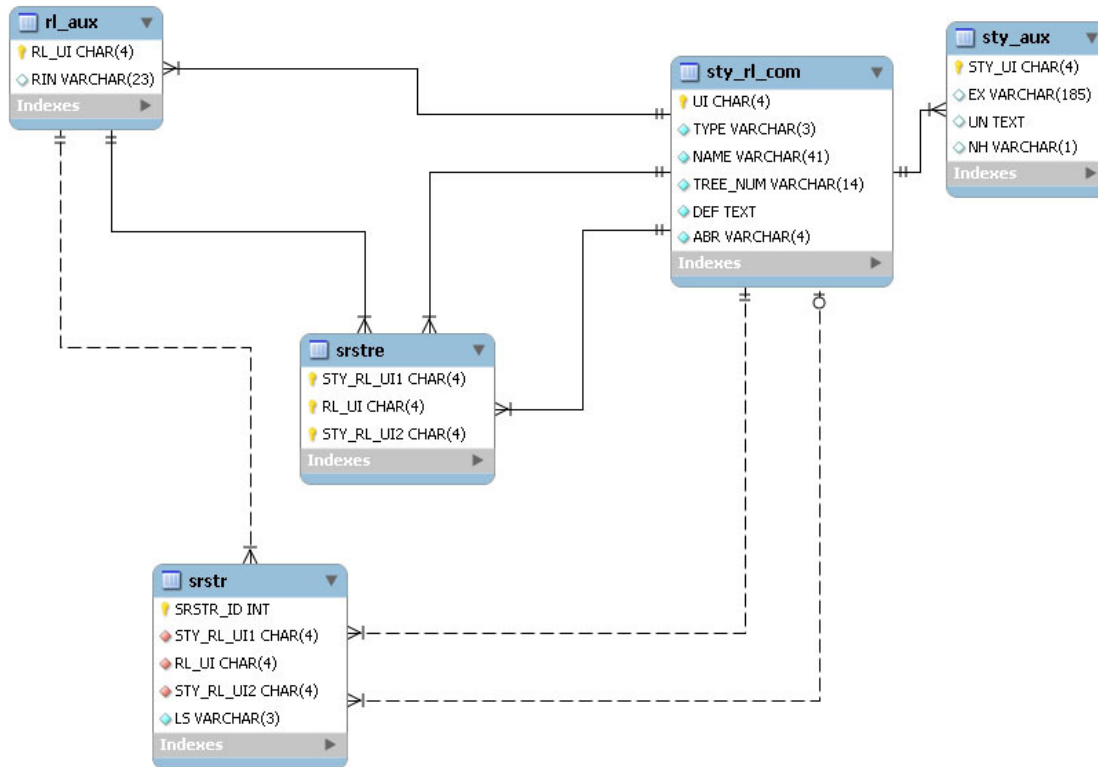


Figure 3.11- The modified Semantic Network database structure.

3.4 Content Submission and Management

It is expected that the majority of media that will be hosted by BioMeRSA will be submitted by users. As such, an interface for users to submit their content was necessary. The content submission interface is built on the same components as the rest of the BioMeRSA system, and is Web-based. This interface enables a user (with the appropriate privileges for content submission, as defined in the user management system already discussed) to upload a file as well as some relevant metadata about the file. At the moment, only single file uploads are supported, and there is no file transfer status indicator. While it would be most desirable if users could fill in all the metadata fields for every submitted file, some of the data may not be known for some files, and thusly the only required metadata field is that for setting the copyright on the file- this determines who can view the file in the system. Other metadata fields include the creation date of the item, the location at which it was generated or captured, the method through which it was generated or captured (lab technique or kind of imaging device), the

creator (which could be a company, individual, lab group, etc.), and a field for a description or other general comments. Figure 3.12 shows this interface.

The content submission form itself is not an AJAX-style form, meaning that the form is submitted synchronously, and the user cannot perform other actions on the page without disrupting the file upload. However, there are some Dojo widgets in use on the page. First, clicking in the creation date selection box brings up a calendar-like date selection box, where the month, day, and year can be selected. Users may still type in the date manually, but some may prefer the familiar calendar-like graphical selection box. The copyright field is an editable drop-down selector, which will fetch a list of copyright options from the server to populate the list upon being clicked. Since the box is user-editable, the user can specify other options that are not in the list. In the case that the connection is slow or the server request to populate the pre-entered options does not complete successfully, the user can still enter something in this field. As the field is required, JavaScript was used to enforce this by disabling the 'Submit File' button if the user removes the text from the copyright selection field. The default value is 'All Rights Reserved', although this default value is configurable in the BioMeRSAConfig file.

BioMeRSA- the Bio Media Repository with Semantic Augmentation
Welcome, Adam!

Home Submit Content Browse and Search for Content Log out

Enter Metadata

↓

Upload File

↓

Modify or confirm submission

Welcome to the content submission interface

Creation Date

Location of Capture or Generation

Method of Capture or Generation

Creator

Other Comments

Copyright Restrictions All Rights Reserved *required field

Select File to Upload Browse...

Submit File

Figure 3.12 The BioMeRSA Content Submission Interface

BioMeRSA- the Bio Media Repository with Semantic Augmentation

Welcome, Adam!

Home Submit Content Annotate Media Search by Metathesaurus Concept Browse by Semantic Network Search Media Metadata Log out ▾

Enter Metadata

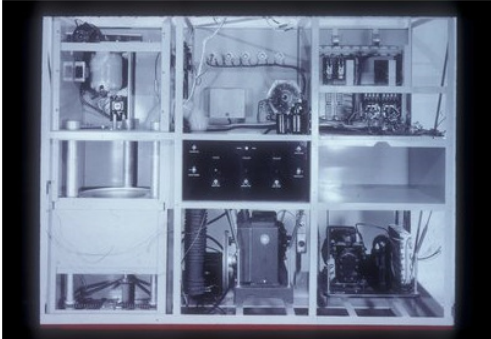
↓

Upload File

↓

Modify or confirm submission

Media content submission confirmation



Submitted file data:

File name:	abc55652009322033443.tif
Original file name:	Virology 79.tif
Date/Time of submission:	2009-3-22 0:34:10
File type : File format:	image : TIFF
File size:	67056 kilobytes
Display dimensions (W x H, in pixels):	3946 x 5782

Submitted metadata:

Creation Date:	<input type="text"/>
Location of Capture or Generation:	<input type="text" value="Unknown"/>
Method of Capture or Generation:	<input type="text" value="Unknown"/>
Creator:	<input type="text" value="Clay Adams, a division of Becton"/>
Copyright Restrictions:	<input type="text" value="All Rights Reserved"/> *required field
Other Comments:	<input type="text" value="Interior of analytical ultracentrifuge"/>

*Note: File will not be fully submitted until confirmed with the button below

Confirm File Submission Don't submit- Remove this file

Figure 3.13- Confirmation page for BioMeRSA Content Submission

After the upload of a file completes successfully, there is a submission confirmation page which presents the user with a preview of the file that they just uploaded (if it is an image file), as well as some file data read from the file and the metadata they entered. This page allows the user to correct and resubmit their metadata if a mistake was made, or delete the file if it turns out that the uploaded file is not the correct one. The preview displayed is a resized (if necessary) version of the submitted file if it is a supported image, or a generic placeholder image indicating the file type if it is an audio or video file. At the moment, in-browser previews are not available for non-image files. For reference, and to help ensure that the correct file was uploaded, the filename as written to the server disk is displayed, along with the filename as

submitted (the users' original name for the file), the time of submission, the type of the file, file size in kilobytes, and resolution in pixels if it is an image. If the user opts to remove the file, all record of it from the disk and database is removed at this stage- this is safe because at this point the file is not considered to have finished the submission process. The 'Confirm File Submission' button advances the file to the next phase of the process, and makes it available for annotation in the Expert Annotation System. Figure 3.13 is a screenshot demonstrating this interface.

As already mentioned in Materials and Methods, the media file binary data is not stored in the database. Files are stored in a directory that is located outside of the `public_html` directory on the server (however this is configurable in the BioMeRSA configuration file and could be changed if needed). The media-related database tables exist to keep track of the file metadata, file properties, metadata submitted by users about the files, and to link them to the semantic data (this last element will be covered more in-depth later). Four tables in the database handle this: *media*, *usermetadata*, *media_data_dic*, and *ontolink*. The *media* table was designed for storing metadata related to the properties of the file itself and the format of the file- storage path on the disk, file name, file size, dimensions (in pixels, if an image), file type (image, audio, video), file format (compression format), MIMEType (stored for use later when the file is redisplayed in the browser), a count of the number of times the file have been downloaded, and the status of the file in the BioMeRSA system. Each file also receives a unique ID, which is autoincremented with every uploaded file and can be up to 10 digits long, yielding an ID namespace capacity of over one billion items- server disk space is much more likely to be a limiting factor in the number of items that can be stored. The status of each file is expected to be one of four states at any moment: 'unconfirmed' (the file has been uploaded by the user, but they have not committed their submission to the next stage of the process), 'annotation' (the user has confirmed the submission and the file is in the expert annotation phase), 'live'

(annotation has been completed, and the file is viewable in the browsing/searching interfaces), and 'review' (the file was live, but was flagged for review for content or inaccuracy of annotation). At the moment, the review system is not implemented. The data which is stored in the *usermetadata* table originates from the file submission interface, as shown earlier in figure 3.12. The fields in the table match the fields in that interface, with the addition of a field to store the ID of the user submitting the file. The *media_data_dic* table is an accessory table used to populate some extra features of the user interface- for example, the list of copyright types which populate the drop-down in the file submission interface is populated from data stored in this table. It could also be used to store definitions and descriptions for various other interface elements, which could be used for making dynamic tooltips and help windows, for example. Finally, *ontolink* is the table which serves to connect media files with the concepts of the UMLS Metathesaurus. This provides the semantic context for the files, which is the primary purpose of this system. This table was designed with the expert annotation system in mind- multiple users can each associate multiple concepts with a single file, making the triple field composite primary key necessary. Additionally, annotating users can add a note along with each concept they associate with a file, and can also give a rating of their confidence in their judgment. The expert annotation system is covered in greater depth later. Figure 3.14 is the ER diagram of these four tables just described.



Figure 3.14- Database tables for storing details for media files

When a file is uploaded, it is initially renamed and placed in a temporary directory in a subdirectory named with the submitting user's username. Files are renamed using the following pattern: username + year + month + day + hour + minute + second + mediaID + original file extension. Note that there are no spaces or delimiters used in the actual names, for example, 'abc5565200922011563116.JPG'. All of these values except for the year are zero-based, so in this example case, the 2 would indicate the file was submitted in March. The files written to the server need to be renamed, as they need to be unique- it is quite foreseeable that a user could have two files in different directories with the same name but different content, which could even be the same file size. For instance, if an imaging procedure in a lab writes a series of bitmap files, with each series in its own directory, and with the files named sequentially as

'1.bmp', '2.bmp', then there are multiple files named '1.bmp' which have different content, but will likely be the same resolution (generated by the same system), and thusly the same file size (because they're uncompressed bitmaps). The server would have to compare the file data directly to be able to tell that the files were actually different, and given that they're being transferred from a remote system, there's then no way to guarantee that the differences are not due to corruption of data in the transfer process. Therefore, it is safest to rename the files. This does introduce the possibility of the user uploading multiple files that are actually the same. Re-naming the files also allows data to be stored in the filename- should the directory structure become corrupt or if some database records are lost, some key info is still stored in the filenames, which can be used to identify the files, or reconstruct the records.

