THE JIKITOU BIOMEDICAL QUESTION ANSWERING SYSTEM:

FACILITATINGTHE NEXT STAGE IN

THE EVOLUTION OF INFORMATION RETRIEVAL

A Dissertation Submitted
to the Graduate School
University of Arkansas at Little Rock

in Partial Fulfillment of the Requirement for the Degree of

DOCTOR OF PHILOSOPHY

in Bioinformatics

from the Bioinformatics Graduate Program of

the University of Arkansas at Little Rock and

the University of Arkansas for Medical Sciences

January 2013

Michael Anton Bauer

B.S., Computer Science, New Mexico Institute of Mining and Technology,

2005

B.S., Biology, New Mexico Institute of Mining and Technology, 2005

M.Sc., Bioinformatics, University of Arkansas at Little Rock and

University of Arkansas for Medical Sciences, 2008

This thesis, The Jikitou Biomedical Question Answering System: Facilitating the Next Stage in the Evolution of Information Retrieval, by Michael Anton Bauer is approved by:

Thesis Advisor:

Daniel Berleant, Ph.D.
Professor of Information Science

Thesis Committee:

Robert E. Belford, Ph.D.
Associate Professor of Chemistry

Alan J. Tackett, Ph.D.
Associate Professor of Biochemistry

William R. Hogan, M.D.
Associate Professor and Chief in Division of Biomedical Informatics

Radhakrishnan Nagarajan, Ph.D.
Associate Professor, Division of Biomedical Informatics at the University of Kentucky

External Member:

Jonathan D. Wren, Ph.D.
Assistant Member of the Arthritis and Clinical Immunology Research Program at the Oklahoma Medical Research Foundation

Program Coordinator:

Elizabeth Peirce, Ph.D.
Associate Professor, Chair of Information Science

Graduate Dean:

Patrick J. Pellicane, Ph.D.
Professor of Construction Management

Fair Use

Duplication

I authorize the Head of Interlibrary Loan or the Head of Archives at the Ottenheimer Library at the University of Arkansas at Little Rock to arrange for duplication of this thesis for educational or scholarly purposes when so requested by a library user. The duplication will be at the user's expense.

Signature_____

I refuse permission for this thesis to be duplicated in whole or in part.

Signature_____

THE JIKITOU BIOMEDICAL QUESTION ANSWERING SYSTEM: FACILITATING THE NEXT STAGE IN THE EVOLUTION OF INFORMATION RETRIEVAL by Michael Anton Bauer, 2013

Abstract

In clinical and biomedical settings researchers often use specialized search engines to acquire answers to technical questions or to verify experimental results from peer reviewed scientific literature. The outcome of such queries typically results in the reading and scanning of multiple Web pages and documents. Information retrieval is the science of retrieving relevant items and question answering (QA) is a specialized type of information retrieval with the aim of returning precise short answers to queries posed as natural language questions. In this dissertation I describe and discuss a QA system, named Jikitou (www.jikitou.com), which creates a dialog with the user that mimics human interaction and utilizes multiple search agents to answer biomedical questions.

Jikitou is designed to be modular to allow for easy modification and evolution of core components. An evaluation system has been devised, which allows for the systematic comparison among different algorithms for finding relevant answers. The system's architecture can be divided into four subsystems: knowledge base, question analysis, answer agents, and user interface. Multiple software agents find possible answers to questions, from which the most relevant are presented to the user. Relevant information is presented to the user which establishes a kind of dialog with the user to obtain feedback to refine the query. Answers are automatically marked up and linked to semantically relevant content

in other databases. The additional information is presented in a popup window that appears when a marked term is clicked.

Jikitou addresses two current requirement gaps in biomedical question answering, namely, incorporating multimedia information and an ability to interact with the user. There is a lack of systems that allow the user to establish context, utilize that information in the process, and automatically return the appropriate answer. Jikitou returns answers to biological questions rather than lists of documents, which reduces the need to read entire documents. In addition to addressing current gaps, the system demonstrates an architecture framework that can continually evolve, maintaining itself as a valuable tool to researchers not only for question answering but also for other information retrieval needs.

Dedication

This dissertation is dedicated to my loving wife Akemi and three daughters Kana, Sana, and Mana who supported me throughout the entire endeavor.

Acknowledgments

There are a number of individuals that I would like to acknowledge for their help and encouragement. I would first like to acknowledge my parents Mark and Lorna Bauer for their love and support and belief in me. They were a strong driving force that kept me pushing on when it felt like there was no end in sight.

I would like to recognize and express my gratitude to my advisor Dr. Daniel Berleant for his guidance throughout both my masters and PhD. His ability to provide guidance, while at the same time still allowing me to take the lead in my education where the perfect balance, which I feel will prove to make me a better researcher. I would also like to acknowledge my dissertation committee members Drs. William Hogan, Radhakrishnan Nagarajan, and Alan Tackett for their guidance. With a special thanks to Drs. Jonathan Wren and Robert Beflord. Dr. Wren provided me with the idea of designing a question answering system and provided invaluable advice and suggestion throughout the dissertation. And Dr. Belford whom I started working with on the HyperGlossary project provided excellent support which has led to my intellectual growth.

I would also like to acknowledge my classmate and dear friend Shweta Chavan for all her support whether if it was proofreading, being there when ever needed to bounce ideas off of, or just having the support of someone going through the same process to be able to talk to during the tough times. I will always treasure the times we were able to take a break and play ping pong.

I would like to give a very special acknowledgement to my wife Akemi who has supported my educational endeavors since undergraduate school and whose

support really made it all possible. Her understanding of late hours and weekends in the office made my ability to focus much easier. She was truly my rock and I cannot thank her enough.

Table of Contents

Chapter 1: Introduction

We live in an age where we have access to more information than ever before. This can be a double edged sword. The access to information allows for more informed and empowered researchers. On the other hand finding relevant information becomes an increasingly more difficult task. Clinical and biomedical researchers often use search engines to find short answers to biological questions or to quickly get validation of genomic and proteomic experimental results. Google (1) and PubMed (2) are well known and successful information retrieval systems, but once the results are returned in the form of a list of documents or sites, it is left to the user to scan the resulting list and linked pages for the relevant information. A simple search can quickly become a time consuming task when one must manually find the answer due to the number of hits returned using these traditional kinds of information retrieval systems. There is a need for intelligent information retrieval systems that can summarize relevant textual information while also incorporating multiple sources of information from reliable sources to satisfy a user's query.

1.1 Background

Question answering (QA) systems are an extension of information retrieval in which precise short answers are built and returned to the user in response to queries posed as a natural language question (3-5) . Currently there are few such QA systems specific to the genomic and proteomic domains. Thus,

there is a need for systems that are able to return short answers extracted from PubMed articles and other such sources that accurately answer genomic and proteomic questions. Reduction of the amount of irrelevant information returned to the user is expected to increase the productivity of researchers in the biomedical field.

There are many reasons why researchers perform literature searches:

- find gaps and limitations in a field
- find and compare results
- validate results against peer's research
- learn about a field

A question answering system might allow researchers interested in biomolecular interactions to fulfill many of these needs. Potential gaps in a field can be found if a particular question cannot be answered. If conflicting answers are presented to the user they can quickly get an idea of the controversies and conflicting opinions. Performing large scale interactome, proteomic, or genomic experiments results in large lists of genes, proteins, and other biomolecules and it is often necessary to review the literature to validate results or see what is known about them (6, 7). The aim of a question answering system like this is to make this task easier.

Grant et al. (6), for example, present a proteomic study of multi-protein complexes in mammalian neuronal synapses. They describe the protocol they followed and believe a systematic literature search should be done when you have a list of proteins and genes obtained during a proteomic or microarray

experiment. The paper states that learning what is already known about the proteins that they have identified can provide insight into what the next step should be in the research plan. This requires an exhaustive search of the literature for the entities of interest.

The author mentions that performing literature research on even a list of just 10 proteins can be a daunting task. Each gene or protein has multiple synonyms and it can be time consuming to create a query to encompass all possible terms. A method is described that utilizes text mining methods to automate the process of finding synonyms and searching the literature databases for papers pertinent to a protein in question. They were looking for papers that mentioned a mutation in a gene of interest that was associated with a human disease. Their method is broad and returns only a small percentage of relevant papers. This type of search still requires a large percentage of the literature research time to be dedicated to reading and skimming papers that do not contain any relevant information.

1.2 Aims and objectives

The main hypothesis in this study is that a question answering system can provide an improved method for information retrieval that deals with specific biological questions. The hypothesis is based on the concept that often the relevant information need can be found in a sentence or phrase contained within a document rather than having to read the entire document (8). Retrieval of this piece of information helps reduce the need for documents to be scanned or read.

This thesis will advance the information retrieval field by filling gaps that are present in information retrieval and question answering systems. Such gaps include the deficiency in the use of multimedia sources to enrich answers and interaction with the user to establish context (9, 10). There is also a lack of systems that enable users to systematically build and test a QA approach without the need to build an entire IR system from the ground up (11). It is the goal of this project to address these gaps and limitations. Multimedia sources are linked to key words matched in the answers that aid in the understanding of the answer. Sources include links to definition of terms, relevant information in other databases, and video. Design decisions taken during the development of the system allow it to systematically evolve and give other researches a base from which they can build novel search strategies immediately (contact the author for source code if you are such a researcher). This QA system advances the field with an application that returns answers to biomedical questions enhanced with data from multimedia sources in addition to providing a QA development tool.

Specific aims

The goal of this thesis is to advance QA in the context of building a text mining system with a question answering system that returns concise answers to genomic and proteomic questions. To achieve this, the following specific aims were fulfilled.

Aim #1: Build components of a base information retrieval system

Build the information retrieval base system components that are necessary for the QA system. Sub-tasks included the selection of a suitable search engine library to incorporate into modules and scripts to provide basic search and indexing capabilities.

Aim #2: Build a QA system onto the base

Build the question answering system on top of the base information retrieval system, also integrating multiple resources together to create a knowledge base which different search strategies can access. This approach will maximize the likelihood that the most relevant information for a particular information need is found using one of the search strategies.

Aim #3: Build a web interface

Build a web interface to the QA system. It is through this interface that standard users will interact with the system. They will submit their questions and answers will be returned to them. The interface will also allow for query refinement through user feedback. As well as providing a mechanism for presenting the content provided by the HyperGlossary. The requirements of the interface are that it be dynamic, responsive, and intuitive.

Aim #4: Enhance the answers using the HyperGlossary

Enhance the answers by incorporating a HyperGlossary module to connect answers to additional resources. Possible answers to questions can be presented to the user enhanced with links to additional information. Passing the

answers to the HyperGlossary increases the understandability of the answers through the addition of links to semantically relevant information.

Aim #5: Evaluate the system

Design and execute a method to evaluate the system. Different search strategies should be developed for use within the system and compared. An evaluation method that minimizes the need for human judges was desired.

1.3 Thesis organization

The rest of this thesis is organized as follows.

Chapter 2 Literature review

This chapter gives background information on current state of the art QA systems and is essentially a paper that we published in the journal Human Genomics entitled “Usability survey of biomedical question answering”.

Chapter 3 Jikitou design and methods

Chapter 3 goes into design and describes the methods and algorithms chosen for the QA system. I describe the software design paradigms followed and the reason for the choices. I continue in the chapter to give background on the basic parts of an information retrieval system and our approach to implementing these into the system. In this project many existing tools, databases and technologies are integrated and their function and use are also

described in detail. The chapter concludes with a detailed description of the interface and how the user interacts with the system.

Chapter 4 System evaluation

Evaluation of the system is explained in Chapter 4. A detailed description of the problem with evaluating IR tools is given and two evaluation methods that I have chosen to implement are detailed.

Chapter 5 Results

Evaluation results are described in this chapter. The results of both evaluation methods are further analyzed with additional evaluation measures calculated and the results presented. Agreement and disagreements between the two methods are highlighted. Performances of the different search strategies are compared.

Chapter 6 Future work

This chapter conveys possible future plans for improving the system. There are several directions that can be taken to further the research, advance the field of IR, and question answering in particular.

Chapter 7 Conclusions

Chapter 7 draws conclusions from the evaluation results as well as making a case for how the Jikitou system has advanced the fields of information retrieval and question answering.

Chapter 2: Literature review

2.1 Usability survey of biomedical question answering systems

Based on a paper published in the journal Human Genomics, 2012, 6:17, http://www.humgenomics.com/content/6/1/17.

Summary

Biologists have access to an ever increasing amount of textual information. Increased access to information allows for more informed and empowered researchers, while information overload becomes an increasingly serious risk. Thus, there is a need for intelligent information retrieval systems that can summarize relevant and reliable textual sources to satisfy a user's query. Question answering is a specialized type of information retrieval with the aim of returning precise short answers to queries posed as natural language questions. I present a review and comparison of three biomedical question answering systems: askHERMES (http://www.askhermes.org/), EAGLi (http://eagl.unige.ch/EAGLi/), and HONQA (http://services.hon.ch/cgi-bin/QA10/qa.pl).

Introduction

There are numerous general purpose search engines available online, but as information sources continue to proliferate, specialized and domain-specific information retrieval tools become more essential. One such domain is the

clinical and biomedical fields, where the body of scientific knowledge is large and increasing. To minimize searching and browsing time while maximizing usefulness of that knowledge and data, we are seeing considerable interest in biomedical/clinical question answering systems (12). Question answering (QA) is a specialized type of information retrieval that returns precise short answers to queries posed as natural language questions (3, 13-15). It is the goal of such systems to move the burden of skimming multiple documents, which can be quite time consuming, from the researcher or clinician to the computer. The recent successes of IBM's Watson on Jeopardy highlight the possibilities and potential power of QA (16). I present a review of three leading biomedical QA systems, askHERMES (17-19), EAGLi (20, 21), and HONQA (22-24), which are all publically accessible online. This paper is organized into sections based on key usability dimensions used to compare the different systems.

Information sources

An important factor for any domain-specific QA system is the accuracy and trustworthiness of the sources against which queries are performed. Most biomedical QA systems make use of MEDLINE abstracts as an information source (25) . Two systems that I reviewed, askHERMES and EAGLi, used MEDLINE as a major source of answers. In addition, askHERMES uses eMedicine,(26) clinical guidelines, PubMedCentral (27) full text documents, and Wikipedia. EAGLi uses Medical Subject Headings to help answer some definitional questions. HONQA uses websites that have been certified by Health

On the Net Foundation (HON) (28), unlike the other two systems that rely heavily on MEDLINE.

Response time and results

First of all, the systems vary in their response times and in the form of answers returned to the user (in particular, single or multiple sentences). All three QA systems return relatively short answers to clinical or biomedical questions instead of entire documents. Response time assessment is based on the relative amount of time it took each system to respond to a typical query.

EAGLi is quite slow and may not truly be ready for high volume traffic. In response to a question that the system 'understands,' a list of possible answers is displayed with corresponding levels of confidence indicated. Links to abstracts are also provided and grouped by which answers to the question they support. If a question is not understood, EAGLi returns a list of abstracts that contained some of the query terms. The program also provides a short snippet of text from the abstract that contains keywords from the query. Next to the text there are links to PubMed and to a page they call a "semantic summary" which displays the entire abstract and a list of all the Gene Ontology and SwissProt terms that were matched, along with the phrase they were mapped to. A score is given to indicate to the user the strength of the mapping. This information gives the user a way to understand why the system has determined that a particular abstract supports an answer or was given as the answer. A link to a matrix is provided on the main results page that can quickly give the user an overview of the terms that were matched in the abstracts. This system provides a degree of transparency to

the retrieval process that traditional information retrieval systems hide from the user. That in turn supports efforts by the user to efficiently figure out how to best phrase a query or question to get the most relevant information.

The askHERMES system responds significantly more quickly than EAGLi or HONQA. It warns that it may take up to 60 s, but more often than not, it returns results in only a few seconds. Query terms are determined first by identifying noun phrases in a question which are then weighted based on several methods. The query is subsequently expanded using the Unified Medical Language System (UMLS), dictionaries, and thesauruses. Answers that are returned in response to a question can be viewed in three different arrangements: clustered answers, ranked answers, and content clustered answers. Clustered answers are first grouped according to different combinations of query and UMLS query expansion terms. They are then sub-clustered by different combinations of synonym concepts. This functionality can be useful in answering a complex question, such as one about a cause and treatment, which may require reading several different passages to find an answer. This is useful because often a sufficient answer cannot be found in just one sentence or short passage. Content clustered answers provide a third method to view answers. Common labels are found for the original clusters, and additional answer passages are found that match these labels. This approach allows a passage to be found under multiple, easy to read labels. A list of related questions is shown and can be used to further refine the one's own query question. The answers returned by the system are short passages or phrases from MEDLINE abstracts which are linked back to

the original citation. The system classifies questions into several categories defined by the National Library of Medicine (NLM) (29), such as diagnosis, treatment and prevention, etiology, pharmacological, management, and others. This classification aids in identifying query terms to use in retrieval.

HONQA is about as slow as EAGLi but it does display a status bar so that you can better tell whether it is working or has hung. Next to each answer, you can indicate whether a response to the question was appropriate or not. This is intended to help improve the quality of the answers provided by the system over time. Answers are linked to cached versions of the websites from which the sentences were obtained. The answers are sentences taken from HON certified websites. A health and medical website can apply to be certified, after which the HON organization will evaluate the site to see that it meets 'The HON Code of Conduct for medical and health Web sites' (HONcode) (30). The use of certified health websites as a source of knowledge is unique to the HONQA system. It was the intent of the designers of HONQA that users with different levels of health and biological knowledge be able to benefit from answers that are understandable and useful. MEDLINE contains high quality peer reviewed literature but can be technically difficult to understand, whereas websites are typically designed and geared for a more diverse audience. However, a significant problem with using the Internet as a source of health information is the lack of oversight of the information that is presented. The HON certification helps alleviate the problem of incorrect and possibly dangerous medical information on the Internet. Another benefit of using websites as a knowledge source is that

there are links to additional information present in most web pages (and absent from MEDLINE abstracts) that can often help answer the question if the sentence returned does not completely answer it.

User interface

EAGLi provides a simple and clean interface which allows users to ask a question and either use the PubMed search tool or their specialized relevance driven search engine. Most of the items on the page can be hovered over with the mouse to display a small tooltip containing a more detailed description of the item. The terms that are selected from the question to be used to query are displayed on the results page. The system appears to reformulate and automatically expand the queries with the addition of Gene Ontology and SwissProt terms.

The interface to askHERMES is also simple and clean with multiple tabs. At the top of the results page are links to clinical question answering tools, which include utilities to browse questions, classify questions, and generate query terms. A question browsing utility allows browsing the NLM collection of clinical questions that they used while developing and tuning the system. A question classifying utility lets the user submit a question and see in which category the question is categorized. An *ad hoc* question can also be submitted to the query term generating utility to get a list of the keywords that would be extracted and used by the system to query the database. These utilities can help the user understand how the system answers questions that are posed, similar to the “Semantic Summary” of EAGLi.

HONQA has a very simple and easy-to-understand interface. When results are returned, information about how the question was interpreted is provided and includes: the number of answers, the language, expected question type, and expected medical type. HONQA does some interpretation of the question to determine the type and kind of medical information being requested. Question types can be definition, factoid, list, and Boolean. The medical types a question may be include definition, diagnostic, physiology, and treatment. This helps the user determine whether the system understands the intent of their question.

Answer quality

The askHERMES system returns passages that could potentially answer all types of questions. A drawback is the consequently high recall; a large number of results are often returned, which tends to defeat the intent of a question answering system in reducing the amount of information that must be read. HONQA returned fewer answers to many biomedical questions and is tuned for medical questions. I observed that HONQA was able to present sentences that answered questions to definitional clinical questions. The sentences returned by the system were clear and easy to understand, and often, following links to the cached source texts for further elaboration was unnecessary. The EAGLi system was unique in that, when it understood a definitional question, it would return a list of target answers with different levels of confidence in addition to supporting abstracts. If a question was not understood, it would just return abstracts that contained the query terms without the list of

possible answers. Thus, while long, complex questions tended to lead to no results from EAGLi and HONQA, askHERMES returned results for any size and type of question posed. This strategy strongly suggests itself as a general architectural feature for future QA systems.

Summary

There are considerable interesting differences between the three systems. HONQA returns single-sentence answers that are clear and easy to understand. Although EAGLi provides single entity answers, it still seems to be often necessary to read the abstracts to validate the answers provided. It also presents the user with many different ratings and views which can be confusing. With its quick responses, askHERMES is currently the most useable of the three systems, especially if it is necessary to make multiple queries. Table 2-1 summarizes the dimensions and comparisons of the different systems. Biomedical question answering systems are improving and will be ready for prime time, perhaps surprisingly, soon. These three systems demonstrate that they are close to becoming valuable tools for the clinical and biomedical fields.

QA Comparison Matrix

	EAGLi	**askHERMES**	**HONQA**
Web address	eagl.unige.ch/EAGLi	www.askhermes.org	services.hon.ch/cgi-bin/QA10/qa.pl
Data sources	MEDLINE abstracts	MEDLINE abstracts, eMedicine, clinical guidelines, PubMedCentral, and Wikipedia	HON certified websites
Answers	Multi-phrase passages and a list of single entities	Multiple sentence passages	Sentence
Language	English	English	English/French
System response	Slow	Fast	Slow
Interface complexity	Complex but many tooltips	Simple	Very simple
Question analysis	Yes	Yes	Yes
Target question types	Definition	All types	Definition, procedure, factoid, who
Key feature	Returns a list of ranked terms to answer "factual" questions	Answers are presented in three ways: answers clustered by terms, simple ranked answer list, and answers clustered by content	Use of certified health websites which allow for information to be geared towards people with varying levels of health literacy

Table 2-1 Question answering system comparison matrix of features for HONQA, askHERMES, and the EAGLi systems.

Chapter 3: Jikitou design and methods

This chapter details each part of the system, describing key functions and algorithms. It also details the external programs and databases that have been integrated into the system and their application. The reader can get an explanation of the design concepts and the reasons for their selection.

3.1 Design rationale

The system that I have developed is called Jikitou and can be found at the web address www.jikitou.com. Jikitou is the Japanese word for “prompt answers/direct personal answer”, the tenets that guided the design and implementation of the system. Question answering, being an advanced information retrieval task, builds upon and combines many basic text-mining tasks that in turn use many methods and tools, see Figure 3-1.

Jikitou incorporates many of the common methods which are used in the more mature text-mining tasks of information retrieval, entity recognition, and information extraction to find answers (15, 31, 32). Information retrieval is the process of identifying text segments relevant to a topic, which is accomplished using methods that find similarity between them. Information extraction is the process of identifying predefined types of fact from the literature. The facts may be specific relationships between biological entities; for example, protein-protein interactions or gene-protein interactions. The most commonly used strategies for automated information extraction are co-occurrence analysis, template matching,

and natural language processing. Entity recognition is the process of identifying specific terms of interest. In the case of biomedical text-mining the usual entities of interest are genes, proteins, or diseases.

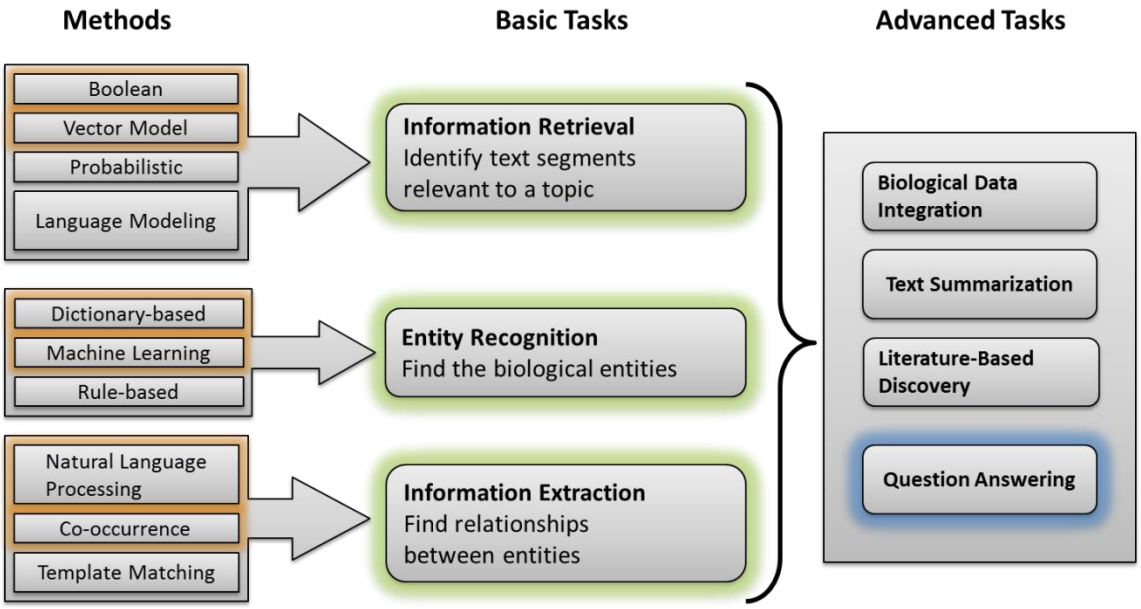


Figure 3-1 The basic text-mining task, common methods to accomplish those task and some of the possible advanced tasks that are built using a combination of them. Highlighted items show which methods and tasks are used in the Jikitou QA system.

As mentioned previously a question answering system is an advanced information retrieval system. A basic information retrieval system has five main components (33) shown in Figure 3-2. Similarly Jikitou contains these components, most of which are implemented using Apache Lucy, an open source search engine. The rest of this section describe, the basic information retrieval components and the Apache Lucy package

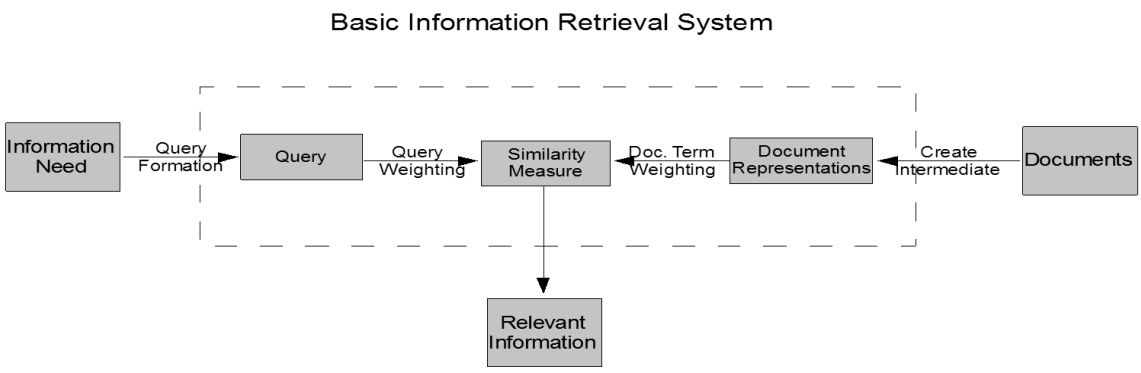


Figure 3-2 Basic organization of an information retrieval system

Apache Lucy

Apache Lucy (34) is a full-text search engine library which is a loose C port of Apache Lucene, which is written in Java. Lucy was chosen as the base search library used by Jikitou because of the availability of Perl bindings, which allow Jikitou to be programmed in Perl and to take advantages of that language and still have the benefits that you are afforded with Lucy being written in C.

