Semantic Annotation and Summarization of Biomedical Literature

A Dissertation Proposal

Submitted to the Faculty

of

Drexel University

by

Lawrence Harold Reeve, Jr.

in partial fulfillment of the

requirements for the degree

of

Doctor of Philosophy

December 2006

Table Of Contents

List of Tables

List of Figures

Abstract

Semantic Annotation and Summarization of Biomedical Literature

Lawrence Harold Reeve, Jr.

Hyoil Han, Ph.D.

Advancements in the biomedical community are largely documented and published in text format in scientific forums such as conferences and journals. To address the scalability of utilizing the large volume of text-based information generated by continuing advances in the biomedical field, two complementary areas are studied. The first area is Semantic Annotation, which is a method for providing machine-understandable information based on domain-specific resources. A novel system architecture is proposed for online matching of concepts defined by a biomedical metathesaurus using a multi-level filter based on both information retrieval and shallow natural language processing techniques to obtain the best term coverage as well as best phrase coherence. The proposed annotator will be evaluated against a state-of-the-art biomedical annotator using the performance measures of time (e.g. number of milliseconds per noun phrase) and precision/recall of the resulting concept matches. The goal is to show that annotation can be performed online, rather than offline, without a significant loss of precision and recall as compared to current offline systems. The second area is Text Summarization, which is a data reduction method. The proposed work is unique in that it focuses exclusively on summarizing biomedical full-text sources, and also exclusively uses domain-specific concepts, rather than terms, to identify important information within a text. Two novel approaches are proposed: one using a concept chaining method based on existing work in lexical chaining, and the other using concept distribution to match important sentences between a source text and a generated

summary. The proposed summarizers are evaluated using the ROUGE summary evaluation tool developed at the University of Southern California. ROUGE compares n-gram co-occurrences between a system summary and one or more model summaries. In this work, the model summaries are produced by three Drexel University College of Medicine students in their final year of study using sentence extraction from the source text. The proposed summarizers are compared to existing summarization systems using the ROUGE tool and the model summaries. Partial results for biomedical text summarization show that the proposed approaches outperform the existing approaches.

1. INTRODUCTION

The output of biomedical research is largely documented as findings in the form of literature written in free-form text format (Nenadic, Mima, Spasic, Ananiadou, & Tsujii, 2002). The written texts are then accumulated in large online databases which have been made readily accessible due to recent advances in software and communications. For example, the PUBMED database provided by the United States National Library of Medicine contains over 14 million citations from over 4,800 journals (United States National Library of Medicine, 2005c). The United States National Institutes of Health clinical trials database contains information on over 13,500 clinical trials (United States National Library of Medicine, 2005a). To use such resources, practicing physicians and biomedical researchers are faced with the task of locating, reading and evaluating relevant biomedical literature. For example, oncologists must find the clinical trial information related to their cancer specialty, evaluate the study for its strength, and then possibly incorporate the new study information into their patient treatment efforts (Brooks & Sulimanoff, 2002), (Jaques, 2002). The large number of clinical trials conducted and the data produced by them makes the information assimilation process time consuming. The proposed research aims to build novel approaches to semantic annotation and text summarization and is an effort to help reduce the time to assimilate the data resulting from large collections of literature in the biomedical domain by reducing the amount of data which must be manually processed.

There are two complementary components to the proposed research: Semantic Annotation and Text Summarization. Semantic Annotation, sometimes called concept matching in the biomedical literature, is the process of mapping phrases within a source

text to distinct concepts defined by domain experts. Text Summarization is a method for reducing the amount of text data while retaining the key ideas of a source text. The proposed research will construct a system for producing summaries of biomedical text using biomedical concepts as the unit for identifying key information. Such a system is expected to allow physicians and researchers to quickly review biomedical documents without requiring a reading of the full source text. The biomedical text used for the proposed research is mainly from randomized controlled trials in oncology.

The first component is Semantic Annotation, which takes a source text and identifies pre-specified concepts within the source text. A key issue with semantic annotation is the variability of human language, which makes the mapping process non-trivial. For example, the biomedical concept *Lung Cancer* has many possible expressions, such as *Cancer of the Lung* and *Pulmonary Carcinoma*. While much research has been done in biomedical semantic annotation, its use is largely designed for indexing documents based on the concepts identified in the text. Such systems are designed to be used in an offline environment, where speed is not critical. In the proposed research, the discovered concept output is directed into a summarizer stage, and both the semantic annotation and resulting summary are envisioned to be used in an online environment, where expected response times may be lower than in an offline system.

The second component of the research is Text Summarization, where the use of domain-specific concepts to find important information within a text is examined. The use of domain-specific concepts is hypothesized to provide a better method for identifying important textual information than the use of terms, which have not been annotated for meaning. Text summarization can be used as a way to reduce the amount of

data presented to readers. The goal of text summarization is to present a subset of the source text which contains the most important points within a source text with minimal redundancy. The summary can then be used by the reader to: (a) make a determination if the source text should be read in its entirety, or (b) act as a surrogate for the source text to obtain information without reading the entire source text. The use of text summarization allows a user to get a sense of the content of a source text, or to know its information content, without reading all sentences within the source text. The reduction of data afforded by text summarization increases scale by (a) allowing users to find relevant source texts more quickly, and (b) assimilating only essential information from many texts with reduced effort. Much work has already been done by the text summarization community, largely dealing in the general domain with genres such as news. While some research work has been done for domain-specific summarization in domains such as legal and medical, the work is incomplete. For example, the use of domain-specific resources in text summarization, such as vocabularies, ontologies and so forth, is largely ignored.

Although the proposed research is focused on the biomedical domain, the resulting system is expected to be useful within other domains which have domain-specific resources available.

The remainder of this chapter identifies the contributions expected of the proposed research and the preliminary work done, as well as explaining the organization of this dissertation proposal.

1.1 Contributions

The primary contributions expected to be produced from the proposed research are as follows:

1. Develop a dictionary-based biomedical annotator for annotating biomedical texts which can be used in an online environment.
2. Develop a single document text summarizer which uses domain concepts to identify relevant sentences within a source to produce an extractive summary.
3. The annotator and the summarizer are expected to be useful within other domains (such as law) which have domain specific resources available, such as thesauri and ontologies.
4. Explore the characteristics of biomedical texts, such as sentence length, concept distribution and summary size, and exploit these characteristics to improve summarization performance.
5. Generate a corpus of biomedical texts which has been generated by three domain experts, and use the corpus to evaluate summarization performance.

1.2 Preliminary Work

Most of the preliminary work for the proposed research completed to-date has been done on the Text Summarization component. The preliminary work on Text Summarization focuses on exploring two different methods for using biomedical concepts to identify important areas within a text which should be extracted as part of a summary. The first approach, concept chaining, uses ideas from research done in lexical chaining and applies the ideas to domain concepts. Existing work in lexical chaining

finds term relationships to identify general concepts from a thesaurus, and then determines which concepts are strongest. The concept chain approach uses concepts discovered through a biomedical annotator to determine which biomedical concepts are strongest. A summarizer built on concept chaining has been implemented and evaluated (L. Reeve, Han, & Brooks, 2006a), (L. Reeve, Han, & Brooks, 2006b). A second approach to text summarization using concepts is based on the frequency distribution of concepts within a source text and its summary. The idea is that the frequency distribution of concepts within a summary should match the frequency distribution of concepts within the source text. A summarizer based on this idea and its evaluation has been implemented and evaluated (L. Reeve et al., 2006).

The work completed to-date for the proposed research on the Semantic Annotation component has largely been a review of current efforts in the general domain (L. Reeve & Han, 2005), (L. Reeve & Han, 2006). Additional work exploring annotation in the biomedical domain has been started and elaborated (see Section 5.1), but has not been published.

1.3 Dissertation Proposal Organization

This dissertation proposal is organized into eight chapters. The first chapter, this chapter, is an introductory chapter. It describes the expected contributions of the work and an overview of the preliminary work completed to date. Chapter 2 provides motivation for performing semantic annotation and text summarization, and how the two can be linked together. Chapter 3 is background information necessary for the reader to understand the rest of the proposal. Biomedical domain concepts are explained, as well as

an overview of extractive text summarization, which is the method used by the proposed summarizers. In addition, the acquisition of an evaluation corpus consisting of biomedical texts, consisting of randomized controlled trials, is explained. Chapter 4 lays out research problems expected to be addressed by the proposed research. The problem statements are drawn from both the Semantic Annotation and Text Summarization components. Chapter 5 describes the approaches to produce a semantic annotation and text summarization system for the biomedical domain, and how the system components will be evaluated. Chapter 6 provides a literature review, broken down into two major areas. The first area is semantic annotation, and it is reviewed in the large (i.e., not domain specific) as well as in the biomedical domain. The second area is text summarization, which has a rich history extending back approximately 60 years. Chapter 7 shows the planned research schedule, and Chapter 8 lists publications related to this research done as part of the preliminary research.

2. MOTIVATION

This chapter motivates the ideas behind the two main components of this research: Semantic Annotation and Text Summarization. In semantic annotation, a domain-specific thesaurus can be used to find concepts within free-form texts. A thesaurus is organized by concept, with one or more synonymous words and/or phrases expressing the concept. Using the thesaurus concepts, words take on meaning rather than being a surface feature. A benefit of semantic annotation in the biomedical domain is merging different ways of expressing the same concept, possibly using different words, into a single concept. This allows for identifying important topics within a document. Text summarization is a data reduction process. A benefit of text summarization is that key ideas can be expressed using a fewer sentences than the source text contains, which allows for the possibility of faster assimilation of content. An example is shown in this chapter how just cancer treatment results can be extracted from a larger randomized controlled trial text. It is sometimes the case an author's abstract is available as a summary of the paper. For example, in the biomedical domain, published clinical trial results usually have an abstract to supplement the full-text. In cases where an abstract is available, any system-generated summary competes with the author's abstract. While system-generated summaries may seem like a duplication of effort, each user is likely to have a unique information need, different background, different motivations, and so forth, and a single summary, even the author's own summary, does not address this issue.

2.1 Semantic Annotation

Semantic annotation is the process of identifying pre-defined concepts and entities within a text. One or more domain-specific resources are used to annotate the text with

the concepts and entities found. For example, Figure 1 shows the annotation of a biomedical sentence using the Unified Medical Language System (UMLS) (United States National Library of Medicine, 2005d). The sentence phrases are shown as text within rectangles, while the corresponding UMLS concept which the text maps to is shown as text contained in ovals. Directed lines indicate a relationship between concepts. Undirected lines indicate a link between a concept instance and a concept. In Figure 1, concept {*Myeloma*} is related to concepts {*Progessive, Cancer*, and *Haematologic Disease*}. Further, the concept {*Haematologic Disease*} is related to {*Plasma Cell*} in addition to {*Myeloma*}. From this relationship graph, it can be seen that the concept {*Myeloma*} is indirectly related to the concept {*Plasma Cell*}. The use of Semantic Annotation allows for such relationships to be discovered in text, and so is more powerful than the use of surface terms alone.

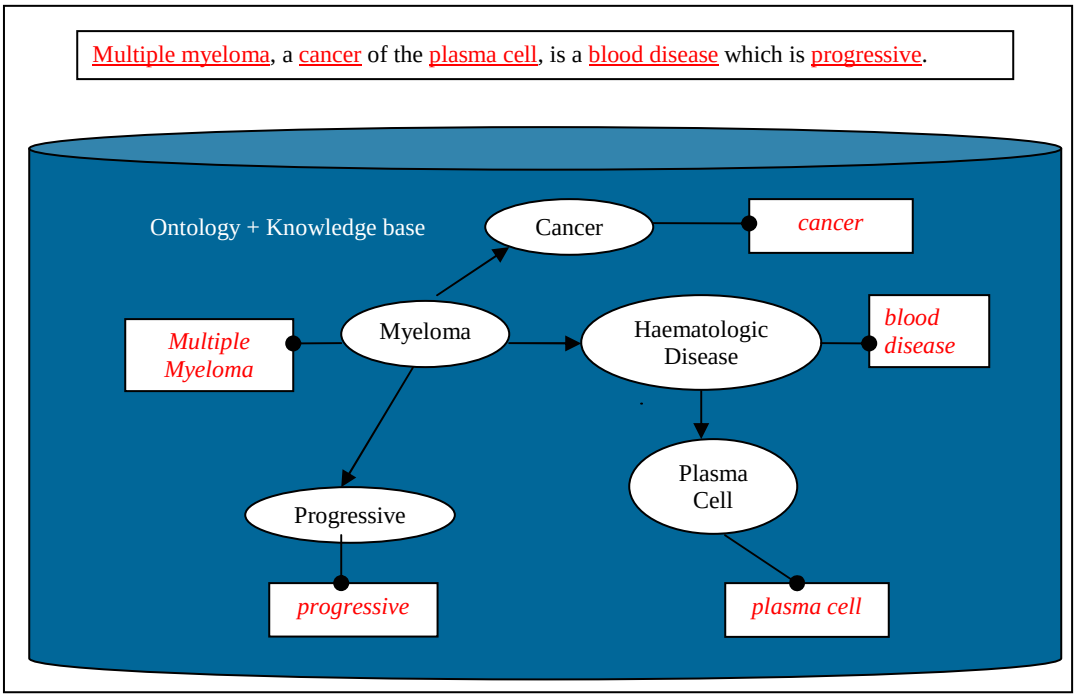

Figure 1: A graphical view of semantic annotation of a sentence. Ovals represent biomedical concepts and rectangles represent instances of the biomedical concepts from the sentence. Directed lines indicate relationships between the concepts, while undirected lines link a concept instance to a concept.

Once the text has been annotated with concepts, the sentence phrases have meaning which is more easily processed by a machine than raw text alone. There are two expected advantages of using biomedical concept annotations, rather than raw biomedical text, in this research: (a) synonym merging, and (b) semantic filtering.