Once a file has been annotated, it is moved from the temporary directory to a directory hierarchy based on the hierarchy of the Semantic Network. The specific directory a file is written to is based on the Semantic Network node for the 'primary concept' assigned to the file, which is the concept with the highest number of expert 'votes', or in case of a tie, the Semantic Network node for the concept with the highest average confidence out of the tied concepts. If consideration of both qualifiers results in a tie again, then the decision is arbitrary as the concepts are then equally suitable. This hierarchy of directories is pre-generated. A set of two utilities were created to check and fix these directories in the case that the decision is made to relocate the media directory, an update is made to the UMLS data which results in the addition of Semantic Types, or some of the directories are removed. One of these utilities shows the list of directories that are expected to exist in the media directory tree and the number of files and folders (combined) that are found there, if the directory exists. Figure 3.15 shows this interface. If any problems are found on this screen (if a directory shows up as not existing) then the 'create missing directories' button runs a method that creates the directories if they are not found, and

will not alter ones which already exist. This will also not alter any directories which may exist in the hierarchy that are not expected to be there. The output from this routine is shown in figure 3.16. After the files have been moved, and their status has been changed in the database, they are available for viewing by other ‘non-expert’ users. This directory hierarchy exists to organize what could be thousands of media files in a well-defined manner. Separating files into multiple directories also helps avert issues related to file system speed when too many files are under a single directory (see Section 2, Materials and Methods for more information).

BioMeRSA- the Bio Media Repository with Semantic Augmentation
 Welcome, Test

Home Submit Content Browse and Search for Content Log out

Check the media content directory structure
 Create missing content directories
 Modify configuration file
 Modify media data dictionary
 Clear the test files folder

Currently viewing the content directory structure.
 Clicking a column name will sort by that column.
 The number of files per directory includes contained directories, and only counts files at that immediate directory level and does not include files in subdirectories.
 If no directories exist or some are missing, try 'Create missing directories'.

Directory Path	Directory Exists?	Number of Files/Dirs
/home/ab-c5565/media/T071/T077	Yes	9
/home/ab-c5565/media/T071/T072/T001	Yes	7
/home/ab-c5565/media/temmedia	Yes	6
/home/ab-c5565/media/T071/T077/T078	Yes	5
/home/ab-c5565/media/T071/T077/T096	Yes	5
/home/ab-c5565/media/T051/T052	Yes	4
/home/ab-c5565/media/T051/T052/T057	Yes	4
/home/ab-c5565/media/T071/T072	Yes	4
/home/ab-c5565/media/T051/T067	Yes	3
/home/ab-c5565/media/T071/T072/T017	Yes	3
/home/ab-c5565/media/T071/T072/T073	Yes	3
/home/ab-c5565/media/T071/T072/T167	Yes	3
/home/ab-c5565/media/T071/T077/T092	Yes	3
/home/ab-c5565/media/T051	Yes	2
/home/ab-c5565/media/T051/T052/T053	Yes	2
/home/ab-c5565/media/T071	Yes	2
/home/ab-c5565/media/T071/T077/T033	Yes	2
/home/ab-c5565/media/T071/T077/T170	Yes	2
/home/ab-c5565/media/T051/T067/T068	Yes	1
/home/ab-c5565/media/T051/T067/T070	Yes	1
/home/ab-c5565/media/T071/T077/T032	Yes	1
/home/ab-c5565/media/T071/T077/T090	Yes	1
/home/ab-c5565/media/T051/T052/T053/T054	Yes	0
/home/ab-c5565/media/T051/T052/T053/T055	Yes	0
/home/ab-c5565/media/T051/T052/T056	Yes	0
/home/ab-c5565/media/T051/T052/T057/T058	Yes	0
/home/ab-c5565/media/T051/T052/T057/T059	Yes	0

Figure 3.15- Utility interface for checking media directory hierarchy

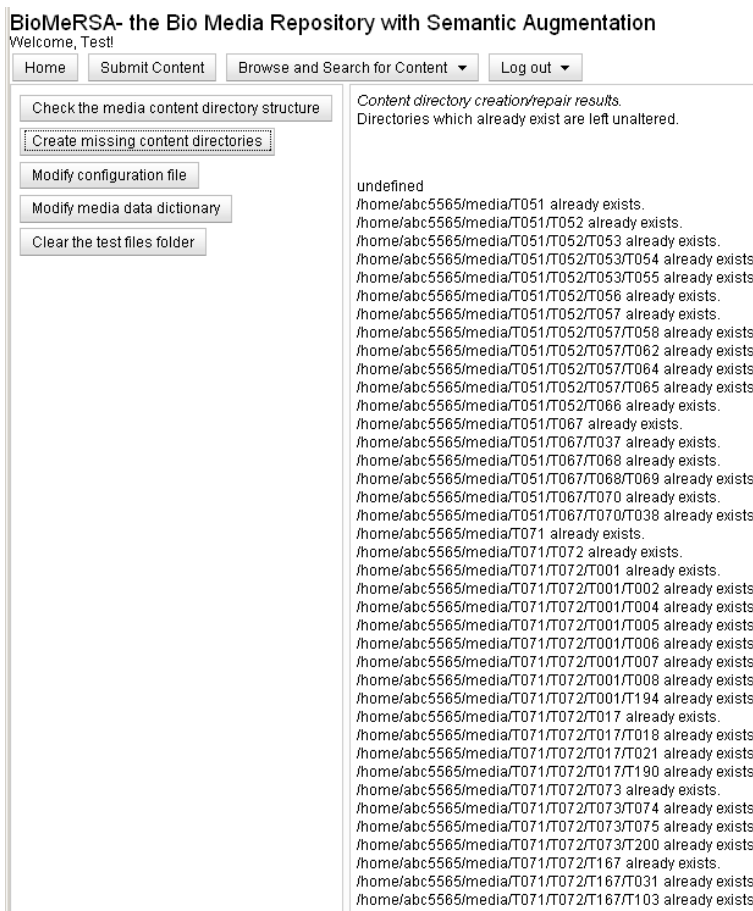


Figure 3.16- Result of utility method which will create missing directories in the media directory hierarchy

3.5 Expert Annotation System

The primary purpose for BioMeRSA is to give semantic context to user-submitted media files. Unfortunately, at present there is no automated method for assigning semantic data to multimedia files that would accurately handle content from across the field of biology. In lieu of such an automated system, the connection between the submitted files and the semantic data must be made by human experts. At some institutions this sort of system would be maintained by a small team of full-time curators- instead this system relies on individuals known to be knowledgeable in biology to contribute some time when they have it available toward making these semantic associations. These 'experts' could be students, professors, lab technicians- this decision is up to the administrators, as the 'expert' users must be granted access to the expert

annotation system by an administrator, or other user with the ability to create user accounts or change user permissions.

The key UMLS semantic entity of concern in this system is the concept- which represents a definable idea or entity for which exists at least one atom, term, and string in the Metathesaurus. Associating concepts with media files allows other similar files to be found which are associated with the same concept, other concepts under the concept's semantic type, and eventually other concepts through various relations represented in the Metathesaurus. As concepts are independent of specific terms, strings, or atoms, the file-concept association allows searching media files by terms which may not be considered to be most often used in a particular circumstance (for more information on this, see Section 3.6 'Searching and Browsing for Media').

There are no additional database tables for this part of the system that have not already been described earlier. The database table with which the annotation system is primarily concerned is *ontolink*, and the structure of this database table is as shown in figure 3.14. Each user can associate multiple concepts with a single file- this makes sense, as a given media item can have more than a single subject, and each subject may be related to multiple concepts. For example, an image of a broken femur might be associated with any number of concepts- that for broken bone, femur, human, male or female if their gender is known, and perhaps the concept for soccer if it is known that the individual was involved in such activity when the bone was broken. It is unlikely that anyone aside from the original submitter will have access to such details about the file, unless they add comments to the description field upon submission, but it shows the breadth of concepts included in UMLS which can be associated with files. Additionally each of those concepts can be found under multiple atoms- for example, 'soccer' could also be found by searching all concept names for 'football'. The *ontolink* table reflects that multiple

users can associate multiple concepts to multiple files- to make a truly unique primary key for the table, a triple key was needed, composed of the ID of the media being annotated, the user making the association, and the ID of the concept being associated. For each association made, the database table also has support for adding a note, and a rating of the users' confidence of the association. The note could include reasons for which they are associating that particular concept, and the confidence rating is used statistically to determine how to rank associated concepts later on.



BioMeRSA- the Bio Media Repository with Semantic Augmentation
 Welcome, Adam!

Home Submit Content Browse and Search for Content Log out

Click the row corresponding to the file you wish the view the annotation information for.

Unannotated Media (click here to open or close this section)

Items found: 3

Media Item	Media ID	Media Type	Media Format	Date Added
	17	image	TIFF	2009-02-23 00:56:32
	18	image	TIFF	2009-02-23 00:57:33
	19	image	TIFF	2009-02-23 00:59:55

Previously Annotated by Me (click here to open or close this section)

Items found: 3


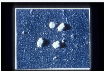

Media Item	Media ID	Media Type	Media Format	Date Added
	20	image	TIFF	2009-02-23 01:01:33
	21	image	TIFF	2009-02-23 01:06:32

Figure 3.17- Interface for selecting media to annotate

The interface for the expert annotation system is divided into two main pages. The first page is where the media to be annotated is selected by the user- this is shown in figure 3.17. Two grids are presented, one which lists items that a user has not yet annotated, and the other has ones that they have already associated at least one concept with. Each of these grids can be hidden by clicking on the title of the grid (to simplify the display), and the columns in each can also be sorted by clicking on the column headers. Each grid will also have the number of items

loaded. Note that this only includes files which are considered to be still available for annotation- ones set with status of 'annotation'. Files that have already been made 'live' and are viewable in the searching and browsing interfaces will not show up under previously annotated items, as those files are no longer available for annotation. Clicking on a row in either of the two grids will load the annotation info page for that file (figure 3.18)- this includes the file and user-submitted metadata, and a list of concepts already associated by the current user, if any (associations made by other users do not appear in this display). Previously associated concepts can be removed from this list as well, by selecting the 'Remove Concept from this File button', and full information for an associated concept can be viewed by selecting the 'Get Detailed Concept Info' button.

Annotation information for file ID 19 Find Concepts by Browsing the Semantic Network Search for Concepts



[Download the file!](#) (please use 'save as...')