Searching

The actual searches are performed by the Lucy::Search::IndexSearcher module which executes a search against a single index. It takes as a parameter the directory path of the index of choice. The index is one that would have been created in an earlier process and is discussed in more detail in the section on indexing later in this chapter. The hits method made available through the IndexSearcher can accept a plain query string or a query object. A query object is created by using the Lucy::Search::QueryParser, which is used when complex queries are desired.

Query formation

This is the process of building a query from the user's input. The user has an information need and inputs an initial string either in natural language or Boolean form depending on the system. Query formation in Jikitou takes a user's question and then modifies the query based on feedback from the user through the selection of additional terms suggested by the system. Additionally, the query may be further modified through the use of the knowledge base for example, and through automatic query expansion by including synonyms of terms identified in the initial question.

Jikitou Query Formation

An important aspect of question answering is the formation of a query that can be used to search an index that returns answers that are semantically relevant to the question posed by the user. In Jikitou, the Lucy module that is used in query formation is a subclass of the Lucy::Search::QueryParser module which can take several syntactical constructs to build complex queries which may include:

- Boolean operators 'AND', 'OR', and 'AND NOT'
- Prepended '+' or '-' to indicate whether a term is required or forbidden in the results
- Phrases indicated by double quotes

The module used in Jikitou is named FlatQueryParser and is used to transform text produced after question analysis into a query object.

FlatQueryParser is a simplified custom search query language which, as of now, only supports simple term queries and phrases identified by double quotes. It takes a string of text and builds the query object that is then used by the search module. The string is tokenized and terms enclosed in double quotes are identified as phrases and a phrase query object is created. Terms without quotes are turned into individual term query objects. After the string has been completely parsed, the query objects are unioned together by adding it as a child of a Lucy "OR" query object which is then used by the searcher to retrieve the closest matching text.

Document representation and document weighting

This is the process by which the documents are converted to a secondary representation known as an index. Here is where the terms to be indexed are identified. The most basic document representation is a matrix with the vocabulary for the collection on one axis and all the documents on the other axis. If a term is present in the document it is represented as a 1 and if it is absent it is represented with a 0. Most often a term is instead represented with a value that represents its relative importance and ability to discriminate between different documents. This involves a weighting scheme to assign different values to different terms depending on their potential document discriminatory power, meaning their ability to distinguish one document from another. The indexed terms can be terms in a document set or a subset based on a controlled vocabulary or at least the removal of stop words and other common words which have very low document discrimination power. It can also be indexed by groups

of terms semantically relevant to one another, such as using Latent Semantic Indexing (LSI). LSI is based on the principle that terms that are used in similar textual context often have similar meanings.

Parsing and Indexing

Parsing is the process of analyzing text to determine structure. Parsing requires tokenization, which is the process of segmenting text into meaningful parts. Parsing can be as simple as finding terms and sentence boundaries or as complex as determining grammatical structure. A parser tells the computer what it should consider a token from the sequence of bytes it receives as it scans a document. The results of parsing are placed in an index.

Indexing is a process of converting a corpus into a form that can be quickly searched. When a document is indexed the entire corpus does not have to be scanned to locate the desired information. The indexing process converts a document to an intermediate form that is structured to allow the use of traditional data mining techniques to discover relevant information. There are many types of intermediate forms, each capturing different levels of semantic information (35). The basic difference between the intermediate forms is the minimal text unit, which can be a word, phrase, sentence, paragraph, or even an entire document. Table 3-1 contains some text units and possible intermediate forms.

Text Unit	Intermediate Form
Word	Bag-of-Words
	N-grams
Concept	Concept Hierarchy
	Conceptual Graph

	Semantic Graph
	Conceptual Dependence
Phrase	N-phrase
	Sentence
	Multi-term text phrase
	Trends
Paragraph	Paragraph
	N-phrases
	Multi-term text phrase
	trends
Document	Document

Table 3-1 Text unit and possible intermediate forms (35).

Standard Inverted Index

A standard inverted index has, for each term, a posting list associated with the documents in the collection (36). Each term is represented by a value or weight. Term weighting during the creation of a document representation is a process for giving terms a value for perceived level of importance. In its simplest form a 1 is assigned if the term is present in the document and a 0 is assigned if the term is not present in the document. The resulting matrix gives a vector for each term and the documents it can be found in, or a vector for each document and the terms that occur in it. A common weighting scheme is the *tf-idf* (term frequency / inverse document frequency) approach, which is described in more detail next.

The following procedure describes the process of calculating *tf-idf* and the theory behind the weighting scheme.

Local weighting – Terms that occur several times in a document have a higher probability of being more meaningful indicators of document topic than terms that occur just once. A local weighting scheme will give these terms a higher weight. This is called term frequency (*tf*).

Global weighting – Terms that are used infrequently in the entire corpus are likely to be more meaningful indicators of document topic than terms that are common or occur frequently. A global weighting scheme gives these terms a higher weight. This weight is called inverse document frequency (*idf*) and is given by

$$idf_t = log \frac{N}{df_t} \quad \textbf{(3.1)}$$

where *t* is a term and *N* is the number of documents in the collection. *df* is the *document frequency* and is defined as the number of documents that contain term *t*. Term frequency-inverse document frequency is then calculated by

$$tf\text{-}idf_{t,d} = tf_{t,d} \times idf_t \quad \textbf{(3.2)}$$

It is this standard indexing scheme that is used in Jikitou. Each sentence in this case is considered a document. The sentences are tokenized where terms are located by the space boundaries between them. The terms are then case folded (make lower case) so that searches will be case insensitive. Next the terms are stemmed which reduces them to their base forms. A Snowball (37) stemming library was used. Three fields are indexed; the sentence id, PMID

number and the sentence. Only the sentence field is available for full text search. Lucy creates a group of flat files that contain the index.

Query weighting

It is this process that gives the weights to the terms in the query. The weights are often identical to the weights given to the terms in the document. Weights can also be assigned based on dialog with the user and the importance put on certain terms.

Similarity measure

The similarity measure determines similarity between the query vector and the document vector. A possible measure is the cosine similarity measure, which finds the similarity between a query and a document. A vector is created with the tf-idf being the value for each term. The score is then calculated using the dot product of each term in the non-judged document *q* that is present in the judged passage *d*. This value is then normalized by dividing this score by the product of the Euclidean distance between *q* and *d*. The cosine similarity measure is used to compare each Jikitou answer to every relevant judged passage and the average score is recorded. Figure 3-3 is a simplified example of some number of sentences in an *n* term vector space. It is the angle between the query and each

$$Sim(q,d) = \cos(\theta) = \frac{q \cdot d}{\|q\|\|d\|} = \frac{\sum_{i=1}^{n} q_i \times d_i}{\sqrt{\sum_{i}^{n}(q_i)^2} \times \sqrt{\sum_{i}^{n}(d_i)^2}} \quad \textbf{(3.3)}$$

of the sentence vectors.

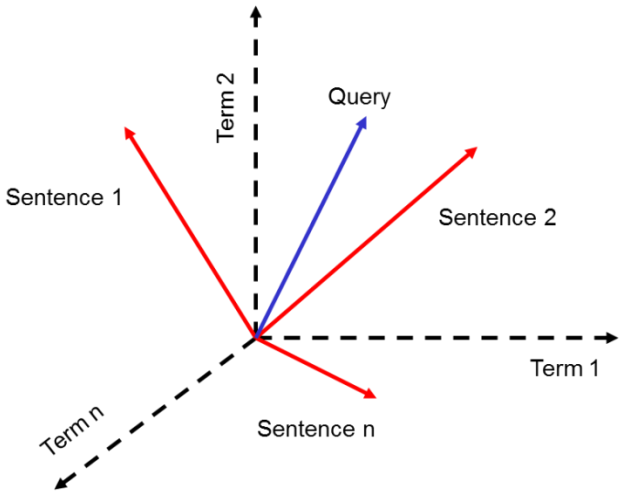


Figure 3-3 A diagram of sentences and a query in a vector space where each dimension is a different term. Cosine similarity measures the distance between the query vector and the different sentence vectors to gauge similarity between them.

Once the similarity measure has been completed the items of information which are most similar to the query are presented to the user in rank order.

3.2 Jikitou design

Jikitou was built generally following the Model, View, Controller (MVC) framework which compartmentalizes the user's interaction with the system from the information/data in the system. The view encompasses any output from the system. The model includes system data and the knowledgebase. It is the controller that implements the "business logic" and is responsible for executing commands for the model or view. Catalyst, an open-source MVC web framework for the Perl language, was implemented in Jikitou. It is a flexible framework which facilitates the separation of the application logic from the display of information. Figure 3-4 is a simplified diagram of the Jikitou system that shows the Catalyst

MVC separation and key technologies and functions that are implemented. Catalyst is used on an Apache server using FastCGI, an improved implementation of the traditional Common Gateway Interface (CGI). When the Apache server is started the entire Catalyst project is brought into main memory and multiple FastCGI processes are started, which increases the speed/response to request to the application from users.

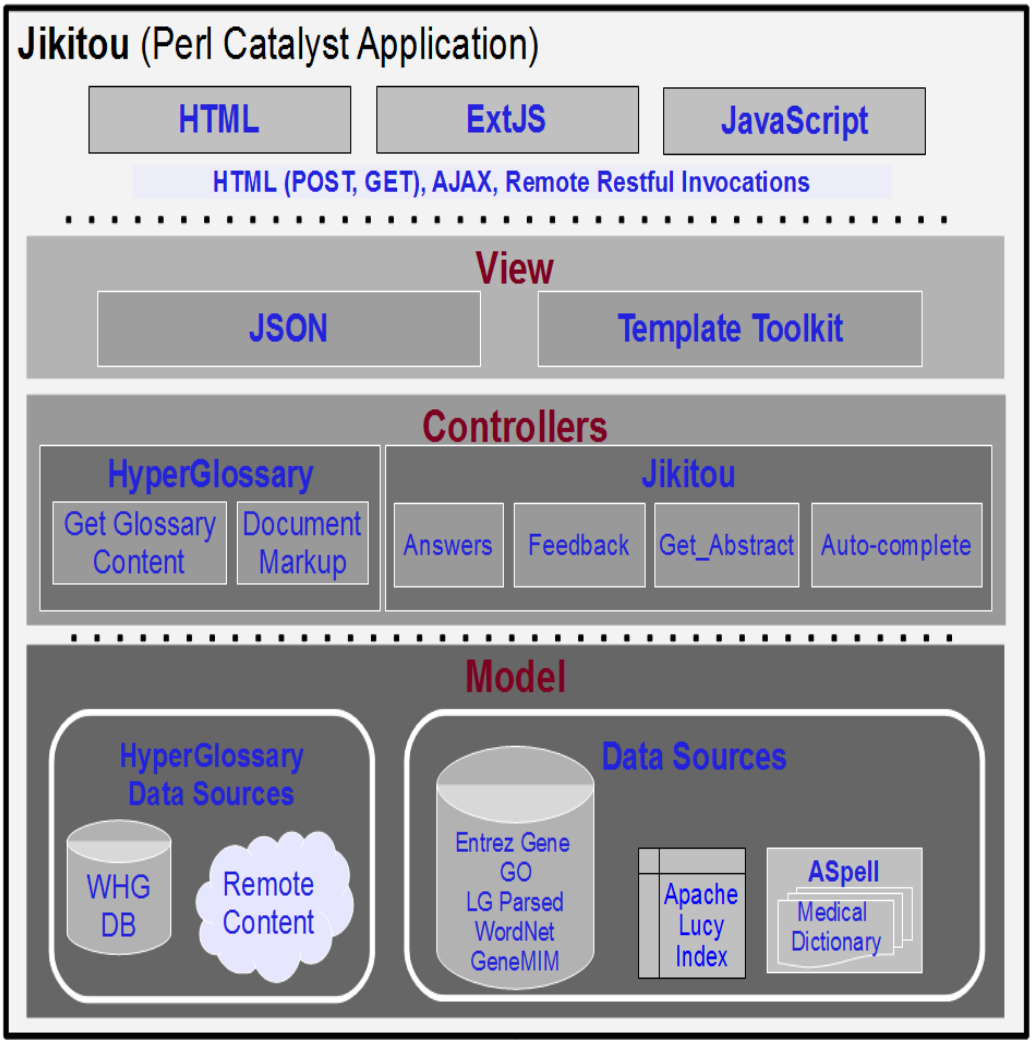


Figure 3-4 A diagram overview of the Jikitou QA components. The separation of the Model, View, and Controller is shown with many of the key parts of each shown.

Model

The model is the part of the application that accesses and modifies the data. Each module in the system that accesses the different databases is considered a component of the model.

Aspell

Aspell (38) is an open source spell checker that has been supplemented with OpenMedSpell. OpenMedSpell (39) is an open source spelling dictionary of 50,000 medical terms. The Aspell dictionary of terms has been enhanced with these medical terms which include terms for diseases and medications which can often be long and difficult to spell. Aspell is used in the autocomplete functionality and other modules that make use of it to gain access to its dictionary.

Natural language processing

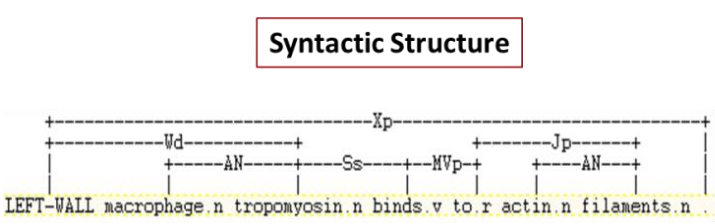
A natural language parser is a program used to determine the grammatical structure of a sentence. Natural language processing parses a sentence into the different parts of speech (POS) and uses a set of rules to identify possible relationships. The advantage of natural language processing is the ability to infer the direction of the relationship and distinguish between different relationships when more than two entities are present in the same sentence for example protein-protein interactions (40, 41). Jikitou uses NLP in a couple of ways. It uses the semantic structure to match answers based on the semantic distances of terms from the query in the potential answers. Many of the

agents use NLP parse structures to help identify POS and potential phrases to modify the user's query.

Link Grammar Parser

Jikitou uses a parser called the Link Grammar Parser (42) which is an English parser based on link grammar; it takes a sentence and calculates a syntactic structure, where pairs of words are connected by labeled links. The parser is written in C and has an API that a Perl module in Jikitou uses to gain access to its functionality. Figure 3-5 is an example sentence and the resulting Link Grammar syntactic tree.

Macrophage tropomyosin binds to actin filaments.



Code	Connects
Xp	periods at ends of sentences
Wd	main clause back to the wall
AN	noun-modifiers to nouns
Ss	subject-nouns to finite verbs
MVp	verbs (and adjectives) to modifying phrases like adverbs
Jp	prepositions to their objects
A	pre-noun ("attributive") adjectives to nouns

(S (NP Macrophage tropomyosin) (VP binds (PP to (NP actin filaments))) .)

Figure 3-5 The tree is the resulting semantic structure after being sent through the link grammar parser. The table contains explanation of some of the labeled links in the semantic structure.

View

The view is comprised of parts written using the ExtJS JavaScript library (43), plain JavaScript, and HTML. Data returned to the view use JavaScript Object Notation (JSON) which is a human readable format for data interchange. Passing data in this format provides a good example of the separation of the data from the view. The controller passes the data to the view with no instructions on how it should be presented to the user. The view can be modified without having to deal with changing the model or controller. The specifics on the Jikitou interface are described in greater detail later in this chapter.

Controller

The controller handles the request to the system. It gathers the required data from the model and converts it to a JSON object and sends it to the view to be rendered. Jikitou's controllers are explained in the section about the user interface because it is the controller that handles the user requests and it is mainly through the interface that request are made.

3.3 Knowledge base

A knowledge base is a collection of specialized database that contains information that is collected and organized. The knowledge base in Jikitou is the repository from which answers are selected and also includes the lexicons used to identify domain-relevant terms such as genes. The database consists of corpora from multiple domains and includes dictionaries, thesauruses, and a

domain specific ontology. The dictionaries and thesauruses allow concepts to be identified and connected to terms in the question. The knowledge base is used to reformulate some queries with the addition of synonyms and additional concepts.

The Interaction Sentence Database (ISDB) (44) is used in Jikitou system's knowledge base as the corpus and is thus the source of the answers. Sentences are indexed as though they were "documents" to be retrieved because they are a minimal textual unit that is often self-contained (8). ISDB contains sentences that were taken from PubMed abstracts. The sentences have been analyzed and filtered so that all sentences contain at least two biomolecule entities and at least one interaction-indicating term. This takes advantage of the fact that entities that co-occur in the same sentence are related. Thus these sentences are more likely to contain a biological interaction. The size of the database is approximately 4.5 million sentences.

3.4 Dictionaries and thesauruses

Jikitou has incorporated many publically available databases which have been locally installed. These are as follows.

- The **Gene Ontology** (GO) (45) is a controlled vocabulary for describing gene product characteristics. This database helps mitigate the inconsistent terminology used to describe gene products. The ontology is organized into three categories: cellular component, biological process, and molecular function. This

database is used in user feedback and query expansion in this system.

- **BioThesaurus** (46) is a protein entity dictionary that enables the retrieval of synonymous names of proteins and the identification of ambiguous names shared by multiple proteins.
- **WordNet** (47) is a database of nouns, verbs, adjectives, and adverbs that are grouped by cognitive relatedness. These groups are linked by conceptual-semantic and lexical relations. This database is mainly used in identifying synonyms for non-biological terms. It can also be an aid in natural language processing which is important in answering queries posed in the form of questions.
- **Entrez Gene** (48) contains a large and diverse body of gene information which includes synonyms, nomenclature, alternative ids, and interaction information.

3.5 Multiple search agent approach

Studies have shown (35, 49) that there is good reason to have multiple document representations and multiple searchers. Das-Gupta concluded, from the four document representations that they investigated, that there was low overlap between pairwise comparisons of the representations (average of 14% overlap) even though performance values differed only slightly. The low overlap suggests that the different document representations encode different information in the document. Any particular document representation may not be

optimal for all information needs. The lack of overlap between representations is the reason that this system is designed to have multiple document representations. The multiple search agents were implemented for the same underlying reason. Different agents formulate the queries differently and in the end match to different sets of documents. See Figure 3-6.

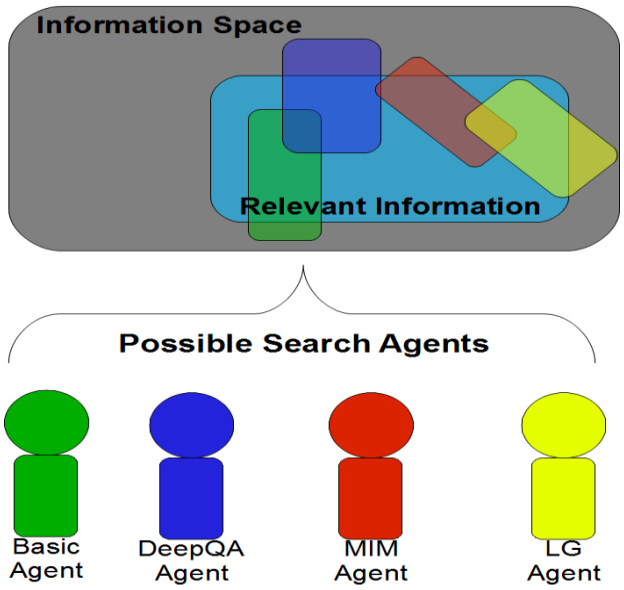


Figure 3-6 Representation of an agent based search approach. Each agent has the potential of getting a different set of relevant information.

Software agents are components that are implemented using goal concepts that are traditionally reserved for humans. They also interact with their environment and with other agents on a user's behalf. Software agents work with each other and the user to accomplish a set of goals. The use of software agents nicely fits the way that we think about complex tasks. The following group of properties is commonly used to define software agents (50, 51). Software agents are *persistent*, meaning the code is always running and not just executed when

needed. They are *autonomous* in their actions, making decisions and prioritizing tasks without human intervention. They have the ability to *socialize* with the other agents to work on common tasks together. The final attribute of software agents is that they are *reactive* and able to adapt to their environment (52). These properties of software agent reduce the need for complex descriptions of complex software designs. It is easy for people to conceptually see software agents as though they were human workers that communicate and work together towards the completion of a common goal in a dynamic environment.

There are several systems for information personalization and information extraction that utilize a multi-agent strategy (52-56). Agents are assigned to major tasks such as searching data sources, information retrieval, and interacting with the user. There are several reasons that make a QA system amenable to a multiple-agents based design (53, 54): The information is available in many distinct locations, the content is heterogeneous, and this content is constantly changing. It also aids in the modularity of the system. New types of analysis and sources of data can easily be added through the creation of new agents or modification of current agents.

In this project, multiple software agents were designed to find possible answers to questions from which the most relevant is presented to the user. The idea is that, just as when working in a team, each member brings different skills to address a problem. Software agents form a convenient and powerful way to describe a software abstraction of a system that is somewhat autonomous and brings a different set of skills to any particular problem. It is similar to Object

Oriented Programming in that pieces of the software are described as being objects that can be acted upon and return results in specific ways. Different agents use different algorithms and techniques to determine the different answers to a query. This multi-model approach takes advantage of the fact that different models are better for finding relevant passages to address different types of queries/questions. After completing the document retrieval, each agent returns its results to a dispatcher, which currently returns the top 50 results from each agent but in the future will be tasked with making decisions pertaining to which answers to present to the user.

Agent toolbox

In Jikitou the agents perform their search using a combination of modules that make up the available tools in the toolbox. Agents can share tools and use them in different combinations. The available tools fall into two categories: tools that aid in refining and the automatic expansion of the query, and tools that help re-score and re-rank answers.

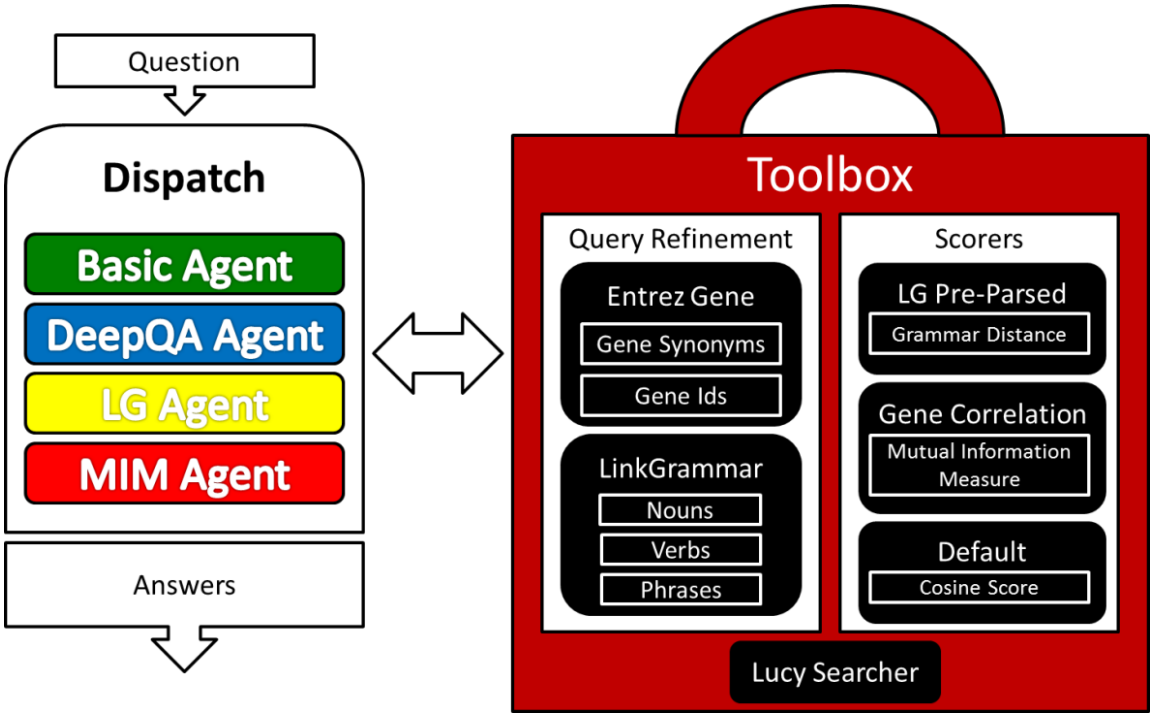


Figure 3-7 A depiction of the tools used by the agents. A question is submitted to dispatch which assigns it to each agent and forks off a new process so that the agents can work in parallel. Dispatch then waits for their completion then returns all the answers.

Automatic Query Refinement Tools

Two modules were created that aid in automatic query refinement. The goal of query expansion is to improve the probability of retrieving relevant content through the addition of terms to a user's query. Studies have shown that query expansion improves information retrieval results (57-60). Interactive and automatic are the two main concepts in query expansion. These two approaches to query expansion are supported in Jikitou. Later in this section I discuss automatic query expansion and the modules that implement it, which is used by several of the agents.

Gene Synonyms Module

The first module identifies possible genes and returns possible synonyms. The module takes a string as input. The string is case folded (transformed to lower-case) and tokenized. The tokens are then sent to an English dictionary to determine if they are actual words. Tokens that are not found in the dictionary are used in a query. Sending the terms to a dictionary first helps reduce the number of possible misidentified genes that are actually non-gene words. The "gene_info" table, located in a local installation of the Entrez Gene database, is queried to find possible gene symbols and gene synonyms. These additional terms are added to the original user's query.

NLP Link Grammar Module

The second module utilizes natural language processing (NLP). The user's question is submitted to the module where it is parsed using the Link Grammar Parser. If a complete linkage can be found, meaning that it is able to parse the entire sentence, the linkage tree is stored for additional analysis. If no linkage is found then no further analysis is performed by this module.

Using the linkage tree, two functions are used to identify the different parts of speech (POS) using regular expressions. The first function identifies all nouns and the second identifies verbs. These POS are identified because they are the most informative words which consequently have a greater chance of discriminatory power in identifying relevant sentences to answer the question. It is also a way to remove words that are common and have low discriminatory power.

The linkage tree contains labeled links which describe the connections among terms. The function that identifies phrases follows these links to build up a list of possible phrases to add to the query. It looks for noun phrases and verb phases. See Figure 3-11 for an example. Phrases in the query are queried as complete strings to look for answers that contain the exact phrase.

Scorers

The scoring modules in the toolbox take a list of *N* sentences returned by the search engine and re-ranks the sentences using additional features identified between the query and the possible answers.

LG pre-parsed Sentence Scoring Module

This scoring method uses the Link Grammar Natural Language Parser to determine the semantic distance between terms of interest. Sentences are parsed by the Link Grammar Parser and the resulting syntactic tree is reassembled into a simple undirected graph. The terms represent the vertices and the semantic links the edges between the terms. The original syntactic tree is then parsed for semantically important terms such as nouns, verbs, and adjectives. These terms are then paired in every combination and returned in an array of pairs. The distance by number of vertices between each pair is found using the previously described undirected graph. Table 3-2 is an example adjacency matrix which is a way to represent a graph in table form. The numbers in the table indicate the semantic distance between the words in the sentence based on the results of the Link Grammar Parser. Only half the table needs to be filled because it is an undirected graph.

	macrophage	tropomyosin	binds	to	Actin	filaments
macrophage	0					
tropomyosin	1	0				
binds	2	1	0			
to	3	2	1	0		
actin	5	4	3	2	0	
filaments	4	3	2	1	1	0

Table 3-2 This adjacency matrix contains pairs of keywords found in the semantic tree produced by the Link Grammar parser and contains the semantic distance between the terms.