Synonym merging is the process of using synonyms identified by various medical vocabularies and collapsing them into a single concept. This is important for identifying salient information for text summarization. If only raw text were used, different expressions of the same concept would be identified as distinct entities. For example, the phrase *heart attack* can be expressed in several ways, such as {*heart attack, coronary attack, myocardial infarction*}. If raw text is used and the author used multiple expressions, these three phrases would be considered distinct. Since reiteration is a strong component of identifying important information within text (Sparck Jones, 1999), it is better to collapse these three synonymous phrases to a single concept, such as {Myocardial Infarction}, to help identify reiteration and thus the most important parts of the text as expressed by an author.

The second major use of annotations in the proposed research is to use them to filter out concepts which are not important to a user of a summarization system. Semantic filtering allows customizing a summary for a user's information need. For example, an experienced physician may not need much background information on a clinical trial, and instead want to focus more heavily on the results and methodology of the clinical trial. In this case, the physician can customize the summary to filter out qualitative concepts, which express primarily opinion and background information. The user can be presented a list of concepts which are not important, as well as a list of concepts which are

important. One example filtering method is concept weighting. In concept weighting, the unimportant concepts selected by the user are negatively weighted, while the important concepts selected by the user are given increased weight over neutral or unimportant concepts.

2.2 Text Summarization

While the result of the proposed work could be used to summarize abstracts, the goal is not to summarize abstracts (summarizing the summary), but instead to use the full source text to produce a summary. There are several reasons for wanting to generate text summaries from a full-text source even in the presence of the author's abstract:

1. There exists no ideal summary. An ideal summary is dependent on each user, including factors such as information need and domain background. An author's abstract is one view of an ideal summary, but users may want alternative summaries.

2. The abstract may be missing content from the full-text (Cohen & Hersh, 2005).

3. Customized summaries can be useful in question-answering systems where they provide personalized information. Such summaries have been described in the literature as *user-focused* (Mani & Bloedorn, 1998) and *query-relevant* (Carbonell & Goldstein, 1998) summaries.

4. The use of automatic or semi-automatic summary generation by commercial abstract services may allow them to scale the number of published texts they can evaluate.

5. The generation and evaluation of summaries allows for evaluation of sentence selection methods that may be useful for use in multi-document summarization. The idea is that if sentence selection methods do not work well for single-document summarization, it is unlikely they will identify important data across multiple documents.

Figures 2 and 3 below motivate the need for text summarization even in the presence of the author's summary (the paper's abstract). Figure 2 shows an abstract from

a biomedical text. While the text itself is relatively short (as compared to the original source text), if the reader wishes to quickly know the outcome of the research described in the abstract, the abstract must be read partially or completely, or skimmed at the very least, to find the information. In contrast, Figure 3 shows a summary which automatically identified the result information and displayed it.

Adjuvant Chemotherapy for Adult Soft Tissue Sarcomas of the Extremities and Girdles: Results of the Italian Randomized Cooperative Trial.

Adjuvant chemotherapy for soft tissue sarcoma is controversial because previous trials reported conflicting results. The present study was designed with restricted selection criteria and high dose-intensities of the two most active chemotherapeutic agents.

Patients and Methods: Patients between 18 and 65 years of age with grade 3 to 4 spindle-cell sarcomas (primary diameter >= 5 cm or any size recurrent tumor) in extremities or girdles were eligible. Stratification was by primary versus recurrent tumors and by tumor diameter greater than or equal to 10 cm versus less than 10 cm. One hundred four patients were randomized, 51 to the control group and 53 to the treatment group (five cycles of 4'-epidoxorubicin 60 mg/m2 days 1 and 2 and ifosfamide 1.8 g/m2 days 1 through 5, with hydration, mesna, and granulocyte colony-stimulating factor).

Results: After a median follow-up of 59 months, 60 patients had relapsed and 48 died (28 and 20 in the treatment arm and 32 and 28 in the control arm, respectively). The median disease-free survival (DFS) was 48 months in the treatment group and 16 months in the control group (P = .04); and the median overall survival (OS) was 75 months for treated and 46 months for untreated patients (P = .03). For OS, the absolute benefit deriving from chemotherapy was 13% at 2 years and increased to 19% at 4 years (P = .04).

Conclusion: Intensified adjuvant chemotherapy had a positive impact on the DFS and OS of patients with high risk extremity soft tissue sarcomas at a median follow-up of 59 months. Therefore, our data favor an intensified treatment in similar cases. Although cure is still difficult to achieve, a significant delay in death is worthwhile, also considering the short duration of treatment and the absence of toxic deaths.

Figure 2: Sample biomedical document abstract

The median disease-free survival (DFS) was 48 months in the treatment group and 16 months in the control group (P = .04); and the median overall survival (OS) was 75 months for treated and 46 months for untreated patients (P = .03).

Figure 3: Summarized version of abstract shown in Figure 2.

3. BACKGROUND

3.1 Biomedical Domain Concepts

One way to provide meaning to biomedical documents is by creating ontologies, and then linking information within each document to specifications contained in the ontology using a markup language (Berners-Lee, Hendler, & Lassila, 2001). Ontologies are conceptualizations of a domain that typically are represented using domain vocabulary (Chandrasekaran, Josephson, & Benjamins, 1999). Automated semantic annotation is the process of mapping instance data to an ontology (S. Handschuh, Staab, & Volz, 2003) , (L. Reeve & Han, 2006). The resulting annotations from the semantic annotation processing are what provide the link between information stored within a document and the ontology (Berners-Lee et al., 2001). In this work, the annotations are then used to identify important areas of a text, such as phrases, sentences, paragraphs, and sections useful for generating a text summary. In the biomedical domain, the National Library of Medicine (http://www.nlm.nih.gov/) provides resources for identifying concepts and their relationships under the framework of the Unified Medical Language System (UMLS) (United States National Library of Medicine, 2005d). UMLS contains many sub-components, but we three are used for the proposed research: Metathesaurus, Semantic Network and MetaMap Transfer.

The UMLS Metathesaurus contains concepts and real-world instances of the concepts, including a concept name and its synonyms, lexical variants, and translations (United States National Library of Medicine, 2004a). The Metathesaurus is derived from over 100 different vocabulary sources resulting in over one million biomedical concepts.

Table 1 shows the example concept *Multiple Myeloma* taken from the Metathesaurus, and displays several of the concept instances (i.e., synonymous words and phrases) associated with the concept. The instances are derived from the vocabulary sources. The key idea is that a single concept may have multiple ways of being expressed (instances). The Metathesaurus organizes the concept instances.

Table 1: UMLS concept and its concept instances

Concept Name	Concept Instances
Multiple Myeloma	Multiple Myeloma
	Myeloma
	Plasma Cell Myeloma
	Myelomatosis
	Plasmacytic myeloma

The UMLS Semantic Network organizes the Metathesaurus concepts into categories called semantic types (United States National Library of Medicine, 2004b). There are currently 135 semantic types.

The MetaMap Transfer (MetaMap) application (United States National Library of Medicine, 2005b) maps biomedical text to concepts stored in the Metathesaurus as follows. The text-to-concept mapping in the MetaMap application is done through a natural language processing approach. Sentences are first identified, and then noun phrases are extracted from each sentence. MetaMap proceeds through several stages to map a noun phrase to one or more concepts. Term variants of the phrase are generated, candidate concepts are generated, and a scoring process is done for each candidate concept. The highest scoring concept is then selected as the concept for the phrase. It is possible a noun phrase can map to more than one concept. In this case, no disambiguation step is performed, and MetaMap returns multiple concepts. Figure 4 shows an example of

MetaMap mapping of the phrase *protein kinase CK2*. The output shows the phrase, the concept candidates preceded by their score ("Meta Candidates"), and the final mapping of the phrase ("Meta Mapping"). In the example, there are six candidate mappings, shown in descending score order. The final mapping takes the highest scoring candidate, shown as "Meta Candidate (1000)" in Figure 4. In cases where a phrase cannot be successfully disambiguated, it is possible for MetaMap to generate a final mapping consisting of more than one concept. Finally, MetaMap indicates the semantic type for the selected concept, shown as a text description in square brackets next to the concept text description in Figure 4.

```
Phrase: "protein kinase CK2."
Meta Candidates (6)
  1000 protein kinase CK2 (casein kinase II) [Amino Acid, Peptide, or Protein,Enzyme]
   901 PROTEIN KINASE [Amino Acid, Peptide, or Protein,Enzyme]
   827 Kinase (Phosphotransferases) [Amino Acid, Peptide, or Protein,Enzyme]
   827 Protein (Proteins) [Amino Acid, Peptide, or Protein,Biologically Active S
ubstance]
   827 Protein NOS (Protein measurement) [Laboratory Procedure]
   827 CK2 [Laboratory Procedure]
Meta Mapping (1000)
  1000 protein kinase CK2 (casein kinase II) [Amino Acid, Peptide, or Protein,Enzyme]
```

Figure 4: MetaMap Transfer mapping of the phrase *protein kinase CK2*.

3.2 Text Summarization

There are two different approaches to generating summaries from text: extractive and abstractive (Afantenos, Karkaletsis, & Stamatopoulos, 2005). The *extractive* approach extracts sentences or parts of sentences verbatim from text and uses them to generate a summary. The extractive approach is the most common way to perform summarization, and is the method used in the proposed research. Extractive approaches to text summarization usually follow a model of scoring sentences based on a set of

features, such as term frequency, keyphrase identification, and sentence location. A set of highest-scoring sentences from the source text is used to form a final summary. The task of sentence selection can be considered an information retrieval task, where the set of all sentences within a text are evaluated (scored), and the highest scoring sentences are selected as being the most relevant to a user. The top-*n* highest-scoring sentences in a text are extracted, using *n* as an upper bound on the number of sentences to select. The extraction summarization task, then, is to identify a minimal subset of sentences from the source text which are relevant to the user and which minimize redundancy. The proposed research uses the extractive approach.

The second and significantly more difficult approach is called *abstractive summary generation*, and involves generating summary text using natural language processing and generation techniques. For example, the template method of abstractive summary generation uses a predefined template where the fields in the template are filled-in from information contained in the source text. A different method for abstraction uses a syntactical analysis of the source text to identify key components of each candidate sentence, and this analysis is used to form new sentences from existing sentences.

3.3 Evaluation Corpus

To provide a set of data for evaluating semantic annotations as well as summary performance, a corpus of 24 biomedical texts was generated from a citation database of oncology clinical trial papers. The database contains approximately 1,200 papers physicians feel are important to the field (Brooks & Sulimanoff, 2002). Of the 1,200 papers cited, 24 were randomly selected. The number of papers chosen (24) was based on

the minimum requirements of the ROUGE summary evaluation tool (C. Lin, 2004) as well as the resources available to complete the manual processing of each paper. The PDF versions of these 24 papers were then obtained and converted to plain-text format. The papers were then manually processed to remove graphics, tables, figures, captions, citation references, and the bibliography section. The resulting texts were further split into an abstract text and a full-text source text (without the abstract). To develop a corpus for semantic annotation, the 24 papers will be processed by MetaMap which will find all noun phrases in the 24 papers. The unique noun phrases will then become the corpus for evaluating semantic annotation accuracy and performance.

4. PROBLEM STATEMENT

The goal of the proposed research is to construct a system to perform semantic annotation and summarization of biomedical texts which has performance competitive with existing systems. The identification and use of domain-specific concepts is used to produce summaries. The methods developed as part of the proposed research, while requiring input from domain resources, are domain-independent, and could be applied to other domains, such as summarizing legal texts. The long-term result of the proposed research is that the system reduces the amount of work required by practicing physicians and researchers to assimilate new knowledge from continuing advancements in biomedicine. The proposed research may also be helpful for future researchers performing multiple-document summarization where concepts, rather than terms, can reduce the variability of language between documents.

The proposed research is divided into two components: Semantic Annotation and Text Summarization. For the Semantic Annotation component, methods for performing faster annotation of biomedical texts are examined. The idea is to improve the performance of annotation so that it can be used in an online environment. For the Text Summarization component, concepts are used in place of terms to identify important areas within the source text which should be extracted into a summary. The idea is to determine if the use of domain-specific concepts improves the performance of summarization. In addition, since the research on the characteristics of biomedical texts is largely absent from existing literature, the proposed research will investigate two important characteristics of biomedical texts using information from domain experts: (a)

finding an optimal size of a biomedical summary (at least for randomized controlled trials), and 2) finding the locations within texts where sentences are extracted from when constructing model summaries. Biomedical texts are usually written with a definite structure, such as Introduction → Background → Methodology → Evaluation → Conclusion. Knowing which sections human summarizers are likely to extract sentences from is expected to be useful in improving summarization performance by weighting sections differently, instead of giving equal weight to all sections.

5. APPROACH

The high-level structure of the proposed system is shown in Figure 5. The system takes a source text and performs analysis in several stages where the final output is a text summary of the original source text. The two primary stages which are part of the proposed research are Semantic Annotation and Text Summarization. Two other stages, Lexical Processing and User Presentation, are constructed as part of the system, but are not the focus of the research. The Lexical Processing stage is the first stage and is responsible for finding sentence boundaries within a source text, and then phrases within each discovered sentence. Lexical Processing in the proposed research is performed using software components from third-party sources. The final stage is the User Presentation stage, which presents to the user the generated summary. For the proposed research, the user presentation consists of generating a simple text file containing the extracted sentences.

The two stages which are the focus of the research, Semantic Annotation and Text Summarization, are between the Lexical Processing and User Presentation stages. Once a source text has been decomposed into sentences and phrases in the Lexical Processing stage, the sentence phrases are fed into the Semantic Annotation stage, and the output is an annotated version of the source text containing sentences, sentence phrases, and sentence phrases annotated with one or more domain-specific concepts (a best mapping is attempted, but in cases of ambiguity more than one mapping may be returned.) The annotated source text from the Semantic Annotation stage is then directed into the Text Summarization stage. The Text Summarization stage uses the concept mapping information to identify the most important content in the source text, and then extract

sentences from the source text which are most representative of the content. The output of the Text Summarization stage is an ordered subset of sentences from the source text. In the proposed research, only single document summarization is considered. In the following sections, the details of each of the two primary stages, Semantic Annotation and Text Summarization, are presented.