File Info:	
File name:	abc556520092230592019.tif
Original file name:	Virology 06.tif
Date/Time of submission:	2009-02-23 00:59:55
File type : File format:	image : TIFF
File size:	67056 kilobytes
Display dimensions (W x H, in pixels):	3946 x 5782
File Metadata:	
Creation Date:	0000-00-00
Location of Capture or Generation:	
Method of Capture or Generation:	
Creator:	Clay Adams, Division of Becton Dickinson and Compa
Copyright Restrictions	All Rights Reserved
Other Comments:	Scanned slide from Osier slide collection. Set: 'virology_1' Slide: 6 Description: 'Bushy stunt virus 65,000x'

Associated Concepts (only those added by abc5565)

All of the concepts you have associated with this file are shown in the table below.
 You can select a row and use the button below the table to remove concepts, or browse/search to associate an additional concept with this file.
 0 Associated Concept(s) Found

Figure 3.18- Interface for annotation of a media file

Concepts are associated with a file by searching for concepts in one of two ways. Both of these methods are covered in more depth in this section under 3.6 ‘Searching and Browsing for Media’, as they are also the same interfaces used to find files already annotated with concepts. The first method is to use the Semantic Network browser to find concepts linked to specific Semantic Types. This can be loaded by going to the ‘Find Concepts by Browsing the Semantic Network’ tab shown on the top of figure 3.18. This will load the page shown in figure 3.19, which includes a tree widget on the left side showing the nodes of the Semantic Network. The tree levels can be expanded by clicking on the ‘+’ boxes, revealing the child nodes for a given node. When a node in the tree is selected, the large frame to the right will load extended

information and a definition (if there is one) of the selected Semantic Network node, as well as a list of the concepts linked to that node. If one of these concepts is selected, the “Selected Concept” indicator in the very bottom frame will update to indicate that a concept has been chosen. If a different concept is selected before the submission of the data, the concept shown as selected will be changed. The notes field is there for the user to add a specific note to this association, which may include the reasons why they are choosing the given concept to be associated with the file. The slider bar on the rightmost side of the bottom frame is for the user to rate their confidence in this association- this will be factored into simple statistical procedures for determining the ranking of concepts that have been added to a file. Zero indicates that the user is very unsure of the relatedness of the concept and the file, and five indicates that there is a certain link between the two.

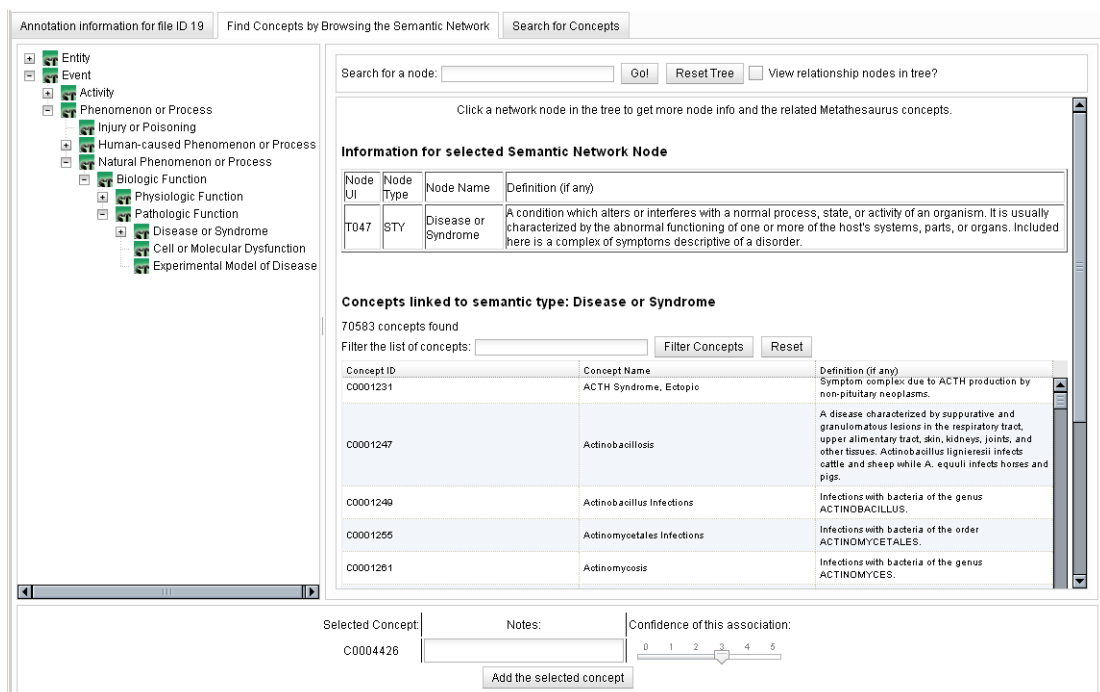


Figure 3.19- Interface for selecting a concept to be associated with a file utilizing the Semantic Network browser

The second way of looking for concepts to add is to directly search for them. The “Search for Concepts” tab will load the page in figure 3.20, which has the same bottom-frame

features as the previous tab. On this page, the desired search method can be selected, and along with a search term, will result in concepts being displayed in the list below, if there are any results. The default search option will look only in the primary concept names. The 'Searching and browsing for media' section has more information on these options. Concepts can be selected in the same manner as in the previous tab, and the behavior of the bottom frame is the same. It is notable that because this interface was assembled with Dojo widget layout components, the tabs on the top can be visited without changing their content, and without having to load the pages again- this means that if a concept is selected in either the Semantic Network browsing tab, or the concept search tab, and the user wants to go back and look at some of the detail on the file again, they can go between the concept selection tabs and the file annotation information tab without losing their concept selections. Furthermore all of the transactions between the client and the server are asynchronous, so while the client is waiting for results from the server, the interface will still be responsive. This is important, because some search responses may take some time.

After the desired concept has been selected (using either method), the 'Add the selected concept' button will immediately send a request to the server to add the concept. The interface will then switch back to the file annotation information tab, and the list of concepts associated with the file will be updated to reflect the change. Figure 3.21 shows this interface after a concept has been added.

Annotation information for file ID 19 Find Concepts by Browsing the Semantic Network Search for Concepts

Search for concepts:


Search IDs
 Search only primary concept names
 Search all concept names
 Search all concept names and definitions

Concept ID	Concept Name	Definition (if any)
C0520503	Tomato bushy stunt virus	
C1221731	Maize bushy stunt phytoplasma	
C1529822	P19 protein, tomato bushy stunt virus	
C1926318	Tomato bushy stunt virus satellite RNA	

Selected Concept:
Notes:
Confidence of this association:

Figure 3.20- Interface for selecting a concept to be associated with a file utilizing concept search

Annotation information for file ID 19 Find Concepts by Browsing the Semantic Network Search for Concepts



[Download the file!](#) (please use 'save as...')

File Info:	
File name:	abc556520092230592019.tif
Original file name:	Virology 06.tif
Date/Time of submission:	2009-02-23 00:59:55
File type : File format:	image : TIFF
File size:	67056 kilobytes
Display dimensions (W x H, in pixels):	3946 x 5782
File Metadata:	
Creation Date:	0000-00-00
Location of Capture or Generation:	
Method of Capture or Generation:	
Creator:	Clay Adams, Division of Becton Dickinson and Compa
Copyright Restrictions:	All Rights Reserved
Other Comments:	Scanned slide from Osier slide collection. Set: 'virology_1' Slide: 6 Description: 'Bushy stunt virus 65,000x'

Associated Concepts (only those added by abc5565)

All of the concepts you have associated with this file are shown in the table below. You can select a row and use the button below the table to remove concepts, or browse/search to associate an additional concept with this file.

1 Associated Concept(s) Found

Concept ID	Preferred Concept Name	Notes	User Confidence Rating
C0520503	Tomato bushy stunt virus	Slide description is labelled with this virus name	5

Figure 3.21- File annotation information page now with an associated concept

3.6 Searching and Browsing for Media Files

As was overviewed in the previous section, there are two primary methods of finding concepts in the BioMeRSA system. Once a user has found a concept of interest, they can browse the media files linked to that concept that have already been annotated by the experts (the file status is marked as 'live' in the database). From there, if a file is selected, they can view the file information page which includes more detail on the file, the preview (for images) and a download link.

The first of the two concept-discovery methods involves browsing (or searching) a tree structure populated with the nodes of the hierarchical UMLS Semantic Network, which exists to classify concepts into broader categories, through which the concepts are then also related. The Semantic Network nodes themselves are also related, making the basic Semantic Network suitable for representation in a tree. To accomplish this however, all non-'isa' relationships and cyclic relationships were ignored, as they could not be represented in the tree. There is a search box on top of the main middle frame for searching the tree for specific network nodes, which may be faster than expanding long branches when the name of the desired node may be already known. When a node is selected in the tree, the main middle frame will load all of the associated information for the selected node, and will also load a list of the concepts linked to that Semantic Type. Above the list of concepts will load another search box, which can be used to search the concept list to narrow the list down to more specific choices- this is useful as there are Semantic Types associated with 30,000 or more concepts. A demonstration of this interface is shown in figure 3.22.

BioMeRSA- the Bio Media Repository with Semantic Augmentation
 Welcome, Adam!

Home Submit Content Browse and Search for Content Log out

- Entity
 - Event
 - Activity
 - Phenomenon or Process
 - Injury or Poisoning
 - Human-caused Phenomenon or Process
 - Environmental Effect of Humans
 - Natural Phenomenon or Process
 - Biologic Function
 - Physiologic Function
 - Pathologic Function
 - Disease or Syndrome
 - Cell or Molecular Dysfunction
 - Experimental Model of Disease

Search for a node: View relationship nodes in tree?

Click a network node in the tree to get more node info and the related Metathesaurus concepts.

Information for selected Semantic Network Node

| Node UI | Node Type | Node Name | Definition (if any) |
|---------|-----------|--------------------------------|---|
| T069 | STY | Environmental Effect of Humans | A change in the natural environment that is a result of the activities of human beings. |

Concepts linked to semantic type: Environmental Effect of Humans

56 concepts found

Filter the list of concepts:

| Concept ID | Concept Name | Definition (if any) |
|------------|---------------------------------|---|
| C0001111 | Acid Rain | Acidic water usually pH 2.5 to 4.5, w/ ecosystem and adversely affects plants mammals. It is caused by industrial sulfur oxides and nitrogen oxides, enter atmosphere and returning to earth in rain water. |
| C0001873 | Air Pollution | The presence of contaminants or pollutants in the air (AIR POLLUTANTS) that interfere with health or welfare, or produce other harmful effects. The substances may include PARTICULATE MATTER, or volatile CHEMICALS. |
| C0001874 | Air Pollution, Radioactive | |
| C0014419 | Environmental Pollution | |
| C0016323 | Water fluoridation | |
| C0016463 | Food Contamination, Radioactive | |
| C0017095 | Garbage | |
| | | The application of heat to raise the t |

Displayed UMLS data is from UMLS version 2008AA

Figure 3.22- Browsing for concepts by Semantic Network nodes

When a concept in the list is selected, a new page loads, such as that in figure 3.23. This displays all of the information available on the concept, and a list of all the media items associated with it (the tabbed interface on the right). Included in this concept information is the ID in the Metathesaurus, the preferred name, definition (if there is one), and a list of all the associated atoms, strings, and terms. The string and term lists can be expanded or hidden by clicking on the title bar for the tables if they are not needed- they are hidden by default when the page initially loads. In the media list, if a media item row is selected, a new tab in this interface will load all of the file's information. Multiple file information tabs can be opened simultaneously, by reselecting the media selection tab, and opening a different file. These tabs can also be closed by the user, by clicking the circular 'x' icon on the right side of each of the media info tabs. The displayed file information includes the file ID number, a preview (if it is an

image, otherwise a placeholder will be displayed), the physical file data, user submitted metadata, and a link from which to download the file. This can be seen in figure 3.24.