Due to the complexity and length of sentences found in biomedical literature, natural language parsing can be a very CPU-intensive process. It was determined that it was necessary to preprocess the calculation of the semantic distance and to store it in a database to be retrieved at the time of sentence ranking.

Preprocessing of the sentence knowledge base required multi-core servers and parallelization of the task to accomplish the processing in a reasonable amount of time. A Perl script was written which started the desired number of child processes and partitioned the approximately 4.5 million sentences into chunks depending on the number of available processors. The script then forked off to that number of processes and the new script in each runs its own instance of the Link Grammar parser to process a different chunk of data and calculate the distances in parallel.

The server used was an HP ProLiant DL980 with 8 CPUs each having 10 cores and able to run 2 threads per core, resulting in the ability to run 160

processes at a time. Eighty processes were run to process the sentence set. The server also had 4 terabytes of main memory which meant that the data could be kept in memory without having to write to the hard drive. Using the HP ProLiant DL980 Server reduced these times from about 150 hours on a Dell PowerEdge, only able to run 8 processes at a time, to 20 hours of processing time. Figure 3-8 is a graph that shows the difference in processing time between two servers, the Dell PowerEdge and the HP ProLiant DL980.

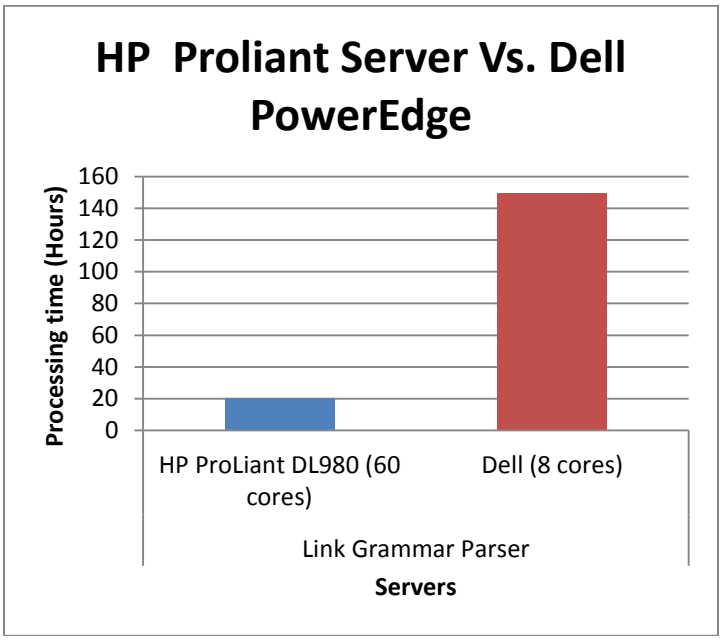


Figure 3-8 Graph showing the preprocessing times of the Dell PowerEdge Vs. the HP ProLiant server. Using the HP server resulted in reducing the processing time from many days to less than one day.

The use of high-performance computing enabled the preprocessing of the sentences in days instead of weeks. This will be invaluable should the system be scaled up to include more sentences.

Mutual Information Measure of Gene Co-Expression Scoring Module

One area of biomedical text mining that has the potential to change the way we mine data from textual sources is the integration of high-throughput biological data (61). In this module information from high-throughput gene expression experiments is used as a resource. Mutual Information Measure (MIM) is one way to show the relatedness between two entities and in this case it is used to show the relation between the co-expression between two genes. The MIM Database of Gene Co-Expressions was created by Wren et al. (62, 63). They performed a meta-analysis on all publically available two-channel human microarray datasets found at GEO(64) (Gene Expression Omnibus). A local installation of the MIM for gene co-expression was created where each row is two gene ids, the 3rd being the MIM between them. This scoring module checks the question and possible answers returned by a deep question analysis and query for genes. If the question and the answer both contain genes the MIM Database of Gene Co-Expressions is queried for different combinations of genes in the question to genes in the answer. If a gene pair is found in the database the sentence is given a boost in its rank by the MIM value. The idea is that if a gene or genes in a question have a high co-expression value with a gene or genes in the answer, this might increase its relevancy compared to answers that do not have pairs of genes with a value or have a pair with a low co-expression value. Below in Figure 3-9, the process taken by the MIMAgent to re-rank answers is explained in a flowchart.

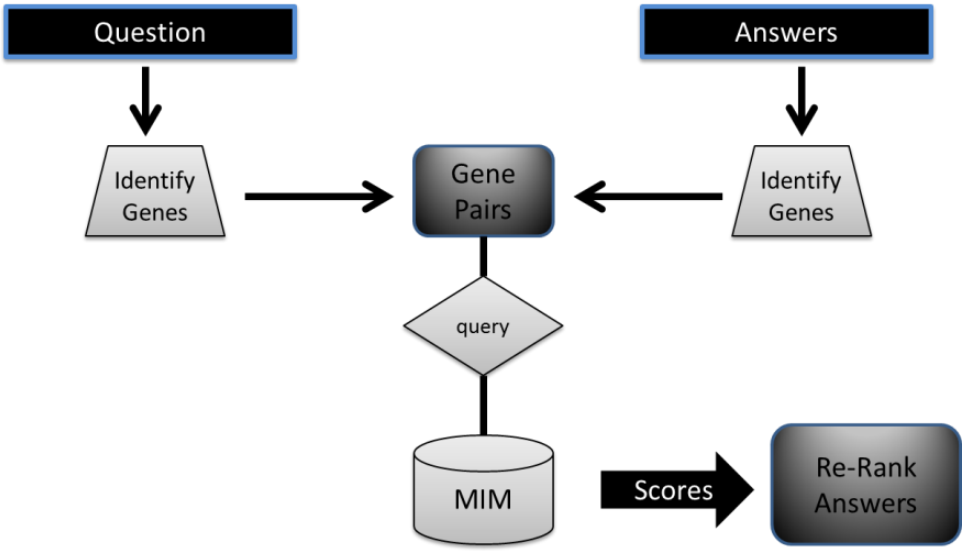


Figure 3-9 A flowchart showing the process used by the MIMAgent. Genes are identified in both the question and the answers. All possible combinations are found between the two sets and the local MIM database is queried. Answers that contain genes that returned a MIM score get a boost in score and upon all answers being processes they are re-ranked according to the new scores.

Dispatch module

The dispatch module takes a list of modules that represent the agents, the user's question and the number of results to return. Inside the module each of the agent modules is dynamically loaded and each is assigned the question and the number of results to return. In order to reduce the processing time each module is forked and run as an individual process. The dispatch module waits for each agent to return its results, compiles the results, and returns them all at once. The processing time is dependent on the longest running agent, which tends to be the agent that uses the LG scorer module.

Agents

Jikitou enlists the search strategies of four different agents. In the following section the tools used to accomplish these different strategies is explained.

Basic Agent (BasicAgent)

The basic agent does no query processing. The question is submitted to the Lucy search module as a string with no query reformulation and its default internal cosine based ranking is used to order the answers. During the evaluation of the system, this agent is used as the control to compare the different search strategies used by the other three agents.

Deep Question Analysis Agent (DeepQAgent)

The deep question analysis agent takes the users' question and performs automatic query refinement. It uses the gene synonym module as well as the NLP link grammar module. It concatenates the nouns, verbs, gene synonyms and phrases and submits them to the Lucy::Search::QueryParser module which returns a query object which is then sent to the Lucy::Searcher. Figure 3-10 is a flowchart which describes this process and in Figure 3-11 we see an actual example of a question being parsed and its resulting parse tree. The tree is then used to identify nouns, verbs, and phrases. The process of identifying genes is also demonstrated and in the example we see that from 2 genes in the original question we get 9 possible synonyms.

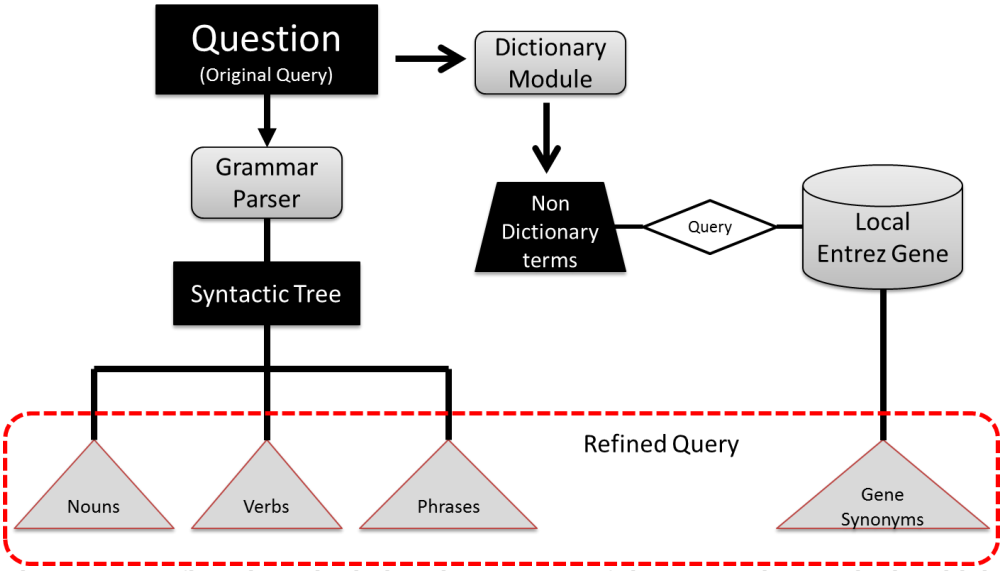


Figure 3-10 A flowchart depicting the process of deep question analysis which is performed by all the agents except the BasicAgent. A question is sent to the module that runs the Link Grammar Parser. The syntactic tree is returned and functions identify the nouns, verbs and phrases. The question is also sent to the dictionary module that uses ASpell to identify terms that are not in the dictionary and they are used to query the local install of the Entrez Gene database for possible gene synonyms.

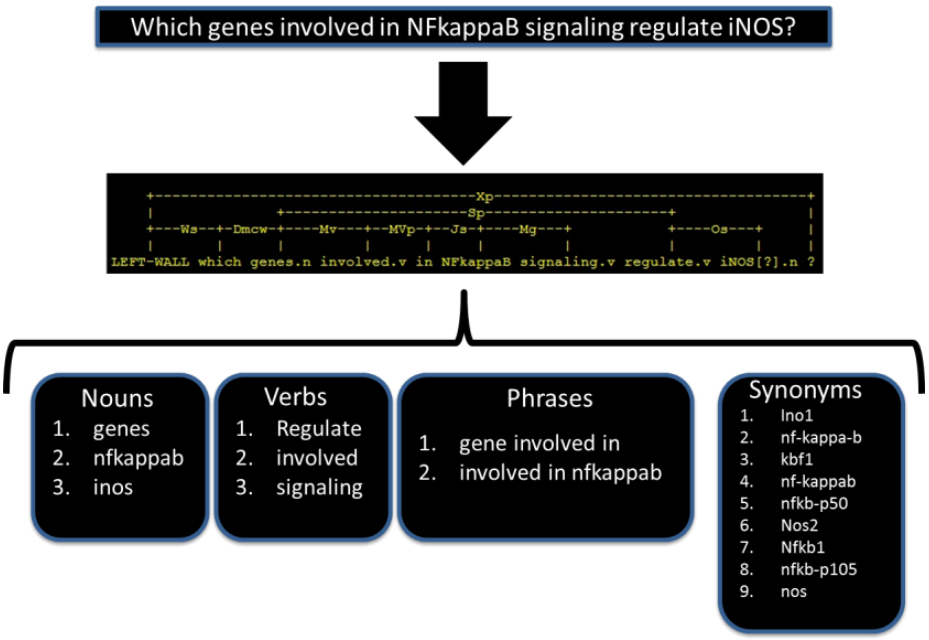


Figure 3-11 An example of a question with its resulting syntactic tree produced by the Link Grammar Parser. The nouns, verbs, phrases, and gene synonyms are the results of further processing shown in Figure 3-10.

Mutual information measure gene co-expression (MIMAgent)

This agent uses the same query refinements as the deep question analysis agent. After the results are returned to the agent it then uses the MIM module value to re-rank the results based on genes that are located in the question and answer and have an MIM value.

Link grammar analysis agent (LGAgent)

This agent also uses deep question analysis but also uses the preprocessed link grammar database to re-score the results. Terms found in the question are used to find their corresponding distances in the returned answers. The LG Agent re-ranks the answers based on the semantic distances of the key terms. A naïve method to assess strength of evidence of interaction in a sentence is determining the lexical separation of key terms in the sentence. Semantic distance is expected to give a more accurate measure of term relatedness because we are looking at the meaning of the words as opposed to just saying terms are related by observing that they co-occur.

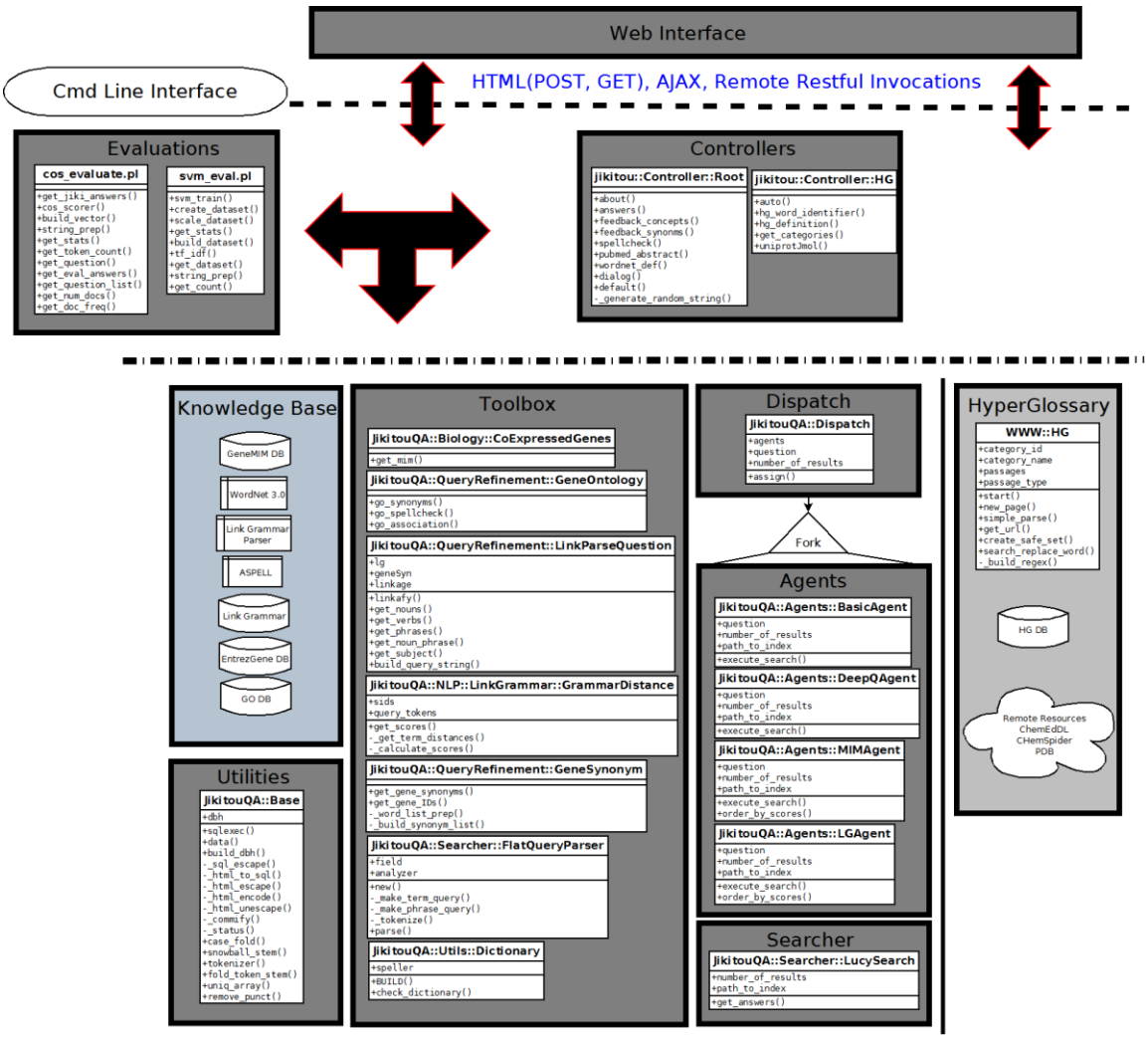


Figure 3-12 A UML diagram showing the major modules, controllers, scripts, and function that make up the Jikitou QA application. User access to the system can be through the web interface or command line interface. The knowledgebase contains all of the database resources that are used. The Jikitou utilities contain functions that are common to all modules. The evaluation methods access Jikitou in two ways. The cosine similarity measure includes the Jikitou modules and calls the necessary functions directly. The SVM evaluation method goes through the controller using a web agent to make a request to the system.

3.6 Interface

An important aspect of any information retrieval is the interface through which a user interacts with the system. The goals of the Jikitou interface are to be simple, informative and dynamic. Being dynamic, all content is brought to the

screen without ever moving to another screen. Asynchronous calls are made to the server and content is returned as a JSON object. It is a tab based interface with four permanent tabs: “about”, “question”, “answer” and “help”. Figure 3-13 shows the different components that make up Jikitou’s web interface. The interface is made of a main viewport with multiple panels, the main panel being the tab panel. In Figure 3-13 we can see the hierarchical structure of the different components along with their key subroutines.

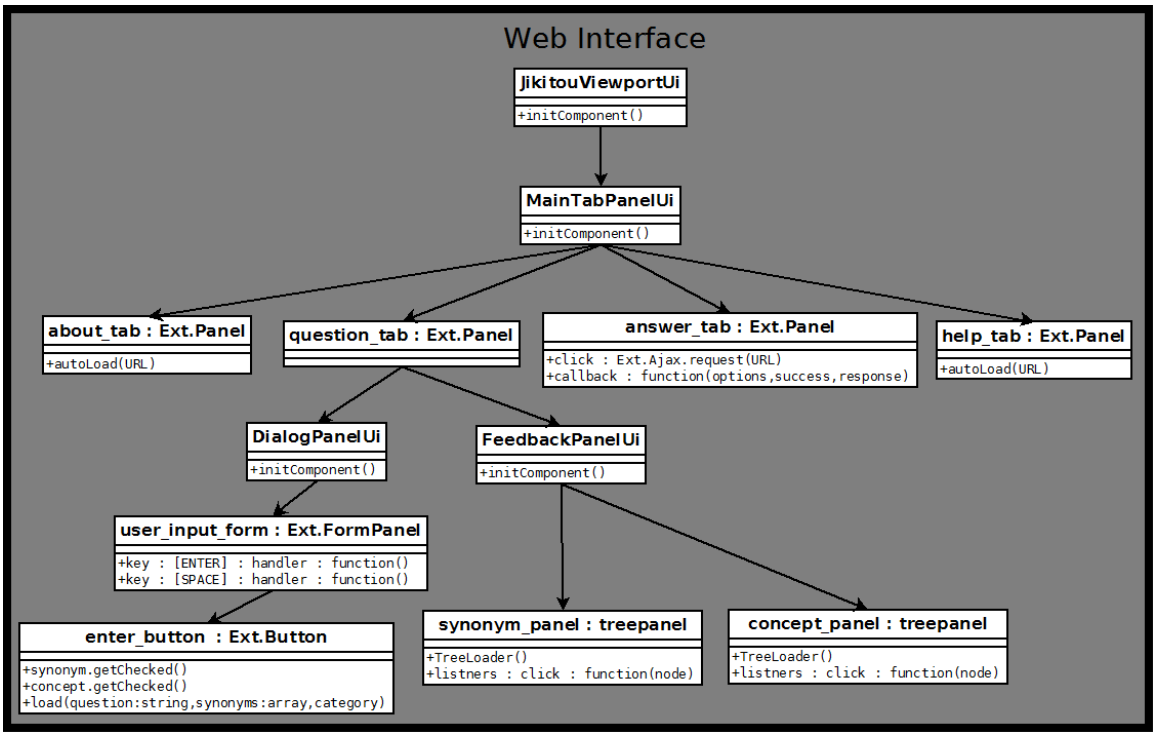


Figure 3-13 A diagram of the main components and sub-components of the Jikitou interface. Important subroutines are written in the retangles of their respective components.

“About” tab

The “about” tab displays a description of the project and the Jikitou QA system.

"Question" tab

The "question" tab is where the user asks a question and interacts with the system to refine the question. The question tab contains three windows.

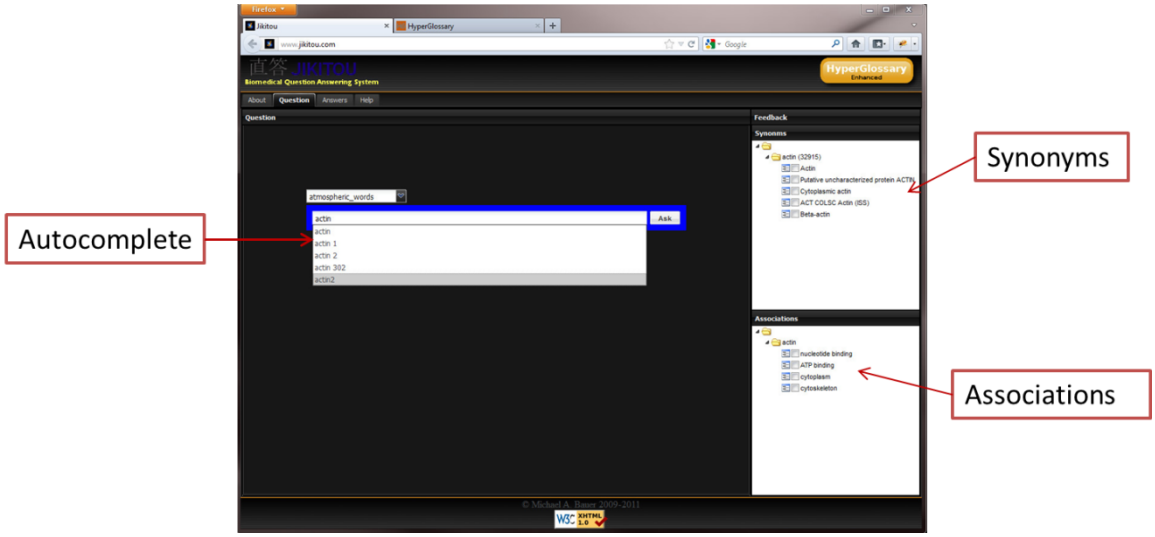


Figure 3-14 As a user types a question the system suggests additional terms that can be added to refine the initial query.

Main window

The main window (Figure 3-14) is the form where the user inputs a question and can choose a glossary that they would like the answers to be parsed against and linked to additional resources. As the user types a drop down menu appears with suggestions for the current word they are typing. This action is initiated after the first 4 characters have been typed to reduce the amount of calls to the server. The suggestions are provided by controllers with two functions in Jikitou. The first one takes the currently typed characters and uses Aspell, the Open Source spell checker, to check the spelling and to provide spelling suggestions for words and these are provided to the user as possible completions of what they are trying to type. The other function takes the currently

typed characters and queries a local installation of the Gene Ontology for gene symbols that start with the characters that the user is typing. These two functions combine their results and the user is presented with up to 9 (4 from Aspell, 5 from Gene Ontology) different suggestions for completing the current word they are typing.

Feedback windows

A dialog with the user is used for interactive query expansion. If a query term is vague the user is presented with a list of possible, more-specific alternatives. If there might be too many results, the idea is to identify broad terms in a query and try to get the user to indicate more specific terms with which to reformulate the query. When a query is too specific and very few or no results are found, a broader term may be substituted into the query. One of the problems with interactive query expansion has been that users are unable to identify good query expansion terms (58). Using the feedback windows the user has the ability to quickly gain an understanding of the term and possibly discover other terms they might like to include in their query.

There are two windows for user feedback, one for word synonyms and the other for Gene Ontology associations. As the user types these windows are also dynamically updated, one with a list of suggestions for synonyms and the other with GO concepts. This suggested content might not have been thought of by the user to include in their query. The user can then check the boxes of the terms that they would like added to their query. The query for both these windows is

activated by the pressing of the spacebar, which indicates that the user has completed typing a word and now that word can be used in subsequent queries.

The controller that provides content to the synonym feedback panel has 3 main functions. The first gets the term frequencies in the sentence database of the terms in the query. This provides the user with information about the quality of their query. A high term frequency may mean the term they have selected has very low ability to provide relevant sentences since it is present in too many sentences. On the other hand, a low frequency term may give a hint as to why few or no results are being presented. The second takes the term and queries the local installation of the WordNet lexical database for possible synonyms. Terms that are not found in the word dictionary are then used in a query to identify possible gene synonyms. Checking the word dictionary for terms that are not English words reduces the number of spurious identification of genes that are not really genes. There is a tradeoff between too many irrelevant synonyms being presented to the user and some legitimate gene names that look like words being missed. The association window is populated by the controller, which submits a query to the Gene Ontology database for possible gene associations relating biological processes, cellular components, and molecular functions. Figure 3-15 shows the feedback presented to the user along with the data source.

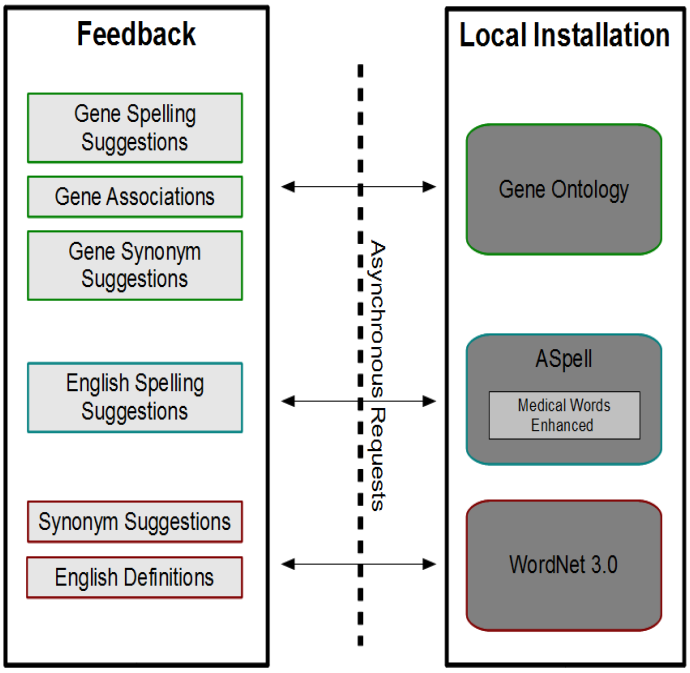


Figure 3-15 . Shows the dynamic feedback process and sources of the data presented to users' as they type their questions. The WordNet and Gene Ontology databases are queried to find synonyms to terms entered. A medical term dictionary has been added to provide biomedical domain specific terminology suggestions with autocomplete. The Gene Ontology database is used to find associations with biological terms relating to biological processes, cellular components, or molecular functions.

"Answer" tab

The "answer" tab provides a table of all the sentences that were found to be possibly relevant to the user's question. Each row contains the rank, the sentence, the score and the PMID of the abstract of its origination. If a glossary had been selected when submitting the question, terms may be marked and appear green. They can be selected and bring up additional content as shown in Figure 3-16. More on that functionality, and types of content, is described in the next section.