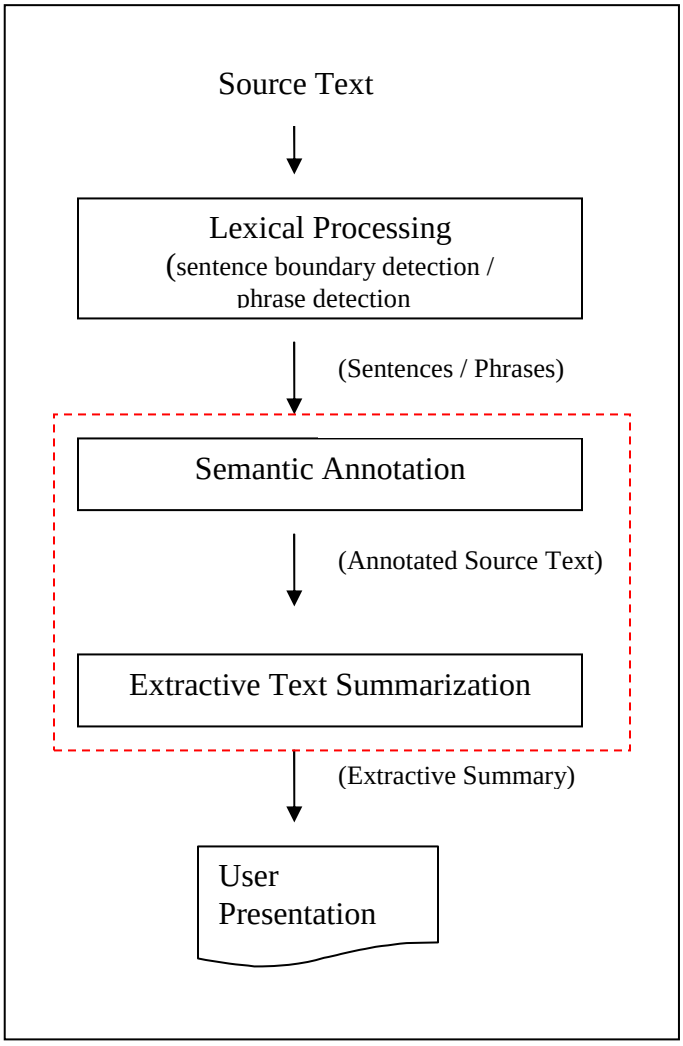

Figure 5: High-level overview of the proposed research system. The dashed box indicates the parts of the system the proposed research focuses on.

5.1 Biomedical Semantic Annotation

A graphical view of the proposed semantic annotation process is shown in Figure 6, which is a detailed view of the Semantic Annotation stage shown in Figure 5. The approach to semantically annotating a biomedical text is to use a multi-level filtering approach based on phrases from the source text. The novel contributions of the proposed semantic annotation component of the system are as follows: (a) a multi-level, extensible filtering architecture to find the best-matching concept, (b) the use of term importance (through weighting) during the filtering process, and (c) the use of skip-bigrams to measure term order.

The proposed annotation system assumes the source phrase has already been determined from prior analysis. Source phrases can be detected using a variety of methods, such as natural language parsing and sliding windows (Wollersheim, Rahayu, & Reeve, 2002) and barrier words (Tersmette, Scott, Moore, Matheson, & Miller, 1988). The proposed system makes no assumptions on the selection of source phrases, other than the input unit must be a phrase. The goal of the Semantic Annotation stage is to find the best UMLS concept match for each phrase in the source text using surface-level features.

In the proposed annotation system, there are several types of phrases. The *source phrase* is a phrase from the source text which the system will attempt to annotate with a biomedical concept. Recall from the Background chapter that UMLS concepts are composed of one or more synonymous phrases, which are known as *concept phrases*. A single UMLS concept may have more than one concept phrase associated with it. *Candidate phrases* are concept phrases having words in common with the source phrase.

A *candidate concept* identifies the UMLS concept a candidate phrase belongs to. A *concept name* is the name given to a particular UMLS concept.

The multi-level filtering approach of the proposed system takes a source phrase from the source text, retrieves a list of concept phrases based on the overlap of words from the source phrase and concept phrases, forming a set of candidate phrases. If only a single candidate phrase exists, its associated concept is returned. If there is more than one candidate phrase generated, an iterative process of filtering out candidate phrases begins. The candidate phrase filters are based on n-gram co-occurrences between the source phrase and the candidate phrases. The multi-level filtering is done to improve computational efficiency by applying successively more computationally complex filters, rather than scoring a candidate phrase with different measures at once. This approach is different than existing approaches, which typically score a candidate phrase completely in one pass and then rank the set of resulting concepts (W. Hersh & Leone, 1995), (Aronson, 2001). The idea is to successively filter out concepts using basic techniques, and compute more complex candidate phrase scores for a small subset of possible candidate phrase matches. The proposed research uses two different types of filters, although other filters can be added: coverage and coherence. Coverage is the number of common words between the source phrase and a candidate phrase, and coherence measures the common word ordering between the source phrase and a candidate phrase (Aronson, 1996). These two filters were chosen because they are the types used in MetaMap. By using similar measures of coverage and coherence, the accuracy of the proposed annotator is expected to be similar to the accuracy of MetaMap but with

reduced annotation time. The MetaMap system output is used as the gold standard for the evaluation.

The proposed annotation system does not, however, use the same scoring mechanism as MetaMap and instead proposes new methods for measuring coverage and coherence. The coverage filter will use term weights rather than term counts, as MetaMap does. The coherence filter will use skip-bigrams, which have proven effective for measuring term order in machine translation and text summarization evaluation. Both methods are described in more detail below. Additional filters may also be explored, such as a concept disambiguation filter based on UMLS concept co-occurrence information, and a filter based on language modeling techniques from the information retrieval field.

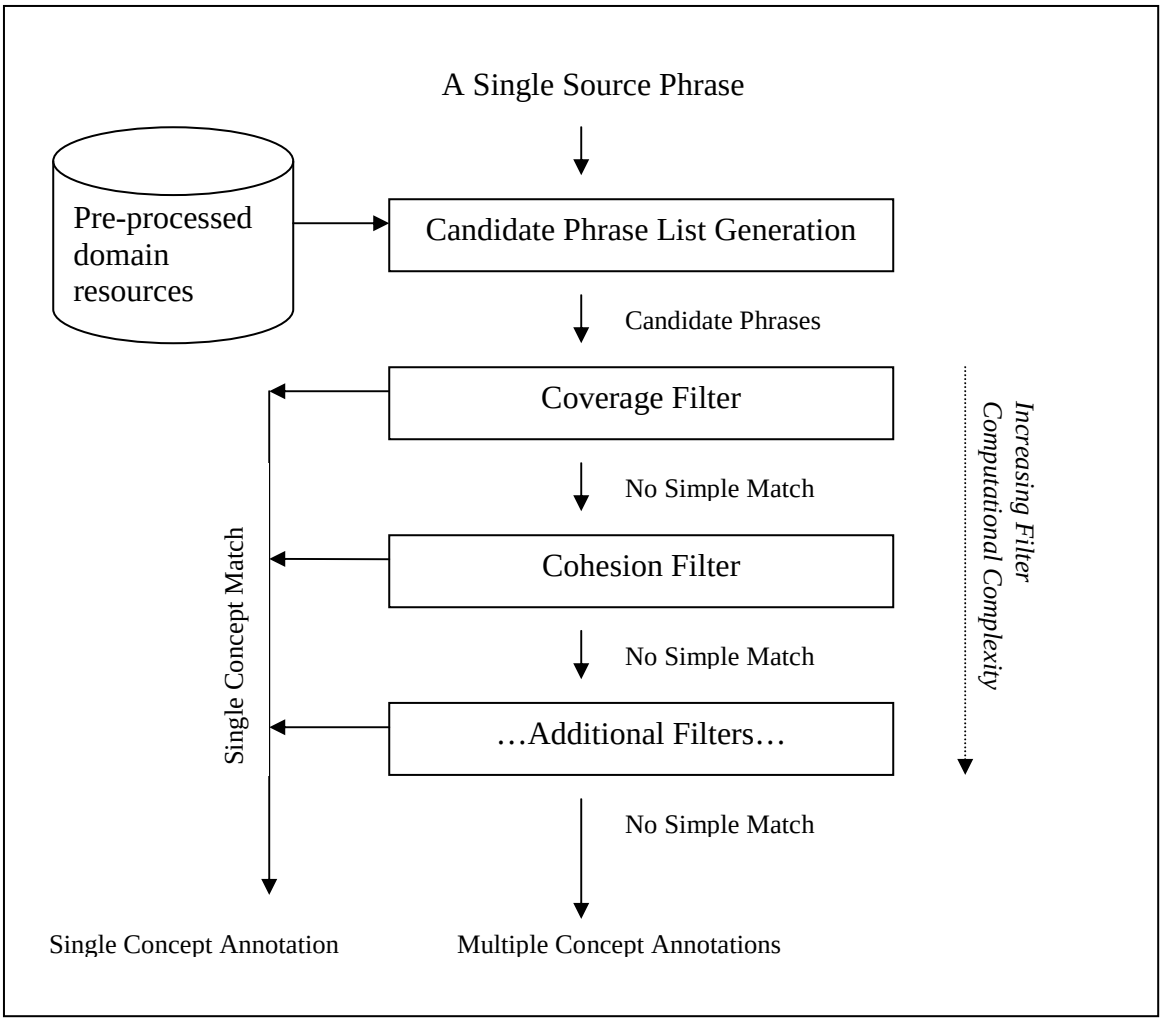

Figure 6: Stages in the proposed multi-level annotator

The proposed annotation system has advantages over existing concept mapping approaches. Systems such as MetaMap (Aronson, 1996) and SAPHIRE (W. Hersh & Leone, 1995) score candidate phrases completely in one pass. This has the disadvantage of computing more complex scores, such as coherence, when the simpler coverage score may have rejected the candidate phrase alone. In addition, by modeling the mapping as a series of filtering stages, the number of candidate phrases is reduced at each stage,

reducing the number of complex calculations which must be performed. The MetaMap system also generates word variants and inflections at run-time. The proposed annotation system maps the concept phrase words to their base form in a pre-processing stage (done once for each UMLS version of the data, not at each run-time). For example, the variants {*eyes*, *eyed*, *eying*} are all mapped to the base word *eye*. This allows concept phrases to map to base words so that variant generation is not required. By reducing the number of candidate phrases which have to be scored, using in-memory table structures, and eliminating time-consuming variant generation, the proposed annotation system is expected to outperform the state-of-the-art MetaMap system in terms of time to annotate a source phrase.

The overall annotation strategy for a single source phrase is as follows:

1. Construct a list of candidate phrases based on the common words between all concept phrases and the source phrase. If only one candidate phrase remains, return its associated candidate concept.

2. Filter the list of candidate phrases based on a weighted coverage score given to each candidate phrase. If only one candidate phrase remains, return its associated candidate concept.

3. If more than one candidate phrase remains, filter the list of candidate phrases based on a weighted coherence score given to each candidate phrase. If only one candidate phrase remains, return its associated candidate concept.

4. If more than one candidate phrase remains, return the list of candidate concepts associated with each candidate phrase.

The following text explains in more detail each of the annotation stages:

Pre-processing Step: Domain Resource Preparation

Preprocessing of UMLS data is done before the proposed annotator is run. The preprocessing organizes the words, word variants, phrases, and concepts stored in UMLS text files into a format which is faster to process. For example, words are mapped to unique integer identifiers to reduce storage space and increase word comparison speed. Also, concept phrase words are mapped to their base form so that variant generation does not have to occur at run-time, as is the case for systems such as MetaMap (Aronson, 1996). For example, word variants within UMLS concept phrases are all mapped to a single base, such as mapping {*eye, oculus, ophthalmic*} to the base form {*ocular*}.

Each word in the UMLS is also weighted based on its usage in all concept phrases. In contrast to existing systems such as MetaMap (Aronson, 2001), SAPHIRE (W. Hersh & Leone, 1995), and IndexFinder (Zou, Chu, Morioka, Leazer, & Kangarloo, 2003) which consider the count of common words between a source phrase and a candidate phrase, the scoring of coverage and coherence in the proposed annotation system considers the contribution of each word in the source phrase by using a weighting mechanism. Information retrieval research uses a family of algorithms called TF*IDF, which uses the frequency of a term (TF) within a document and the frequency of a term across all documents (IDF) to find a similarity value between a user query and a document. A frequently occurring term within a document better indicates the content of the document, while a frequently occurring term across all documents is thought to give little discriminatory power, since a high proportion of documents contain the term

(Baeza-Yates & Ribeiro-Neto, 1999). The inverse document frequency value (IDF) uses the frequency of a word across all documents as a way to identify words which are semantically focused (Manning & Schutze, 1999). Semantically-focused words are those words which do not frequently occur across all documents within a collection, and thus are more likely to have more discrimination power than words which frequently occur. To apply the ideas of TF*IDF from information retrieval to the proposed annotation system, each concept phrase is substituted for document. The weight of each unique word in all concept phrases is calculated using the inverse document frequency idea from information retrieval (Sparck Jones, 1972), substituting concept phrase for document, as shown in Figure 7:

$$inverse\,phrase\,frequency = \log\frac{N}{n_i}$$

Figure 7. Inverse phrase frequency (IPF) value. N is the total number of phrases in UMLS and n_i is the total number of phrases a particular word occurs in.