Concepts can also be found more directly by searching for them. There are four options, which are able to search different concept-related data. These are, in order of the specificity of the results, from specific to general (which also correlates to number of search results in most cases): 'Search IDs', 'Search only primary concept names', 'Search all concept names', 'Search all concept names and definitions'. If a concept ID is already known, or part of an ID is known, then it might be best to try to search by ID first. Searching by primary concept is the default, and will only look in the database table containing the atoms for the 'preferred names' for each concept. The next option, 'search all concept names', will search every atom for every concept, and the last option will search all atoms as well as their definitions. The last option is most likely to return a result for any given string, but will also take the longest to execute- queries may take thirty seconds (or more, depending on client internet connection speed) to return a result. Searching only the primary concept names is a good balance of speed and specificity, and is thus selected by default. A search will result in a concept list similar to that which was shown when browsing by Semantic Network. Once again, selecting a concept row will load a new page with the concept info and associated media item list as previously described for the Semantic Network browsing interface. The search interface can be seen in figure 3.25.

BioMeRSA- the Bio Media Repository with Semantic Augmentation

Welcome, Adam!

[Home](#)
[Submit Content](#)
[Browse and Search for Content](#)
[Log out](#)

Concept ID: C0520503

Preferred concept name: 'Tomato bushy stunt virus'

Definition(s):

No definitions for this concept.

Atoms found: 4

| Atom ID | Atom | Source |
|----------|-------------------------------------|--------|
| A2042596 | Tomato bushy stunt virus | NCBI |
| A7815745 | Tomato bushy stunt virus | MSH |
| A1180965 | tomato bushy stunt virus | CSP |
| A2227788 | tomato bushy stunt tombusvirus TBSV | NCBI |

More information on strings for this concept

Strings found: 3

| String ID | String |
|-----------|-------------------------------------|
| S2187678 | Tomato bushy stunt virus |
| S1214522 | tomato bushy stunt virus |
| S2377356 | tomato bushy stunt tombusvirus TBSV |

More information on terms for this concept

Terms found: 2

| Term ID | Term |
|----------|-------------------------------------|
| L0250333 | Tomato bushy stunt virus |
| L1961649 | tomato bushy stunt tombusvirus TBSV |

Media Selection


| Media Item | Media ID | Media Type | Media Format | Date Added |
|---|----------|------------|--------------|---------------------|
|  | 19 | image | TIFF | 2009-02-23 00:59:55 |

Figure 3.23 Concept information view

BioMeRSA- the Bio Media Repository with Semantic Augmentation

Welcome, Adam!

[Home](#)
[Submit Content](#)
[Browse and Search for Content](#)
[Log out](#)

Concept ID: C0520503

Preferred concept name: 'Tomato bushy stunt virus'

Definition(s):

No definitions for this concept.

Atoms found: 4

| Atom ID | Atom | Source |
|----------|-------------------------------------|--------|
| A2042596 | Tomato bushy stunt virus | NCBI |
| A7815745 | Tomato bushy stunt virus | MSH |
| A1180965 | tomato bushy stunt virus | CSP |
| A2227788 | tomato bushy stunt tombusvirus TBSV | NCBI |

More information on strings for this concept

Strings found: 3

| String ID | String |
|-----------|-------------------------------------|
| S2187678 | Tomato bushy stunt virus |
| S1214522 | tomato bushy stunt virus |
| S2377356 | tomato bushy stunt tombusvirus TBSV |

More information on terms for this concept

Terms found: 2

| Term ID | Term |
|----------|-------------------------------------|
| L0250333 | Tomato bushy stunt virus |
| L1961649 | tomato bushy stunt tombusvirus TBSV |

Media Selection Info For Item ID 19

Extended Information for Media File ID # 19



File data:

| | |
|--|---------------------------|
| Submitted file name: | abc556520092230592019.tif |
| Date/Time of submission: | 2009-02-23 00:59:55 |
| File type : File format: | image : TIFF |
| File size: | 68666052 |
| Display dimensions (W x H, in pixels): | 3946 x 5782 |

Other metadata:

| | |
|------------------------------------|--|
| Creation Date: | 0000-00-00 |
| Location of Capture or Generation: | |
| Method of Capture or Generation: | |
| Creator: | Clay Adams, Division of Becton Dickinson and Compa |
| Copyright Restrictions: | All Rights Reserved |
| Other Comments: | Scanned slide from Osier slide collection. Set: 'virology_1' Slide: 6 Description: 'Bushy stunt virus 65,000x' |

[Download the file!](#)

(please use 'save as...')

Figure 3.24- File information loaded inside concept information viewer

BioMeRSA- the Bio Media Repository with Semantic Augmentation
 Welcome, Adam!

Home Submit Content Browse and Search for Content Log out

Search for concepts:

Search IDs
 Search only primary concept names
 Search all concept names
 Search all concept names and definitions

| Concept ID | Concept Name | Definition (if any) |
|------------|---|--|
| C0005375 | Bicycle | |
| C0005375 | Bicycle, device (physical object) | |
| C0005375 | Bicycle, device | |
| C0005376 | Bicycle Ergometry Test | |
| C0005376 | Bicycle Ergometry Tests | |
| C0005376 | Ergometry Tests, Bicycle | |
| C0005376 | Tests, Bicycle Ergometry | |
| C0005376 | Ergometry Test, Bicycle | |
| C0005376 | Test, Bicycle Ergometry | |
| C0180749 | Bicycle ergometer | |
| C0180749 | Bicycle Ergometers | |
| C0180749 | Ergometers, Bicycle | Bicycles very similar to common stationary exercise bikes in that they have adjustable seats and handlebars, pedals with toe clips, and variable pedaling resistance. Bicycle ergometers subject the patient to a constant, quantifiable work load and/or provide a quantitative measure for the rate of work output. The effort or rate of work output is usually given in units of watts (W) or kilopound meters/min (kpm/min). Bicycle ergometers are used in many cases to subject patients with suspected pulmonary or circulatory system disease to standardized exercise. |
| C0180749 | Bicycle ergometer, device (physical object) | |
| C0180749 | Bicycle ergometer, device | |

Figure 3.25- Searching directly for concepts

3.7 Program Code and System Environment

With the current server environment (on the Bucatini server), the application files are in a cgi-bin directory under public_html, as is customarily found on web servers running CGI systems under Unix or Linux. All of the Perl CGI and template files are in this single directory. The template files contain HTML and JavaScript code that typically corresponds to one 'view' or page in the application. Where possible, HTML and JavaScript code was placed in the template files instead of being integrated with the Perl code, this results in maintainability which is orders of magnitude greater, and simplifies the process of writing the HTML, as one does not have to be so concerned with string concatenations- this is likely also faster and less memory intensive on the server side. As some of the files contain a large number of subroutines or functions, these were usually separated from each other using some sort of indicator, usually a comment line with a long string of some character, such as '=' equal signs. Where possible, functions corresponding to a certain kind of functionality or feature were grouped together, and otherwise appear in the file in alphabetical order. In-line comments were made in the Perl, HTML, and JavaScript code where it seemed to be helpful, and all subroutines and functions

have blocks of comments with their details. For the Perl code, comments were later ensured to meet the POD (Plain Old Documentation) standard, so that the in-line documentation can be parsed and browsed independently of the code. Both BioMeRSA and UserDB also read many options from an easily editable 'configuration file', itself a Perl class, which provides a way of easily changing settings that seemed likely to need some flexibility, which reduces the need to scour the code to change such values. These efforts toward increasing maintainability will help make the system useful into the future, and will aid future developers in expanding it.

4. Discussion

4.1 The Implemented Product- BioMeRSA

During the course of this project a computer software system was developed to associate semantic data with multimedia files pertaining to the general field of biology. This system was named BioMeRSA- the Biology Media Repository with Semantic Augmentation. The Unified Medical Language System was utilized as a source of the semantic data. UMLS integrates information from many sources across most of biology, making it a nearly comprehensive and cohesive single source, generally known to be reliable and updated regularly. The resulting system consists of a web-based user interface for end-user access across multiple computing platforms, and server-side components for control logic and a database for storage of metadata for submitted media and semantic information. Users are able to submit files, add semantic annotation to existing media, as well as search and browse for items based on annotations. The user can then download media to their local system. Some of these operations depend on a specific users' access to the system, which is determined by the administrators or other individuals who have been granted the ability to create new user accounts.

The system that was developed includes all of the base functionality necessary to satisfy the original goal of having a media database incorporating semantic data related to biology and curated by experts. The interface to the system for end users was developed with technologies enabling dynamic user interface elements to be easily combined to create a logical and intuitive workflow. Users only wishing to find media can go directly to the searching and browsing interfaces, those wishing only to upload their content can do so, and experts can navigate easily through the system for annotating media items with semantic data. To find the appropriate concepts to associate with files, the expert annotation system uses the same general interfaces as the browsing and searching interfaces in the media browsing workflow.

4.2 Issues During Development

As already mentioned, the learning curve associated with implementing the technologies that were chosen was steeper than expected at first. Although Perl had been utilized previously by the author in web-based CGI applications, this experience was not extensive, and many additional plugins were used to extend functionality for BioMeRSA. Additionally, on top of Perl were layered HTML, JavaScript, JSON, and the Dojo Framework. While the author's familiarity with Java made the use of JavaScript syntax a smoother transition, the decision to use the Dojo Toolkit introduced multiple challenges throughout much of the course of the project. The decision to utilize the Dojo Toolkit instead of comparable JavaScript frameworks such as jQuery, Prototype, MooTools, and Yahoo! UI Library was made after browsing various comparisons and feature lists of the two, and looking at various support materials such as documentation and community forums (Schiemann, 2009). It is worth mentioning that most of the available JavaScript frameworks and libraries are open-source projects, with the primary exception being the Yahoo! UI Library, which is developed internally at Yahoo!. Thus in many of these cases, not only the code, but also the documentation may be edited collaboratively. The scope of the Dojo toolkit is more encompassing than many of the other toolkits and libraries- in addition to the transport methods for AJAX functionality, there is also a built-in user-interface element set (the Dijit widgets), very flexible objects for handling data in datastores (dojo.data), and a great deal of other functionality. While loading the entire toolkit into the browser would be unreasonably slow, this is unnecessary because a package system is used, so that only the needed components are downloaded by the clients. Dojo is also known for being very fast- in this case speed refers to rendering and responsiveness on the client- although this also depends greatly on the web browser being used (Brandon, 2008) (Velichkov, 2009). In practice it was found that constant reloading of BioMeRSA pages which heavily utilize Dojo components in Firefox 3/Windows XP led to high memory usage of the

Firefox process, indicating that there may be significant memory leaks associated with the JavaScript, but this may also be due to the author's inexperience with cleanup routines in Dojo and JavaScript in general. The Dojo documentation initially appeared to be quite complete, including a full API reference, and online 'manual' with a number of code examples. Many initial issues stemmed from the Dojo developers being in transition to the release of a new version at around the time that development on this project commenced. The version used for development, 1.2.3 turned out to be somewhat of an incremental build toward 1.3, and had already introduced many features and altered the syntax for using many components, which had not been updated in the documentation (which was mostly for version 1.0). As the Dojo developers progressed toward their release of 1.3 (which was released on March 31st, 2009), updated documentation began to appear on the new documentation site (Dojo Toolkit, 2009) (The Dojo Toolkit Community). At that point, the combination of the new documentation and acquired experience made development less frustrating, also indicating that the issues found with documentation early on were a significant contributor to the learning curve. While for the most part, these issues would not have been found without actually trying to use Dojo, a more in-depth initial comparison of the available frameworks could have led to the use of different tools to avoid the issues that were encountered. However if that were to be the case, the interface of BioMeRSA would have likely been significantly altered due to different user interface components being provided by each framework.