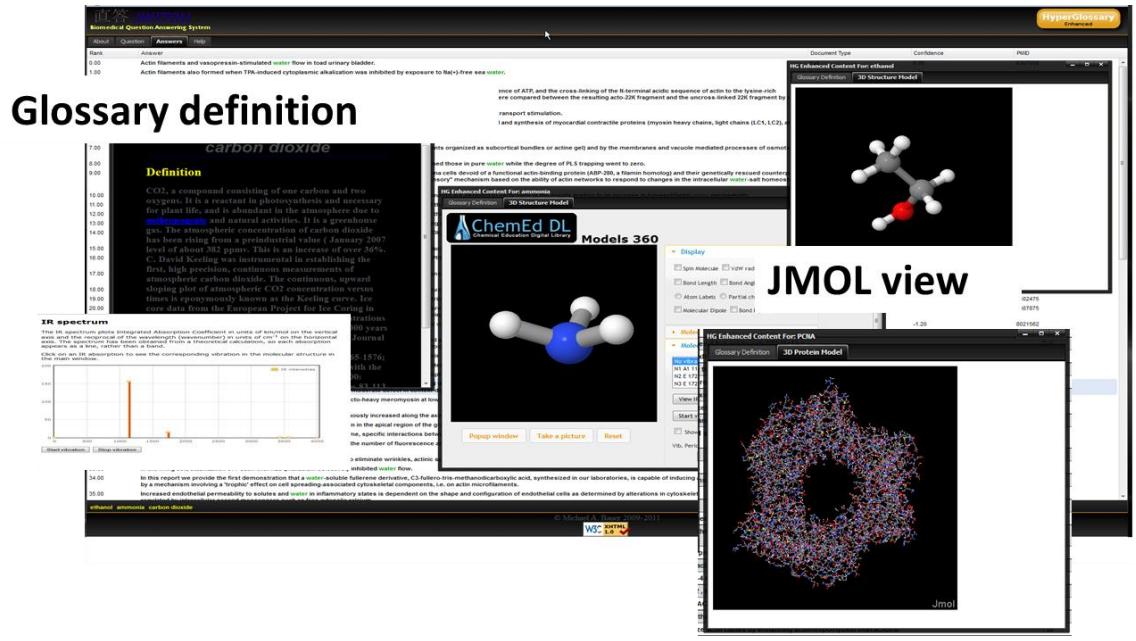


Figure 3-16 Connection to the ChemEd Digital Library returns JmolS. This allows you to interact with molecules in multiple ways beyond simple measurements, like connecting molecular vibrations to IR Spectra. The data is presented in JavaScript overlays, which can be opened as well as be minimized. Depicted are a term definition, a plain chemical Jmol, a ChemEdDL enhanced Jmol with IR spectra, and a protein Jmol.

"Abstract" tab

Additional tabs are created when a user clicks on an answer. These tabs contain the abstract from which the sentence was extracted. The controller that populates this tab uses PubMed's Efetch (65) to download the abstract using the PMID. The sentence that has been selected as the answer is highlighted in the abstract.

3.7 HyperGlossary

The HyperGlossary is an information literacy tool that we developed for a chemistry education project and which I integrated with Jikitou. It automates the insertion of hyperlinks into the text answers and connects them to textual

definitions, multimedia content, and in the case of many molecules, 2D and 3D representations. The HyperGlossary takes advantage of authoritative knowledge sources on the Internet such as ChemSpider (66) and RCSB Protein Data Bank (67).

After the search results are returned by the agents the answers are automatically marked up and linked to semantically relevant content in other databases using the HyperGlossary. An overview of the core functionality of the HyperGlossary is shown in Figure 3-17. When a user reads a word or phrase in an answer that is connected to a glossary term, the information associated with the term can be viewed without leaving the original document. The additional information is presented in a popup window that appears when a marked term is clicked. The types of extra content include 2-dimensional and 3-dimensional structures, definitions of terms, and WWW content.

The answer is not only linked back to the original document, but keywords and phrases are linked to additional sources of information. An example is that proteins mentioned in the answer are linked to a proteomics database, which, when clicked, reveals the structure and the sequence of the protein in a pop-up window. This information allows the answers to be more accessible to users of varying backgrounds (68). The system brings together traditional text information and dedicated biological databases to present a concise answer to the user.

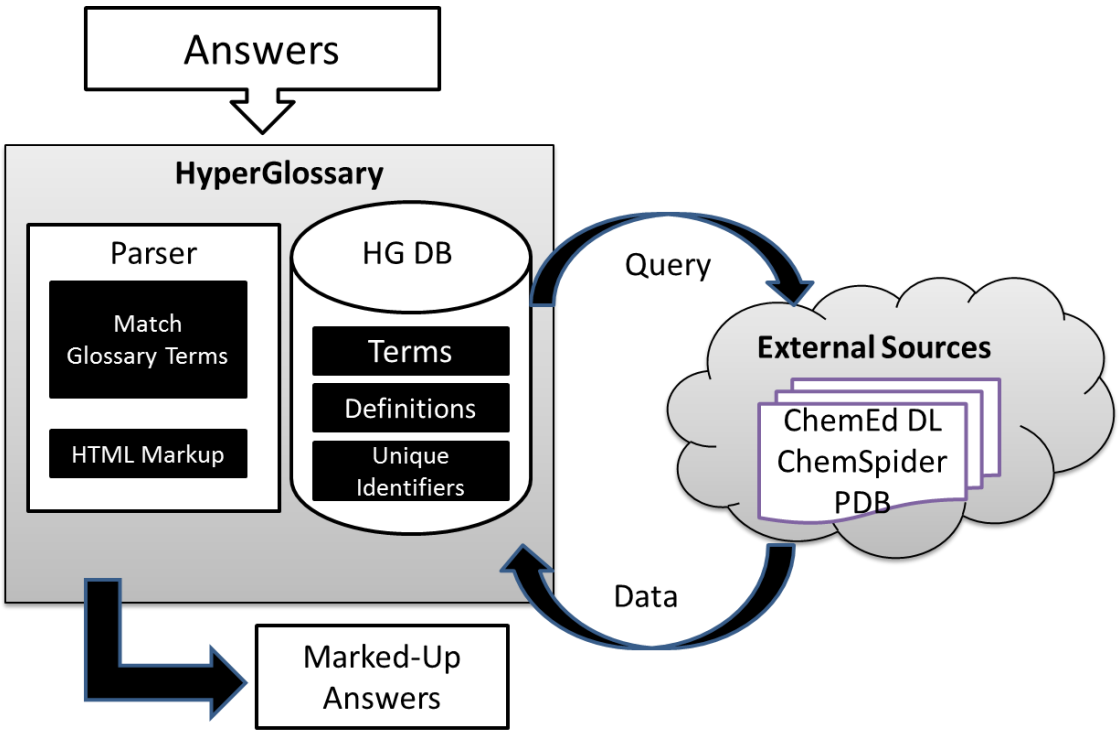


Figure 3-17 Answers are automatically marked up based on the selected glossary and linked to semantically relevant content in other databases using the HyperGlossary.

Creating a glossary

In order to have a glossary geared towards the biomedical field I created a protein glossary. The id mapping table for *Homo sapiens* was downloaded from the UniProt website. A script was written that grabbed all the Protein Data Bank (PDB) (67) and UniProtKB (69, 70) protein identifiers from the downloaded id mapping table. The list of UniProt identifiers was then submitted to UniProt for a batch web retrieval of available protein information for each supplied identifier. Another script was written that parses the protein information file and retrieves the general annotations about the proteins. It is these annotations that are used as the definitions in the protein glossary. The UniProt Id is used as the unique identifier in the HyperGlossary. A mapping database of the UniProt and PDB

identifiers was created and a function added to the HyperGlossary that retrieves the PDB id for a protein that is selected and it is the PDB id that is used to retrieve the PDB file from the RCSB website (71). This file is submitted to the Jmol applet to render the 3D structure of the selected protein.

JavaScript overlays

When a marked term is clicked a JavaScript overlay pops up and acts as the portal to additional information about the selected term. A number of tabs are presented to the user for the different types of content available. The number of tabs and types of content that are provided depends on the chosen glossary and the type of term. Currently the HyperGlossary contains three types of terms: “no type”, “chemical”, and “protein”. Terms of the type protein or chemical have associated with them an identifier that not only uniquely identifies them but in the case of InChIs (72) also provides structural information. The identifiers can then be used to link the term to many more remote information resources. Also they themselves can be used in a web applet to view and manipulate the resulting molecules.

Definitions

Most terms in the HyperGlossary have a definition and this is the first tab that is presented when the overlay comes up. They have a citation stating the source of the definition and there is the possibility that they have additional fields of user defined definitions. It is also possible that while the original definition is

the canonical definition the additional fields can contain a rewrite of the definition for varying levels of knowledge.

ChemSpider search

A ChemSpider (66) tab has been added to the JavaScript overlay. ChemSpider is a large database of chemical structures that is free to access over the internet. ChemSpider provides several services for accessing information in the collection. When a term that is identified as being a chemical is selected the term is used to perform a simple search of ChemSpider, which can return a list of ChemSpider identifiers. Another search is performed using the GetCompoundThumbnail service. The ChemSpider identifiers are used to query for the thumbnails of the compounds. Each thumbnail is returned as a 64 bit string which must next be decoded. The Perl module MIME::Base64::Perl is used to decode the string into a PNG file that is saved to the server. The image is then displayed and used as a link to the ChemSpider webpage where additional information on the compound can be found.

3D proteins

Terms that are in glossaries and are identified as being a chemical or a protein have a unique identifier assigned to them. If a chemical term is selected and the 3D tab is active, its InChI is queried from the database. Jmol (73) is an open-source java applet for viewing 3D structures of chemicals. This is converted to an InChI Key which is used to query ChemEdDL (74) for enhanced Jmols. In the event that the Jmol is not available at ChemEdDL the system can generate it on the fly locally. The file is created dynamically by first converting the InChI to a

SMILES string using Open Babel (75). The SMILES string is then sent to Balloon (76) which creates the mol2 file with the 3D coordinates. The mol2 file is saved so that it only needs to be created once. The location of the file is then sent to the Jmol applet. This process is represented diagrammatically in Figure 3-18.

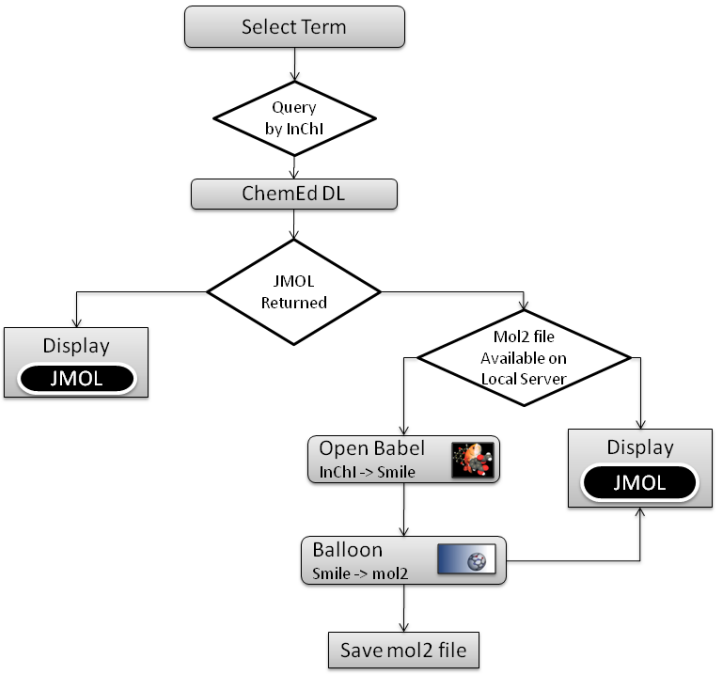


Figure 3-18 A flowchart that depicts the process of presenting a Jmol when a term with a qualifying ID is selected. In the event that a resource does not have the information requested for a chemical term, the hyperglossary has the ability to generate its own structure file from the ID. Open Babel and Balloon, two open source chemistry resources, are used to create a Jmol file on the fly.

2D structure

If the term selected is a chemical and the 2-D structure tab is selected the InChI string is converted to a SMILESS string and submitted to the Java Applet JChemPaint (77). The 2-D structure of the molecule is drawn. JChemPaint allows the molecule to be edited. The clickable link at the bottom of the window submits the SMILES string for the resulting chemical to ChemSpider. JavaScript was written to use JChemPaint's API (getSMILESs()) to grab the SMILES string of the

current chemical structure in the applet. The string is sent back to the server where the sting is converted to an InChI string using ChemSpider's Open Babel Web Service. The InChI string is then used to query ChemSpider's database and have it return a PNG thumbnail of the compound if it exists in the database. The thumbnail is linked back to ChemSpider with additional information on the newly created structure. The information is presented in a new tab that can be closed. See Figure 3-19.

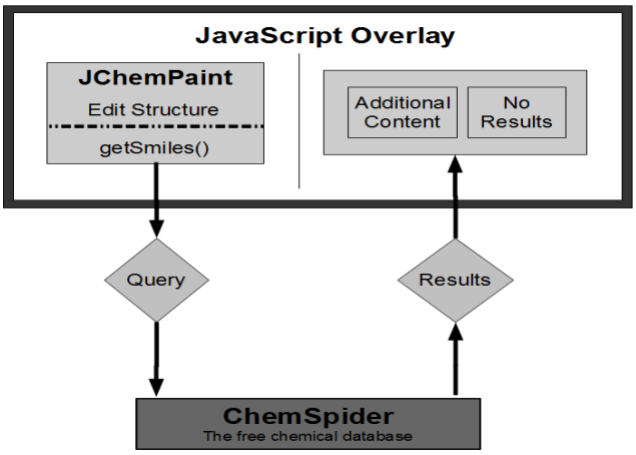


Figure 3-19 An overview of the process of querying ChemSpider using the JChemPaint applet. In the 2D structure tab in the JavaScript overlay for a chemical term, the JChemPaint applet allows the user to edit/create a chemical. A link at the bottom of the overlay grabs the resulting SMILESs string using JChemPaint API and uses it to query ChemSpider. If the resulting string is an actual chemical in the database, the results are shown. Otherwise a message saying "no results" is shown.

Chapter 4: System evaluation

An important component to any information retrieval system construction is evaluation of the system. In this chapter the strategies used for evaluation of Jikitou is described and the rationale behind their implementation.

4.1 Evaluation methods: User study vs Automated

There are two common methods used to evaluate an information retrieval system evaluation. The first method is to perform a user study. The users can be recruited to evaluate the results of a system. In other words judge the relevancy of information returned. They can also be users of the system and given a task to perform and based on their experience with the system they can fill out a rubric on different aspects of the system such as responsiveness of the system or ease of use. Preforming a user study does have its drawbacks. Each time changes are made to the system the time consuming task of setting up a user study to evaluate the results must be redone. The second method to evaluate an information retrieval system is to compare results to previously judged answers. Using this method we have the ability to quickly evaluate the system as different components are tweaked. It lends its self to automation and can easily be inserted into a system development pipeline.

Jikitou is designed to be complete information retrieval system, which in addition to a question answering system, includes tools to evaluate the system to aid the development of additional agents. To that end an evaluation method that

can be automated fits in with the desired capabilities of the system. The rest of this chapter describe two automated methods and the resources used.

4.2 Text Retrieval Conference Resources

The Text Retrieval Conference (TREC) Genomics Track was a conference with the goal of evaluating text retrieval systems aimed at the biomedical field. The last two years of the track focused on question-answering in the biomedical domain (78, 79). Evaluation of the systems was performed by providing participants a list of topics/questions that they would use as queries to submit to their systems. The participants then returned their results which were placed in a pool of possible answers to the queries. Experts would then judge them for relevance while isolating the minimum information that answers the question. This pool of judged documents was then used to evaluate the performance of systems that contributed to the pool.

The topics and relevance judgments are still available at the TREC Genomics track website (80) and it is this resource that is utilized to evaluate Jikitou. The test collection contains 36 questions (see Table 9-1 in appendix A), the documents used as the corpus for the TREC participants, and a file with the document numbers, character offsets, and spans of the submitted passages that have been judged relevant or not. I created a database with one table containing the questions and the other table containing the passages, each with a question id to identify which question it refers to and its relevancy or not to that question. To get the passages a script was written that read the judged passage file,

identified the document file, the offset and span, retrieved the text from the file, and inserted the passage into the database.

Due to the fact that Jikitou was not one of the original systems that contributed to the pool of possible answers and it is not using the same corpus it is not possible evaluate Jikitou in the same manner or really compare it to the other TREC participants. Answers returned from Jikitou are not likely to be present in the pool of judged passages and therefore we would not know if the answer was relevant or not. The relevance judgments are biased towards systems that contributed to the pool.

It is expensive to create a large pool of judged documents, so to take advantage of this valuable resource we needed to figure out a new way to evaluate Jikitou. Here I present two types of evaluations that were devised to test the performance of Jikitou. For both of these evaluations each of 36 questions was submitted to Jikitou and the top 50 results returned by each agent were analyzed.

4.3 Similarity-judged vs. Jikitou-produced answers

When working with and analyzing text we often create vectors as described in the parsing and indexing section, where each vector represents a string of text which can include a document, sentence, or phrase. Representing text in vector form gives us the ability to perform vector bases analyses. The cosine similarity measure is an analysis that uses the angle between two vectors

and is one way to measure the similarity between them. A useful feature of this measure is that the lengths of the documents are normalized.

In this method I take all the relevant passages for a topic and compare them one by one to answers returned from each agent. Each answer is compared to every other judged relevant passage for the topic and the average score is calculated and recorded. This is repeated against all the non-relevant passages as well for comparison.

The theory is that answers that have a high cosine similarity score will be, on average, more closely related to the passages that were judged relevant than the passages that are judged not-relevant. This method is a quick way to compare the performances of the different agents within the Jikitou system. Every time an agent or tool is modified this evaluation can be run to judge how performance has been affected.

4.4 SVM: extending the judged passages

To take advantage of the pool of judged passages it was necessary to decide on a way to extend its judgments to Jikitou's answers. An issue that often arises is the biased effect of the pooled judgments against a new system and there has been considerable research with the goal of minimizing that bias in pooled judgments (81-83). Two measures proposed are the bref (81) and RankEff (82) measures, both of which take only judged documents into account when assessing a system's performance. The problem with this method when it comes to Jikitou is that there is little chance that any of the documents retrieved

would have been judged by TREC since the document sources are different. The works of Aslam et al. (84) and Buttcher (85) discuss methods that extend the pool to include the non-judged documents by predicting relevancy. Aslam et al. accomplished this by inferring document relevancy by analyzing document rankings by several different information retrieval systems. The Buttcher approach was to use a classifier to predict the relevancy of non-judged documents. They experimented with Kullback-Leibler divergence and support vector machines (SVMs), two different classifiers. In this project I decided to use SVMs to predict relevancy, described below.

Using SVMs is a supervised classification method, meaning that an SVM needs to be trained on a dataset that has been labeled as to its relevancy. The resulting model is then used to tag new documents as either relevant or not relevant. SVM works by trying to draw a hyperplane between a dataset that separates the two classes in a high-dimensional space. Usually, the larger the distance between the nearest training set data point and the hyperplane, the better the classifier.

The evaluation of Jikitou uses the LIBSVM (86) SVM software library, written in C/C++, to perform the training and classification of the relevancy of sentences returned by each agent. The first step was to create a set of features that would be used by the SVM. This is called the feature space. A database table was created that contained the terms from the corpus after stemming and their frequency in the corpus, post-stemming. The terms used in the feature space were selected from this table with the following criteria: A term had to be

longer than three characters and have a term frequency of at least two. This resulted in the selection of 120 thousand terms. It was this list of terms which constituted the feature space that was then used to train the model. The training dataset was built by selecting 2/3 of the TREC passages for a particular topic that were judged relevant and not relevant. The returned passages were then converted to vectors of features where the value for each term/feature is the IDF-TF of the term. These vectors were then converted to the correct format using the Perl module Algorithm::SVM::DataSet, which assigns a vector a class label. The dataset was then submitted to the training algorithm of the SVM. The resulting model is written to a file to be used later. This process is repeated for each of the 36 topics. The parameters in Table 4-1 were ultimately used in the SVM after a process of trial and error found these produced the models with relatively high accuracy.

SVM Parameters	
SVM Type	C-SVC
Kernel Type	Linear
Gamma	0.01674
C	19.835
Degree	3

Table 4-1 The parameters that were used to train the SVM models.

A Perl script was written that performs all of the tasks of training a model and classification. The task that the script performs is controlled by the supplied command line arguments when the script is executed. Table 4-2 lists the possible

command line arguments. The script can be set to train a model by setting the action to “train” and then selecting the SVM parameters to set or to leave blank (to use default values) and then set the question id parameter for the question to train the model against the TREC-judged relevant and not-relevant passages. The script can then be used to predict the relevancy of answers returned by Jikitou. This is done by setting the action parameter to “load” and the model parameter to the path to the model you would like to use, setting the question id parameter to the question you would like to use, and setting the “jikitou” option to submit the question to Jikitou.

Option	Values	Description
--action	train, load	Train a model or load a model
--model	File path	Takes a file path to the model
--option	validate (v), retrain (r), predict (p), Jikitou (j)	Set to “v” to validate a model Set to “r” to retrain the model Set to “p” to predict on the remaining 1/3 test set Set to “j” to predict on results returned from Jikitou
-t	'C-SVC', 'nu-SVC', 'one-class', 'epsilon-SVR' and 'nu-SVR'	Sets the SVM type
-k	'linear', 'polynomial', 'radial' and 'sigmoid'	Sets the type of kernel to use in the SVM
-g	Float	Sets the gamma function in the kernel function
-c	Float	Sets the cost a penalty parameter
-d	Integer	Sets the degrees in the kernel function
-i	200-235	Sets the TREC question to either train the model to or submit to Jikitou for answers to classify

Table 4-2 svm_eval.pl command line parameters. The user can train a new model with a different set of SVM parameters or load a model to classify answers using a particular model.

Chapter 5: Results

In this chapter the results of the SVMs and cosine similarity evaluations are described. The first section compares the evaluation results of both methods. The rest of the chapter uses the evaluation results to compare the performance of the agents.

5.1 Evaluation comparison

The first objective was to see how the two evaluation methods compare to one another. Figure 5-1 is a graph of the results of both evaluation methods. The bar graph indicates the number of Jikitou answers that were predicted relevant for each question by the SVM classifiers. The two line graphs are the cosine similarity measures comparing the Jikitou-produced answers with the TREC judged relevant and not-relevant passages.

Looking at both evaluation methods on a query by query basis we see similar trends. On topics that have a low average cosine score between the Jikitou answers and the TREC judged relevant passages there appears to be a corresponding low number of predicted relevant answers. There are several queries where they diverge meaning the SVM did not classify any or very few answers as relevant but the average cosine similarity measure for answers as compared to the TREC judged relevant passages is relatively high. This can be observed with the following question IDs: 214, 216, 217, 221, 229, 233, and 235.

Another observation is, for question IDs for which there are no elements judged relevant we see that the average cosine score between Jikitou answers and the TREC judged relevant passages tends to be higher than the score between the Jikitou answers and the judged not-relevant passages. These IDs include: 201, 202, 205, 206,207, 208, 209, 210, 219, 220, 222, 223, 224, 225, and 233. This agreement suggests that these two evaluation methods are actually able to distinguish between relevant and not-relevant answers and may provide alternatives to always having a human judge the relevancy of answers in the developments and tuning of question answering systems and perhaps in the evaluation of other types of information retrieval tools.

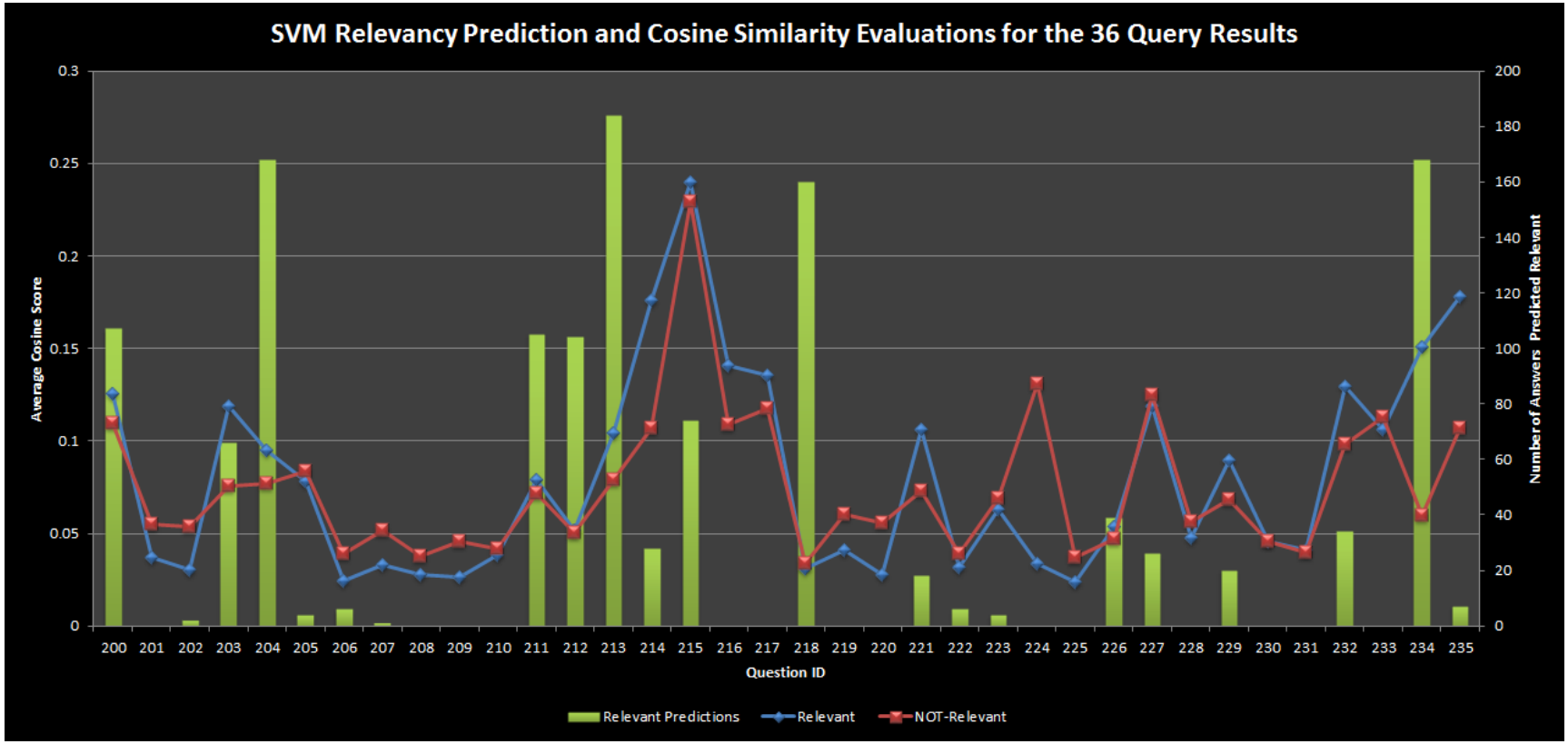


Figure 5-1 A graph of the results of both evaluation methods. The bar graph indicates the number of answers classified as relevant for all agents' answers by SVM and the 2 line graphs are the cosine similarity scores comparing relevant and not-relevant TREC passages with the Jikitou answers.