Each unique UMLS word *i* is assigned a weight w_i based on its inverse phrase frequency (IPF) value. The importance (or weight) w_i of a word *i* is represented by its IPF value. Words which are more semantically focused will be given a higher weight than words which are not semantically focused. For example, assume there are 1,000 concept phrases, the word *plasma* occurs in 200 of the phrases, and the word *myeloma* occurs in 50 of the phrases. The IPF value of *plasma* will be 0.70 (log 1000/200) and the IPF value of *myeloma* will be 1.30 (log 1000/50). Therefore, if the words *plasma* and *myeloma* occur in the same phrase, the word myeloma is considered a more discriminative word than *plasma*. The idea is to give some indication of the importance

of a word based on its usage within all concept phrases. Term frequency, which is typically combined with inverse document frequency for document information retrieval, is not considered since it is highly likely the frequency for each word will be one because the input unit of the proposed annotation system is a phrase, which usually does not include the same word multiple times in it. Eliminating term frequency from scoring reduces computational complexity.

Figure 8 shows a minimal list of tables which are expected to be generated from the pre-processing. The tables are all simple key-value lookup tables which map a unique key to a value. The advantage of creating the lookup tables is to speed access to key information, such as the words belonging to concept phrases, and to pre-calculate key information used in the stages, such as the Inverse Phrase Frequency value for each word, in order to increase runtime performance. The first column in the each of the tables shown in Figure 8 is the key, and the second column is the value. Figure 8(a) shows that each unique UMLS word is mapped to a unique identifier. Similarly, Figure 8(b) shows a unique concept identifier can be used to find a concept name. Figure 8(c) indicates a mapping from a unique concept phrase identifier to a unique UMLS concept identifier. Figure 8(d) maps a unique word identifier to a list of concept phrase identifiers the word occurs in. Figure 8(e) shows the mapping from a unique word identifier to the word's inverse phrase frequency (IPF) value. Figure 8(f) is the mapping from a concept phrase identifier to the ordered list of word identifiers belonging to a concept phrase. Table 2 lists an example usage for each mapping table. Each mapping table is useful for getting additional information about a word, phrase or concept. For example, given a word in the source text, its unique word identifier can be retrieved using the WordToWordId table,

and the list of concept phrases the word appears in can then be obtained using WordIdToPhraseIdList table. To get the concept name a phrase belongs to, the PhraseIdToConceptId table can be used to find the phrase's concept identifier, and then the ConceptIdToConceptName table is used to find the concept name.

Word	WordId
lung	1
cancer	2

(a) *WordToWordId*

ConceptId	Concept Name
0242379	Malignant Neoplasm of the Lung
0684249	Carcinoma of the Lung

(b) *ConceptIdToConceptName*

PhraseId	ConceptId
100	0242379
200	0684249

(c) *PhraseIdToConceptId*

WordId	PhraseIdList
1	100,200
2	100,200

(d) *WordIdToPhraseIdList*

WordId	WordIPF
1	.5
2	.3

(e) *WordIdToWordIPF*

PhraseId	WordIdList
100	1,2
200	2,1

(f) *PhraseIdToWordIdList*

Figure 8: Mapping tables resulting from UMLS pre-processing: (a) mapping a word to a unique identifier; (b) mapping a concept identifier to a concept name; (c) mapping a unique phrase identifier to a unique concept identifier; (d) mapping a unique word identifier to a list of concept phrase identifiers; (e) mapping a unique word identifier to the word's IPF value; (f) mapping a unique phrase identifier to an ordered list of words in the phrase.

Table Name	Example Usage
WordToWordId	Given a word, find its unique identifier
ConceptIdToConceptName	Get the concept name for a given concept identifier
PhraseIdToConceptId	Retrieve the concept associated with a UMLS phrase
WordIdToPhraseIdList	Find all concept phrases having the given word
WordIdToWordIDF	Find the inverse phrase frequency value for a given word
PhraseIdToWordIdList	Find all words belonging to a concept phrase

Table 2: Example Usage of Mapping Tables

Step 1: Candidate list generation

When a source phrase is presented to be annotated, it is first processed to remove all words which do not appear in UMLS, as well as removal of stop words. The words in the source phrase are mapped to their UMLS base form. This is done to eliminate word variation, and to allow exact matching of concept phrase words, which had the same base-form mapping done in the pre-processing step. A list of candidate phrases is then generated by finding all concept phrases which contain one or more of the base-form words in the source phrase. For example, the phrase *lung cancer* will find all candidate phrases having the words *lung* and *cancer*, which will return candidate phrases such as {*lung, chronic obstructive lung disease, lung cancer, liver cancer*} and so forth. Table 3 shows a partial list of candidate concepts generated based on the concept phrases having words in common with the source phrase *lung cancer*. It is not required that a candidate phrase have all words in common, since exact mappings between a source phrase and concept phrases are expected to be rare.

Source Phrase: *lung cancer*		
Concept Id	Concept Name	Concept Phrase
0024109	Lung	*Lung*
0024117	Chronic Obstructive Airway Disease	*Chronic Obstructive Lung Disease*
0242379	Malignant Neoplasm of the Lung	*Lung Cancer*
0684249	Carcinoma of the Lung	*Cancer of the Lung*
0279000	Liver and Intrahepatic Biliary Tract Carcinoma	*Liver Cancer*

Table 3: Example candidate phrase list generation.

If the source phrase is long (consisting of five or more words), the number of candidate phrases generated by the words in the source phrase may be very large. One way to overcome this is to select only the most important words in the source phrase, and then use these words to select candidate phrases. The proposed method of finding the most important words is to find the IPF value of each word in the source phrase, calculate the standard deviation of the retrieved IPF values, and then use all source phrase words whose IPF value is greater than a threshold to find candidate phrases. The chosen threshold is one standard deviation of the mean IPF value. Selecting greater than or equal to one standard deviation allows approximately 32% (Kiess, 2002) of the words in the source phrase to be used to retrieve candidate phrases. Figure 9 shows an example of the long source phrase *chronic obstructive pulmonary disease finding* having word IPF values of {1.5, 2, 2, 1, 1}. The mean is 1.5, the standard deviation is 0.5, and the minimum IPF value is therefore 2.0. To generate candidate phrases in this example, only the words {*obstructive*, *pulmonary*} are used since they are the most discriminative words based on their IPF value. In the example, all concept phrases having either the words *obstructive* or *pulmonary* will be passed to the next filter (i.e., Coverage filter). The idea is to use the only the most important words in the source phrase to limit the

number of candidate phrases retrieved. Each stage of the filter should seek to find a single best matching phrase, and if not possible, reduce the number of candidate phrases passed to the next filter stage, which is assumed to be more computationally complex.

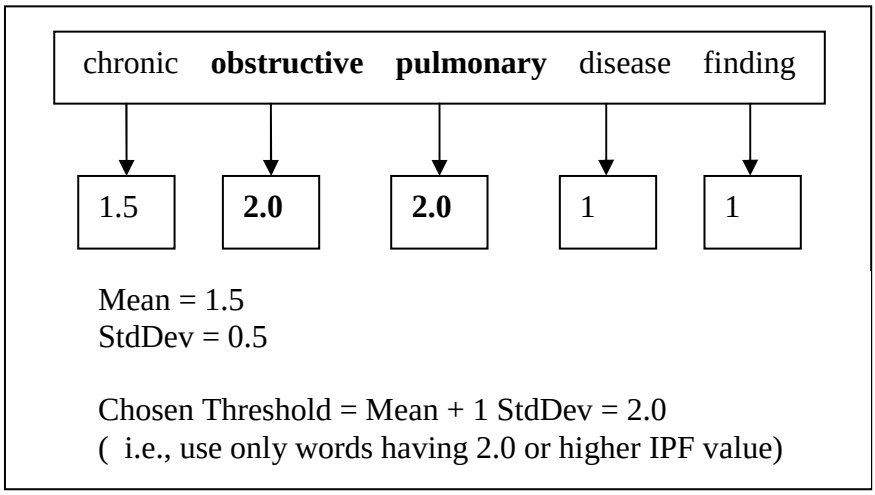

Figure 9: Example of long source phrase using IPF values to find significant words

Step 2: Coverage Filter (Stage 1)

Once a list of candidate phrases is retrieved, coverage (overlap of common words) is measured to filter out less important candidate phrases. The idea is to find the list of candidate phrases having the best coverage of the source phrase words, based on the IPF value of each word in common between the source phrase and each candidate phrase. The Coverage score for each candidate phrase can be computed quickly using table lookup operations. In existing work, a coverage score for a candidate phrase is measured using a count of the number of common words (Aronson, 1996), (Zou et al., 2003). In the proposed research, weighted unigram filtering is used to measure coverage. The combined IPF value of all common words between a source phrase and a candidate

phrase is used as the coverage score for each candidate phrase and is called the PhraseCoverageIPF value, defined in Figure 10 as:

$$\text{PhraseCoverageIPF} = \sum_{i=1}^{N} \text{IPF}_i$$

Figure 10: Phrase Coverage Score (PhraseCoverageIPF). N is the total number of common words between the source phrase and the candidate phrase. IPF is the Inverse Phrase Frequency of a common word *i* between the source phrase and the candidate phrase.

Once the PhraseCoverageIPF values are computed for all candidate phrases, the standard deviation of the PhraseCoverageIPF values for the set of candidate phrase is calculated. A threshold value is chosen as the mean plus one standard deviation. All candidate phrases whose PhraseCoverageIPF values is greater than or equal to the threshold value are passed to the second stage filter (i.e., Coherence filter). There are two exceptions to consider: (a) if there is an exact match between a source phrase and one of the candidate phrases, the candidate concept associated with the candidate phrase is returned; and (b) if no candidate phrases have a PhraseCoverageIPF value greater than or equal to the threshold, the candidate phrases with the highest PhraseCoverageIPF value are passed to the second stage (if there is only one such candidate phrase with a high PhraseCoverageIPF, its candidate concept is returned).

Examples of each case are shown in Tables 4 through 6. Table 4 shows an example of the coverage filter processing for the source phrase *lung cancer* where there is an exact match. The IPF values for each word in the source phrase are first retrieved.

Each candidate phrase from the candidate generation step is scored by summing the IPF values for *lung* and *cancer* when they exist in the candidate phrase. After all candidate phrases are scored with this PhraseCoverageIPF value, exact candidate phrase matches with source phrase PhraseCoverageIPF value and source phrase word ordering are returned. In the Table 4 example, *Lung Cancer* is returned because it is an exact match both with the source phrase PhraseCoverageIPF value and with the source phrase *lung cancer*. *Cancer of the Lung* is not returned because it is not an exact match with the source phrase, even though it has the same PhraseCoverageIPF value as the source phrase. In Table 4, the mean PhraseCoverageIPF value is 0.90, the standard deviation is 0.35, and the threshold value, the mean plus one standard deviation, is 1.24. Since an exact match was found, the threshold value is not used since no candidate phrases need to be passed to the second stage.

Source Phrase: *lung cancer*
(IPF values: lung=0.75, cancer=0.50, total=**1.25**)

Candidate Phrase	PhraseCoverageIPF value
Lung	0.75
Chronic Obstructive Lung Disease	0.75
Lung Cancer	**1.25** **← Exact Value & String**
Cancer (of the) **Lung**	**1.25** **← Exact Value**
Liver Cancer	0.50

Scoring Details:
Mean PhraseCoverageIPF value = 0.90
StdDev of PhraseCoverageIPF values = 0.34
Mean PhraseCoverageIPF value + 1 StdDev = 1.24

Exact match in PhraseCoverageIPF value and
in source phrase string for *Lung Cancer*

Table 4: Example Stage 1 Coverage filtering for exact match with source phrase

Table 5 shows the case where there is no exact match between the source phrase and the candidate phrases. In this case, the function of the Stage 1 (i.e., Coverage) filter is to reduce the size of the candidate list. This is accomplished by retaining all candidate phrases whose PhraseCoverageIPF value is greater than or equal to a threshold value. The mean PhraseCoverageIPF value is 0.90, the standard deviation is 0.16, and the threshold value, the mean plus one standard deviation, is 0.58. Therefore, any candidate phrase which has a PhraseCoverageIPF value greater than or equal to 0.58 is passed to the second stage filter. In Table 5 there are two candidate phrases having a PhraseCoverageIPF value of 0.60, so both of these candidate phrases (shown in bold) are passed to the second stage filter (i.e., coherence filter). If only one candidate phrase had a PhraseCoverageIPF value equal to or greater than 0.60, its candidate concept is returned and no candidate phrases are passed to the Stage 2 filter.

Source Phrase: *lung cancer disease*
(IPF values: lung=0.30, cancer=0.30, disease=0.30, total=**0.90**)

Candidate Phrase	PhraseCoverageIPF value
Lung	0.30
Chronic Obstructive Lung Disease	0.60 **>= 0.58**
Liver Cancer	0.30
Lung Cancer	0.60 **>= 0.58**
Cancer	0.30

Scoring Details:
Mean PhraseCoverageIPF value = 0.42

StdDev of PhraseCoverageIPF values = 0.16

Mean PhraseCoverageIPF value + 1 StdDev = 0.58

Two PhraseCoverageIPF values >= 0.58 are passed to Stage 2 filter:
Chronic Obstructive Lung Disease
Lung Cancer

Table 5: Example Stage 1 Coverage filtering with no exact match for source phrase

Table 6 shows a variation of the no-exact-match case shown in Table 5, where there is no exact match between the source phrase and the candidate phrases, and none of the candidate phrases have a PhraseCoverageIPF value greater than or equal to the threshold value. If the strict logic shown in Table 5 is followed, then no candidate phrases will be passed to the second stage filter. To resolve this, the approach used is to find the candidate phrases with the highest PhraseCoverageIPF value. If only one such candidate phrase exists, its corresponding candidate concept is returned. In Table 6, the mean PhraseCoverageIPF value is 0.68, the standard deviation is 0.26, and the threshold value, the mean plus one standard deviation, is 0.94. Therefore, any candidate phrase which has a PhraseCoverageIPF value greater than or equal to 0.94 is passed to the second stage. As

Table 6 shows, no candidate phrase has a PhraseCoverageIPF value greater than or equal to 0.94. In this case, the highest PhraseCoverageIPF value is found, which is 0.90. There are two candidate phrases having a PhraseCoverageIPF value equal to 0.90, so the two corresponding candidate phrases are passed to the second stage filter (i.e., coherence filter). If only one candidate phrase had the highest PhraseCoverageIPF value, its candidate concept is returned and no candidate phrases are passed to the Stage 2 filter.