The choice of development environments could have also had an impact on progress through the project. The Kate text editor installed on the development server was used for the majority of editing tasks, which supports some basic syntax highlighting, and has decent handling of multiple files (The Kate Development Team, 2008). Editing directly on the development server also means that code can be edited, saved, and tested (from a local web

browser) without having to upload to the server every single time, or set up a local environment to replicate that of the server which could be used to development. There are some alternative IDEs (Integrated Development Environments) which could have provided some more feedback during code development (autoindenting, syntax checking and highlighting for instance), while still keeping all editing within a single program. Eclipse is a large and mature IDE, which is open-source and popular for working with a number of languages, including Java and Perl, as well as numerous plugins for support for other languages (The Eclipse Foundation). There is another open-source editor which happens to be based on Eclipse, Aptana Studio, which could serve all the desired functions (Aptana Inc.). Aptana Studio is targeted primarily toward those working with 'Web 2.0' systems, and includes by default, support for ten AJAX/JavaScript libraries, HTML, CSS, JavaScript, PHP, Ruby, and Python- which covers more than all the technologies utilized in this project with the major exception of Perl. Being based on Eclipse means that Aptana can be used as a plugin for Eclipse, and other Eclipse plugins can be used in Aptana- so if a plugin such as EPIC were used for Aptana, it may be possible to get Perl syntax checking and highlighting (EPIC). Additionally, Aptana is available for Windows, OS X, as well as Linux, so editing using this combination could have been done locally in Linux or Windows, or remotely on the development server under Linux.

A final major area with room for improvement during the development process is in testing. Being somewhat unfamiliar with dynamic web programming, and specifically with JavaScript and Dojo, it was not possible to work on the code without frequently refreshing a browser window to see the results. For this, Firefox 3 and the Firebug extension were used. Firebug makes diagnosing JavaScript errors (which include those related to Dojo) a little easier by giving more information than the browser would normally provide upon error conditions. Firebug is also able to show server requests and responses. Due to this, very little testing was

done in other browsers, despite the target platform for this application being any modern web browser with JavaScript and cookies enabled. It was quickly found upon testing in Internet Explorer that Firefox is very tolerant of some errors that other browsers do not handle. In some cases this resulted in pages which failed to even render. Debugging some of these issues required the use of the W3C HTML validator, as a number of these problems were simple bugs overlooked in the HTML (World Wide Web Consortium). Other errors in JavaScript were diagnosed through trial and error, and some were found to be likely originating from Dojo. Doing more thorough testing throughout development would have reduced the debugging load later on and perhaps identified some common mistakes earlier. Utilization of an IDE with more debugging and highlighting functionality would have also likely assisted with this.

4.3 Advantages of the BioMeRSA Design

Although there many elements of BioMeRSA that have been identified as having room for improvement, there are certain other areas in which the design decisions are advantageous for present and future needs. The database in the current environment utilizes MySQL 4, which is very well-supported, popular, fast, and includes all of the functionality so far needed for this system. If more advanced functions are needed, such as triggers and stored procedures, an upgrade to MySQL 5 will be needed, and is available. Some of the database tables have already been normalized, and these are expected to be the ones which will remain consistently most used. Thus if the decision is made to normalize more of the UMLS tables, it will not be necessary to rewrite all of the UMLS-related business logic in the application. While not all of the queries in that business logic are fully optimized, they were written to be faster when possible- this is most true when data from multiple tables was needed, and in this case JOINS were used instead of making multiple queries on each of the separate tables. The MySQL 'EXPLAIN SELECT'

command was utilized to ensure no unexpected bottlenecks were being encountered on the database side.

While Dojo does not currently have the same level of community support and enthusiasm behind it as other projects like jQuery, it is still in active development and has just reached a milestone version with the most recent major update (version 1.3). As Dojo has a commercial support company behind it, it is a popular solution for some major business customers including Apple, IBM, and Wal-Mart. This momentum, along with the improving documentation, and a comprehensive but modular approach should make Dojo a more attractive choice in the future, which should also improve the available third party resources (books, examples, IDE plugins, and additional widgets)(SitePen, 2009)(Dojo Toolkit, 2009). Thus the selection of Dojo as the framework of choice does make some sense despite the initial frustration. As an added bonus, it's easy to test and transition to new Dojo versions as they are released, because a local copy of Dojo is not even needed. For testing purposes, all that is needed is a reference to one of the third-party servers providing public access to various Dojo versions, such as the AOL Content Developers Network (CDN) and the Google AJAX Libraries API(AOL, 2009)(Google, 2009). Dojo Dijit widgets look decent even with the default stylesheets and no additional styling. There is plenty of room for future customization for the interface 'look and feel', and there is a good deal of documentation available on creating and styling widgets, which will be necessary if the application is restyled.

Perl, in the back end, may not be as popular as it once was for web-based applications but still remains plenty capable and well-supported. In some areas, such as security, Perl may have significant advantages in default configurations over other languages like PHP. Perl is also very mature, and has many good plugins and modules which can be found on CPAN (CPAN, 2009). Some of these were heavily utilized in this project, such as CGI::Application, to achieve

more easily the Model View Controller design pattern. Where possible, values are read from a configuration file which can be edited so that the target values do not have to be found in the application code itself to be modified. This also allows for easier migration of server environments when necessary, as the current paths can be changed in the configuration file. Otherwise, relative paths are used where appropriate, such as in URLs to avoid the possible issues that can arise with hardcoded paths.

4.4 Possible Future Work

As already mentioned, there were some elements presented in the proposal for the project which were not able to be incorporated into the final results. Additionally, as the work progressed on implementation, many additional features were envisioned that would not be reasonable to incorporate as part of this project in the available time frame. Many of these features would increase the usability, usefulness, and manageability of the system- hence it would be beneficial if at least a subset could eventually be developed. To aid in facilitating that, some of the possible ideas for future work are presented here.

4.4.1 Semantic Data

Early in the initial research for the project, it was found that UMLS provides semantic data compiled from a large number of major sources of semantic data in the domain of biology. It was determined that data from the majority of the desired sources were already integrated into UMLS, which also is able to provide these resources in a consistent and unified format, which greatly simplifies the transformation of the data into the necessary database tables. Although the scope of the data contained in UMLS is expansive, and the semantic data tables in the database are already designed with UMLS in mind exclusively, the need to add data from sources not included in UMLS is not impossible. Hence, it may eventually be desirable to alter the database tables so that the semantic data is stored in a less source-dependant format, into

which data from UMLS as well as other sources could be transformed. This will require research into the data formats that other semantic information providers utilize, as well as development of methods to perform a transformation of that nature. Such an alteration would also be beneficial if a new data format for UMLS becomes preferred over RRF. Similarly, it was originally planned that experts with the ability to annotate would be able to add a concept to associate with a file in the case of the desired concept (or atom) not being present in UMLS. This feature was also left unimplemented, but would be reasonable to add in conjunction with the semantic data database structure revisions.

The functionality of the system could be expanded by implementing semantic data not directly pertaining to the subject area of biology. UMLS already includes some concepts which do not appear to have direct biological connections- such as those for bicycles, ladders, cars, etc., but there are many other ontologies and thesauri specializing in data outside of biology which could be integrated. For example, semantic data from a geology or Earth science ontology could be used to associate images depicting wildlife outdoors with concepts representing the terrain of the area, or the location. This could also be extended to geo-tagging with coordinates of specific locations, if the creator of the media was using a GPS. Just as with biology, there are semantic systems for Earth science covering both specific topics and more general domains which could be utilized. For example, there is an ontology developed at the University of Southern California specifically for information pertaining to faults, and a more inclusive system, the Semantic Web for Earth and Environmental Terminology (SWEET) for Earth science developed by the NASA Jet Propulsion Laboratory (Juve, et al.) (Raskin, 2004). Having a standardized semantic data format for the database would, as already mentioned, aid in the addition of such data.

UMLS is a resource that has demonstrated significant longevity, having been initiated in 1986 and continuing to be maintained (Kleinsorge & Willis, Unified Medical Language System Basics, 2008). Updates are regularly compiled and released for UMLS, and are typically made available two to four times per year, as data sources are updated and new sources are added (National Library of Medicine, 2009). Acquiring and integrating updated UMLS data is going to be essential to keep the system relevant and useful as advancements in biology continue to be made. No solution for updating the UMLS data in BioMeRSA currently exists. There are a few factors that can be expected to aid in the development of an update tool. Although the default UMLS database tables were modified in the process of normalization, the changes were documented so that perhaps an automated method of manipulating updated data can be developed. New subsets can be generated using updated versions of the same data sources with MetamorphoSys if the saved subset configuration file is loaded using new UMLS source files. However, in that case sources not present in the previous version that are desired would need to be manually selected for inclusion in the new subset. There are files in the Metathesaurus which track changes in sources between Metathesaurus versions which could be used to determine the data that need to be added to (or perhaps removed from) the current semantic data set. At present, that data of concern includes Semantic Network nodes, definitions, and relationships, as well as concept names, definitions, and the concept/Semantic Network associations, but this list is likely to expand as more Metathesaurus data is utilized during the addition of new features.

Integration with the MEDLINE literature database from the National Library of Medicine was suggested early on as a feature that would be useful. It was found that Medline is listed as a source- both in the online source list and in MetamorphoSys; however no concepts appear to be associated with it. This was confirmed as being intentional, as MEDLINE is utilized exclusively as

a source of statistical co-occurrence data for the elucidation of relationships between concepts, and is not used a source of concept data itself (Emrick, 2009). Even if MEDLINE was a concept source, it would likely be unreasonable to include citation information on individual papers in the Metathesaurus sources file due to the number of items included in MEDLINE. Thusly, regardless of how MEDLINE is utilized for UMLS, items from MEDLINE would have to be added to BioMeRSA as content in the same way that other media items are- manually or via a batch import interface, and will need to be annotated by the pool of experts, unless automated methods are developed.