The SVM evaluation classified many of the answers as not relevant. Out of the 36 questions, 14 questions submitted to Jikitou resulted in the SVM classifier classifying all answers returned by all agents as not-relevant (questions 200, 208, 209, 210, 216, 217, 219, 220, 224, 225, 228, 230, 231, and 233). The results of 10 of the questions (202, 205, 206, 207, 221, 222, 223, 226, 229, and 235) had few answers classified as relevant or only one agent's answers that were classified as relevant. This may be due to inability of the agents to identify answers, inability of the SVM classifier to correctly classify, or insufficient answers in the database. The remaining 12 questions (200, 203, 204, 211, 212, 213, 214, 215, 218, 227, 232, and 234) are used in a majority of the following analyses.

5.2 Cosine similarity measure evaluation results

This section deals with the results of the cosine similarity measure between the TREC judged passages and Jikitou-produced answers. First, cosine scores of relevant and not-relevant TREC passages with Jikitou-produced answers are compared. The rest of the section compares the cosine score of the individual agents.

Cosine score, relevant vs. not-relevant, by rank

The 36 question were used to retrieve 50 candidate answers each, ranked 1 to 50. A comparison by rank of the average cosine similarity by rank between Jikitou answers and relevant and not-relevant TREC passages shows that the cosine scores differ for relevant versus not-relevant passages. To determine if

the results of the similarity measure for each of the agents followed a normal distribution, a Shapiro-Wilk test of normality was performed in R and the data was shown to be normally distributed. Once the data was verified as being normally distributed a t-test was performed to see if the difference between the two data sets was significant with the null hypothesis being that the means are equal. The results of the t-test is a t-statistic value of -4.951 with a p-value of <.05. This means we reject the null hypothesis that the means are the same and accept the alternative that the means of the datasets are different. Appendix B describes the data analysis.

It is important for an answering system to show a difference between relevant and not-relevant passages to be considered effective, see Figure 5-2. The size of the difference between the two data sets may not be as large as possible due to the fact that even though the not-relevant passages were judged not-relevant they were in fact classified by some TREC participant's IR systems as relevant to the query. Nevertheless, out of 50 ranks, 49 had higher cosine scores for the relevant condition than for the not-relevant condition.

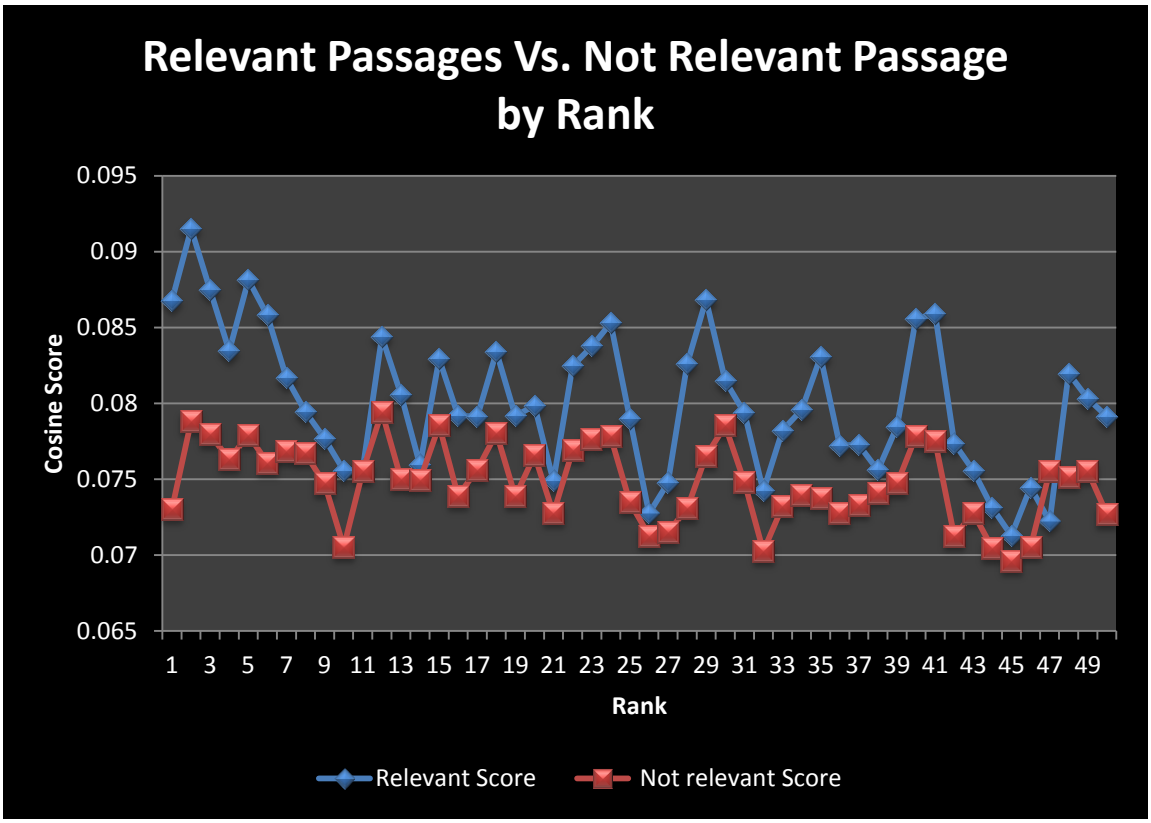


Figure 5-2 A graph showing the average cosine scores comnparing Jikitou answers to relevant passages and to Not-relevant passages by rank.

Agent relevant cosine scores by rank

Next I looked at the agents individually. The average cosine score of each agent by rank gives us an idea of the performance of each agent as the answers increase in rank. The cosine scores contributing to the average for each rank are the ones classified as relevant by SVM evaluation. We can see in **Error! eference source not found.** a graph that shows the average score over the 36 questions at each answer rank, for each agent. A moving average with a window size of 3 was used to remove some of the volatility of the results. Every point is an average of the current rank and the 2 previous ranks. This reduces the dataset by 2 data points. Each of the agents still shows volatility but over a

smaller range. All four agents do show a general trend downward in score similarity with the relevant answers, which we would expect to see due to the fact that the most similar answers should also be ranked highest.

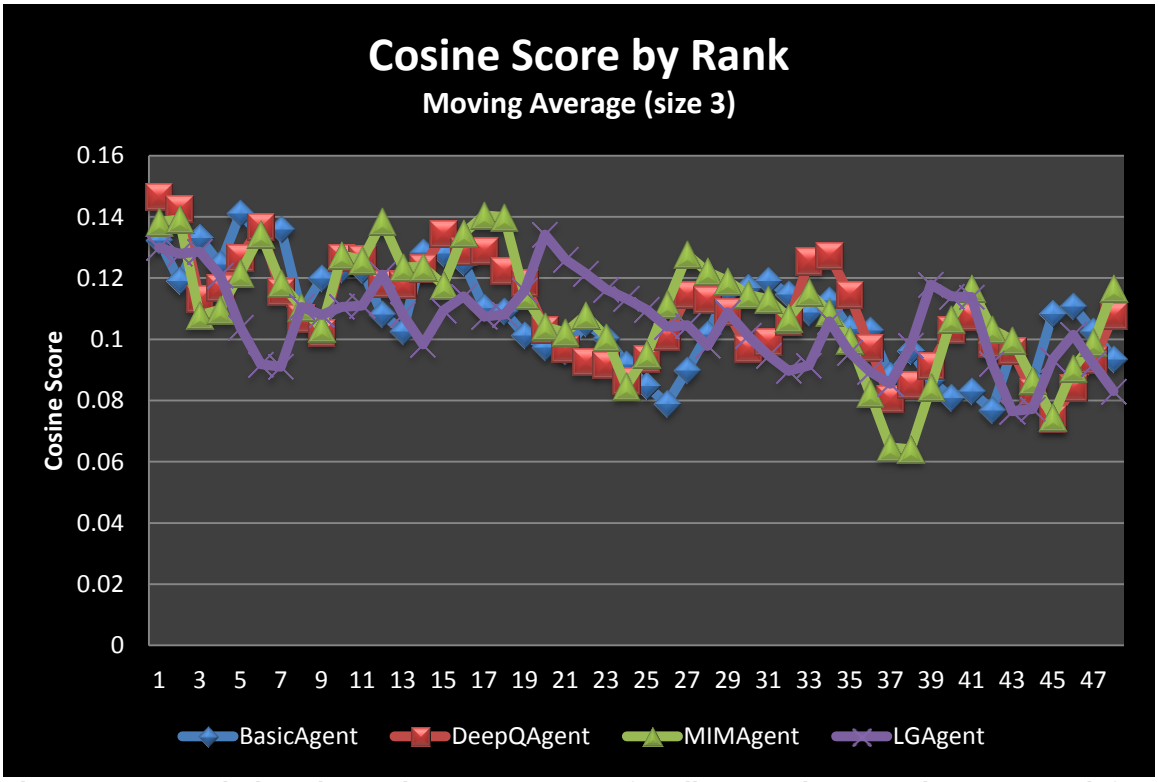


Figure 5-3 A graph that shows the average score for all 36 queries at each answer rank for the four agents. The cosine scores contributing to the average for each rank are the ones classified as relevant by SVM evaluation. A moving average with a window size of 3 was used to remove some of the volatility of the results.

Difference between agent per query

The results of the cosine similarity to relevant passages evaluation were next used to evaluate the performance of the agents on a query by query basis over the 12 queries. This analysis uses the 12 questions for which the SVM evaluation classified the most answers as relevant. An analysis of variance (ANOVA) was performed on each set of query results. When performing an

ANOVA the null hypothesis is that the mean results for each agent are equal while the alternative hypothesis is that one or more of the agents' means are significantly different, with significance being identified here with a p-value of .05 or less. Table 5-1 shows the ANOVA results for each set with its p-value. If the ANOVA resulted in a significant result the null hypothesis was rejected and the alternative accepted. In these cases Tukey's honest significance test was performed in R. It is a multiple comparison of means procedure. The test compares the mean of each group to the mean of every other group. This test allows us to determine which agents' means were different from the others. In the table each significant result is highlighted in yellow. Appendix D contains the complete software output for both the ANOVA and Tukey tests.

Question ID	P-Value (ANOVA)	Significant Pair-wise Comparison of Means (Tukey)	
200	0.421167		
203	0.258027		
204	1.11E-05	DeepQAgent-BasicAgent	0.0012299
		MIMAgent-BasicAgent	0.0012299
		LGAgent-BasicAgent	0.0000129
		MIMAgent-DeepQAgent	1.0000000
		LGAgent-DeepQAgent	0.6826629
		LGAgent-MIMAgent	0.6826629
211	1.25E-26	DeepQAgent-BasicAgent	0.0000000
		MIMAgent-BasicAgent	0.0000000
		LGAgent-BasicAgent	0.0000000
		MIMAgent-DeepQAgent	0.8248632
		LGAgent-DeepQAgent	0.0307114
		LGAgent-MIMAgent	0.2262059
212	0.003969	DeepQAgent-BasicAgent	0.0085656
		MIMAgent-BasicAgent	0.0085656
		LGAgent-BasicAgent	0.2311872
		MIMAgent-DeepQAgent	1.0000000
		LGAgent-DeepQAgent	0.5624999
		LGAgent-MIMAgent	0.5624999
213	0.205935		

214	5.91E-07	DeepQAgent-BasicAgent MIMAgent-BasicAgent LGAgent-BasicAgent MIMAgent-DeepQAgent LGAgent-DeepQAgent LGAgent-MIMAgent	0.0591663 0.0591663 0.0994347 1.0000000 0.0000159 0.0000159
215	0.216707		
218	2.61E-13	DeepQAgent-BasicAgent MIMAgent-BasicAgent LGAgent-BasicAgent MIMAgent-DeepQAgent LGAgent-DeepQAgent LGAgent-MIMAgent	0.0000000 0.0000000 0.0001026 1.0000000 0.0191799 0.0191799
227	0.90762		
232	1.57E-08	DeepQAgent-BasicAgent MIMAgent-BasicAgent LGAgent-BasicAgent MIMAgent-DeepQAgent LGAgent-DeepQAgent LGAgent-MIMAgent	0.0000043 0.0000009 0.0000006 0.9884800 0.9789030 0.9998513
234	9.91E-40	DeepQAgent-BasicAgent MIMAgent-BasicAgent LGAgent-BasicAgent MIMAgent-DeepQAgent LGAgent-DeepQAgent LGAgent-MIMAgent	0.0000000 0.0000000 0.0000000 0.9038357 0.9444567 0.6026047

Table 5-1 An analysis of variance (ANOVA) was performed on each set of query results using the cosine similarity scores to the relevant passages. The table shows the resulting ANOVA p-values. If the resulting p-value was deemed significant with a value < 0.05 a Tukey's honest significance test was performed, which is a multiple comparison of means procedure to determine which agent or agents was significantly different. All p-values that are significant are highlighted.

5.3 SVM evaluation results

In this section the quality of the SVM produced models is calculated using cross-validation on a separate test set. The results are then used to calculate additional measures of accuracy. After that the performances of the Jikitou agents are evaluated based on the results of the classification of their answers as to their relevancy using the SVM models.

SVM performance estimate

When the SVM evaluation Perl script is run to build an SVM model for a particular question, the script is set up to automatically check the accuracy of the learned model. First cross validation is performed on the training set where the data set was partitioned into 5 subsets and each subset validated against the others. The model is written to the file and the accuracy is output. Remember that 2/3 of the dataset was used as the training set leaving 1/3 for a test set. So in addition to the cross validation the remaining 1/3 of the dataset was used as a test set as another way to determine the accuracy of the model. A different model was created for each of the 36 TREC questions and the accuracies of the models as measured by both the cross validation and the test set results are shown in Table 5-2.

The models were used to predict the relevancy of the answers returned by the system. Each agent returned 50 ranked answers to each of the questions which were converted to a vector of features as in the training stage and classified as being either relevant or not-relevant by the SVM model for that question.

Question ID	Accuracy	True Positive	False Negative	True Negative	False Positive
200	71.60	28	2	21	9
201	97.03	24	0	27	3
202	94.27	28	1	28	2
203	70.03	30	0	24	6
204	89.66	23	7	20	10
205	87.09	28	2	29	1
206	94.78	22	3	29	1
207	95.58	8	1	27	3
208	96.85	12	1	30	0
209	95.34	30	0	30	0
210	88.91	30	0	28	2
211	92.59	8	22	30	0
212	86.72	29	1	23	7
213	87.39	23	7	17	13
214	89.70	21	9	27	3
215	83.96	26	4	26	4
216	94.57	23	1	27	3
217	92.93	26	0	29	1
218	86.85	26	4	20	10
219	94.95	15	0	30	0
220	99.03	11	0	30	0
221	82.38	30	0	24	6
222	92.46	30	0	29	1
223	95.32	10	1	29	1
224	99.22	3	0	30	0
225	99.83	1	0	30	0
226	83.22	17	13	25	5
227	79.02	28	2	28	2
228	97.57	11	0	30	0
229	90.83	29	1	24	6
230	90.81	28	2	30	0
231	96.05	10	0	27	3
232	78.78	22	8	27	3
233	95.45	13	0	29	1
234	89.04	29	1	19	11
235	80.24	30	0	19	11

Table 5-2 Results of the SVM model analysis of each question. The accuracy column is the result of the 5-fold cross validation. The true positive, false negative, true negative and false positive columns were determined from the 1/3 of the judged passages that were reserved as a test set. The results of the test set were used in additional measures of effectiveness of the SVM models.

Measures of effectiveness

The following section describes four different evaluation measures; precision, recall, Mathews Correlation Coefficient, and F-measure.

Precision (Specificity)

Precision is the ratio of relevant to total retrieved answers. In other words of all the items that are retrieved, how many are relevant.

$$\boldsymbol{precision} = \frac{\boldsymbol{True\ Positive}}{(\boldsymbol{True\ Positive} + \boldsymbol{False\ Positive})} \qquad \textbf{(5.1)}$$

Recall (Sensitivity)

Recall is a measure of a system's ability to retrieve all possible relevant items and is the fraction of all relevant items that are retrieved. Precision and recall are measures that are often used in many more sophisticated measures of efficiency, and are used later in the chapter to evaluate the Jikitou agents.

$$\boldsymbol{recall} = \frac{\boldsymbol{True\ Positive}}{(\boldsymbol{True\ Positive} + \boldsymbol{False\ Negative})} \qquad \textbf{(5.2)}$$

Mathews Correlation Coefficient

The Mathews Correlation Coefficient (MCC) is a measure of the quality of binary classification. It is a correlation coefficient based on the predicted classification and the observed. A perfect prediction has a coefficient value of 1, a value of 0 for no better than random prediction, and -1 for complete

disagreement between observation and prediction. MCC takes into account true positives, false positives, true negatives, and false negatives.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}} \quad (5.3)$$

F-Measure

The F-measure, which is the weighted harmonic mean of precision and recall, is a single measure that provides a tradeoff between recall and precision:

$$F = \frac{1}{\alpha \frac{1}{precision} + (1 - \alpha)\frac{1}{recall}}$$

$$= \frac{(\beta^2 + 1)precision * recall}{\beta^2 precision + recall} \; where \qquad \beta^2 = \frac{1 - \alpha}{\alpha} \quad (5.4)$$

where αϵ[0,1] and then β ϵ[0,∞]. Setting α=1/2 or

β=1 equally weights recall and precision and is known as the balanced F

$$F_1 = 2 \times \frac{precision \times recall}{precision + recall} \; where \; \beta = 1 \quad (5.5)$$

measure and is denoted F_1.

Measuring SVM models' effectiveness

One way to evaluate the effectiveness of an SVM classification is to check the quality of the model using measures that incorporate TP, FP, TN, and FN values, which are the results of the test set found in Table 5-2, to identify different performance qualities of a model. The measures of precision, recall, Mathews

Correlation Coefficient, and F-measure were calculated for each of the 36 models. Figure 5-4 is a graph that has all 4 measures for each question against the test set data from TREC.

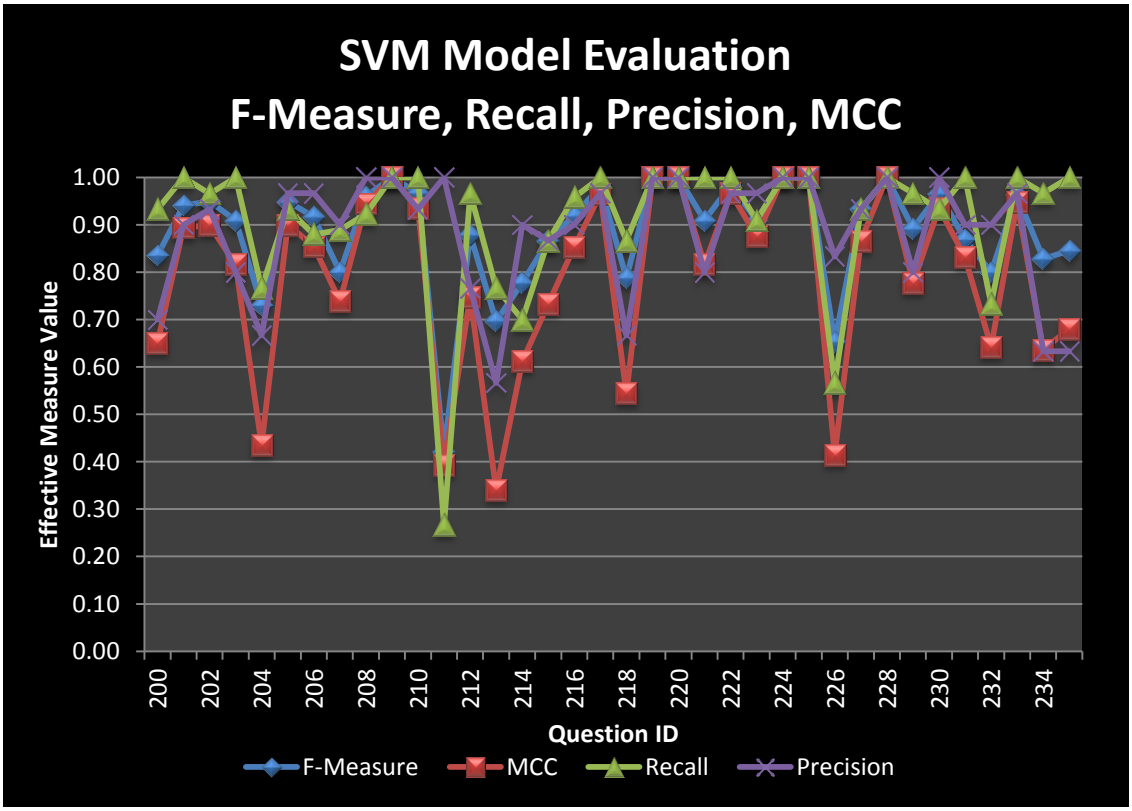


Figure 5-4 SVM model evaluation measure; precision, recall, Mathews Correlation Coefficient, and balanced F-measure overlaid in a single graph. We see that some measures like MCC and recall are more volatile.

Measuring Jikitou's effectiveness

Precision and recall are common evaluation measures for information retrieval systems. The problem with determining recall for information retrieval systems with large corpora is that we do not know how many relevant answers are available for each query. The Jikitou answers were classified using the SVM classifier to predict their relevancy and since I did not run the entire ISDB

collection through each model to determine their relevancy to each question here too we do not know the total number of relevant answers in the database. To get around this I took the aggregate of relevant items retrieved by each of the four different agents as the total relevant set.

Recall and precision values were calculated for each of the 12 query sets that had enough answers judged relevant for the 4 agents. A plot was made for each query set where recall is on the x-axis and precision is plotted on the y-axis. As the number of retrieved results increases, recall increases but precision decreases. The individual query precision-recall results can be found in appendix D.

Interpolated precision-recall

Interpolating a recall/precision curve is a way to visualize the change in precision and recall as the number of ranked results increases. The precision value is interpolated for each of the method's standard recall values (36). The interpolated precision at the *i*-th standard recall is the maximum of the known precision values at that recall level or above. To get an idea of overall performance I took the average precision at each standard recall level over all queries, separately for each agent. Tables that contain the interpolated average precision values at the 11 standard recall values for each of the agents can be found in Appendix E. A graph (Figure 5-5) for these four curves for average interpolated precision versus standard recall was plotted. The line that is closest to the upper right hand corner represents the agent that performed the best.

$$P(r_i) = \max_{r, r_i \le r \le 1.0} P(r) \qquad (5.6)$$

$$r_i \in \{0.0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0\}$$

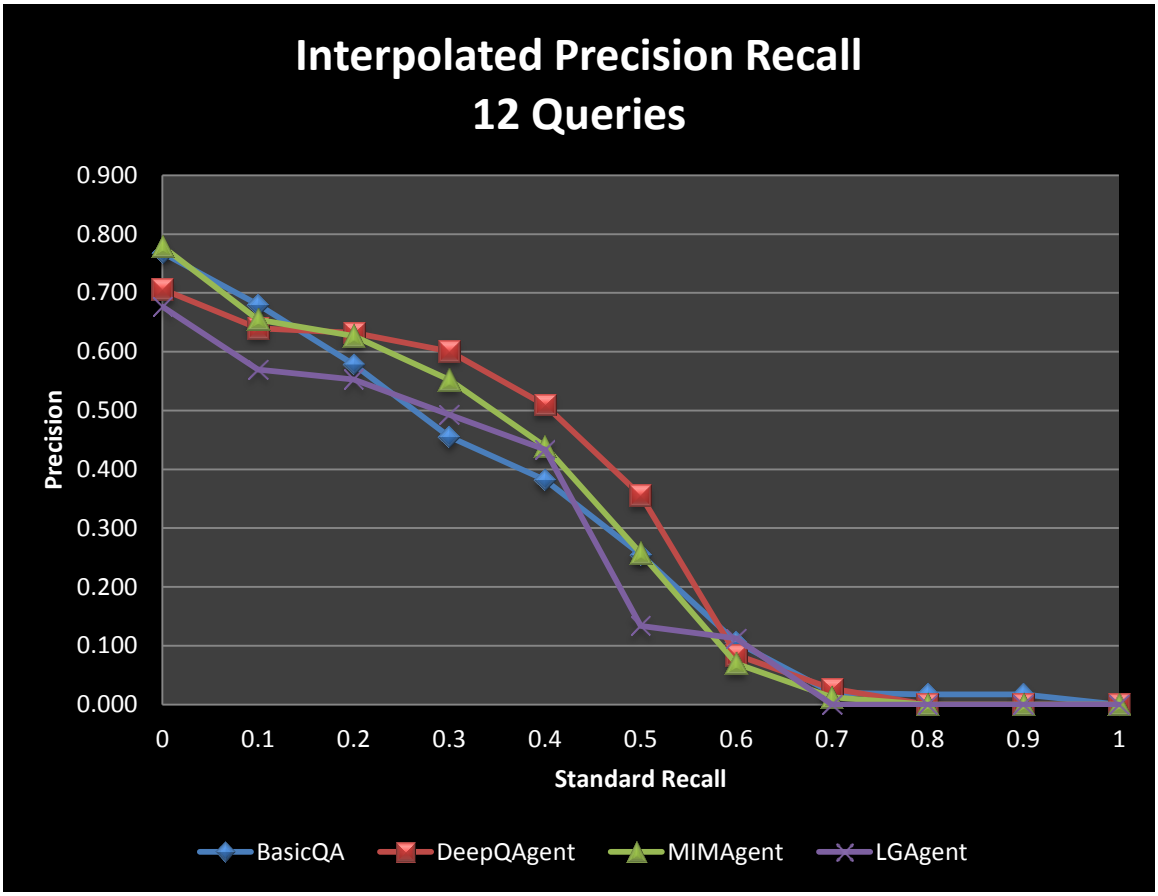


Figure 5-5 To get an idea of overall performance we take the average precision at each standard recall level for all queries for each agent. The line that is closes to the upper right hand corner is the agent that has performed the best by this measure.

The interpolated precision vs. recall graph shows that the DeepQAgent performs generally the best according to this evaluation measure. The MIMAgent appears second best. Recall that the DeepQAgent works by using NLP and deep question analysis, and the MIMAgent uses the same deep question analysis with the addition of using the mutual information measure of genes found in the

questions and the answers. By this measure MIM addition reduced the performance of the agent.