Source Phrase: *lung cancer disease*
(IPF values: lung=0.50, cancer=0.40, disease=0.40, total=1.30)

Candidate Phrase	PhraseCoverageIPF value
Lung	0.50
Chronic Obstructive Lung Disease	0.90 **← Highest Value**
Liver Cancer	0.40
Lung Cancer	0.90 **← Highest Value**

Scoring Details:
Mean PhraseCoverageIPF value = 0.68

StdDev of PhraseCoverageIPF values = 0.26

Mean PhraseCoverageIPF value + 1 StdDev = 0.94
(No PhraseCoverageIPF values >= 0.94, so candidate phrases with highest scores are passed to Stage 2 filter:
Chronic Obstructive Lung Disease
Lung Cancer

Table 6: Example Stage 1 Coverage filtering with highest scores used

Step 3: Coherence Filter (Stage 2)

The second stage filter measures coherence, where coherence is the ordering of terms in the phrase. Coherence is measured by looking at the order of the common words

between the source phrase and each candidate phrase. The idea is that the common syntactic ordering of the source and candidate phrases will remove candidate phrases which have some words in common but are in a different order, indicating the candidate phrase may be expressing a different concept than the source phrase. For this stage, pairs of ordered words which allow for intervening words, known as skip-bigrams (C. Y. Lin & Och, 2004a), are used. The bigrams are generated by walking the candidate phrase words from beginning to end and pairing each word with the word that follows it. For example, the phrase *peripheral plasma cell myeloma* has the set of complete skip-bigrams as shown in Figure 11:

peripheral plasma
peripheral cell
peripheral myeloma
plasma cell
plasma myeloma
cell myeloma

Figure 11: Complete skip-bigram (i.e., no gap defined) for phrase *peripheral plasma cell myeloma*

The number of intervening words, called a gap, can be limited. The skip-bigram gap can be set from 0 to any specified number, and indicates the number of allowed intervening words. The lower the gap size, the more restrictive the order of words is enforced. For a given gap size *n*, the skip-bigrams generated include all skip-bigrams for lower levels of *n*. For example, a gap size of two will include skip-bigrams with gap sizes zero, one and two. Figure 12 shows the skip-bigrams generated for a gap size of zero:

peripheral plasma
plasma cell
cell myeloma

Figure 12: Skip-bigrams with gap zero for phrase *peripheral plasma cell myeloma*

A gap size of one produces the list shown in Figure 13 (which includes skip-bigrams of gap size zero as well as skip-bigrams of gap size one):

peripheral plasma
peripheral cell
plasma cell
plasma myeloma
cell myeloma

Figure 13: Skip-bigrams with gap one for phrase *peripheral plasma cell myeloma*

The skip-bigram statistic can be computed as precision and recall measures (C. Y. Lin & Och, 2004a), as shown in Figure 14. In both measures, the number of common skip-bigrams between the source phrase and a candidate phrase within a specified gap is computed. For precision, this common skip-bigram count is divided by the total number of skip-bigrams of the source phrase within a specified gap. For recall, the common skip-bigram count is divided by the total number of skip-bigrams of the candidate phrase within a specified gap. The skip-bigram precision measures the degree of skip-bigram matching with the source phrase, while the skip-bigram recall measures the degree of skip-bigram overlap with the candidate phrase.

$$\text{Precision} = \frac{\text{CommonSkipBigramsWithinGap}(\text{SourcePhrase}, \text{CandidatePhrase})}{\text{CounSkipBigramsWithinGap}(\text{SourcePhrase})}$$

$$\text{Recall} = \frac{\text{CommonSkipBigramsWithinGap}(\text{SourcePhrase}, \text{CandidatePhrase})}{\text{CountSkipBigramsWithinGap}(\text{CandidatePhrase})}$$

Figure 14: Skip-bigram Precision and Recall metrics. CommonSkipBigramsWithinGap(SourcePhrase, CandidatePhrase) is the number of the common bigrams between the source phrase and the candidate phrase within the specified gap, and the CountSkipBigramsWithinGap is the number of skip-bigrams within the specified gap distance. (C. Y. Lin & Och, 2004a)

For example, the source phrase *cancer of the lung* demonstrates how the skip-bigram filter works. There are two potential UMLS concepts for this phrase: *C0242379: Malignant Neoplasm of the Lung* and *C0684249: Carcinoma of the Lung*. The candidate phrase for concept C0242379 is *Lung Cancer*. The candidate phrase for concept C0684249 is *Cancer of the Lung*, which becomes *Cancer Lung* after stop word removal. The source phrase after stop word removal is *cancer lung*. The skip-bigrams with a gap of zero for C0242379 is *lung cancer*, for C0684249 is *cancer lung* and for the source phrase is *cancer lung*. The skip-bigram recall score for C0242379 is 0 (0/1) while the skip-bigram recall score for C0684249 is 1 (1/1). Therefore, the returned UMLS concept is C0684249: *Carcinoma of the Lung*.

Figure 15 shows an example of how the skip-bigram filter works using the source phrase *cancer of the lung*. There are two potential UMLS concepts for this phrase: *C0242379: Malignant Neoplasm of the Lung* and *C0684249: Carcinoma of the Lung*. The candidate phrase for C0242379 is *Lung Cancer*. The candidate phrase for C0684249 is *Cancer of the Lung* which becomes *Cancer Lung* after stop word removal. The source phrase after stop word removal is *cancer lung*. The skip-bigrams with a gap of zero for C0242379 is *lung cancer*, for C0684249 is *cancer lung* and for the source phrase is

cancer lung. The skip-bigram recall score for C0242379 is 0 (0/1) while the skip-bigram recall score for C0684249 is 1 (1/1). Therefore, the returned concept is C0684249.

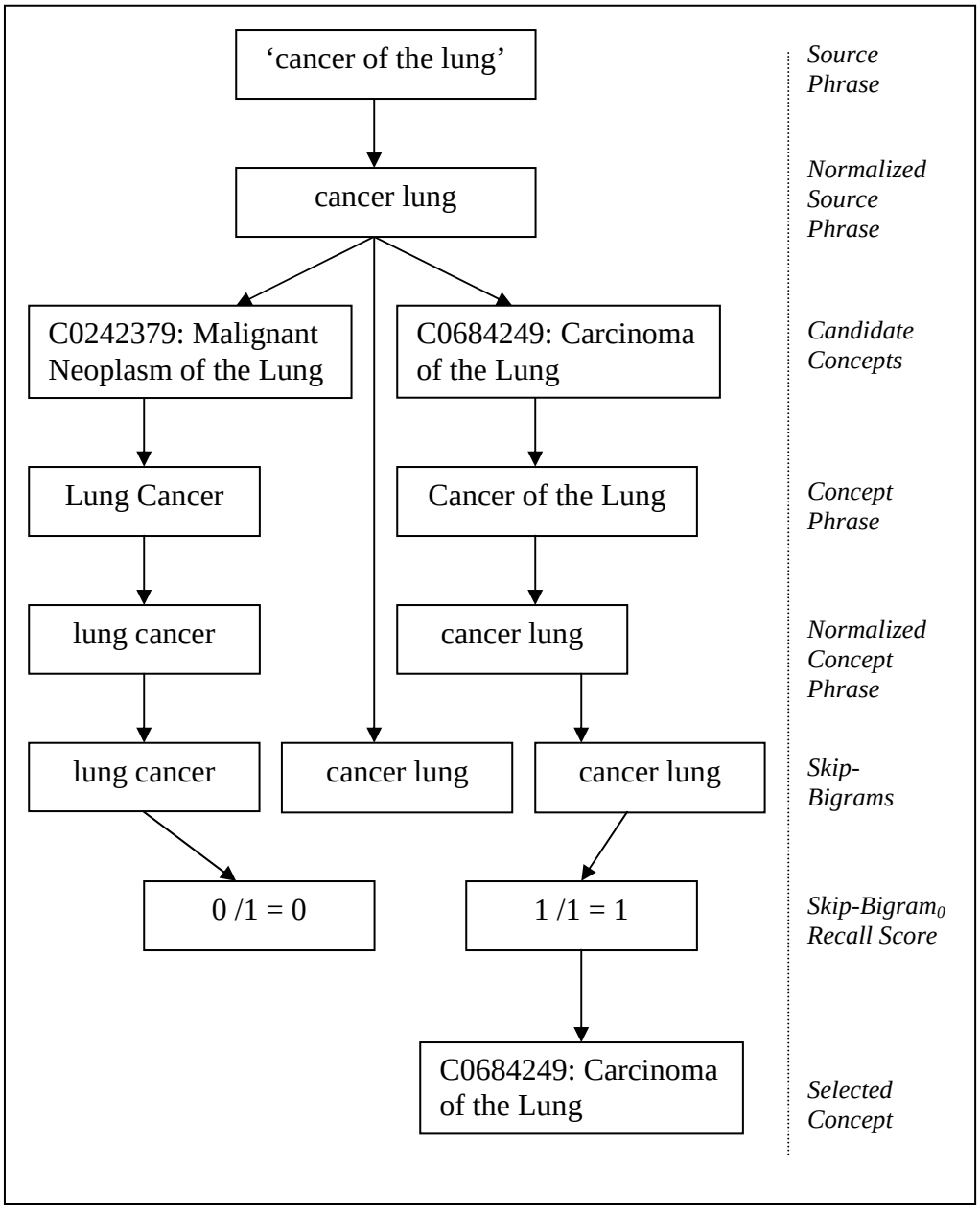

Figure 15: Example Skip-Bigram filtering.

The performance of skip-bigrams has been evaluated in machine translation evaluation and summary evaluation, and has been shown to perform at or above state-of-the-art measures with less complexity (C. Y. Lin & Och, 2004b). The proposed annotator will use the Recall measure, since it has been shown in machine translation evaluation research that n-gram recall is the biggest factor in evaluations using n-gram measures (Lavie, Sagae, & Jayaraman, 2004). In addition, the original SAPHIRE system used a high-precision approach to match all words in the source phrase in their original order, and found that this resulted in missed concept mappings (W. Hersh & Leone, 1995). The original SAPHIRE system used exact word order to try and eliminate false-positive matches where all words appeared in a phrase but in a different order resulting in finding a different meaning than the source phrase intended. The proposed annotation system will calculate the skip-bigram recall scores for all candidate phrases with a gap size of zero, which is the strictest matching of word order, but is less strict than the high-precision approach. Whereas the original SAPHIRE system required exact word order, the proposed annotation system will allow for gaps between words, which imposes word order as a requirement, but does not completely require all words to match between the candidate phrase and the source phrase. This enforcement of word order while allowing intervening words is the primary advantage of using the skip-bigram approach. The concept associated with the highest-scoring candidate phrase is returned. If there are ties in candidate phrase scores, the concepts associated with the tied candidate phrases are returned.

Step 4: Additional Stages

The proposed system is not limited to the two filtering stages presented above. The coverage and coherence stages were designed to give similar performance as the MetaMap system. It is anticipated future work will provide additional filters, and may even modify the coverage and/or coherence filters. Examples of additional filters are a concept disambiguation filter and a language modeling filter. The concept disambiguation filter could be implemented by using UMLS-provided concept co-occurrence information. The general idea would be to make two passes over the source text, annotating first all unambiguous concepts, and then using the unambiguous discovered concepts along with UMLS co-occurrence metrics to disambiguate remaining concepts. A language modeling filter could use the information retrieval approach of first building a language model as part of the pre-processing step for all UMLS concepts based on all synonymous phrases within a concept. The filter would then calculate the probability of each candidate concept generating the source phrase based on the language used in all synonymous concept phrases within a single concept. This approach differs from existing approaches, which consider each synonymous concept phrase within a single concept to be independent.

5.1.2 Biomedical Semantic Annotation Evaluation

Evaluation of the annotation system will be done in two ways, using intrinsic and extrinsic evaluations. The intrinsic evaluation is done by comparing the proposed system's concept output to the concept output of the MetaMap system (Aronson, 2001), and determining precision and recall values. The extrinsic measure evaluates the performance of proposed system's concept output on a text summarization task.

5.1.2.1 Intrinsic Annotation Evaluation

The MetaMap system output is used as the gold standard to measure against. MetaMap takes a phrase as input and generates the best matching UMLS concepts. The proposed annotator output for the same phrase is then generated and compared to the concept(s) generated by MetaMap.

The corpus of 24 randomized controlled trial biomedical texts (see Section 3.3) is used to produce a set of noun phrases for evaluation. To eliminate variability introduced by various natural language processing techniques in the identification of sentences and phrases, MetaMap will produce a set of unique noun phrases from the corpus. Once the set of unique noun phrases is produced, MetaMap is used to map the concepts and its performance is measured based on the amount of time it takes to annotate phrases. The output of MetaMap is a list of the noun phrases and their concept mappings. The proposed annotator is then executed against the same set of noun phrases and its performance time measured. The proposed annotator's output is also a list of the noun phrases and their concept mappings. The proposed annotator concept output is then

compared against the MetaMap concept output, and precision and recall values determined.

There are two measures for the intrinsic evaluation: (a) precision and recall, and (b) phrase annotation time. The first measure looks at the accuracy of the concept mapping, and the second measure looks at the speed of the concept mapping.

Precison and recall measures are adapted to fit concept mapping rather than document retrieval (W. R. Hersh, Mailhot, Arnott-Smith, & Lowe, 2001). Precision is defined as the fraction of mapped concepts which are correct, and recall is defined as the fraction of correct concepts mapped, as shown in Figure 16:

$$\text{Precision} = \frac{\#\text{of correct concepts}}{\text{total \# of concepts mapped}}$$

$$\text{Recall} = \frac{\#\text{of correct concepts}}{\text{total \# of MetaMap concepts}}$$

Figure 16: Annotation Precision and Recall metrics

The timing performance of the mapping is defined as the amount of time needed to annotate a phrase. For a set of phrases, the average amount of time per phrase can be provided, as well as the shortest and longest times required to annotate a single phrase. Since the UMLS domain knowledge resources are large, even after preprocessing, they can impact the annotation system startup time since they are usually fully or partially loaded at runtime. To account for this, two metrics will be produced: (a) concept mapping time including annotation system startup time, and (b) concept mapping time without

annotation system startup time. The idea is to give some indication of the impact of annotation system startup time on the speed of annotation.