As mentioned before, the files containing the data of the generated subset which are utilized to create the database tables are not provided with their structure normalized. Much of the data is duplicated between tables, and the creation scripts generated by MetamorphoSys assign primary keys for only seven tables, without any foreign key constraints. There were 34 UMLS tables generated which were non-empty, and due to this large volume of data and table count, normalization was not performed until a table was needed. In most cases, a new table was created containing the needed data in a normalized form, instead of modifying the columns of existing tables. Due to this fragmented approach, normalization was done based on the specific need of the table, and so the altered or added tables are not normalized to any specific normalization form (1st normal form, 2nd normal form, etc.). Normalization of all of the UMLS-related tables would provide the structure needed to maintain referential integrity, be easier to visualize in entity-relationship diagrams, and could have some database speed and disk usage benefits if unneeded fields are removed from the newly normalized tables.

4.4.2 Searching and Browsing

There is still considerable room for improvement and additional flexibility in the various search options provided by BioMeRSA. All searches currently function in one of three ways:

using Dojo to query a Dojo datastore, MySQL 'LIKE' statements in a database query, and a MySQL fulltext search in a database query. Queries on Dojo datastores are utilized when all of the data being searched exists on the client side as the result of a request to the server. These queries allow multiple searches without having to send multiple requests to the server for data, and are likely faster because of that. It is unlikely that these queries will need to be altered, or be changed to server requests, unless a specific search algorithm is developed to optimize results, and existing Dojo query methods are overridden. In all other cases, requests are made to the server with a search parameter specified in the browser query string. For simpler searches 'LIKE' is used with a wildcard on each end of the specified search term to try to return as many results as possible. For more complex queries, like queries on TEXT-type fields, the built-in MySQL fulltext search support is utilized. In the case of 'LIKE' queries, partial word matches are not supported. When partial word matching is enabled for fulltext queries, there are usually too many results to be useful. A more effective search routine for some of these situations, particularly if it were to be 'semantically-aware', would save a good deal of time by producing a smaller set of more highly related results particularly when searching directly for concepts. The Metathesaurus provides a table with many relationships between concepts, and that data is currently not being utilized. Those relationships and the relationships between concepts and Semantic Types could be used to provide greater focus to search results, as well as provide results which are related, but would not be returned by direct string queries. UMLS also provides some lexical tools that could also be used to produce multiple query strings to return more results if necessary. These lexical tools are currently written in Java, but the desired methods could likely be ported to Perl, or else interfaced to by other means (National Library of Medicine, 2008). Highlighting the search term in the results would also be particularly useful in

queries involving definitions and TEXT type strings; the attention of the user is drawn to the word, and they can determine the quality of the results more quickly.

4.4.3 Visualization

The tree structure representing the nodes of the Semantic Network is the only visualization of the semantic data included in the system. Considering the multitude of relationships existing between concepts in the Metathesaurus, and even the relationships between concepts and Semantic Types, it could be possible to use visualizations as both a means of browsing the semantic data, as well as representing search results. If combined with AJAX transport methods, interactive SVG or Adobe Flash would work very well for this task. This would be a good fit into the existing interface, which already utilizes AJAX for asynchronous requests on most pages. Considering the volume of data in UMLS, visualizations may make some of the relationships existing between concepts more clear. It could also be possible to view only certain types of relationships. A representation of the media files might be used to show the associated concepts, and find other media sharing some of those concepts. Many tools for the visualization of complex networks already exist. The additional dimension added by interactive visualizations would improve the user experience, and could expose results and information that might otherwise be overlooked by users.

4.4.4 Metadata

Due to time constraints, the original intention to adhere to the IEEE Learning Objects standard (IEEE 1484.12.1 – 2002) for metadata of the media items was deemed to be outside of the scope of the project, and would not be implemented. Considering the education-centric nature of this resource, if it is seen that learning management systems or course creation assistant software packages gain popularity, then it will likely be worthwhile to add support for learning objects. The stored media items in BioMeRSA could then be utilized by the

aforementioned systems in an automated manner, with obvious benefits to those faculty and students under biological sciences utilizing such learning software. At present, only a small number of metadata fields are available through which users can add details for the files that they upload, and only one of these fields is actually required for each file (the copyright or license type). Despite their being seemingly few fields, it could end up being adequate- it is unlikely that each user would add details for more fields, even if they were available. However, it may become apparent over time that the current level of detail for the user-entered metadata is inadequate, and more fields may need to be added. There could perhaps also be methods for expert users to modify certain metadata parameters for a file, or metadata fields only for the experts, instead of the sole 'comments' field which currently exists on the annotation pages.

4.4.5 Media Files and Management

Although very solid support for creating previews of large image files or image files not typically supported natively in web browsers was implemented through the utilization of ImageMagick, no support exists for previewing any other kind of files (video, audio). To have any sense of the content of the file without downloading the full file, the user must rely on the provided metadata and associated UMLS concepts. This could be an issue for large video files, where if a user downloads a large file, only to find that it does not contain any content of interest, it is a frustrating situation for the user, and a waste of bandwidth both at the client and the server. Video and audio compression schemes are more complex than that for images and are also rapidly evolving, making it necessary that such a tool be mature and frequently updateable to be considered for use in a production environment. These previews would likely have to be streamable files that would be highly compressed and loaded in a small Adobe Flash-based player, to keep the focus on the web interface, and so that users will not have to download the preview files and play them using locally installed software. There are a few

solutions to this- the first being that ImageMagick does support the creation of screenshot images or animated GIF files from videos, but does not support a wide range of file types, nor audio, and a screenshot or short animation may not be a useful indication of the content of a video file, as it depends on where in the file the preview would be from. The most complete ImageMagick-like solution at present would be to utilize an existing interface to FFMPEG, an open-source and multi-platform project with libraries for the decoding and encoding of audio and video(FFMPEG, 2009). Using an existing Perl interface to FFMPEG, audio and video files from nearly any common format could be re-encoded into a format that would be more suitable for embedded streaming web display, such as Flash Video (FLV, for video) or MP3 (for audio)(Day, 2006)(FFMPEG). Note that practical use of such a feature would probably require an upgraded server, or perhaps even a cluster of servers, to handle the conversion process for multiple files in a timely manner. In the current single-server environment, a processing queue would almost certainly be needed, and if many large files were to be submitted in a short period, there would be likely be long delays before the processing of preview files was complete. Additional disk storage would also be needed, but much less than for the original files.

The potential integration of content from the MEDLINE database was discussed earlier in this section. Although there are other issues associated with the importation of documents from MEDLINE, at the moment there is some support in BioMeRSA for document-type file formats. Most MEDLINE documents could likely be obtainable as PDF files, which would be easy to work with as their ubiquity on the web has led to support of PDF in multiple tools. ImageMagick as well supports working with PDF files as if they were images. If working with PDF, previews of files could even be as text if a given document utilizes PDF's support for text objects and is not simply an image embedded in a PDF (such as with older documents which

may have just been scanned and not also processed through OCR). Other simple document formats could be supported as well, such as plaintext files or rich text, if needed. Support for such preview functionality could be implemented by utilizing a service such as iPaper from Scribd (Scribd, 2009). Overall, some further development on the media file handling procedures would be needed to handle document-type files well.

There is also room for significant improvement in the management of files in BioMeRSA. In particular, there is no way to remove or edit files that have already finished the annotation process without having to do so manually. It seems that a good system for this would involve a kind of reporting and re-evaluation procedure. A single user (expert or other users) could click a link or button on a media file information page which would be similar the 'report this' links commonly found on many sites that openly allow the uploading of content (an example using Google Video is in figure 4.1). The user would then be able to specify if they are flagging the file to be reviewed for the purpose of deletion (inappropriate content, corrupt file, or otherwise violating service terms) or for editing (if an annotation is thought to be inaccurate, or if metadata needs to be corrected) along with a field for comments stating their reason for reporting the file (which would probably be a required field). The file status could then be changed from 'live' (viewable in the content and browsing interfaces) to something like 'review', and would then appear to experts for them to either approve deletion or make the appropriate changes and re-approve the file for 'live' status.

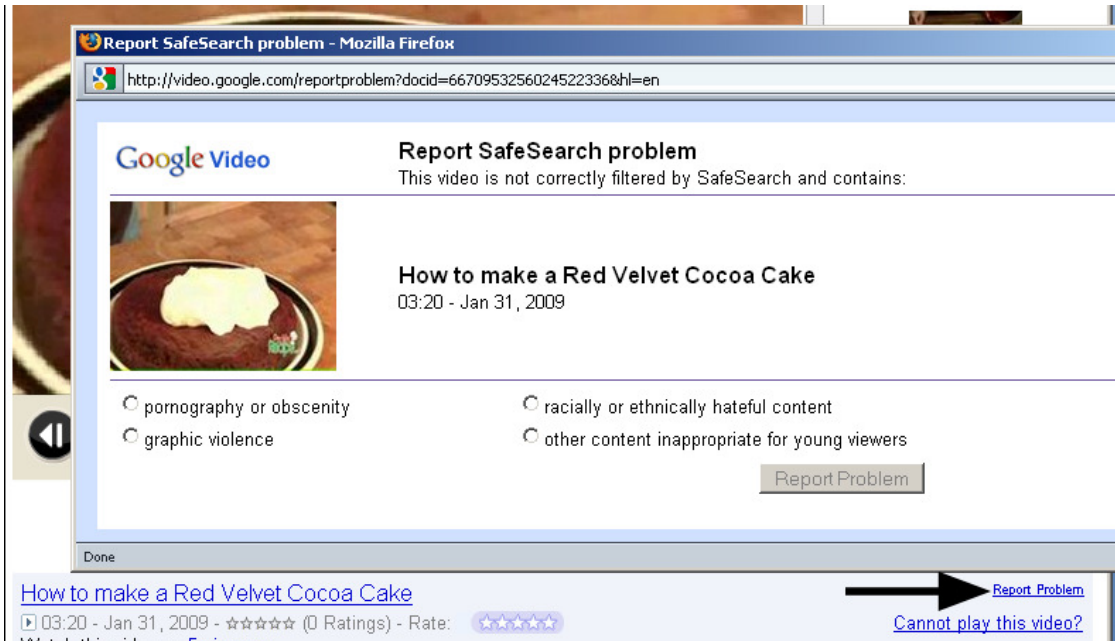


Figure 4.1 Example of links to report files in Google Video, which brings up the window in the foreground. The arrow points to the link, the text of which is "Report Problem"

4.4.6 Other Application Improvements

Altering the current configuration file system used in BioMeRSA would greatly increase the flexibility of configuring those options. At present, the configuration 'file' storing the settings is actually a Perl script which returns takes a string as an argument corresponding to the configuration parameter to return, and then returns a variable which is set to the desired value within the script. Thus, it is necessary to modify the script manually to alter these values. However, this arrangement means that the logic that deals with returning the configuration values is already abstracted away from the rest of the system, so that if the method of obtaining the values changed inside the BioMeRSAConfig script, none of the current calls to fetch values from the configuration file would need to be changed in the rest of the system. This makes it much simpler to expand on the current functionality by altering the current configuration script to read the values from a formatted flat file instead. Using a flat file, such as a file with XML data or with key-value pairs, instead of a database table for storing and retrieving configuration values allows for more options in utilizing existing Perl modules for dealing with configuration

files. Such a module, which deals with XML configuration files, would be XML::Conf (Nielson, 2000). There are also plugins for CGI::Application which handle working with configuration files, like CGI::Application::Plugin::ConfigAuto and CGI::Application::Plugin::ConfigSimple(Stosberg, 2005)(Peters, 2005). If one of the CGI::Application plugins were to be used, the calls to the plugin methods would likely end up replacing the calls to the current configuration script. If the module utilized supports changing configuration values, then it would be possible for a web interface to the configuration file to be developed, so that an administrator could change the values without editing the file directly in a text editor, or without having to connect to the server at all outside of the web browser.