Mean Average Precision (MAP) and Geometric Mean Average Precision (GMAP)

Mean average precision (MAP) and geometric mean average precision (GMAP) are two measures that take into account a collection of queries. This measure gives us an idea of overall effectiveness of an information retrieval system. MAP is calculated using the average precision (AP) over m recall points and then taking the mean of AP (MAP) over *Q* queries. Table 9-22 and Table 9-23 in appendix G contain the raw AP and MAP values.

$$AP = \frac{1}{m}\sum_{j=1}^{m} Precision(Recall_j) \qquad (5.7)$$

$$MAP = \frac{1}{|Q|}\sum_{i=1}^{Q} AP_i \qquad (5.8)$$

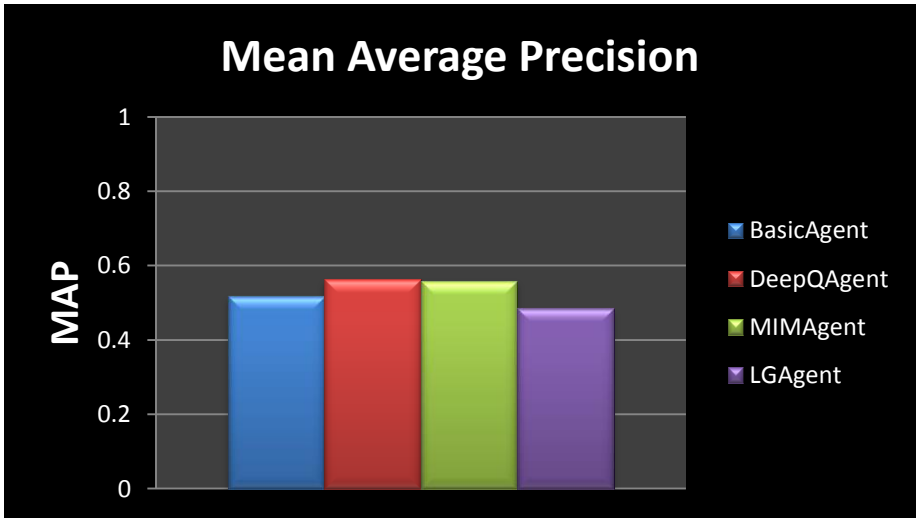


Figure 5-6 Mean Average Precision (MAP) calculated for each agent. MAP is a measure that takes into account a collection of queries.

Figure 5-6 shows the MAP measure for each of the agents. Although the results are not significant we can see that the DeepQAgent and MIMAgent had the highest MAP measure.

GMAP is a measure that emphasizes improvements in queries that have a low average precision. Figure 5-7 shows the results of calculating the GMAP and again we see that the DeepQAgent and MIMAgent have the highest values.

$$GMAP = exp\frac{1}{|Q|}\sum_{i=1}^{Q}\log AP_i \qquad (5.9)$$

Table 9-24 contains the calculated GMAP values.

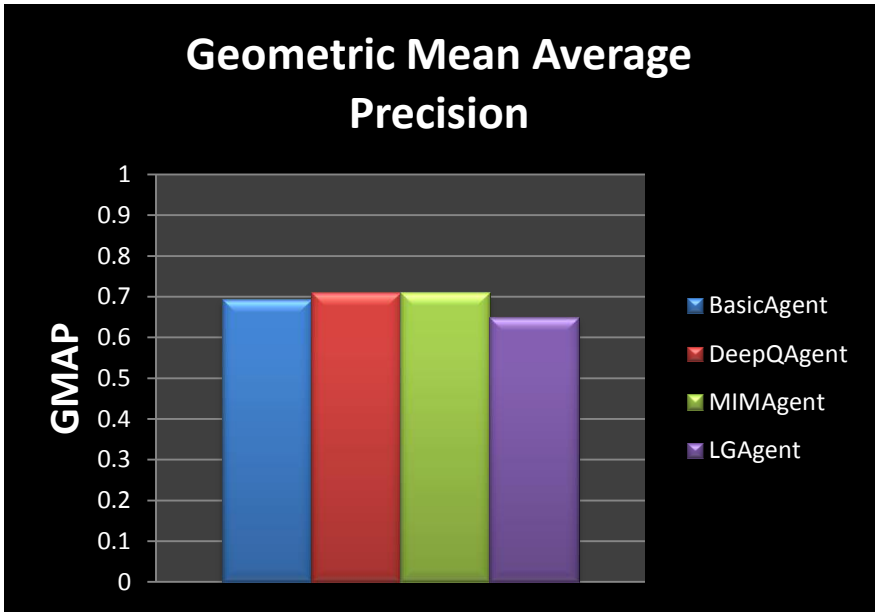


Figure 5-7 The Geometric Mean Average Precision (GMAP) for each of the agents. GMAP is a measure that emphasizes improvements in queries that have a low average precision.

F-measure comparison

The average F-measure was calculated for each of the agents. See Figure 5-8. As with the previous measures we see that DeepQAgent and MIMAgent have the highest F-measures. In this measure comparison we see that the LGAgent is third highest and separation from the other agents is not as great as compared to the other measures. Table 9-25 in appendix G contains the values for the 12 queries for each agent and the average values that were used to create Figure 5-8. Also in appendix G is Figure 9-14 which shows the max F-measure value for each query.

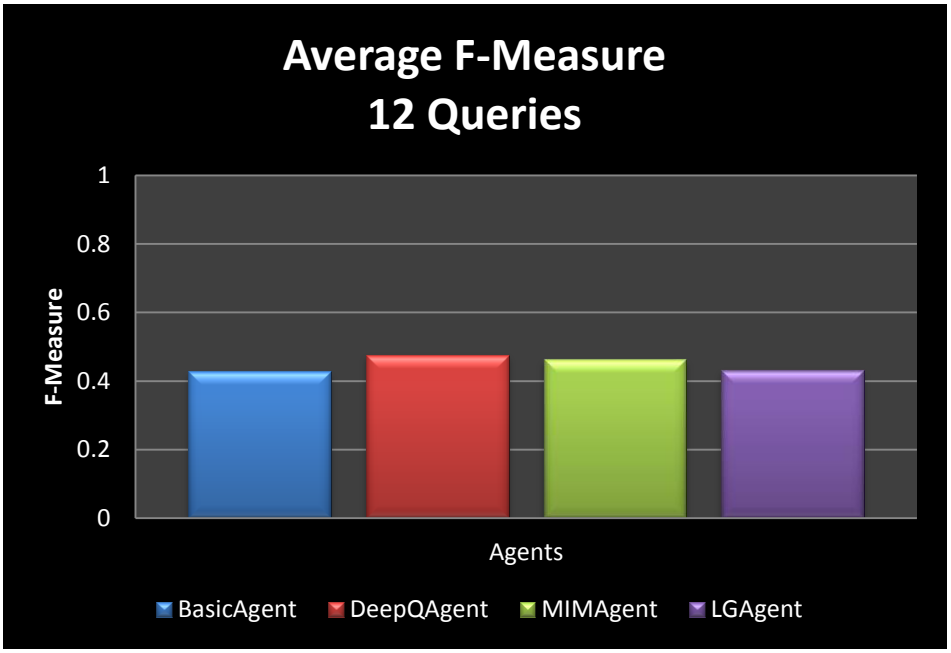


Figure 5-8 The average F-measure for each of the agents. In this case we are using a balanced F-measure meaning that recall and precision are weighted equally.

Agent answer overlap

Overlap in the sentence sets retrieved for each query was analyzed. Venn diagrams for each of the 12 queries was created, see Figure 5-9, which show the

amounts of overlap among the different agents. Summing over all the Venn diagrams, the BasicAgent and LGAgent have the largest distinct set of answers they classified as relevant. The DeepQAgent and MIMAgent have the most overlap which is to be expected since the only difference between the two is that MIMAgent boosts the scores of answers mentioning genes with an MIM score between the query and answer. Often no MIM score exists and in these cases they both return the same answer set, thus creating the high degree of overlap between the two. Overall, the results show that answers are frequently unique to an agent, lending validity to the idea of having multiple search agents tuned to get different relevant answers from the knowledge space.

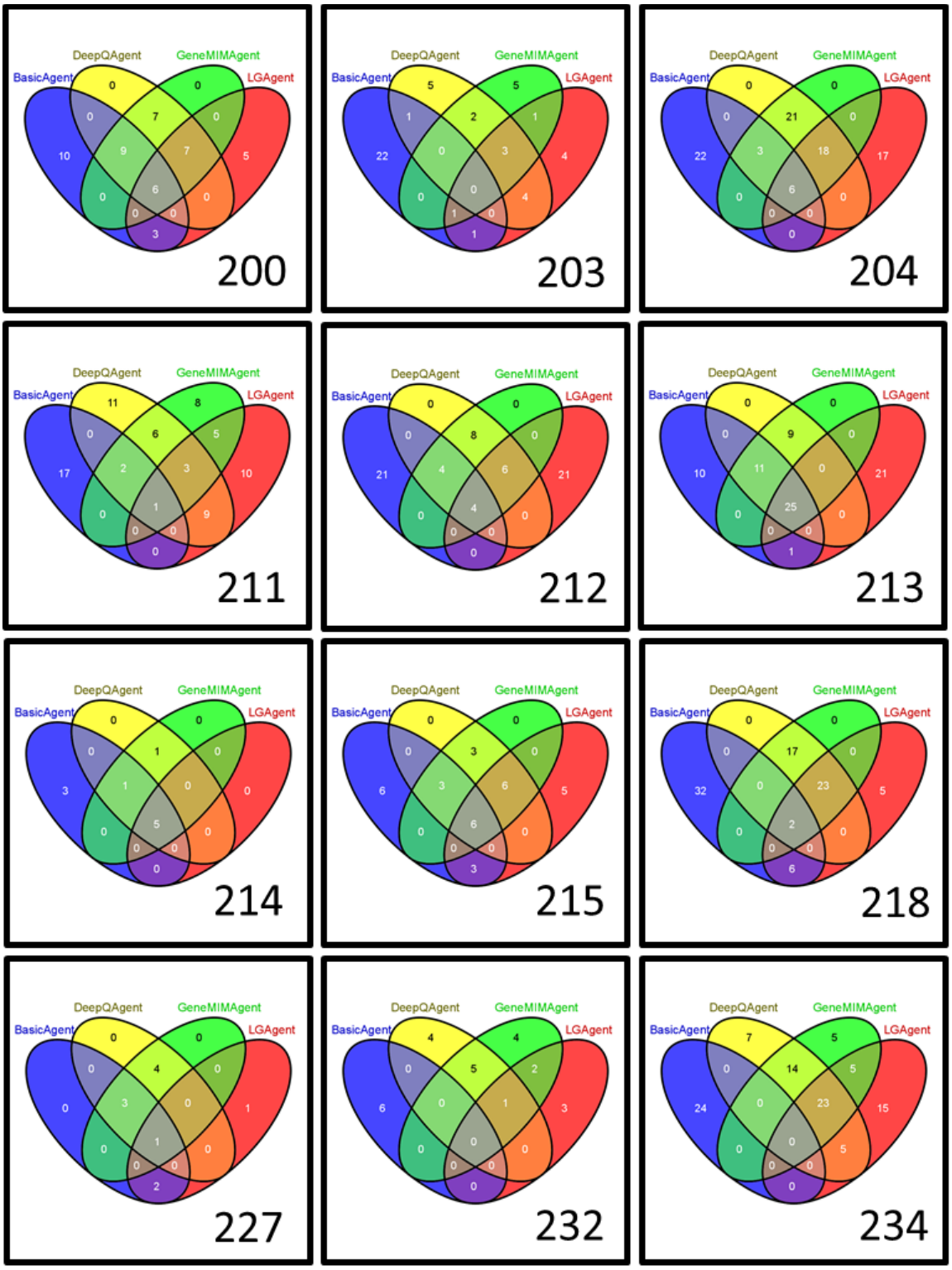


Figure 5-9 A Venn diagram for each of the 12 question showing the amount of answer overlap for answers classified as relevant.

Chapter 6: Future

There are multiple directions to expand and improve the system. In this section I will expand and explain in detail some plans for future research.

6.1 Additional agents

New types of agents should be evaluated. The ISDB sentence database has already been parsed using the Stanford Parser (87). The Stanford Parser is a probabilistic natural language parser which uses knowledge of language gained from hand-parsed sentences to assign the most probable structures to new sentences. An agent that leverages the syntactic information produced by this parser should be created and evaluated. It would also be interesting to identify other types of biological data that could be useful in identifying answers and design agents accordingly.

In addition to new agents, we hope to devise a method to combine the results from the agents to present the best subset from each agent to the user. This would allow for the user to benefit from integrating the information retrieval strengths of all of the agents.

6.2 Additional resources

There are many additional resources that have been shown to improve query results and are being considered for implementation in a future version of Jikitou. The list below is a few of the resources and their descriptions:

- The Medical Subject Headings (MeSH) (88) terms are a controlled vocabulary biomedical lexicon which is organized hierarchically. The headings at the highest level are broad, and they become more specific as one moves down the hierarchical structure. It is freely available from the National Library of Medicine.
- The Unified Medical Language system (UMLS) (89) includes three knowledge sources:
    - The Metathesaurus (90), a multilingual vocabulary database that contains information about biomedical concepts, names, and the relationships among them.
    - The Semantic Network, a consistent organization of all the concepts that are present in the UMLS Metathesaurus which provides a set of relationships among the concepts.
    - The Specialist Lexicon, containing information useful for natural language processing systems. It contains a general English lexicon which includes biomedical terms. Terms include multi-word items that form a lexical item. Abbreviations and acronyms are also included in this resource. Each entry includes syntactic, morphological, and orthographic information.

Another issue facing the knowledge base is that the data can quickly become outdated. The knowledge base needs to be updated periodically because new terminology and literature are constantly being introduced. It is

desirable to automate the process of checking for changes in external databases and taking the appropriate actions.

6.3 Current agent modification

It would also be useful to experiment with different scoring equations for the agents to tune them and seek out their maximum potential. Many of the current agents had issues with answers classified as being relevant not being ranked above answers that were classified as non-relevant.

6.4 Future evaluation

The evaluation system showed that it could be a viable way to evaluate the agents. It would be useful to improve the SVM classifier by taking into account additional document features such as POS, grammatical relations, and bi-grams. The similarity measure for measuring performance could also be improved by using a more robust scoring algorithm to replace the cosine similarity measure which does not take into account any actual available textual/NL clues as to the relatedness of two documents.

Although the main reason for the design of the evaluation system was to avoid user studies, in the future it would be helpful to do one, it would be a way to validate or identify weakness in the current automated method. It would also be useful to get human relevancy judgments for answers from the Jikitou corpus and compare them to SVM classified answers. User studies would also help to evaluate Jikitou's user interface which is really the only way to evaluate the UI.

Chapter 7: Conclusions

Jikitou is a complete information retrieval system, which in addition to a question answering system, includes tools for indexing as well as tools to evaluate the system. Jikitou is designed to return short answers to biological questions. The QA system is intended to decrease the effort required to find answers when compared to methods that use traditional information retrieval systems (e.g., PubMed). Short answers can eliminate the need to read or scan entire documents to obtain desired information. The Jikitou QA system is designed to be a tool used by biomedical domain experts as well as useful and informative to students. The system has the potential to impact the design of future question answering systems, thereby advancing the field. It is not only a live information retrieval tool but also an IR research system. Jikitou combines multiple natural language processing techniques, data resources and technologies to create a unique system to help researchers navigate a biomedical corpus.

7.1 Evaluation

The evaluation techniques provide a method to extend the use of manually pre-judged passages to evaluating systems that were not among the original systems that contributed to the pool of passages and indeed use different corpuses altogether. The two evaluation methods were shown to validate each other where often a higher cosine score trend was observed when more answers

were classified as being relevant and a lower cosine score trend was seen when more answers where classified as not relevant. They provide a way to identify performance differences among agents and agent algorithm variations.

There were occasions where the two evaluation methods differed. The TREC passages were often comprised of multiple sentences so during model training the models were tuned to longer answer strings. This may have resulted in the shorter Jikitou answers being incorrectly classified as not-relevant because of length. The cosine score evaluation normalizes so that the effect of different string length was minimized. So the fact that one method controlled for length and the other did not may be a reason for seeing a difference in results.

7.2 Agents

Jikitou uses an agent based architecture in which different agents search the information space using different retrieval strategies. The amount of overlap among the agents shows that the agents retrieve significantly different sets of relevant information. Although the LGAgent often had the lowest performance according to the evaluation measures used, a higher percentage of the answers that it returned and that were also deemed relevant were different from the answers returned by the other agents. It was able to retrieve answers that the other agents missed. This is additional evidence that implementing a multiple search strategy is an ideal method to maximize the recall of relevant answers from the information space. The LGAgent's scoring equation should be further investigated and tuned to return more relevant answers with higher ranks. The

BasicAgent is a good agent to compare to the other agents. For all measures of performance and quality it performed well. The DeepQAgent was shown to be the best by most of the evaluation measures. It is clear that providing phrase queries that take into account key phrases from the questions, and gene synonyms increased its performance over the BasicAgent. The MIMAgent was basically the DeepQAgent with an added module to use the correlation value of gene co-expression. Although not consistently higher, for certain answers, a boost was seen in cosine score over the answers from the DeepQAgent, meaning that the MIM boost brought back answers that were more closely related to the judged relevant passages in those cases. There were occasions where using MIM boosted answers up in the rank that were then found to be not relevant or had a relatively low cosine scores, however. The next step would be to tweak each of the agent's search algorithms to optimize their performances, and compare the difference in evaluation results.

7.3 User interface

Jikitou addresses two current gaps in current QA systems through the integration of the HyperGlossary. The first gap is answer generation and presentation. It is through the HyperGlossary system that the user is connected to multimedia information that has the potential to add value to the text answers returned by the agents. The second gap is a lack of systems that allow the user to choose his or her environment, establish context, have the system take that information into account, and automatically return the appropriate answer.

The choice of the glossary changes the context per the user's preference. The HyperGlossary gives the user the ability to choose a glossary that allows him or her to customize the information that is returned. The HyperGlossary-enhanced answers make the information more accessible to a wider audience and reduces the need to search other resources. This approach can enable users with varying levels of knowledge to have access to their choice of glossary to aid in the understanding of the answer. The terms in the answer that get enhanced are determined by the glossary the user chooses.

Answers are parsed for keywords that are linked to external sources of information. The enhancement of answers is achieved through the use of heterogeneous multimedia sources such as ChemSpider, ChemEdDL, and RCSB Protein Data Bank. The fusion of the HyperGlossary and Jikitou helped to create a unique system to assist researchers in navigating the current information deluge.

7.4 Query refinement

The system's ability to have a dialog with the user is likely to result in more relevant answers being retrieved compared to systems that rely solely on the user to supply the query unguided. It is through a dialog with the user that essential parameters can be communicated to the system. In Jikitou the dialog is created through the terms suggested in the question drop down box as the user types and in both feedback panels, all of which are dynamically updated as the

user types. These components are used to establish context and help refine the question in order to return better answers.

7.5 Architecture

The system is designed to be easily updated and improved. This ability is achieved with modular components that permit swapping of different algorithms. An agent design was adopted not only for algorithmic purposes, but also to keep the system modular and allow it to be distributed to increase speed performance. Agents can be run in parallel on the same processing unit or could be easily assigned to their own unit. The Catalyst framework for Perl was used because it facilitates modular code by keeping the application logic separate from the user interface code. Ultimately it would be good to keep track of versioned protocols to help the system evolve and improve. The ability of agents in the system to evolve depends on our ability to understand what happened, through the tracking of results of different algorithms. The methods implemented in this system ensure that these goals are achieved, such that the system could continually evolve to become and stay an invaluable tool to researchers.

7.6 Summary

This work has focused on three synergistic contributions. Firstly, Jikitou serves as a model QA architecture, one which integrates the important properties of agents, the MVC framework, incorporating biological data, NLP, and deep question analysis. Other system builders can incorporate these ideas into their own designs. Secondly, Jikitou provides a model user interface strategy. The

flexible and diverse palette of UI features serves to demonstrate a user-centered UI approach that other system builders could borrow. These features include dynamic feedback panels, medical term focused autocomplete, and connections to a multitude of additional resources through the HyperGlossary. Thirdly, the evaluation strategy I designed, which is based on a method suggested by Büttcher et. al. (85), includes using the cosine similarity measure as an innovative contribution. The approach cleanly trades the noise and subjectivity of a user study for the objectivity and reproducibility of a method based on standard evaluation data. Other system builders could benefit by using this evaluation strategy as well.

Chapter 8: List of references

1. Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. Comput Networks and ISDN Syst. 1998;30(1):107-17.

2. Home - PubMed - NCBI [Internet].; cited 1/4/2013]. Available from: http://www.ncbi.nlm.nih.gov/pubmed/.

3. Zweigenbaum P. Question answering in biomedicine. EACL. 2003.

4. Zweigenbaum P, Demner-Fushman D, Yu H, Cohen KB. Frontiers of biomedical text mining: Current progress. Brief Bioinform. 2007;8(5):358-75.

5. Allan J, Aslam J, Belkin N, Buckley C, Callan J, Croft B, et al. Challenges in information retrieval and language modeling: Report of a workshop held at the center for intelligent information retrieval, university of massachusetts amherst, september 2002. SIGIR Forum. 2003;37(1):31-47.

6. Grant S, Marshall M, Page K, Cumiskey M, Armstrong D. Synapse proteomics of multiprotein complexes: En route from genes to nervous system diseases. Hum Mol Genet. 2005; #14:R225-R23.

7. Uetz P, Giot L, Cagney G, Mansfield TA, Judson RS, Knight JR, et al. A comprehensive analysis of protein-protein interactions in saccharomyces cerevisiae. Nature. 2000 February;403(6770):623-7.

8. Ding J, Berleant D, Nettleton D, Wurtele E. Mining MEDLINE: Abstracts, sentences, or phrases. Proceedings of the pacific symposium on biocomputing; 2002.

9. Berger A, Lafferty J. Information retrieval as statistical translation. In proceedings of the 1999 ACM SIGIR conference on research and development in information retrieval; August 15-19, 1999; University of California at Berkeley. New York, NY, USA: ACM; 1999.

10. Mayburry M. New direction in question answering. In: Strzalkowski T, Harabagui SM, editors. Advances in Open Domain Question Answering. Springer Netherlands; 2008. p. 533.

11. Ferrucci D, Nyberg E, Allan J, Barker K, Brown E, Chu-Carroll J, et al. Towards the open advancement of question answering systems. 2009.

12. Wren JD. Question answering systems in biology and medicine - the time is now. Bioinformatics. 2011 jul;27(14):2025-6.

13. Athenikos SJ, Han H. Biomedical question answering: A survey. Comput Methods Programs Biomed. 2010;99(1):1.

14. Lee M, Cimino J, Zhu HR, Sable C, Shanker V, Ely J, et al. Beyond information retrieval - medical question answering. AMIA. 2006:469-73.

15. Cohen AM, Hersh WR. A survey of current work in biomedical text mining. Brief Bioinform. 2005 Mar;6(1):57-71.

16. IBM watson [Internet]. [cited 1/4/2013]. Available from: http://www-03.ibm.com/innovation/us/watson/.

17. Cao Y, Cimino JJ, Ely J, Yu H. Automatically extracting information needs from complex clinical questions. J. of Biomedical Informatics. 2010 December;43(6):962-71.

18. Cao Y, Liu F, Simpson P, Antieau L, Bennett A, Cimino JJ, et al. AskHERMES: An online question answering system for complex clinical questions. J Biomed Inform. 2011 4;44(2):277-88.

19. AskHermes - the clinical question answering system [Internet]. [cited 1/4/2013]. Available from: http://www.askhermes.org/.

20. Gobeill J, Tbahriti I, Ehrler F, Ruch P. Vocabulary-driven passage retrieval for question-answering in genomics. TREC; 2007; Gaithersburg, Maryland, USA. National Institute of Standards and Technology (NIST); 2007.

21. EAGLi: The EAGL project's biomedical question answering and information retrieval interface [Internet]. [cited 1/4/2013]. Available from: http://eagl.unige.ch/EAGLi/.

22. Cruchet S, Gaudinat A, Boyer C. Supervised approach to recognize question type in a QA system for health. Stud Health Technol Inform. 2008;136.

23. Cruchet S, Gaudinat A, Rindflesch T, Boyer C. What about trust in the question answering world. AMIA 2009 annual symposium; ; 2009.

24. Honqa [Internet]. [cited 1/4/2013]. Available from: http://services.hon.ch/cgi-bin/QA10/qa.pl.

25. MEDLINE/PubMed resources guide [Internet]. [cited 1/4/2013]. Available from: http://www.nlm.nih.gov/bsd/pmresources.html.

26. Diseases & conditions - medscape reference [Internet]. [cited 1/4/2013]. Available from: http://emedicine.medscape.com/.

27. Home - PMC - NCBI [Internet]. [cited 1/4/2013]. Available from: http://www.ncbi.nlm.nih.gov/pmc/.

28. Health on the net foundation [Internet]. [cited 1/4/2013]. Available from: http://www.hon.ch/.

29. National library of medicine - national institutes of health [Internet]. [cited 1/4/2013]. Available from: http://www.nlm.nih.gov/.

30. HONcode: Principles - quality and trustworthy health information [Internet]. [cited 1/4/2013]. Available from: http://www.hon.ch/HONcode/Conduct.html.

31. Hearst MA. Untangling text data mining. ACL. 1999:3-10.

32. Rebholz-Schuhmann D, Kirsch H, Couto F. Facts from text - is text mining ready to deliver? PloS Biology. 2005;3(2):0188-91.

33. McGill M, Koll M. An evaluation of factors affecting document ranking by information retrieval systems. Syracuse Univ , School of Information Studies: Tech. Rep; 1979.

34. Apache lucy [Internet]. [cited 1/4/2013]. Available from: http://lucy.apache.org/.

35. Torre CJdl, Martín-Bautista MJ, Sánchez D, Miranda MAV. Text mining: Intermediate forms on knowledge representation. EUSFLAT conf.; 2005.

36. Manning CD, Raghavan P, Schütze H. Introduction to information retrieval. 1st ed. Cambridge University Press; 2008.

37. Porter MF. An algorithm for suffix stripping. . 1980.

38. GNU aspell [Internet]. [cited 1/4/2013]. Available from: http://aspell.net/.

39. OpenMedSpel | world class open source medical spelling tools | [Internet]. [cited 1/4/2013]. Available from: http://www.e-medtools.com/openmedspel.html.

40. Abul Seoud R, Youssef A, Kadah YM. Extraction of protein interaction information from unstructured text using a link grammar parser. Computer

engineering & systems, 2007. ICCES'07. international conference on; IEEE; 2007.

41. Ding J, Berleant D, Xu J, Fulmer AW. Extracting biochemical interactions from MEDLINE using a link grammar parser. Tools with artificial intelligence, 2003. proceedings. 15th IEEE international conference on; IEEE; 2003.

42. Sleator DD, Temperley D. Parsing english with a link grammar. 1991.

43. Industry leading JavaScript framework for building desktop web apps | sencha ext JS | products | sencha [Internet]. [cited 1/4/2013]. Available from: http://www.sencha.com/products/extjs.

44. Bauer M, Belford R, Ding J, Berleant D. ISDB: Interaction sentence database. BMC Research Notes. 2010;3(1):122.

45. Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, et al. Gene ontology: Tool for the unification of biology. Nat Genet. 2000 May; 25(1):25-9.

46. Liu H, Hu ZZ, Zhang J, Wu C. BioThesaurus: A web-based thesaurus of protein and gene names. Bioinformatics. 2006 Jan; 22(1):103-5.

47. Fellbaum, Fellbaum C. WordNet: An electronic lexical database (language, speech, and communication). The MIT Press; 1998.

48. Maglott DR, Ostell J, Pruitt KD, Tatusova TA. Entrez gene: Gene-centered information at NCBI. Nucleic Acids Res. 2007:26-31.

49. Das-Gupta P, Katzer J. A study of the overlap among document representations. SIGIR Forum. 1983;17(4):106-14.

50. Bradshaw JM. An introduction to software agents. In: Bradshaw JM, editor. Software Agents. Cambridge MA: MIT Press; 1997. p. 1.

51. Jennings NR, Wooldridge M. Applications of intelligent agents. In: Agent Technology: Foundations, Applications, Markets. Springer-Verlag; 1998. p. 3-28.