5.1.2.2 Extrinsic Annotation Evaluation

The extrinsic evaluation is designed to measure the performance of the annotator by using its concept mapping output to perform a task. The selected task is text summarization. The FreqDist and SumBasic summarizersare the summarizers used (see Section 5.2.4.2). Both summarizers only use concepts as the feature to select salient sentences, and so their performance is entirely reliant on the concepts identified in the texts. It is expected if the concept output is accurate, summarization performance will improve because the concepts will have identified important areas within a text.

The evaluation will be done by first annotating the evaluation corpus of 24 texts using the proposed annotator. The FreqDist and modified version of the SumBasic summarizers are then used to generate a summary of each of the 24 texts using the concept output from the proposed annotator. As a baseline comparison for the evaluation, the FreqDist and SumBasic summarizers will also be run in the same manner using the concept output from MetaMap. All of the summary output will be evaluated against manual summaries generated from domain experts using the ROUGE tool, as described in the summarization evaluation section (see Section 5.2.4).

5.2 Biomedical Text Summarization

For biomedical text summarization, two different approaches are adapted to use concepts rather than terms to identify salient portions of text within a source text. The

first is concept chaining, which uses ideas from existing research work done in lexical chaining and applied to summarization applications. The basic idea is to link together the concepts found in a text based on semantic types. The semantic types with the strongest chains are then representative of the text's main topics. The second approach uses the idea of reiteration, where an author repeats important points. Reiteration is reflected in the frequency of content terms used. Instead of using terms, the use of frequently occurring concepts is chosen instead. The novel contributions of the proposed semantic annotation component of the system are as follows: (a) the use of concepts and an associated semantic network to chain concepts together to find text themes (BioChain); (b) the use of concept, rather than term, frequency to identify text themes; (c) the development of a text summarization algorithm which matches the concept or term distribution of the source text to the generated summary (FreqDist algorithm); (d) the determination of the optimal length of a summary; and (e) the identification of the location within texts human summarizers draw text, and the incorporation of this information into a summarizer.

5.2.1 Biomedical Text Summarization Using Concept Chaining

This section describes a proposed summarization system called *BioChainSumm*, which utilizes the concept chaining approach. BioChainSumm applies the concepts and methods of lexical chaining to biomedical text using concepts rather than terms.

BioChainSumm uses identified biomedical concepts in the source text and chains them based on their biomedical semantic type(s). Figure 17 shows the flow of the proposed text summarizer BioChainSumm with concept chaining processing. The basic

idea is to first identify the strongest chains (as indicated by the number and type of concepts found in the source text), and then score each sentence in the source text by counting the number of strong chain concepts each sentence contains. The highest-scoring sentences are then extracted to form a summary.

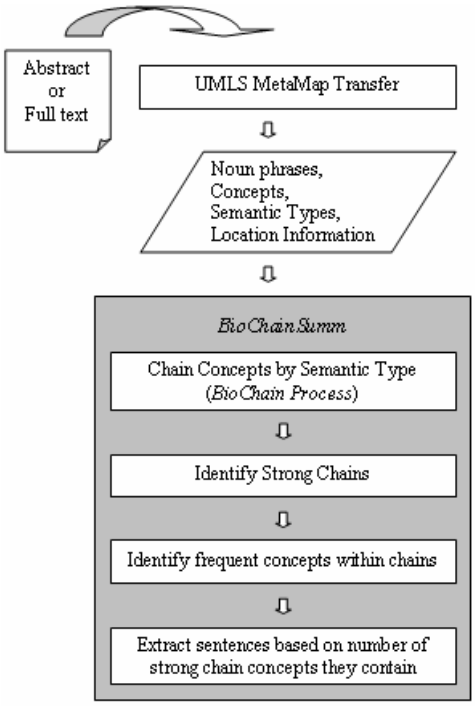

Figure 17: Proposed Text Summarization Process for BioChainSumm.

A concept chain is created for each semantic type defined in the UMLS Semantic Network (135 total). In this paper, a concept chain is also called a semantic type chain or a semantic chain. Each entry in a semantic type chain contains a list of concepts belonging to the semantic type. Each concept entry in a semantic chain contains the concept, sentence number, section number (roughly paragraph), and source noun phrase. If a concept belongs to multiple semantic types (i.e., multiple concept chains), the concept appears in multiple chains.

Once all concepts within a source text have been identified and linked into semantic type chains, the chains are then scored to identify the strongest chains. Each chain is scored by multiplying the frequency of the most frequent concept in the chain by the number of distinct concepts in the chain. This formula incorporates a combination of features as proposed by (W. P. Doran, Stokes, Dunnion, & Carthy, 2004) and Barzilay/Elhadad (Barzilay & Elhadad, 1997).

Once all chains are scored, strong chains, which identify the semantic types occurring most often, are determined. Lexical chaining research generally uses two standard deviations above the mean of all chain scores (Barzilay & Elhadad, 1997), and we follow that method. The strong chains are sorted into descending order based on their score. Strong concepts within the strongest chains are then identified using two different methods: (a) most frequent concept within each chain (multiple concepts having the same frequency count are considered equal) (abbreviated as MostFrequentStrongChainConcept) and (b) all concepts within a chain (abbreviated AllStrongChainConcepts). Sentences from the source text are then scored based on the number of strong concepts they contain. After sentences have been scored, sentences are sorted into descending order based on their score. The top-*n* sentences in the sorted list are extracted, re-sorted into their order of appearance in the original text, and presented to the user. Figure 18 presents the pseudocode for the summarization algorithm, which consists of several stages: concept mapping, concept chaining, strong chain identification, and sentence scoring and extraction. Each major stage in the process is detailed in the following sections.

```
Concept Chaining (Section 5.2.1.2):
    FOR EACH concept and semantic type found
        APPEND the concept to the semantic type chain
    END FOR

Strong Chain Identification (Section 5.2.1.3):
    // Score each chain
    FOR each semantic type chain
        FIND the concept with the highest frequency
        FIND the number of distinct concepts within the chain
        SET the chain score to concept frequency count * number of distinct concepts
    END FOR

    // Find minimum score required to be a strong chain
    COMPUTE ChainScoreAvg = AVG(all chain scores)
    COMPUTE ChainScoreStdDev = STDDEV(all chain scores)
    COMPUTE StrongChainMinScore = ChainScoreAvg + (2 * ChainScoreStdDev)

    // Find all strong chains
    FOR EACH semantic type chain
        IF (chain score >= StrongChainMinScore) THEN
            Chain is a strong chain
        END IF
    END FOR

Sentence Scoring and Extraction (Section 5.2.1.4):
    // Score sentences
    //      Two variations
    //       Variation #1: use most frequent concept in the strong chain
    //       Variation #2: use all concepts in the strong chain
    FOR EACH sentence
        IF (sentence CONTAINS strong chain concept)
            INCREMENT sentence score by number of
                strong chain concept instances in sentence
        END IF
    END FOR

    // Sentence Extraction
    //      Note: N is the number of sentences to output
    SORT sentences into descending order by sentence score
    EXTRACT top N sentences
    SORT top N sentences into original appearance order
    PRESENT top N sentences as summary
```

Figure 18: BioChainSumm summarization algorithm

5.2.1.1 Text-To-Concept Mapping

In the preliminary work on BioChainSumm, the UMLS MetaMap Transfer application is responsible for finding UMLS Metathesaurus concepts in biomedical text (United States National Library of Medicine, 2005b). It processes text through a series of stages (Aronson, 2001). The text is first split into sections, sentences are identified, and words are tokenized. Lexical resources or patterns are used to identify entities such as dates and locations. The part-of-speech tagger tags each word with its part-of-speech. The parser breaks sentences into phrases. The variant generation step identifies variants of a phrase, such as acronyms, synonyms, and derivational and spelling variations. The candidate retrieval stage retrieves all UMLS Metathesaurus concepts containing the variants. The retrieved candidate concepts are then evaluated, scored, and a final mapping determined by the highest scoring concept.

5.2.1.2 Concept Chaining

Identified concepts are chained based on their semantic type(s). A concept chain is created for each semantic type defined in the UMLS Semantic Network (135 total). Each concept chain contains a list of concepts belonging to the semantic type. Each concept entry in a concept chain (or semantic chain) contains the concept, sentence number, section number (roughly paragraph number), and source text noun phrase. If a concept belongs to multiple semantic types (i.e., multiple concept chains), the concept appears in multiple chains. Concept disambiguation is not explicitly implemented.

5.2.1.3 Identification of Strong Chains

There has been no definitive measure for scoring chains, and the literature suggests changes in scoring methodology do not adversely impact chaining results (W. P. Doran et al., 2004). The original lexical chain paper by (Morris & Hirst, 1991) defines three types of strong chain features: (a) reiteration, (b) density, and (c) length. Reiteration is repetition of concepts throughout a text. Density is physical proximity of concepts; that is, concepts closer together are more likely to be related. Length is the number of concept instances within a chain. Our scoring method, shown in Figure 19, includes a combination of features as proposed by (W. P. Doran et al., 2004) and Barzilay/Elhadad (Barzilay & Elhadad, 1997). Once all chains are scored, strong chains, which identify the semantic types occurring most often in the source text, are computed. Lexical chaining research generally uses two standard deviations above the mean of all chain scores (Barzilay & Elhadad, 1997), as shown in Figure 20.

*Score(Chain) = Frequency of most frequent concept * number of distinct concepts*

Figure 19: Chain scoring

*Strong(Chain) = Score(Chain) > (Average(Scores) + 2 * StandardDeviation(Scores))*

Figure 20: Strong chain identification

5.2.1.4 Identification of Frequent Concepts and Summarization

Summarization identifies sentences most likely capture the main ideas of a text. BioChainSumm uses the sentence extraction method to generate a summary. Sentence extraction begins by first sorting the strong chains into descending order based on chain score explained in Section 4.3. In each strong chain, either the most frequent concept in

the chain is used (MostFrequentStrongChainConcept) or all of the concepts are used (AllStrongChainConcepts). Each sentence is assigned a score based on how many concepts it contains from each strong chain.

For MostFrequentStrongChainConcept, a sentence is scored higher if it contains the most frequent concept from a strong chain. Multiple concepts having the same frequency count are considered equal. When using AllStrongChainConcepts, a sentence receives a higher score if it contains any of the concepts from a strong chain. Each strong chain concept found increases the sentence score value by one. Once all sentences have been scored, the sentence list is sorted into descending order based on the computed sentence score. The top-*n* sentences are then extracted, where *n* is a user-defined upper bound on the number of sentences to select for a summary of the original text. After the specified number of sentences has been extracted, the sentences are re-sorted into their order of appearance in the original text, and presented to the user.

5.2.2 Biomedical Text Summarization Using Concept Frequency

Figure 21 shows an outline of a proposed algorithm called *FreqDist* to generate a summary given the full-text of some source (source text) using a frequency distribution approach. There are two stages: Initialization and Summary Generation. In the initialization stage, the unit items (terms, concepts, etc.) of the source text are counted to form a frequency distribution model of the text, and a pool of sentences from the source text is created. A summary frequency distribution model is created from the unit items found in the source text, and their frequency counts are initialized to zero. In the Summary Generation stage, new sentences are selected to be added to the summary.

Identifying the next sentence to be added to the summary is accomplished by finding the sentence which most closely aligns the frequency distribution of the summary to the frequency distribution of the original source text. For each sentence in the sentence pool, a candidate summary is first initialized to the summary generated so far, and then the sentence is added to the candidate summary. The candidate summary frequency distribution is then compared for similarity to the original source text frequency distribution. This similarity score is assigned to the sentence. After all sentences from the sentence pool have been evaluated for their contribution to the candidate summary, the highest scoring sentence is added to the summary and removed from the sentence pool. This process is iterative, and repeats until the desired length of the summary is reached.

```
Initialization:
// Note: '-model' means  'frequency distribution model'
INITIALIZE source-model to unit-items in source-text
INITIALIZE summary-model,
            candidate-model from source-model
  SET all frequency values to 0

INITIALIZE sentence-pool to source-text sentences

Summary Generation:
REPEAT
  INITIALIZE sentence-pool scores to 0
  INITIALIZE best-score to 0
  INITIALIZE best-sentence to first sentence in pool

  FOR each sentence-entry in sentence-pool
    INITIALIZE candidate-model from summary-model

    ADD sentence unit-item frequencies to candidate-model

    SET sentence-entry.score =
          similarity(source-model, candidate-model)

    IF sentence-entry.scorescore > best-score
      SET best-score to sentence-entry.score
      SET best-sentence to sentence-entry
    ENDIF
  ENDFOR

  ADD unit-items from best-scoring sentence
          to summary-model

  REMOVE best-sentence from sentence-pool

UNTIL desired summary size reached or
          sentence-pool exhausted
```

Figure 21: FreqDist - an algorithm for generating summaries using a frequency distribution approach.