There is significant room for improvement in the handling of errors which occur during the use of the system in the Perl application code. This includes normal errors, such as data not being found in the data for a given search term, as well as unexpected errors which could be the result of tampering attempts. Currently, there is not a unified method of handling these errors across the program, and thus also the methods of communicating this status to the user interface are varied, if they exist. Creating a single framework for the handling of these errors, perhaps through the development of subroutines in a separate class to handle that functionality, would not only provide a robust error handling system, but also streamline adding code to deal with errors when the system is later extended with additional features available to users.

Developing other user interfaces in addition to the existing browser-based one would expand the possible applications of the system. The model-view-controller design paradigm utilized in the existing implementation reduces the amount of code which would need to be modified for supporting another interface without having to rewrite or port the entire back-end.

While more capable web browsers are appearing on devices such as the iPhone, which can render full web pages and supports JavaScript, the current BioMeRSA layout is not by any means optimal for viewing on the smaller screens of mobile devices, and would need to be customized. For devices where a fully-featured browser is not present, a Web Service approach could be utilized, along with the development of a full native front-end application for the device. The Web Service approach would likely be the method of choice if working with Windows Mobile-based devices. This could allow a specialized submission interface to be created, which could enable submission of media from the field. With the Web Service approach, more of the logic would be moved to the client-side, as multiple server requests from a wireless mobile device are going to be much slower than executing the logic natively on the device itself, and so server requests would be used exclusively for getting data from or sending data to the database. A full desktop application could also be useful for the bulk submission of media- particularly if a user wishes to regularly, or perhaps even automatically (generated from a lab imaging procedure for instance) submit multiple items, it would be more effective to do so from a program running locally, which would also likely utilize a Web Service on the server.

5. Conclusion

Work on this project led to the development of a semantic system for multimedia files which was named the Biology Media Repository with Semantic Augmentation, BioMeRSA. Although many biology-related media databases exist, many depend on extensive manual curation or accurate metadata from the original submitter. This system is an attempt to strike a balance, making use of existing semantic data to provide a simple yet collaborative framework for the annotation of submitted files. BioMeRSA is based on some newer Web technologies, such as AJAX, which helps to provide a user interface that is useful yet not difficult or overwhelming to use. With some additional refinement, if public awareness of the system is eventually raised, and the means to support it exist, the system could become a popular resource in the target community.

As already discussed, the current state of the system provides a basis which can be extended significantly in the future. It is expected that this work will improve manageability such that less intervention on the part of administrators is necessary, expand functionality available to users, and make more use of the existing semantic data in UMLS. More complete support for increasingly complex media may also play a part in the success of the system as the move is increasingly made toward video and interactive media. With further adoption of Semantic Web principles, it may eventually be possible to connect this resource with other similar semantic-aware resources so that steps may be made toward fulfilling the vision of a Web with ubiquitous semantics.

6. Sources Cited

Adiscon GmbH. (2001, May 10). *Programming in Stateless Environments*. Retrieved March 2009, from Adiscon: <http://www.adiscon.com/iis/isapi005.htm>

AOL. (2009, April 22). *Dojo and AOL*. Retrieved April 2009, from AOL Developer Network: <http://dev.aol.com/dojo>

Aptana Inc. (n.d.). *Aptana Studio -- The Web 2.0 IDE for Ajax, PHP, Ruby on Rails, Jaxer and more*. Retrieved April 2009, from Aptana Studio -- The Web 2.0 IDE for Ajax, PHP, Ruby on Rails, Jaxer and more: <http://aptana.com/studio>

Barend, M., Ashburner, M., Chichester, C., & van Mulligen, E. (2008). Calling on a million minds for community annotation in WikiProteins. *Genome Biology*, 9 (R89).

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The Semantic Web. *Scientific American*, 284 (5), 34-44.

BIODIDAC Group. (n.d.). *BIODIDAC: A bank of digital resources for teaching biology*. Retrieved 2007, from <http://biodidac.bio.uottawa.ca/>

Blackboard Inc. (2005, May 10). *Blackboard Embraces IMS Global Learning Consortium*. Retrieved June 20, 2008, from <http://www.blackboard.com/company/press/release.aspx?id=707807>

Brandon, J. (2008, November 24). *Which Web Browser is King?* Retrieved April 2009, from Extremetech: <http://www.extremetech.com/article2/0,2845,2335254,00.asp>

Brent. (2007, January 14). *real-world postgresql vs mysql benchmark*. Retrieved from Bio and Geo Informatics: <http://hackmap.blogspot.com/2007/01/real-world-postgresql-vs-mysql.html>

Buitelaar, P., Eigner, E., & Declerck, T. (2004). OntoSelect: A Dynamic Ontology Library with Support for Ontology Selection. *Proc. of the Workshop on Knowledge Markup and Semantic Annotation*. Hiroshima, Japan: International Semantic Web Conference.

Burton, K. R. (2002, October 8). *MultiNode.pm -- a multi node tree object. Most useful for modeling heirarchical data structures*. Retrieved from search.cpan.org: <http://search.cpan.org/~krburton/Tree-MultiNode-1.0.10/MultiNode.pm>

Candler, C. S., Uijdehaage, S. H., & Dennis, S. E. (2003). Introducing HEAL: The Health Education Assets Library. *Academic Medicine*, 78 (3), 249-253.

Carazo, J. M., & Stelzer, H. K. (1999). The Bioimage Database Project: Organizing Multidimensional Biological Images in an Object-Relational Database. *Journal of Structural Biology*, 125, 97-102.

Carazo, J. M., Stelzer, E., Engel, A., & Fita, L. (1999). Organising multi-dimensional biological image information: The Bioimage Database. *Nucleic Acids Research*, 27 (1), 280-283.

Conrad, T. (2004, April 12). *PostgreSQL vs. MySQL vs. Commercial Databases: It's All About What You Need*. Retrieved from DevX: <http://www.devx.com/dbzone/Article/20743>

CPAN. (2009, April). *The CPAN Search Site*. Retrieved April 2009, from CPAN: <http://search.cpan.org/>

Day, A. (2006, July 10). *FFmpeg - Perl interface to FFmpeg, a video converter written in C*. Retrieved April 2009, from search.cpan.org: <http://search.cpan.org/~allenday/FFmpeg-6036/FFmpeg.pm>

Denny, M. (2004, July 14). *Ontology Tools Survey, Revisited*. Retrieved June 6, 2008, from <http://www.xml.com/pub/a/2004/07/14/onto.html>

Dojo Toolkit. (2009, March 31). *Dojo 1.3 now available*. Retrieved April 2009, from Dojo The Javascript Toolkit: <http://www.dojotoolkit.org/2009/03/31/dojo-1-3-now-available>

Dojo Toolkit. (2009, April 22). *Sites, Companies, and Products Succeeding With Dojo*. Retrieved April 2009, from The Dojo Toolkit: <http://dojotoolkit.org/node/6652>

Emrick, S. (2009, April). personal communication.

EPIC. (n.d.). *EPIC - Eclipse Perl Integration*. Retrieved April 2009, from EPIC - Eclipse Perl Integration: <http://www.epic-ide.org/>

FFMPEG. (2009, March 26). *FFMPEG*. Retrieved April 2009, from FFMPEG: <http://ffmpeg.org/>

FFMPEG. (n.d.). *General Documentation*. Retrieved April 2009, from FFMEG: <http://ffmpeg.org/general.html#SEC3>

Flickr/Yahoo inc. (n.d.). *Flickr: Help: Tags*. Retrieved March 08, 2008, from Flickr: <http://www.flickr.com/help/tags/>

Garrett, J. J. (2005, February 18). *Ajax: A New Approach to Web Applications*. Retrieved from adaptive path: <http://www.adaptivepath.com/ideas/essays/archives/000385.php>

Gasevic, D., Jovanovic, J., & Devedzic, V. (2007). Ontology-Based Annotation of Learning Object Content. *Interactive Learning Environments*, 15 (12), 1-26.

Gonzalez-Cuoto, E., Hayes, B., & Danckaert, A. (2001). The life sciences Global Image Database (GID). *Nucleic Acids Research*, 29 (1), 336-339.

Google. (2009, April 22). *Google AJAX Libraries API*. Retrieved April 2009, from Google: <http://code.google.com/apis/ajaxlibs/>

Gruber, T. (2008). *Ontology (Computer Science)*. Retrieved 11 25, 2008, from <http://tomgruber.org/writing/ontology-definition-2007.htm>

Gruber, T. (2007, September). *What is an Ontology?* Retrieved June 8, 2008, from <http://www-ksl.stanford.edu/kst/what-is-an-ontology.html>

Hannyaharamitu, M. (2009, February 23). *JSON - JSON (JavaScript Object Notation) encoder/decoder*. Retrieved from [search.cpan.org: http://search.cpan.org/~makamaka/JSON-2.14/lib/JSON.pm](http://search.cpan.org/~makamaka/JSON-2.14/lib/JSON.pm)

Harvey, P. (2009, April 11). *ExifTool by Phil Harvey*. Retrieved from ExifTool by Phil Harvey: <http://www.sno.phy.queensu.ca/~phil/exiftool/>

Health Education Assets Library. (n.d.). *Health Education Assets Library - About HEAL: History*. Retrieved June 19, 2008, from <http://www.healcentral.org/about/aboutHistory.jsp>

Health Education Assets Library. (2008, June). *Health Education Assets Library*. Retrieved June 19, 2008, from <http://www.healcentral.org/index.jsp>

Hek, C. (2006, May 23). *CGI::Application::Plugin::Session - Add CGI::Session support to CGI::Application*. Retrieved April 21, 2009, from [search.cpan.org: http://search.cpan.org/~ceeshek/CGI-Application-Plugin-Session-1.03/lib/CGI/Application/Plugin/Session.pm](http://search.cpan.org/~ceeshek/CGI-Application-Plugin-Session-1.03/lib/CGI/Application/Plugin/Session.pm)

Henning, E. (2006, August 24). *SHA-1 hash function under pressure*. Retrieved March 2009, from The H Security: <http://www.h-online.com/security/SHA-1-hash-function-under-pressure--/news/77244>

Hillman, D. (2005, 11 7). *Using Dublin Core*. Retrieved June 7, 2008, from <http://dublincore.org/documents/usageguide/>

ImageMagick. (n.d.). *ImageMagick: PerlMagick, Perl API for ImageMagick*. Retrieved from ImageMagick: <http://www.imagemagick.org/script/perl-magick.php>

Japan Electronics and Information Technology Industries Association. (2002). *Exchangeable image file format for digital still cameras: Exif Version 2.2*. Standard Document.

jQuery. (2009, February 19). *jQuery: The Write Less, Do More, JavaScript Library*. Retrieved April 19, 2009, from jQuery: <http://jquery.com/>

jQuery UI. (n.d.). *Home*. Retrieved April 19, 2009, from jQuery UI: <http://jqueryui.com/>