52. Petit-Rozé C, Grislin-Le Strugeon E. MAPIS, a multi-agent system for information personalization. Information and Software Technology. 2006 2;48(2):107-20.

53. Lhotská L, Prieto L. A multi-agent system for information retrieval. EUROCAST; 2007.

54. Decker K, Zheng X, Schmidt C. A multi-agent system for automated genomic annotation. International conference on autonomous agents proceedings of the fifth international conference on autonomous agents; Montreal, Quebec, Canada. New York, USA: ACM; 2001.

55. Chu-Carroll J, Czuba K, Prager J, Ittycheriah A, Blair-Goldensohn S. IBM's PIQUANT II in trec 2004. Proceedings TREC; 2004.

56. Chu-Carroll J, Prager J, Welty C, Czuba K, Ferrucci D. A multi-strategy and multi-source approach to question answering. In proceedings of text REtrieval conference; 2003.

57. Hersh WR, Cohen AM, Ruslen L, Roberts PM. TREC 2007 genomics track overview. TREC; 2007.

58. Ruthven I. Re-examining the potential effectiveness of interactive query expansion. SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on research and development in informaion retrieval; Toronto, Canada. New York, NY, USA: ACM; 2003.

59. Vakkari P. Subject knowledge, source of terms, and term selection in query expansion: An analytical study. ECIR; 2002.

60. Belkin NJ, Cool C, Head J, Jeng J, Kelly D, Lin S, et al. Relevance feedback versus local context analysis as term suggestion devices: Rutgers' TREC-8 interactive track experience. TREC-8, proceedings of the eighth text retrieval conference; Harman; 2000.

61. Jensen Juhl L, Saric J, Bork P. Literature mining for the biologist: From information retrieval to biological discovery. Nature Reviews GENETICS. 2006;7:119-29.

62. Wren JD. A global meta-analysis of microarray expression data to predict unknown gene functions and estimate the literature-data divide. Bioinformatics. 2009 July 01;25(13):1694-701.

63. Mikhail D, Jonathan W, Dumas EK, Dozmorov IM, Benbrook DM, Marcia S. High-throughput processing and normalization of one-color microarrays for transcriptional meta-analyses. BMC Bioinformatics;12.

64. Edgar R, Domrachev M, Lash AE. Gene expression omnibus: NCBI gene expression and hybridization array data repository. Nucleic Acids Research. 2002 January 01;30(1):207-10.

65. Entrez programming utilities help - NCBI bookshelf [Internet]. [cited 1/4/2013]. Available from: http://www.ncbi.nlm.nih.gov/books/NBK25501/.

66. ChemSpider [Internet]. [cited 2012]. Available from: http://www.chemspider.com/.

67. Berman HM, Westbrook J, Feng Z, Gilliland G, Bhat TN, Weissig H, et al. The protein data bank. Nucleic Acids Research. 2000 January 01;28(1):235-42.

68. Belford RE, Bauer MA, Berleant D, Holmes JL, Moore JW. ChemEd DL WikiHyperGlossary: Connecting digital documents to online resources, while coupling social to canonical definitions within a glossary. 22nd international conference on chemisty education; July15th-20th 2012; Italy: Editrice di chimica; 2012.

69. Apweiler R, Bairoch A, Wu CH, Barker WC, Boeckmann B, Ferro S, et al. UniProt: The universal protein knowledgebase. Nucleic Acids Res;32:115-9.

70. Consortium TU. Reorganizing the protein space at the universal protein resource (UniProt). Nucleic Acids Research. 2012;40(D1):D71-5.

71. RCSB protein data bank - RCSB PDB [Internet]. [cited 1/4/2013]. Available from: http://www.rcsb.org/pdb/home/home.do.

72. Stein SE, Heller SR, Tchekhovskoi D. An open standard for chemical structure representation: The IUPAC chemical identifier. 2003.

73. Jmol: An open-source java viewer for chemical structures in 3D [Internet]. [cited 2012]. Available from: http://www.Jmol.org/.

74. Welcome to the chemical education digital library [Internet]. [cited 1/4/2013]. Available from: http://www.chemeddl.org/.

75. O'Boyle N, Banck M, James C, Morley C, Vandermeersch T, Hutchison G. Open babel: An open chemical toolbox. Journal of Cheminformatics. 2011;3(1):33.

76. Vainio MJ, Johnson MS. Generating conformer ensembles using a multiobjective genetic algorithm. Journal of Chemical Information and Modeling. 2007;47(6):2462-74.

77. Krause S, Willighagen EL, Steinbeck C. JChemPaint - using the collaborative forces of the internet to develop a free editor for 2D chemical structures. Molecules. 2000;5:93-8.

78. Hersh W, Voorhees E. TREC genomics special issue overview. Inf.Retr. 2009 feb;12(1):1-15.

79. Hersh W, Cohen A, Roberts P, Rekapalli HK. TREC 2006 genomics track overview. The fifteenth text retrieval conference; 2006.

80. TREC genomics track [Internet]. [cited 1/4/2013]. Available from: http://ir.ohsu.edu/genomics/.

81. Buckley C, Voorhees EM. Retrieval evaluation with incomplete information. Proceedings of the 27th annual international ACM SIGIR conference on research and development in information retrieval; Sheffield, United Kingdom. New York, NY, USA: ACM; 2004.

82. Grönqvist L. Evaluating latent semantic vector models with synonym tests and document retrieval. ELECTRA workshop: Methodologies and evaluation of lexical cohesion techniques in real-world applications beyond bag of words; in association with ACM SIGIR, pages 86-88, 2005..

83. Ahlgren P, Gronqvist L. Retrieval evaluation with incomplete relevance data: A comparative study of three measures. Conference on information and knowledge management: Proceedings of the 15 th ACM international conference on information and knowledge management; 2006.

84. Aslam JA, Yilmaz E. Inferring document relevance via average precision. Proceedings of the 29th annual international ACM SIGIR conference on research and development in information retrieval; ACM; 2006.

85. Büttcher S, Clarke CLA, Yeung PCK, Soboroff I. Reliable information retrieval evaluation with incomplete and biased judgements. Proceedings of the 30th annual international ACM SIGIR conference on research and development in information retrieval; ACM; 2007.

86. Chang C, Lin C. LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology. 2011;2(3):27:1,27:27.

87. Klein D, Manning CD. Accurate unlexicalized parsing. Proceedings of the 41st annual meeting on association for computational linguistics - volume 1; Sapporo, Japan. Stroudsburg, PA, USA: Association for Computational Linguistics; 2003.

88. Rogers F. Medical subject headings. Bull Med Libr Assoc. 1963;51:114.

89. Bodenreider O. The unified medical language system (UMLS): Integrating biomedical terminology. Nucl.Acids Res. 2004 January;32(suppl_1):D267-270.

90. Schuyler PL, Hole WT, Tuttle MS, Sherertz DD. The UMLS metathesaurus: Representing different views of biomedical concepts. Bull Med Libr Assoc. 1993 April;81(2):217-22.

Chapter 9: Appendices

9.1 Appendix A: Question list

ID	Questions
200	What serum proteins change expression in association with high disease activity in lupus?
201	What mutations in the Raf gene are associated with cancer?
202	What drugs are associated with lysosomal abnormalities in the nervous system?
203	What cell or tissue types express receptor binding sites for vasoactive intestinal peptide (VIP) on their cell surface?
204	What nervous system cell or tissue types synthesize neurosteroids in the brain?
205	What signs or symptoms of anxiety disorder are related to coronary artery disease?
206	What toxicities are associated with zoledronic acid?
207	What toxicities are associated with etidronate?
208	What biological substances have been used to measure toxicity in response to zoledronic acid?
209	What biological substances have been used to measure toxicity in response to etidronate?
210	What molecular functions are attributed to glycan modification?
211	What antibodies have been used to detect protein PSD-95?
212	What genes are involved in insect segmentation?
213	What genes are involved in Drosophila neuroblast development?
214	What genes are involved axon guidance in C.elegans?
215	What proteins are involved in actin polymerization in smooth muscle?
216	What genes regulate puberty in humans?
217	What proteins in rats perform functions different from those of their human homologs?
218	What genes are implicated in regulating alcohol preference?
219	In what diseases of brain development do centrosomal genes play a role?
220	What proteins are involved in the activation or recognition mechanism for PmrD?
221	Which pahtways are mediated by CD44?

222	What molecular functions is LITAF involved in?
223	Which anaerobic bacterial strains are resistant to Vancomycin?
224	What genes are involved in the melanogenesis of human lung cancers?
225	What biological substances induce clpQ expression?
226	What proteins make up the murine signal recognition particle?
227	What genes are induced by LPS in diabetic mice?
228	What genes when altered in the host genome improve solubility of heterologously expressed proteins?
229	What signs or symptoms are caused by human parvovirus infection?
230	What pathways are involved in Ewing's sarcoma?
231	What tumor types are found in zebrafish?
232	What drugs inhibit HIV type 1 infection?
233	What viral genes affect membrane fusion during HIV infection?
234	What genes make up the NFkappaB signaling pathway?
235	Which genes involved in NFkappaB signaling regulate iNOS?

Table 9-1 A list of the 36 TREC question and their IDs used to evaluate the Jikitou system.

9.2 Appendix B: Agent relevant cosine scores by rank

An ANOVA was performed to determine if there was a significant difference between agents' cosine scores by rank. The results of the ANOVA were a p-value of .55 which is well above a p-value ≤ .05 so we accept the null hypothesis that the means are equal. See Table 9-2.

Anova: Single Factor

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	48	5.159888	0.107498	0.000272
DeepQAgent	48	5.226204	0.108879	0.000295
MIMAgent	48	5.292254	0.110255	0.000354
LGAgent	48	5.066484	0.105552	0.000191

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.000581	3	0.000194	0.695602	0.555805	2.652646
Within Groups	0.052312	188	0.000278			
Total	0.052893	191				

Table 9-2 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using the moving average cosine similarity measure (window size of 3)

9.3 Appendix C: relevant vs. not-relevant data analysis

Before determining the difference seen between the cosine measure scores of the relevant and not-relevant, it was first determined if the data was normal. A Shapiro-Wilk test for normality was performed in R. The results are shown in Table 9-3. Both test resulted in non-significant results meaning that both datasets conform to being normally distributed.

Shapiro-Wilk normality test		
Dataset	**W**	**p-vlaue**
Relevant	0.9837	0.7156
Not-Relevant	0.9699	0.2297

Table 9-3 The results of the Shapiro-Wilk normality test on relevant and not-relevant average cosine score by rank. The results of both test were not significant at the p-value of <.05 meaning that both datasets are normally distributed.

A t-test was then performed in Excel, the results of which can be seen in Table 9-4. The t-test was done to determine the difference between the relevant and not-relevant average cosine score by rank. The null hypothesis being that the difference between the two means is the same. The alternative hypothesis is that the means are different. The results of the t-test is a t-statistic value of -4.951 with a p-value of <.05. This means we reject the null hypothesis and accept the alternative that the means of the datasets are different.

t-Test: Two-Sample Assuming Equal Variances

	NOT Score Average	*Relevant Score Average*
Mean	0.074790593	0.080031355
Variance	0.002650613	0.005417105
Observations	7200	7200
Pooled Variance	0.004033859	

Hypothesized Mean Difference	0
df	14398
t Stat	-4.950913491
P(T<=t) one-tail	3.73511E-07
t Critical one-tail	1.644959466
P(T<=t) two-tail	7.47022E-07
t Critical two-tail	1.960128762

Table 9-4 The results of the t-test to determine if the difference between the relevant and not-relevant average cosine score by rank. The results of the test is that the difference between the two means is significant at a p-value of <.05.

9.4 Appendix D: Difference among agents per query: data analysis

To determine the difference between agents by query an ANOVA was performed on each query. The ANOVA was done in excel on the 12 different queries.

Anova: Single Factor (QID: 200)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	5.9063	0.1181	0.00166
DeepQAgent	50	6.4355	0.1287	0.001301
MIMAgent	50	6.4355	0.1287	0.001301
LGAgent	50	6.3765	0.1275	0.001314

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.003941	3	0.001314	0.942422	0.421167	2.650677
Within Groups	0.273197	196	0.001394			
Total	0.277138	199				

Table 9-5 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 200.

Anova: Single Factor (QID: 203)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	5.5222	0.1104	0.001745
DeepQAgent	50	6.4029	0.1281	0.003948
MIMAgent	50	5.5784	0.1116	0.002633
LGAgent	50	6.2356	0.1247	0.003623

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.012138	3	0.004046	1.354451	0.258027	2.650677
Within Groups	0.585479	196	0.002987			

Total	0.597617	199

Table 9-6 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 203.

Anova: Single Factor (QID: 204)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	6.4912	0.1298	0.001845
DeepQAgent	50	4.3836	0.0877	0.003979
MIMAgent	50	4.3836	0.0877	0.003979
LGAgent	50	3.7620	0.0752	0.002701

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.085524631	3	0.02851	9.12019	1.11E-05	2.650677
Within Groups	0.612663654	196	0.00313			
Total	0.698188285	199				

Table 9-7 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 204.

Anova: Single Factor (QID 211)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	7.1505	0.1430	0.002182
DeepQAgent	50	3.3587	0.0672	0.002228
MIMAgent	50	3.0053	0.0601	0.001955
LGAgent	50	2.2203	0.0444	0.000375

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.289520621	3	0.09651	57.27172	1.25E-26	2.650677

Within Groups	0.330273757	196	0.00169
Total	0.619794379	199	

Table 9-8 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 211.

Anova: Single Factor (QID 212)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	2.1604	0.0432	0.000446
DeepQAgent	50	2.8374	0.0567	0.000439
MIMAgent	50	2.8374	0.0567	0.000439
LGAgent	50	2.5621	0.0512	0.000463

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.006149338	3	0.00205	4.586946	0.003969	2.650677
Within Groups	0.087586987	196	0.00045			
Total	0.093736325	199				

Table 9-9 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 212.

Anova: Single Factor (QID: 213)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	5.29048	0.10581	0.003184
DeepQAgent	50	5.50599	0.11012	0.002347
MIMAgent	50	5.50599	0.11012	0.002347
LGAgent	50	4.50229	0.09005	0.001689

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.013644861	3	0.00455	1.901615	0.130632	2.650677
Within Groups	0.468793213	196	0.00239			
Total	0.482438074	199				

Table 9-10 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 213.

Anova: Single Factor (QID: 214)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	8.4648	0.1693	0.002756
DeepQAgent	50	9.7057	0.1941	0.001651
MIMAgent	50	9.7057	0.1941	0.001651
LGAgent	50	7.3301	0.1466	0.0036

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.078268411	3	0.02609	10.80438	1.33E-06	2.650677
Within Groups	0.473283568	196	0.00241			
Total	0.551551979	199				

Table 9-11 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 214.

Anova: Single Factor (QID: 215)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	11.8285	0.2366	0.008375
DeepQAgent	50	12.5439	0.2509	0.005165
MIMAgent	50	12.5439	0.2509	0.005165
LGAgent	50	11.0496	0.2210	0.008444

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.030477598	3	0.01016	1.496779	0.216707	2.650677
Within Groups	1.330325092	196	0.00679			
Total	1.360802689	199				

Table 9-12 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 215.

Anova: Single Factor (QID: 218)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	3.0122	0.0602	0.002171
DeepQAgent	50	0.7789	0.0156	5.67E-05
MIMAgent	50	0.7789	0.0156	5.67E-05
LGAgent	50	1.6728	0.0335	0.001416

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.066833442	3	0.02228	24.07964	2.61E-13	2.650677
Within Groups	0.181333739	196	0.00093			
Total	0.248167181	199				

Table 9-13 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 218.

Anova: Single Factor (QID: 227)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	5.7820	0.1156	0.002757
DeepQAgent	50	6.0542	0.1211	0.002127
MIMAgent	50	6.0542	0.1211	0.002127
LGAgent	50	5.8487	0.1170	0.001609

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.001185457	3	0.0004	0.183358	0.90762	2.650677
Within Groups	0.422397758	196	0.00216			
Total	0.423583215	199				

Table 9-14 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 227.

Anova: Single Factor (QID: 232)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	9.8511	0.1970	0.00796
DeepQAgent	50	5.5553	0.1111	0.007579
MIMAgent	50	5.2856	0.1057	0.006849
LGAgent	50	5.2232	0.1045	0.005756

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.304515095	3	0.10151	14.42674	1.57E-08	2.650677
Within Groups	1.379035261	196	0.00704			
Total	1.683550356	199				

Table 9-15 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 232.

Anova: Single Factor (QID: 234)

SUMMARY

Groups	*Count*	*Sum*	*Average*	*Variance*
BasicAgent	50	2.3226	0.0465	0.000205
DeepQAgent	50	9.3194	0.1864	0.003124
MIMAgent	50	8.9849	0.1797	0.003412
LGAgent	50	9.5928	0.1919	0.002881

ANOVA

Source of Variation	*SS*	*df*	*MS*	*F*	*P-value*	*F crit*
Between Groups	0.733777139	3	0.24459	101.6831	9.91E-40	2.650677
Within Groups	0.471465642	196	0.00241			
Total	1.205242782	199				

Table 9-16 ANOVA results for BasicAgent, DeepQAgent, MIMAgent, and LGAgent using question ID 234.

ANOVA tests that resulted in a significant result at the p-value <.05 level, meaning that not all 4 agents had equal means, were then tested using Tukey

multiple comparison. This test is used to determine which of the agents were significantly different from the others. Questions 204, 211, 212, 214, 218, 232, and 234 were found to not have equal means by ANOVA. The complete results of the Tukey test can be found in Table 9-17.

Tukey multiple comparisons of means 95% family-wise confidence level Fit				
aov(formula = lm(value ~ variable, data = rdata))				
Pair-wise Comparison	**diff**	**lwr**	**upr**	**Adjusted P-Value**
204				
DeepQAgent-BasicAgent	-4.22E-02	-0.0711	-0.0132	0.0012299
MIMAgent-BasicAgent	-4.22E-02	-0.0711	-0.0132	0.0012299
LGAgent-BasicAgent	-5.46E-02	-0.0836	-0.0256	0.0000129
MIMAgent-DeepQAgent	-5.55E-17	-0.0290	0.0290	1
LGAgent-DeepQAgent	-1.24E-02	-0.0414	0.0165	0.6826629
LGAgent-MIMAgent	-1.24E-02	-0.0414	0.0165	0.6826629
211				
DeepQAgent-BasicAgent	-0.07583473	-0.0971	-0.0546	0
MIMAgent-BasicAgent	-0.082904	-0.1042	-0.0616	0
LGAgent-BasicAgent	-0.09860406	-0.1199	-0.0773	0
MIMAgent-DeepQAgent	-0.00706927	-0.0283	0.0142	0.8248632
LGAgent-DeepQAgent	-0.02276933	-0.0440	-0.0015	0.0307114
LGAgent-MIMAgent	-0.01570006	-0.0370	0.0056	0.2262059
212				
DeepQAgent-BasicAgent	1.35E-02	0.0026	0.0245	0.0085656
MIMAgent-BasicAgent	1.35E-02	0.0026	0.0245	0.0085656
LGAgent-BasicAgent	8.04E-03	-0.0029	0.0190	0.2311872
MIMAgent-DeepQAgent	1.39E-17	-0.0110	0.0110	1

LGAgent-DeepQAgent	-5.51E-03	-0.0165	0.0054	0.5624999
LGAgent-MIMAgent	-5.51E-03	-0.0165	0.0054	0.5624999
214				
DeepQAgent-BasicAgent	2.48E-02	-0.0006	0.0503	0.0591663
MIMAgent-BasicAgent	2.48E-02	-0.0006	0.0503	0.0591663
LGAgent-BasicAgent	-2.27E-02	-0.0482	0.0028	0.0994347
MIMAgent-DeepQAgent	2.78E-17	-0.0255	0.0255	1
LGAgent-DeepQAgent	-4.75E-02	-0.0730	-0.0220	0.0000159
LGAgent-MIMAgent	-4.75E-02	-0.0730	-0.0220	0.0000159
218				
DeepQAgent-BasicAgent	-4.47E-02	-0.0604	-0.0289	0
MIMAgent-BasicAgent	-4.47E-02	-0.0604	-0.0289	0
LGAgent-BasicAgent	-2.68E-02	-0.0426	-0.0110	0.0001026
MIMAgent-DeepQAgent	3.47E-18	-0.0158	0.0158	1
LGAgent-DeepQAgent	1.79E-02	0.0021	0.0336	0.0191799
LGAgent-MIMAgent	1.79E-02	0.0021	0.0336	0.0191799
232				
DeepQAgent-BasicAgent	-0.085916587	-0.1294	-0.0424	0.0000043
MIMAgent-BasicAgent	-0.091309932	-0.1348	-0.0478	0.0000009
LGAgent-BasicAgent	-0.092559525	-0.1360	-0.0491	0.0000006
MIMAgent-DeepQAgent	-0.005393344	-0.0489	0.0381	0.98848
LGAgent-DeepQAgent	-0.006642938	-0.0501	0.0368	0.978903
LGAgent-MIMAgent	-0.001249594	-0.0447	0.0422	0.9998513
234				
DeepQAgent-BasicAgent	0.139937421	0.1145	0.1654	0
MIMAgent-BasicAgent	0.133247255	0.1078	0.1587	0
LGAgent-BasicAgent	0.145404328	0.1200	0.1708	0
MIMAgent-DeepQAgent	-	-	0.0187	0.9038357

	0.006690166	0.0321		
LGAgent-DeepQAgent	-0.005466908	-0.0200	0.0309	0.9444567
LGAgent-MIMAgent	-0.012157073	-0.0133	0.0376	0.6026047

Table 9-17 Tukey's honest significance test was performed in R, which is a multiple comparison of means procedure. The test compares the means of each group to the mean of every other group. This test allows us to determine which agent or agents' means was different from the other.

9.5 Appendix E: Individual query precision vs. recall graphs

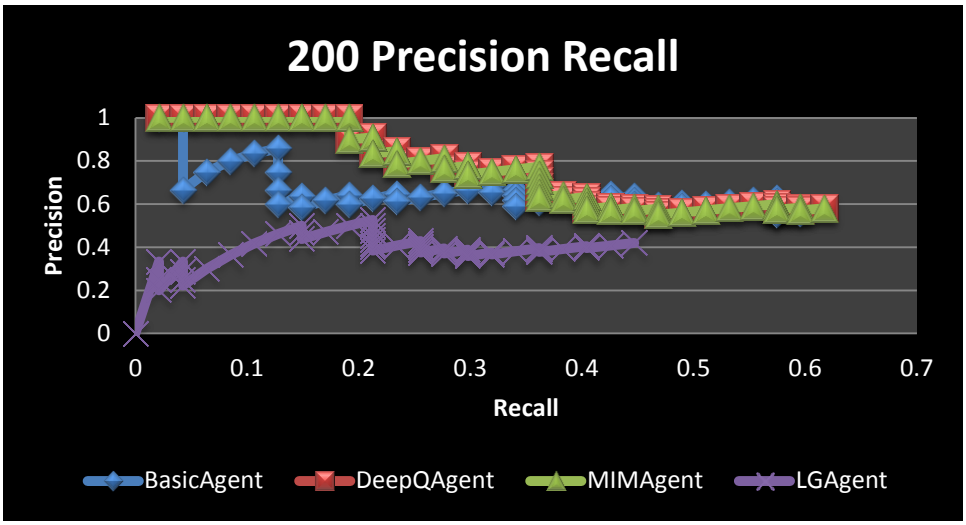


Figure 9-1 Precision-recall graph for query 200.

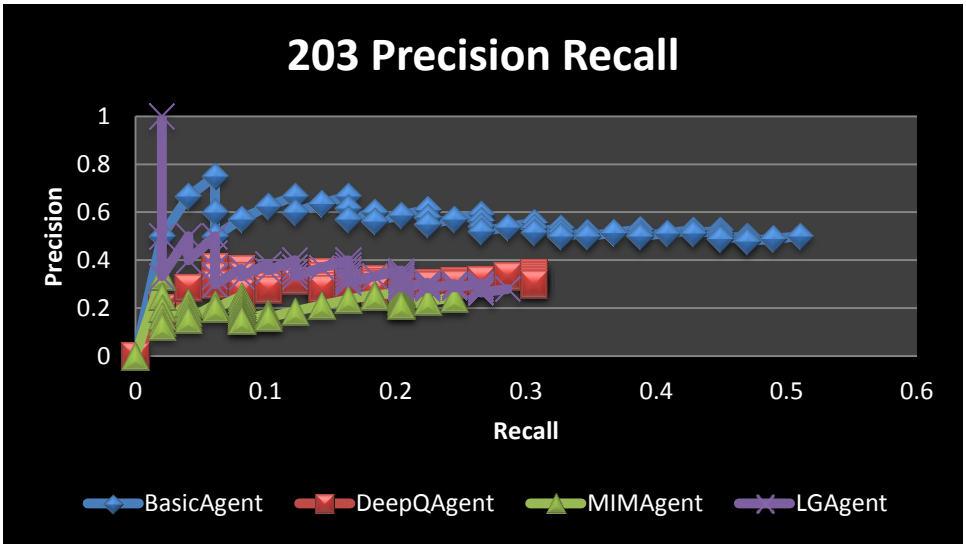


Figure 9-2 Precision-recall graph for query 203.

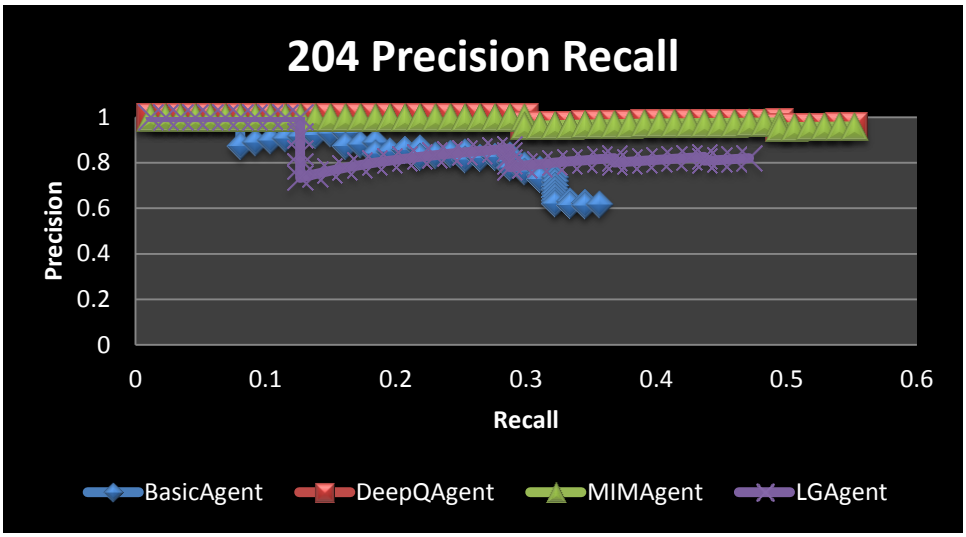


Figure 9-3 Precision-recall graph for query 204.