Five similarity functions were compared to find which type of function worked best to evaluate a candidate summary's frequency distribution to the original source text frequency distribution. Each frequency distribution (candidate summary and original source text) is modeled as a vector of unit items. Similarity functions are then applied to the two vectors. Figure 22 shows the five similarity functions used. The notations are as follows: *ui* is unit item; *srcUIs* and *sumUIs* is all unit items in source text or candidate summary, respectively; *src(ui)* and *sum(ui)* is indexed unit item in the source text or candidate summary, respectively. Cosine similarity (Baeza-Yates & Ribeiro-Neto, 1999), Dice's coefficient (Dice, 1945), Euclidean distance and vector subtraction (Subhash, 1996) are all well-known vector comparison methods. In addition, an approach to vector model comparison considering only unit item frequency was tried (Lee, Chuang, & Seamons, 1997). Cosine similarity uses the cosine angle value between the vectors for similarity. Dice's coefficient looks at the number of common terms between the two vectors. Euclidean distance measures the distance between the vectors in Euclidean space. For vector subtraction, the absolute value of the difference of each unit item in each vector is summed to form a distance score. The unit item frequency approach attempts to simulate cosine similarity without the computational complexity by only considering unit item frequency (Lee et al., 1997).

$$score = \frac{\sum_{ui=1}^{srcUIs} sum(ui) \times src(ui)}{\sqrt{\sum_{ui=1}^{srcUIs} sum(ui)^2 \times \sum_{ui=1}^{srcUIs} src(ui)^2}}$$

(a) Cosine similarity

$$score = \frac{2 * count(srcUIs \cap sumUIs)}{count(srcUIs) + count(sumUIs)}|$$

(b) Dice's coefficient

$$score = (\sum_{ui=1}^{srcUIs} (sum(ui) - src(ui))^2)^{1/2}$$

(c) Euclidean distance

$$score = \sum_{ui=1}^{srcUIs} |(src(ui) \times sum(ui))|$$

(d) Unit item frequency

$$score = \sum_{ui=1}^{srcUIs} |(src(ui) - sum(ui))|$$

(e) Vector subtraction

Figure 22: Similarity functions to evaluate a candidate summary's frequency distribution to the original source text frequency distribution. (a) cosine similarity, (b) Dice's coefficient, (c) Euclidean distance (d) unit item frequency, and e) vector subtraction. Notations used: ui is unit item; srcUIs and sumUIs is all unit items in source text or candidate summary, respectively; src(ui) and sum(ui) is indexed unit item in the source text or candidate summary, respectively.

5.2.3 Biomedical Text Structure

To explore the text structure of biomedical texts, two studies are proposed. Both studies uses the corpus of biomedical texts annotated by domain experts with the sentences they would extract from the source text to form a summary. The sentences are selected from one sentence up to 20% of the number of sentences in the text, as well as the major section of the paper each selected sentence appears in.

The first study will determine the ideal size of a biomedical text summary. The study will generate system summaries at different compression ratios, starting at 20% and

working downward to 1%. The system summaries are evaluated using against model summaries, as described in the Biomedical Text Summarization Evaluation section. The model summaries will match the size of the system-generated summaries. Compression ratio (a percentage of the sentences from the source text), and a fixed number of sentences will be used.

The second study explores how people draw summary sentences from different sections of a text. This information can be used to weight different sections of a text more heavily than others if people draw sentences from one section more than another section. The sentences selected in the corpus by domain experts will be annotated with the section they were pulled from. Using this information, a study will be developed to see if there are any commonalities between the domain experts in what sections they pull summary sentences from. After this information is gathered, the concept frequency distribution summarizer, which has frequency as its only feature, will be modified to include weighted section information and the generated summaries will be evaluated to show the performance increase or decrease caused by the section weighting.

5.2.4 Biomedical Text Summarization Evaluation

Summarization evaluation is typically done by comparing a system-generated summary to summaries generated by people (C. Lin & Hovy, 2003), (Harnly, Nenkova, Passonneau, & Rambow, 2005). For the proposed research, the 2005 Document Understanding Conference (National Institute of Standards and Technology (NIST), 2005) was reviewed for its approaches to evaluating system-generated summaries. DUC2005 used two different approaches: NIST and ISI/Columbia. Both approaches

require the use of model summaries, which are manually created summaries of a source text. The system-generated summary and the model summaries are then compared to form an evaluation of the system-generated summary.

The NIST approach uses two manual methods for evaluation and one automated method. The manual methods are subjective measures of *quality* and *responsiveness*, and no comparison is done with the model summaries. The quality method is implemented by asking an evaluator to read the system-generated summary, and then answer a series of five questions about it which inquire about grammaticality, information redundancy, anaphora resolution, focus, and structure and coherence (National Institute of Standards and Technology, 2005b). The responsiveness evaluation (National Institute of Standards and Technology, 2005a) looks at the quality of information in the system-generated summary, and the granularity of the information supplied. The idea is to understand how well the system-generated summary responded to a specified information need. The automated method uses a tool called ROUGE (Recall Oriented Understudy for Gisting Evaluation) (C. Lin, 2005) for comparing a system-generated summary to a set of model summaries. The ISI/Columbia approach uses a manual method called the Pyramid method (A. Nenkova & Passonneau, 2004). The Pyramid method evaluates the common information content between a system-generated summary and a set of model summaries. Information content is defined as an expression of an idea without regard to the terms used to express it. This analysis is difficult to do automatically, and so a person is used to annotate all summaries with their information content, and the overlap of the information content between a system-generated summary and its model summary is measured using

frequency of occurrence. Both systems generate a score indicating how well the system-generated correlates with the summaries produced by people for the same text.

In order to evaluate the proposed research work in summarization, a corpus of biomedical texts and a set of associated model summaries produced from domain experts has been acquired. This was done by asking three domain experts (three Drexel University College of Medicine medical students in their final year of study) to manually generate extractive summaries from 24 biomedical texts (see Section 3.3). The students used the same sentences as the source text to produce their summaries. To evaluate a system-generated summary, the output of the summarizer is compared to the domain experts' (model) summaries using an automated tool. The proposed research uses the ROUGE evaluation method because (a) it is completely automated as compared to Pyramid, which requires at least some manual annotation, and (b) requires fewer summaries as compared to Pyramid, which requires about five summaries (Harnly et al., 2005).

The ROUGE (Recall-Oriented Understudy for Gisting Evaluation) tool (version 1.5.5) (C. Lin & Hovy, 2003) developed by the Information Science Institute at the University of Southern California is an automated tool which compares a system-generated summary from an automated system with one or more ideal summaries produced by people. No manual annotation of the system-generated summary or the domain-experts' summaries is required. Because of this, and because of its usage in recent DUC conferences, which gives it credibility in the text summarization community, the proposed research plans to use ROUGE. The ideal summaries are called models, and were generated by three medical students in their final year of study. ROUGE uses n-

gram co-occurrence to determine the overlap between a summary and the models. An n-gram can be considered as 1 or more consecutive words. ROUGE was used in the 2004 and 2005 Document Understanding Conferences (DUC) (National Institute of Standards and Technology (NIST), 2005) as the evaluation tool. The ROUGE parameters from the DUC 2005 conference (National Institute of Standards and Technology (NIST), 2005) will be used to evaluate system summarizer performance. Two recall scores are extracted from the output of ROUGE to measure each summarizer: ROUGE-2 and ROUGE-SU4. ROUGE-2 evaluates bigram co-occurrence while ROUGE-SU4 evaluates *skip-bigrams* with a maximum distance of 4 words. ROUGE-2 and ROUGE-SU4 are also the measures used by DUC 2005. The recall scores indicate the n-gram overlap between the source text and the model summaries. It is difficult to compare ROUGE results outside of the corpus and model summaries used in the evaluation. For this reason, we gathered several summarizers from publicly-available sources in order to provide some meaningful comparison among them using the same corpus and set of model summaries.

The Pyramid method (A. Nenkova & Passonneau, 2004) is mostly a manual evaluation effort, although there has been some recent work done to automate part of the evaluation (Harnly et al., 2005). Pyramid evaluation begins by identifying all units of model summary text no bigger than a clause which express an idea. These clauses are called contributors. The contributors all expressing the same idea are gathered together into a Semantic Content Unit (SCU). An SCU is composed of a unique index (i.e., SCU1, SCU2, etc.), a weight which is the number of model summaries it appears in, and a label which expresses the idea of the contributors (A. Nenkova & Passonneau, 2004). After all SCUs have been identified, a pyramid is formed based on the SCU

weights, where each level of the pyramid has SCUs of the same weight, with the highest weighted SCU at the top of the pyramid and descending weighted SCUs forming lower levels of the pyramid. A system-generated summary is then also annotated in the same way as the model summaries. The pyramid expresses the content which should be in the summary, with the top level expressing the most important content. The system-generated summary is given a score computed by first counting the SCU overlap between the model summary SCUs and the system-generated SCUs at each level of the pyramid, multiplying the overlap count by the tier in the pyramid where the SCU occurs, and summing the overlap scores for each level of the pyramid (A. Nenkova & Passonneau, 2004).

5.2.4.1 Model Summaries

The proposed summary evaluation requires that the system-generated summaries be compared with summaries produced by people. A corpus of manual summaries of the selected 24 texts has been acquired by having three students from the Drexel University College of Medicine in their final year of study select 20% of the sentences from each paper to form a manual summary of each paper. The selection of sentences forms a model summary. For each of the 24 papers, a corpus has been built which has three model summaries (at 20% compression) as well as an abstract, which also serves as a model summary, for a total of four model summaries. In future work, the medical students will take the sentences they selected from each of the 24 papers and order them by importance and also indicate the section of the paper each sentence is from (e.g., Introduction, Method, Result, Conclusion, and so forth). This information will allow for studies

exploring the ideal size of a summary (from 1%-20%, as well as fixed amounts, such as five sentences) and the location of extracted sentences. While the work in summarization of news has found the ideal size of a summary to be 20% (Goldstein, Kantrowitz, Mittal, & Carbonell, 1999), the ideal size of a biomedical text summary, particularly randomized controlled trials, has not yet been determined.

5.2.4.2 Summarizers used for evaluation

Eight extractive summarizers are used for our evaluation and briefly explained in each subsection that follows. The BaseLine and SumBasic summarizers were implemented for this evaluation, and each have multiple variations. The Lemur MMR, MEAD, AutoSummarize in Microsoft Word, OTS, and SWESUM summarizers are publicly available. eight additional extractive summarizers were randomly selected based on the type of summarization method and availability. There are roughly four categories of summarizers selected: baseline, frequency-based, multiple feature, and redundancy-sensitive, and we select two summarizers in each category. The two baseline summarizers are Baseline-Lead, which sequentially selects the first 20% of sentences in the source text, and Baseline-Random, which randomly selects 20% of the sentences in the source text. The frequency-based summarizers are AutoSummarize in Microsoft Word (Microsoft Coporation, 2002)and Open Text Summarizer (OTS) (Rotem, 2003). AutoSummarize is a feature of the Microsoft Word (Microsoft Coporation, 2002) word processing software, and although exact details of the algorithm are not documented, online help for the product indicates sentences using frequently-used words are given a higher score than sentences containing low frequency words. OTS is an open source

project where stemming can also be performed to eliminate word variations. The two summarizers using multiple features to identify sentences are SweSum (Dalianis, 2000) and MEAD (Radev et al., 2004). SweSum is a multi-lingual summarizer for Swedish and English text using features such as sentence position and numerical data identification. MEAD is a single and multi-document summarizer using features such as position of sentence within the text, overlap of each sentence with the first sentence, sentence length, and a centroid method based on a cluster of related documents. Finally, the two summarizers which reduce information redundancy are Lemur Maximal Marginal Relevance (MMR) (The Lemur Project, 2006)and SumBasic (A. Nenkova & Vanderwende, 2005). Lemur MMR iteratively selects sentences having a high query similarity to an automatically-generated query, and which are also maximally dissimilar to sentences already included in the summary. SumBasic uses a probability distribution of terms in the text, and reduces term probability as sentences containing the terms are selected. SumBasic was also adapted to use concepts as the input source, rather than terms.

Each summarizer generated a summary that was equal to 20% of the length of the source text. For example, if a source text consists of 100 sentences, then 20 sentences are selected and extracted by each summarizer and presented as the summary. Selecting a summary size was problematic. The news summarization domain typically selects a size of less than five sentences. This represents about 20% of the size of a typical news story (Goldstein et al., 1999). It has been generally thought that a summary should be no shorter than 15% and no longer than 35% of the source text (E. H. Hovy, 2005). The following is a brief description of the approaches used by each summarizer.

5.2.4.2.1 BaseLine

The purpose of the baseline summarizers is to give some indication of the level of performance of a naïve summarization implementation. Two baseline summarizers were implemented. The first baseline summarizer is called Baseline-Lead, and it sequentially selects the first 20% of sentences in the source text. The second baseline summarizer is called Baseline-Random, and it randomly selects 20% of the sentences in the source text.

5.2.4.2.2 BioChainSumm

The BioChain summarizer (i.e., BioChainSumm) uses the concept chaining approach described in Section 4. Several different implementations are evaluated. BioChain is divided into two primary approaches based on the concept selection criteria in strong chains for the sentence scoring: 1) using the most frequent concept in the strongest chains (MostFrequentStrongChainConcept), and 2) using all of the concepts within the strongest chains (AllStrongChainConcepts). Each of these primary approaches also implements variations by providing each sentence (or each chain) with a certain weight. The first variation is adding a sentence position heuristic where each sentence is initially scored as 1/N, where N is the number of sentences in the source text. The concept chain score is then added to this base score. The idea is that sentences in the beginning of the text are more important than sentences at the end of a text (Dalianis, 2000). It was also added in response to informal qualitative feedback from one of our biomedical domain experts, a final-year medical student, that the generated summaries did not include enough introductory information. A second variation is to filter out semantic types. The idea is that not all semantic types are important to focus on for a particular user's information

needs. Our domain expert identified the semantic types important within the oncology clinical trial domain. A chain is scored as zero if not in the list shown in Table 7. The final variation is to combine the sentence position heuristic with semantic type filtering.

Table 7: Important Semantic Types for oncology clinical trials

UMLS Semantic Type	UMLS Semantic Type Name
T37	Injury or Poisoning
T51	Event
T52	Activity
T61	Therapeutic or Preventative Procedure
T62	Research Activity
T67	Phenomena or Process
T81	Quantitative Concept
T169	Functional Concept
T170	Intellectual Product
T191	Neoplastic Process

5.2.4.2.3 Lemur MMR

The Lemur MMR application (The Lemur Project, 2006) is a summarizer built using the idea of Maximal Marginal Relevance (MMR) (Carbonell & Goldstein, 1998). Carbonell describes marginal relevance as finding relevant sentences which contain minimal similarity to previously selected sentences. Maximal marginal relevance in text summarization attempts to maximize the dissimilarity of the information content between sentences within a summary. Lemur MMR iteratively selects sentences based on similarity to a query, and then selects sentences having a high query similarity which are also maximally dissimilar to sentences already included in the summary. In the evaluation no query was specified, so Lemur MMR automatically generated a query based on the source text. No domain specific knowledge sources were provided to the summarizer.