Juve, G., Francoeur, H., Gordon, L., Jhatakia, S., Sharma, N., Jordan, T., et al. (n.d.). *Creating a Virtual Fault Database Using Ontologies*. Retrieved Apr; 2009, from SCEC Community Modeling Environment: <http://epicenter.usc.edu/cmportal/docs/OntologyPoster.pdf>

Kleinsorge, R. (2005, May 16 & 17). *Unified Medical Language System (ULMS)*. Retrieved 11 25, 2008, from National Library of Medicine: http://www.nlm.nih.gov/research/umls/pdf/mla_nlmtheater_2005.pdf

- Kleinsorge, R., & Willis, J. (2008, December 5). *Unified Medical Language System Basics*. Retrieved 2009, from Unified Medical Language System: www.nlm.nih.gov/research/umls/pdf/UMLS_Basics.pdf
- Learning Technology Standards Committee of the IEEE. (2002, June 12). *IEEE LTSC / WG12 / Final LOM Draft Standard*. Retrieved June 6, 2008, from IEEE Learning Technology Standards Committee: http://ltsc.ieee.org/wg12/files/LOM_1484_12_1_v1_Final_Draft.pdf
- Lindesay, V. (n.d.). *SchemaWeb - RDF Schemas Directory*. Retrieved June 9, 2008, from <http://www.schemaweb.info/>
- Marlow, C., Naaman, M., Boyd, D., & Davis, M. (2006). HT06, tagging paper, taxonomy, Flickr, academic article, to read. *Proceedings of the seventeenth conference on Hypertext and hypermedia*, (pp. 31-40). Odense, Denmark.
- McCray, A. (2003). An upper-level ontology for the biomedical domain. *Comparative and Functional Genomics* (4), 80 - 84.
- McGreevy, P., Shaw, T., Burn, D., & Miller, N. (2007). OLIVER: An Online Library of Images for Veterinary Education and Research. *Journal of Veterinary Medical Education* , 34 (4), 510-516.
- Microsoft. (2003, March 28). *How NTFS Works: Local File Systems*. Retrieved from Microsoft TechNet: <http://technet.microsoft.com/en-us/library/cc781134.aspx>
- National Library of Medicine. (2009, April 3). *2009AA Section 2*. Retrieved April 19, 2009, from Metathesaurus Documentation: http://www.nlm.nih.gov/research/umls/meta2.html#s2_7_1_1
- National Library of Medicine. (2009, April 2). *2009AA Section 3 UMLS Semantic Network*. Retrieved April 2009, from UMLS: <http://www.nlm.nih.gov/research/umls/meta3.html>
- National Library of Medicine. (2008, November 20). *Metathesaurus Documentation*. Retrieved from Unified Medical Language System: http://www.nlm.nih.gov/research/umls/metathesaurus_doc.html
- National Library of Medicine. (2008, November 21). *The SPECIALIST NLP Tools* . Retrieved March 2009, from The SPECIALIST NLP Tools : <http://lexsrv3.nlm.nih.gov/SPECIALIST/index.html>
- National Library of Medicine. (2006, March 28). *UMLS Metathesaurus Fact Sheet*. Retrieved 11 25, 2008, from Unified Medical Language System (UMLS): <http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html>
- National Library of Medicine. (2009, April 8). *UMLS Release Documentation Archive*. Retrieved April 2009, from UMLS: http://www.nlm.nih.gov/research/umls/archive/archive_home.html

Nielson, J. (2000, August 19). *XML::Conf - a simple configuration module based on XML*. Retrieved April 2009, from search.cpan.org: <http://search.cpan.org/~jonasbn/XML-Conf-0.04/lib/XML/Conf.pm>

NLM. (2008, January 14). *Medical Subject Headings - Home Page*. Retrieved March 5, 2008, from Medical Subject Headings: <http://www.nlm.nih.gov/mesh/meshhome.html>

NLM. (2008, March 05). *Unified Medical Language System (UMLS)*. Retrieved March 05, 2008, from <http://www.nlm.nih.gov/research/umls/>

NSF. (n.d.). *NSDL.org - About NSDL - The National Science Digital Library*. Retrieved March 5, 2008, from NSDL.org - The National Science Digital Library: <http://nsdl.org/about/>

Peters, M. (2005, December 18). *CGI::Application::Plugin::Config::Simple - Add Config::Simple support to CGI::Application*. Retrieved April 2009, from search.cpan.org: <http://search.cpan.org/~wonko/CGI-Application-Plugin-Config-Simple-1.01/lib/CGI/Application/Plugin/Config/Simple.pm>

Rajas, M. (2004, July 19). *Why can't I allocate 2GB of heap to the JVM on Windows, Part 2*. Retrieved 2009, from garbage collection II: http://blogs.sun.com/moazam/entry/why_can_t_i_allocate

Raskin, R. (2004). *Enabling Semantic Interoperability for Earth Science Data*. Retrieved April 2009, from SWEET Ontologies: <http://sweet.jpl.nasa.gov/EnablingFinal.doc>

Schiemann, D. (2009, October 27th). *Debunking Dojo Toolkit Myths*. Retrieved March 15, 2009, from SitePen Blog: <http://www.sitepen.com/blog/2008/10/27/debunking-doj-toolk-it-myths/>

Schneier, B. (2008, February 18). *Cryptanalysis of SHA-1*. Retrieved March 2009, from Schneier on Security: http://www.schneier.com/blog/archives/2005/02/cryptanalysis_o.html

Scribd. (2009, April 22). *iPaper Viewer*. Retrieved April 2009, from Scribd: <http://www.scribd.com/ipaper>

Seebach, P. (2008, July 22). *The stateless state*. (IBM) Retrieved March 2009, from IBM developerWorks: <http://www.ibm.com/developerworks/web/library/wa-state/>

SitePen. (2009, April 22). *Web Application Development*. Retrieved April 2009, from SitePen Inc.: <http://www.sitepen.com/>

Stosberg, M. (2005, July 23). *CGI::Application::Plugin::ConfigAuto - Easy config file management for CGI::Application*. Retrieved April 2009, from search.cpan.org: <http://search.cpan.org/~markstos/CGI-Application-Plugin-ConfigAuto-1.30/lib/CGI/Application/Plugin/ConfigAuto.pm>

Stosberg, M. (2009, January 2). *CGI::Session - persistent session data in CGI applications*. Retrieved April 21, 2009, from search.cpan.org: <http://search.cpan.org/~markstos/CGI-Session-4.41/lib/CGI/Session.pm>

The Dojo Toolkit Community. (n.d.). *The Official Dojo Documentation*. Retrieved April 21, 2009, from DojoCampus: <http://docs.dojocampus.org/>

The Eclipse Foundation. (n.d.). *Eclipse.org home*. Retrieved April 2009, from Eclipse: <http://www.eclipse.org/>

The Gene Ontology Consortium. (2000). Gene Ontology: a tool for the unification of biology. *Nature Genetics*, 25, 25-29.

The Kate Development Team. (2008, April 15). *Kate*. Retrieved April 2009, from Kate: <http://kate-editor.org/>

Tom's Hardware. (2004, December 14). *File/Folder Limits*. Retrieved from <http://www.tomshardware.com/forum/227251-46-file-folder-limits>

UMBC Ebiqurity Group. (2008, June 9). *Swoogle Semantic Web Search Engine*. Retrieved June 8, 2008, from <http://swoogle.umbc.edu/>

UMLS. (2009, April 3). *2009AA Section 2*. Retrieved April 2009, from UMLS: http://www.nlm.nih.gov/research/umls/meta2.html#s2_7_1_1

UMLS. (Unknown). The Unified Medical Language System.

Uschold, M., & Jasper, R. (1999). A Framework for Understanding and Classifying Ontology Applications.

Velichkov, P. (2009, February 3). *MooTools vs JQuery vs Prototype vs YUI vs Dojo Comparison Revised*. Retrieved April 2009, from Peter Velichkov's Blog: <http://blog.creonfx.com/javascript/mootools-vs-jquery-vs-prototype-vs-yui-vs-dojo-comparison-revised>

Vos, P. L. (2007, April 14). *BIODIC*. Retrieved 12 2007, from <http://www.ulb.ac.be/sciences/biodic/homepage2.html>

Windows Skills. (2007, January 16). *PostgreSQL vs MySQL benchmark*. Retrieved from Windows Skills: <http://wskills.blogspot.com/2007/01/postgresql-vs-mysql-benchmark.html>

World Wide Web Consortium. (2004, February 10). *OWL Web Ontology Language Overview*. Retrieved June 7, 2008, from <http://www.w3.org/TR/2004/REC-owl-features-20040210/>

World Wide Web Consortium. (2008, January 15). *Sparql Query Language for RDF*. Retrieved June 7, 2008, from <http://www.w3.org/TR/rdf-sparql-query/>

World Wide Web Consortium. (n.d.). *The W3C Markup Validation Service*. Retrieved April 21, 2009, from W3C: <http://validator.w3.org/>

World Wide Web Consortium. (n.d.). *The W3C Markup Validation Service*. Retrieved April 2009, from World Wide Web Consortium - Web Standards: <http://validator.w3.org/>

World Wide Web Consortium. (2008, May 22). *W3C Semantic Web Activity*. Retrieved Jun 7, 2008, from <http://www.w3.org/2001/sw/>

Wunderlich, M., Ott, A., & Bernauer, J. (2003). MEDIANOVO - A Media Database for Medical Education Research and Health Care. *AIMA 2003 Symposium Proceedings*, (p. 1080).

Yahoo! (n.d.). *The Yahoo! User Interface Library (YUI)*. Retrieved April 19, 2009, from The Yahoo! User Interface Library (YUI): <http://developer.yahoo.com/yui/>

6. Appendix A – Perl Modules Utilized

All of the following modules may be obtained at the Comprehensive Perl Archive Network, <http://www.cpan.org/>

CGI

Provides support for handling Common Gateway Interface (CGI) requests and sending back responses according to functionality implemented by the developer. Typically used to provide a basis for Web applications in Perl.

CGI::Application

A Web application development framework built on top and extending the functionality of CGI. Allows for the Model View Controller (MVC) development paradigm to be easily utilized, and uses 'run modes' for program execution control and flow.

CGI::Application::Plugin::Session

A plugin for CGI::Application for more tightly integrating CGI::Session into CGI::Application. CGI::Session provides management of user sessions.

Date::Calc

Provides methods for calculations related to dates based on the Gregorian calendar.

File::Copy

File::Copy has functions for copying, as well as moving files.

File::Path

Simplifies working with directories by creating or removing directory trees based on paths.

HTML::Template

Allows for cleaner, easier-to-maintain Perl CGI development by separating HTML from Perl code.

Image::ExifTool

An interface to the ExifTool Perl application, which is able to read Exif metadata tag information from images, and other metadata from some other file types.

Image::Magick

A Perl interface to the ImageMagick image manipulation suite.

JSON

An encoder and decoder for the JavaScript Object Notation data interchange format.

Tree::MultiNode

An implementation of an object for representing a multi node tree.