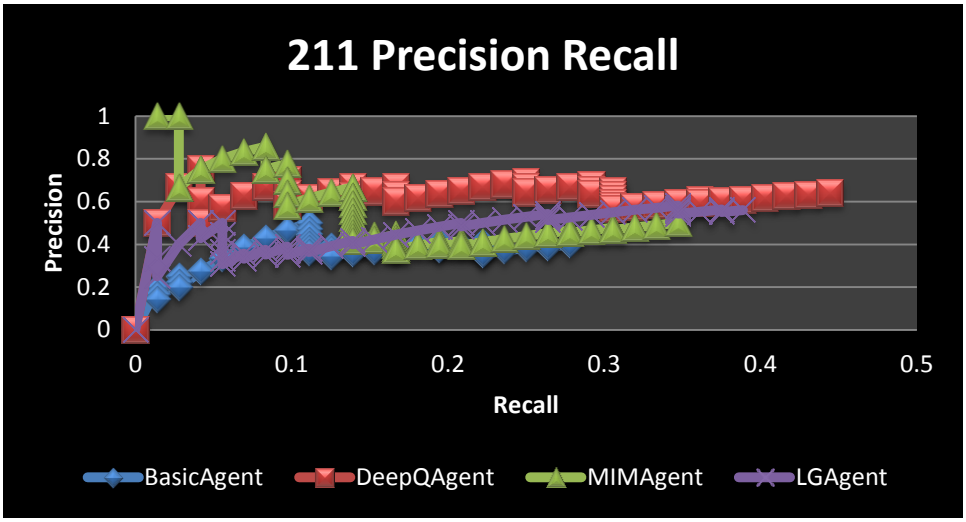


Figure 9-4 Precision-recall graph for query 211.

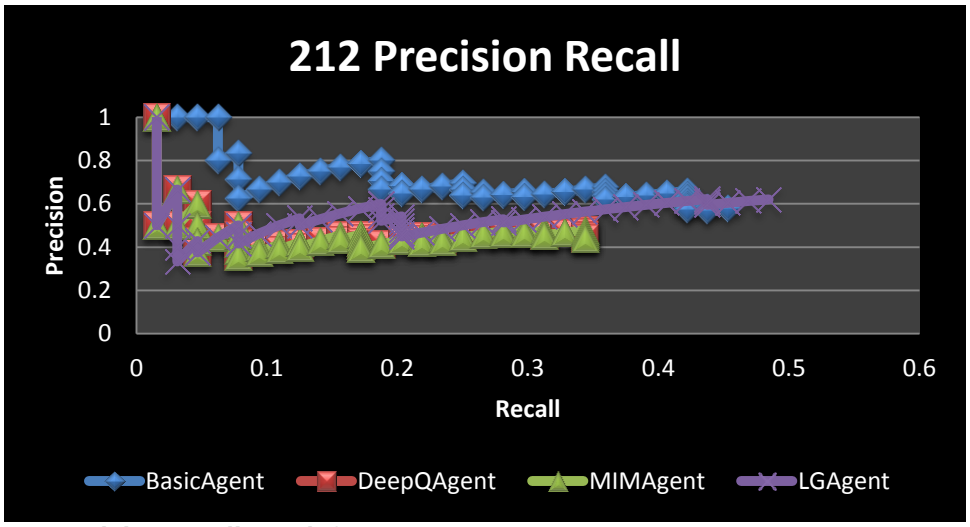


Figure 9-5 Precision-recall graph for query 212.

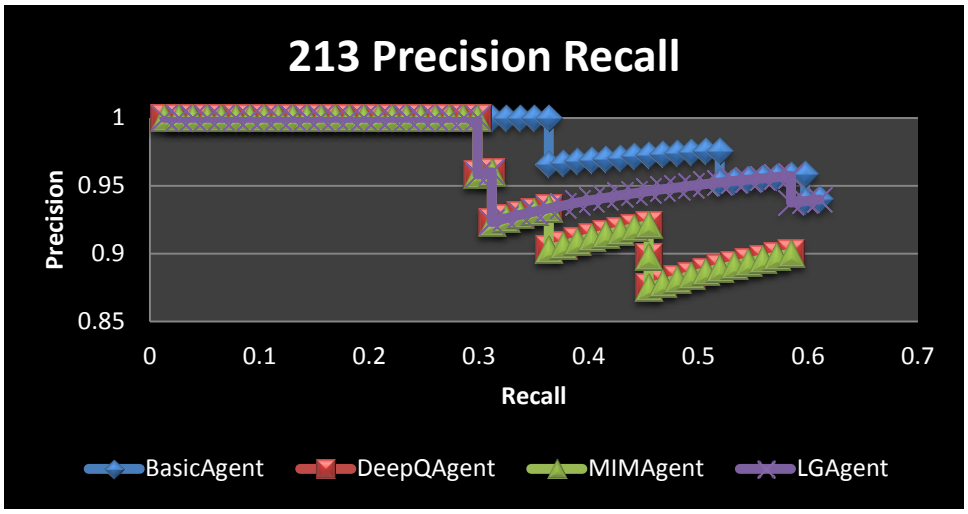


Figure 9-6 Precision-recall graph for query 213.

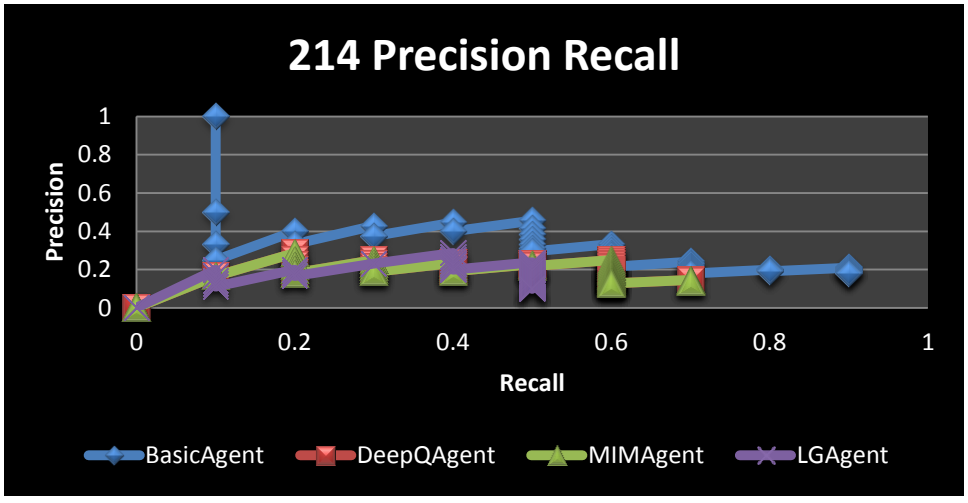


Figure 9-7 Precision-recall graph for query 214.

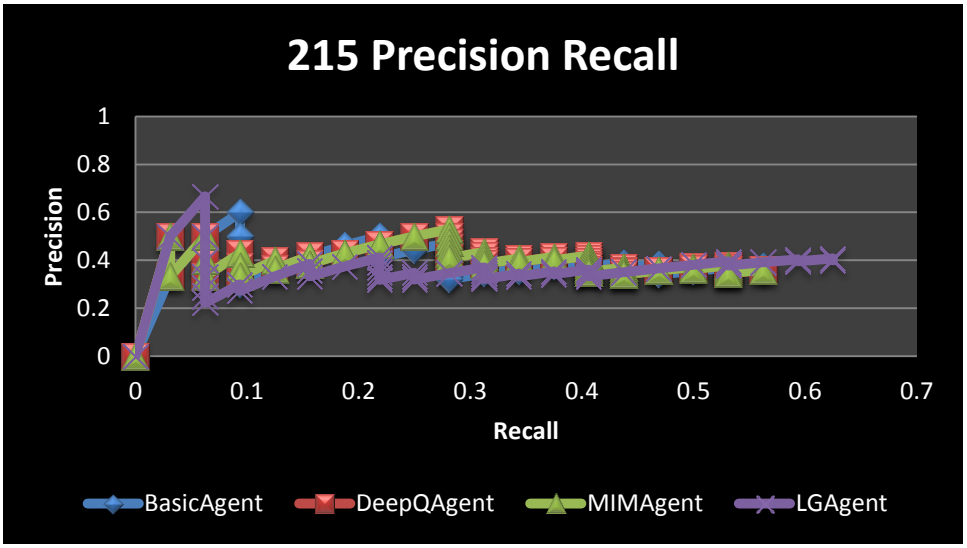


Figure 9-8 Precision-recall graph for query 215.

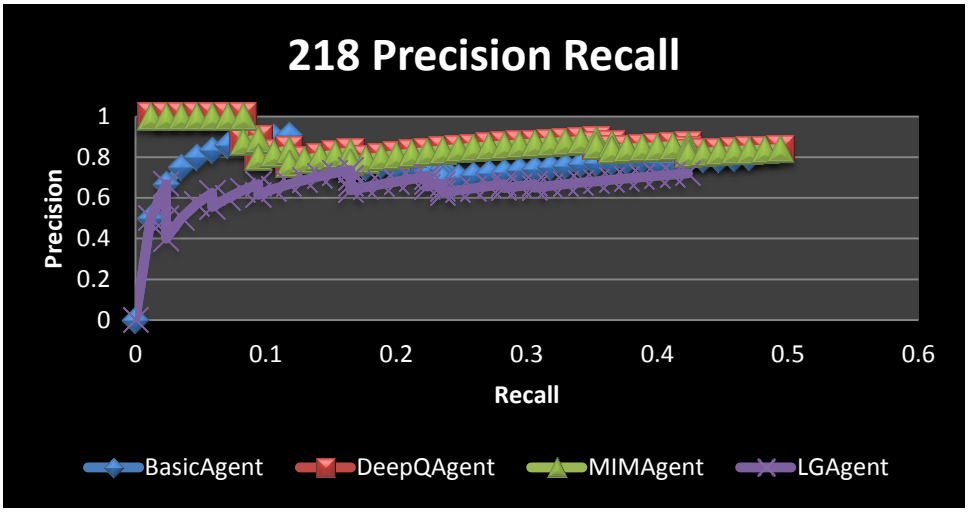


Figure 9-9 Precision-recall graph for query 218.

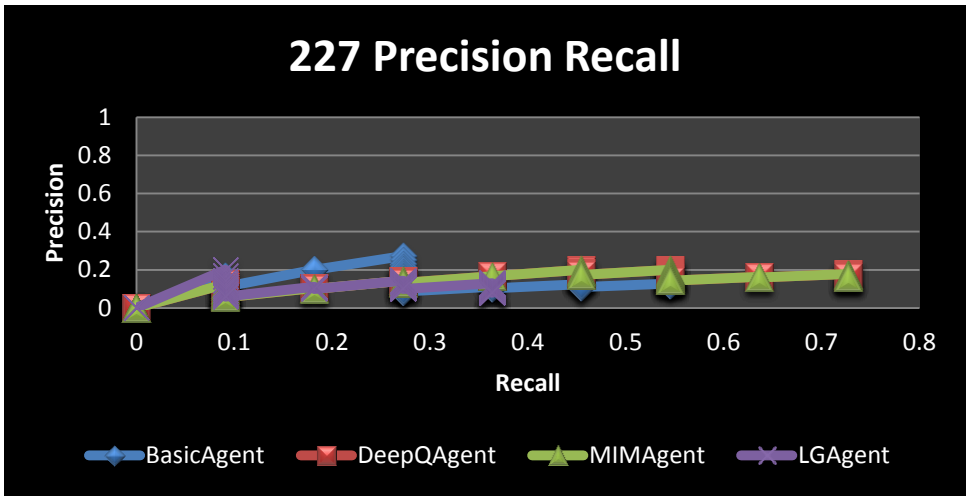


Figure 9-10 Precision-recall graph for query 227.

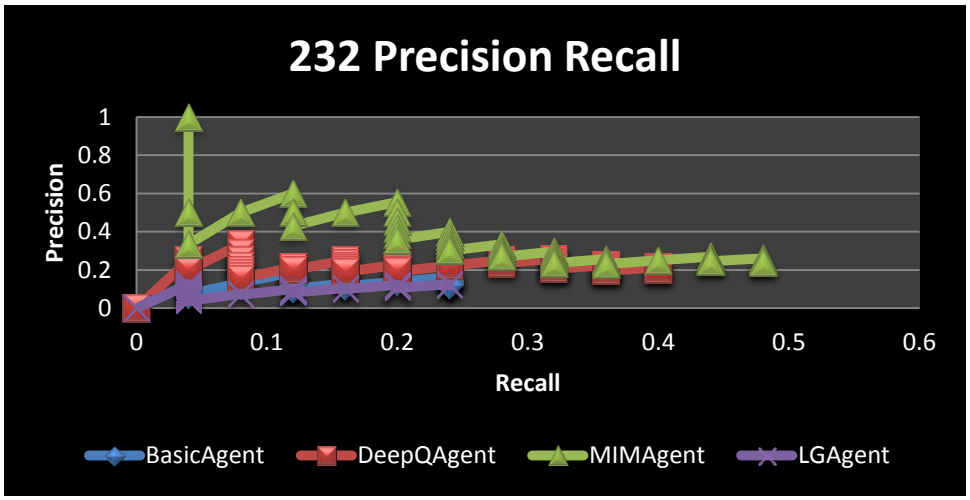


Figure 9-11 Precision-recall graph for query 232.

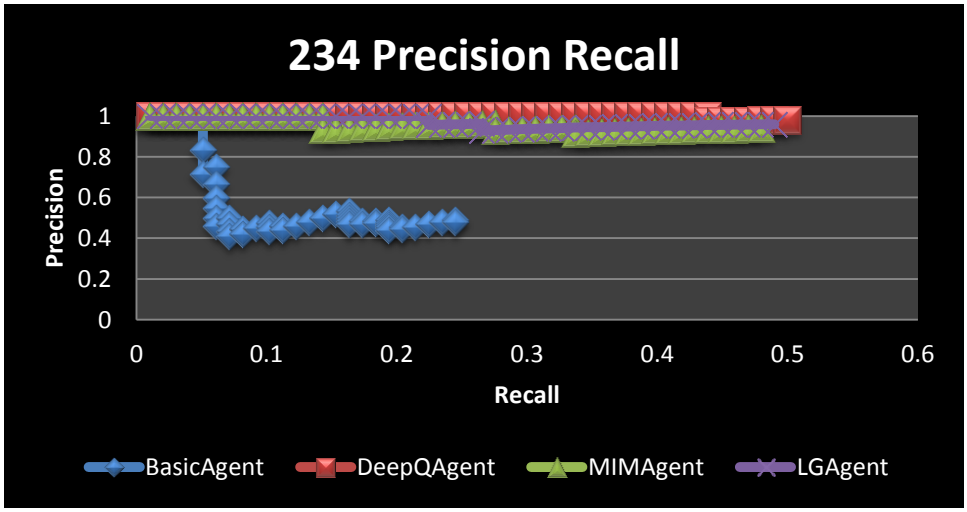


Figure 9-12 Precision-recall graph for query 234.

9.6 Appendix F: Calculation of the interpolated average precision

.	BasicQA												
Std Recall	200	203	204	211	212	113	214	215	218	227	232	234	Average
0	1.000	0.750	1.000	0.500	1.000	1.000	1.000	0.600	0.909	0.273	0.188	1.000	0.768
0.1	0.857	0.667	0.933	0.500	0.800	1.000	1.000	0.500	0.909	0.273	0.188	0.533	0.680
0.2	0.682	0.611	0.864	0.410	0.696	1.000	0.455	0.500	0.800	0.273	0.162	0.490	0.578
0.3	0.682	0.556	0.771	0.000	0.676	1.000	0.455	0.394	0.800	0.128	0.000	0.000	0.455
0.4	0.645	0.525	0.000	0.000	0.659	0.976	0.455	0.394	0.800	0.128	0.000	0.000	0.382
0.5	0.628	0.500	0.000	0.000	0.000	0.976	0.455	0.375	0.000	0.128	0.000	0.000	0.255
0.6	0.000	0.000	0.000	0.000	0.000	0.940	0.333	0.000	0.000	0.000	0.000	0.000	0.106
0.7	0.000	0.000	0.000	0.000	0.000	0.000	0.241	0.000	0.000	0.000	0.000	0.000	0.020
0.8	0.000	0.000	0.000	0.000	0.000	0.000	0.209	0.000	0.000	0.000	0.000	0.000	0.017
0.9	0.000	0.000	0.000	0.000	0.000	0.000	0.209	0.000	0.000	0.000	0.000	0.000	0.017
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9-18 Calculation of the eleven-point interpolated average precision for the BasicQAgent.

	DeepQAgent												
Std Recall	200	203	204	211	212	113	214	215	218	227	232	234	Average
0	1.000	0.375	1.000	0.750	1.000	1.000	0.286	0.529	1.000	0.200	0.333	1.000	0.706
0.1	1.000	0.350	1.000	0.692	0.478	1.000	0.286	0.529	0.882	0.200	0.258	1.000	0.640
0.2	0.909	0.349	1.000	0.692	0.478	1.000	0.286	0.529	0.882	0.200	0.258	1.000	0.632
0.3	0.773	0.349	0.977	0.647	0.478	0.960	0.250	0.435	0.882	0.200	0.258	1.000	0.601
0.4	0.633	0.000	0.977	0.640	0.000	0.921	0.250	0.419	0.857	0.200	0.213	1.000	0.509
0.5	0.600	0.000	0.960	0.000	0.000	0.900	0.250	0.378	0.000	0.200	0.000	0.980	0.356
0.6	0.580	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.000	0.178	0.000	0.000	0.084
0.7	0.000	0.000	0.000	0.000	0.000	0.000	0.146	0.000	0.000	0.178	0.000	0.000	0.027
0.8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9-19 Table 9 18 Calculation of the eleven-point interpolated average precision for the DeepQAgent.

	MIMAgent												
Sd Recall	200	203	204	211	212	113	214	215	218	227	232	234	Average
0	1.000	0.333	1.000	1.000	1.000	1.000	0.286	0.529	1.000	0.200	1.000	1.000	0.779
0.1	1.000	0.270	1.000	0.667	0.478	1.000	0.286	0.529	0.882	0.143	0.600	1.000	0.655
0.2	0.909	0.270	1.000	0.500	0.478	1.000	0.286	0.529	0.882	0.143	0.556	0.964	0.626
0.3	0.773	0.000	0.977	0.500	0.478	0.960	0.250	0.435	0.882	0.129	0.296	0.943	0.552
0.4	0.633	0.000	0.977	0.000	0.000	0.921	0.250	0.419	0.857	0.000	0.268	0.940	0.439
0.5	0.600	0.000	0.960	0.000	0.000	0.900	0.250	0.378	0.000	0.000	0.000	0.000	0.257
0.6	0.580	0.000	0.000	0.000	0.000	0.000	0.250	0.000	0.000	0.000	0.000	0.000	0.069
0.7	0.000	0.000	0.000	0.000	0.000	0.000	0.146	0.000	0.000	0.000	0.000	0.000	0.012
0.8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9-20 Table 9 18 Calculation of the eleven-point interpolated average precision for the MIMAgent.

	LGAgent												
Std Recall	200	203	204	211	212	113	214	215	218	227	232	234	Average
0	0.526	1.000	1.000	0.581	1.000	1.000	0.286	0.667	0.737	0.200	0.125	1.000	0.677
0.1	0.526	0.400	1.000	0.581	0.622	1.000	0.286	0.412	0.737	0.143	0.125	1.000	0.569
0.2	0.526	0.357	0.862	0.581	0.622	1.000	0.286	0.412	0.720	0.143	0.125	1.000	0.553
0.3	0.420	0.000	0.826	0.581	0.622	0.960	0.286	0.408	0.720	0.129	0.000	0.960	0.493
0.4	0.420	0.000	0.826	0.000	0.622	0.957	0.286	0.408	0.720	0.000	0.000	0.960	0.433
0.5	0.000	0.000	0.000	0.000	0.000	0.957	0.238	0.408	0.000	0.000	0.000	0.000	0.134
0.6	0.000	0.000	0.000	0.000	0.000	0.940	0.000	0.408	0.000	0.000	0.000	0.000	0.112
0.7	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.8	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
0.9	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table 9-21 Table 9 18 Calculation of the eleven-point interpolated average precision for the LGAgent.

9.7 Appendix G: Average Precision, MAP, GMAP, and F-Measure

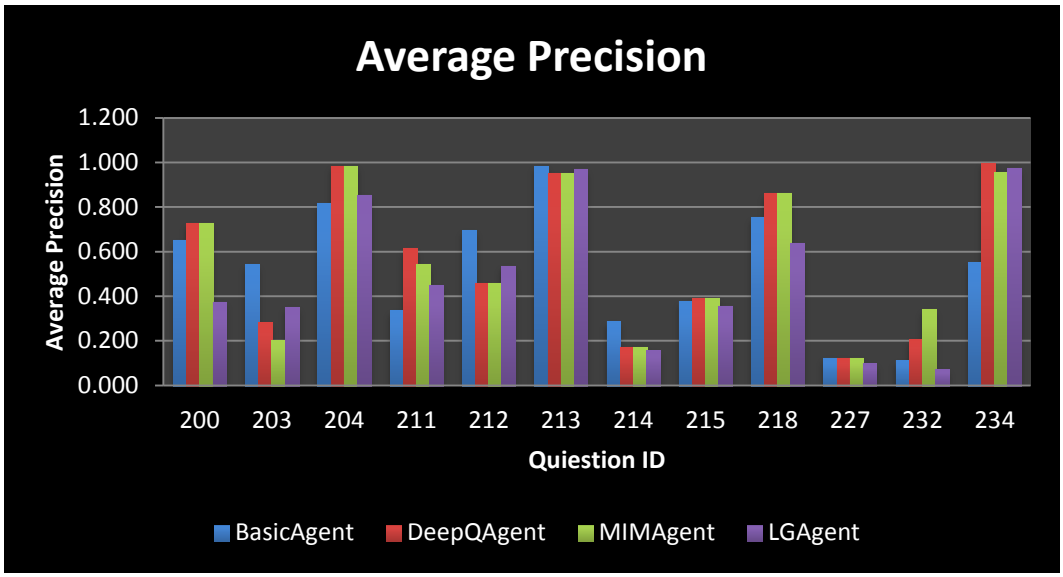


Figure 9-13 Average precision for the agents for each of the queries under investigation.

	BasicAgent		DeepQAgent		MIMAgent		LGAgent	
QID	**AP**	**log-AP**	**AP**	**log-AP**	**AP**	**log-AP**	**AP**	**log-AP**
200	0.651	-0.186	0.727	-0.139	0.727	-0.139	0.375	-0.426
203	0.541	-0.267	0.280	-0.552	0.200	-0.698	0.352	-0.454
204	0.818	-0.087	0.985	-0.007	0.985	-0.007	0.854	-0.068
211	0.335	-0.475	0.615	-0.211	0.542	-0.266	0.448	-0.349
212	0.694	-0.158	0.459	-0.339	0.459	-0.339	0.534	-0.272
213	0.984	-0.007	0.951	-0.022	0.951	-0.022	0.970	-0.013
214	0.287	-0.542	0.171	-0.767	0.171	-0.767	0.154	-0.812
215	0.377	-0.424	0.391	-0.407	0.391	-0.407	0.356	-0.449
218	0.754	-0.123	0.859	-0.066	0.859	-0.066	0.636	-0.196
227	0.123	-0.911	0.123	-0.911	0.123	-0.911	0.098	-1.010
232	0.110	-0.957	0.206	-0.685	0.340	-0.468	0.073	-1.135
234	0.549	-0.261	0.997	-0.001	0.957	-0.019	0.972	-0.012

Table 9-22 Average precision calculated value and the log of the average precision used in the MAP and GMAP calculations.

MAP			
BasicAgent	**DeepQAgent**	**MIMAgent**	**LGAgent**
0.518523889	0.563594839	0.55867549	0.48509

Table 9-23 Values calculated from the Mean Average Precision.

GMAP			
BasicAgent	**DeepQAgent**	**MIMAgent**	**LGAgent**
0.693117572	0.710124763	0.710051767	0.648444

Table 9-24 Values calculated for the Geometric Mean Average Precision.

Query	BasicAgent	DeepQAgent	MIMAgent	LGAgent
200	0.577319588	0.597938144	0.597938144	0.432989691
203	0.505050505	0.303030303	0.242424242	0.282828283
204	0.452554745	0.700729927	0.700729927	0.598540146
211	0.327868852	0.524590164	0.409836066	0.459016393
212	0.50877193	0.385964912	0.385964912	0.543859649
213	0.74015748	0.708661417	0.708661417	0.74015748
214	0.3	0.233333333	0.233333333	0.166666667
215	0.43902439	0.43902439	0.43902439	0.487804878
218	0.592592593	0.622222222	0.622222222	0.533333333
227	0.196721311	0.262295082	0.262295082	0.131147541
232	0.16	0.266666667	0.32	0.16
234	0.324324324	0.662162162	0.635135135	0.648648649
Average	**0.427032143**	**0.47555156**	**0.463130406**	**0.432082726**

Table 9-25 Values F-measure values and the average value for the 12 queries.

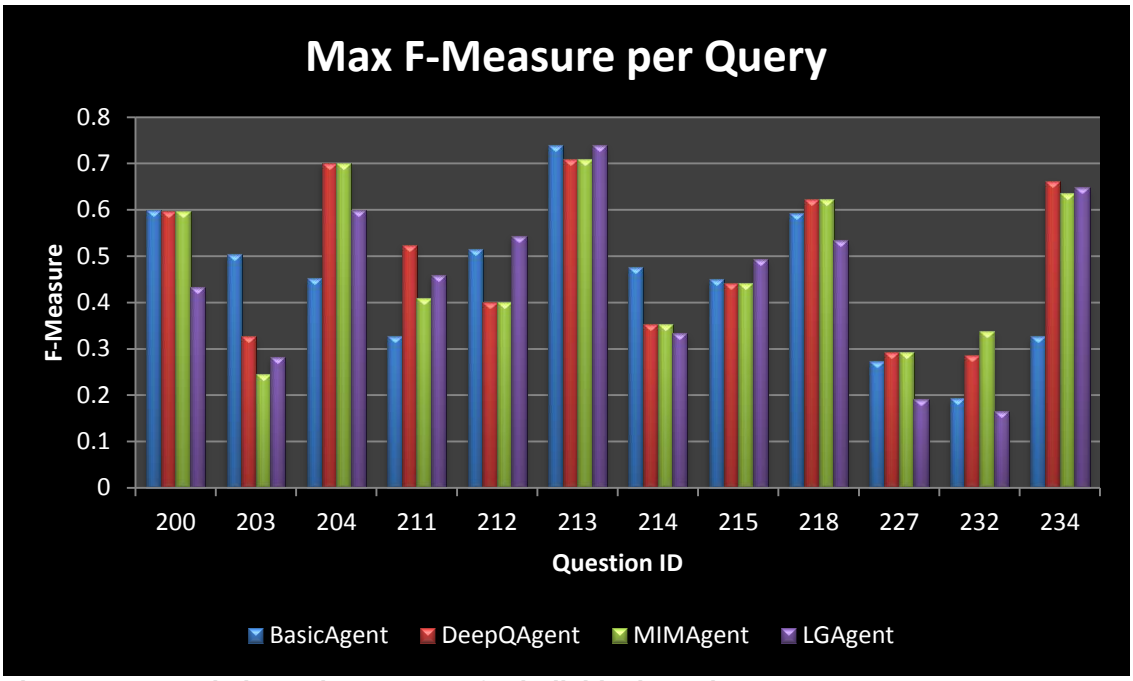


Figure 9-14 Max balanced F-measure for individual queries.