5.2.4.2.4 MEAD

MEAD (Radev et al., 2004) is a single- and multiple-document summarizer using multiple features to score sentences. Some of the features include position of sentence within the text, overlap of sentence with the first sentence, sentence length, and a centroid method based on a cluster of related documents. For the evaluation, we used the MEAD Demo located at http://tangra.si.umich.edu/clair/md/demo.cgi. No domain specific knowledge sources were provided to the summarizer.

5.2.4.2.5 AutoSummarize

The AutoSummarize is a feature of the Microsoft Word (Microsoft Coporation, 2002) word processing software. AutoSummarize is based on a word frequency algorithm. Each sentence in a document is given a score based on the words the sentence contains. Although the exact details of the algorithm are not documented, the online help for the product states that sentences using frequently-used words are given a higher score than sentences containing low frequency words. No domain specific knowledge sources were provided to the summarizer.

5.2.4.2.6 Open Text Summarizer (OTS)

The Open Text Summarizer (OTS) is an open source project which provides a library for summarizing arbitrary texts (Rotem, 2003). It is based on a frequency-based approach, where the most frequently occurring words are assumed to be indicators of the text theme. Stemming can also be performed to eliminate word variations, so that for

example, *year* is not distinguished from the plural *years*. Users can provide their own set of stemming rules. For the evaluation, the pre-built summarization library was used with no change to the stemming rules. No domain-specific resource is used.

5.2.4.2.7 SumBasic

The SumBasic algorithm (A. Nenkova & Vanderwende, 2005) is a recent frequency-based algorithm. The original algorithm works using terms as the unit item to count. For this evaluation, we have modified it so that the unit items can be terms or concepts. SumBasic incorporates a component for ensuring coverage of weaker concepts within a text. There are four steps in the algorithm. The first is to determine the probability distribution of all concepts found within a source text by computing the number of times a unit item appears in the text divided it by the total number of unit items found in the text. The second step is to score each sentence by summing the probabilities of all unit items within a sentence. The third step determines the sentence to be extracted by finding the highest-scoring sentence. The fourth step then reduces the probability of each unit item appearing in future extracted sentences by multiplying each probability of each unit item in the last extracted sentence by itself. The implementation using terms as unit items first had a stop word list applied. For the implementation using concepts, the UMLS Metathesaurus was used as the domain-specific resource (the same as the BioChainSumm summarizer).

5.2.4.2.8 SWESUM

SweSum (Dalianis, 2000) is a multi-lingual summarizer for Swedish and English text. SweSum uses multiple features for scoring sentences, such as sentence position and numerical data identification. Sentences located earlier in a text are scored higher than sentences at the end of the text. Sentences containing numerical data are given additional weight. User-specified keywords can also be provided to boost sentence scores for those sentences containing the keywords. For the evaluation we used the online version located at http://swesum.nada.kth.se/index-eng-adv.html. The text type was set to *Academic* and the summarization size was to 20%. No other parameters were set, and no domain specific knowledge sources were provided to the summarizer.

6. LITERATURE REVIEW

The review of literature is divided into four sections. The first two sections describe semantic annotation. Domain-independent semantic annotation is first described followed by semantic annotation for biomedical texts. The last two sections describe two recent text summarization methods: lexical chaining of terms and term frequency. These two methods are adapted in the proposed research to use concepts rather than terms.

6.1 Semantic Annotation

Semantic Annotation is a method for providing machine-understandable information based on meaning. One way to provide meaning by creating ontologies, and then linking information within a document to specifications contained in the ontology using a markup language (Berners-Lee et al., 2001). Ontologies are conceptualizations of a domain that typically are represented using domain vocabulary (Chandrasekaran et al., 1999). Semantic annotation is the process of mapping instance data to an ontology. Benefits of adding meaning to the Web include: query processing using concept-searching rather than keyword-searching (Berners-Lee et al., 2001); custom Web page generation for the visually-impaired (Yesilada, Harper, Goble, & Stevens, 2004); using information in different contexts, depending on the needs and viewpoint of the user (Dill et al., 2003); and question-answering (Kogut & Holmes, 2001).

6.1.1 Semantic Annotation for General Text-based Documents

Manual annotation can be done using tools such as Semantic Word (Tallis, 2003), which provides an environment for authoring as well as marking up documents from within a single interface. However, manual approaches suffer from several drawbacks. Human annotators can provide unreliable annotation for many reasons: complex ontology schemas, unfamiliarity with subject material, and motivation, to name a few (Bayerl, Lüngen, Gut, & Paul, 2003). It is expensive to have human annotators markup documents (Cimiano, Handschuh, & Staab, 2004). A human annotator may not consider using multiple ontologies (Dingli, Ciravegna, & Wilks, 2003). Documents and ontologies can change, requiring new or modified markup, which leads to document markup maintenance issues (Dingli et al., 2003). Finally, the volume of existing of existing documents on the Web can lead to an overwhelming task for humans to manually complete (Kosala & Blockeel, 2000). For all these reasons, manual efforts have been identified as a "knowledge acquisition bottleneck" (Maedche & Staab, 2001).

Semantic Annotation Platforms (SAPs) are systems for performing semi-automatic semantic annotation. Semi-automatic systems, rather than completely automatic systems, are used because it is not yet possible to automatically identify and classify all entities within source documents with complete accuracy (Popov et al., 2003). There are many advantages of semi-automatic annotation, such as providing document volume scalability by reducing or potentially eliminating the human workload (Dill et al., 2003), and providing annotation services where the source document is not available for write access (Dill et al., 2003). SAPs vary in their architecture, information extraction tools and methods, initial ontology, amount of manual work required to perform

annotation, performance and other supporting features, such as storage management of ontologies, knowledge bases, and annotations. Some SAPs were designed for a specific domain, but usually can be adapted to fit new domains.

Figure 23 shows the general abstraction layer of a SAP. The Application layer is responsible for providing an end-user interface to the annotation services provided by a SAP. Examples include facilities for annotating a document or document set and then potentially confirming the annotations before committing them, providing a query interface for searching annotations, and providing a user interface for configuring the information extraction component. The Upper Interfaces layer is primarily the application programming interface (API) layer. A set of programmatic interfaces are described in this layer. Applications call the defined APIs in order to perform actions on behalf of an application. The APIs can be quite numerous, covering annotation, information extraction, search, storage management, and many other provided services. The Upper Interface APIs are designed to shield the applications from changes in the Lower Interface. The Lower Interface contains the actual components that perform work for an application. The Upper Interface will remain consistent to an application, but the Lower Interface is expected to change based on the various components used. For example, the Information Extraction component may switch from a pattern-based tool to a statistical tool, and it is unlikely the programmatic interface is the same for both. The Upper Interface implementation will need to change to accommodate the various Lower Interface components. Finally, the Storage Layer is designed to provide storage and storage management facilities for storing long-term data such as annotations and knowledge bases.

Examples of existing annotation platforms include AeroDAML (Kogut & Holmes, 2001), Armadillo (Dingli et al., 2003), MnM (Vargas-Vera et al., 2002), MUSE (Maynard, 2003), Ont-O-Mat (S. Handschuh, Staab, & Ciravegna, 2002), and SemTag/Seeker (Dill et al., 2003). The platforms are primarily distinguished by (a) the features offered, (b) the information extraction method used to find entities within documents, and (c) whether or not they are extensible. Features offered by SAPs include ontology and knowledgebase management (storage, editors), access APIs, annotation storage to allow multiple ontologies/annotations per document, and information extraction methods.

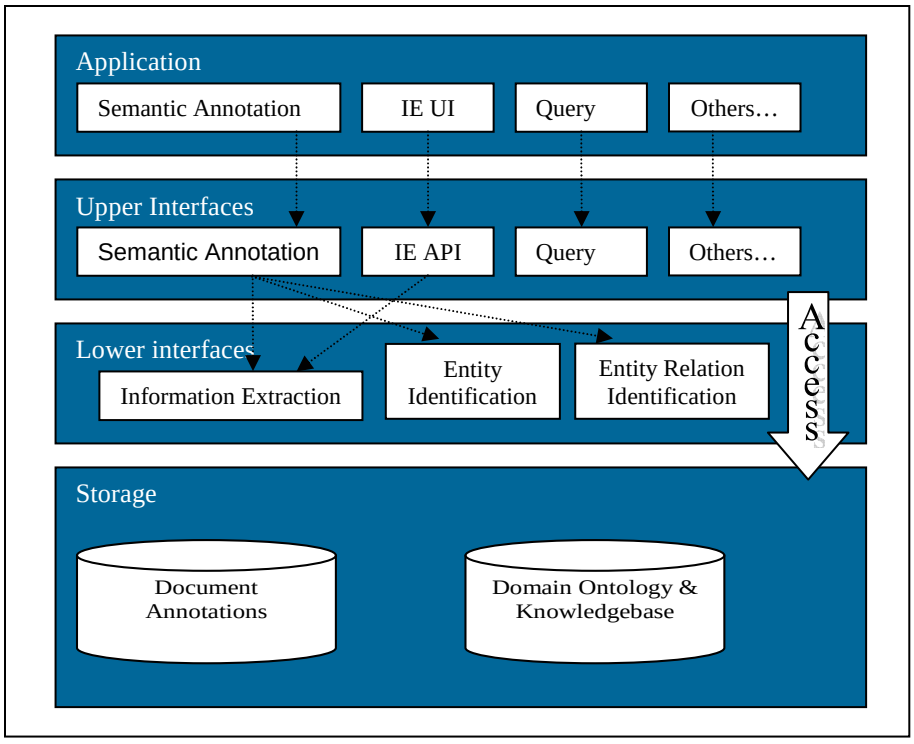

Figure 23: General architecture of a semantic annotation platform.

Semantic annotation requires an ontology in order to perform concept instance mapping. Ontologies are usually architected using levels, such as upper and lower. The upper ontology consists of general concepts, while the lower ontology has a deeper specialization of the upper ontology concepts (Missikoff, Navigli, & Velardi, 2002). Some semantic annotation platforms place the responsibility on the user for constructing an initial ontology. Examples include MUSE (Maynard, 2003) and Ont-O-Mat (S. Handschuh et al., 2002). Other platforms provide an initial ontology as part of their development. The KIM platform provides an ontology called KIMO that is designed to provide a minimal open-domain ontology, and is based on OpenCyc, WordNet, DOLCHE and other upper-level resources (Popov et al., 2003). KIMO is composed of approximately 250 classes and 100 attributes and relations, and the specialization of classes is derived from an analysis of a corpus of general news (Popov et al., 2003). The Seeker platform uses TAP, which is a shallow knowledgebase that contains information about a broad range of popular culture subjects, such as movies, sports, and so forth (Dill et al., 2003). The TAP knowledgebase has about 72,000 labels that are used to tag instances found in documents. The MnM platform uses a hand-crafted ontology called KMi (Knowledge Management Institute) (Vargas-Vera et al., 2002). The AeroDAML platform uses the commercial product Aderotext, and utilizes an upper-level ontology based on Wordnet, while the lower-level ontology uses the common knowledgebase of AeroText (Kogut & Holmes, 2001). Armadillo provides an example of a platform where the initial ontology is very light weight, consisting of an address-book type of ontology where members of a computer science department are discovered and populate address information, such as name, phone number, address, and so forth (Dingli et al., 2003).

6.1.1.1 Classification of Platforms

Current semantic annotation platforms use several methods for information extraction (IE) from Web documents. Figure 24 shows a hierarchical classification of annotation platforms, and this classification can be used to organize the platforms performing semantic annotation. While semantic annotation platforms have many aspects, the information extraction approach currently used to find entities within text has the most impact on the effectiveness of the platform. For this reason, the IE approach of each platform is used to organize the platforms. As semantic platforms develop, it is anticipated that the classification structure will adapt to newer approaches as well.

The top-level approach is multi-strategy, which uses a combination of the lower level approaches. A platform using a multi-strategy approach is able to adapt its IE methods based on the text it is processing in order to obtain the best results. The multi-strategy approach uses a high-level identification of text genre, and then executes the appropriate IE methods. This is in contrast to lower-level approaches using newer IE algorithms such as LP^2 (Ciravegna, 2001) in platforms such as KIM (Popov et al., 2003), which are able to use machine-learning to perform rule induction using both structural and linguistic information. No semantic annotation platform to date is using a complete multi-strategy approach incorporating both pattern and machine-learning approaches. The MUSE system comes the closest by using text features and then conditionally executing rules based on the text features (Maynard, 2003).

The two primary lower levels are pattern-based methods and machine-learning methods. Pattern-based methods are systems composed of manual rules. The rules are typically hand-crafted rules that define how entities can be found in text (Maynard,

2003). Examples of such systems are AeroDAML (Kogut & Holmes, 2001), MUSE (Maynard, 2003) and SemTag/Seeker (Dill et al., 2003). A limiting factor on the scalability of such systems is that the manual rule generation process can be maintenance intensive. Each time a data source changes, the pre-defined rules may also need to be changed. Machine-learning approaches use pre-annotated examples to learn how to identify entities. Rules are learned automatically, and this type of rule learning is currently used in platforms using the Amilcare toolkit (University of Sheffield, 2002), which implements the LP^2 algorithm (Ciravegna, 2001). Examples of systems using rule learning are Ont-O-Mat (S. Handschuh et al., 2002) and MnM (Vargas-Vera et al., 2002). Hidden Markov Model (HMM) is an example of another machine-learning approach that can also be used. HMM is not currently being used by any of the semantic annotation platforms as an information extraction method.

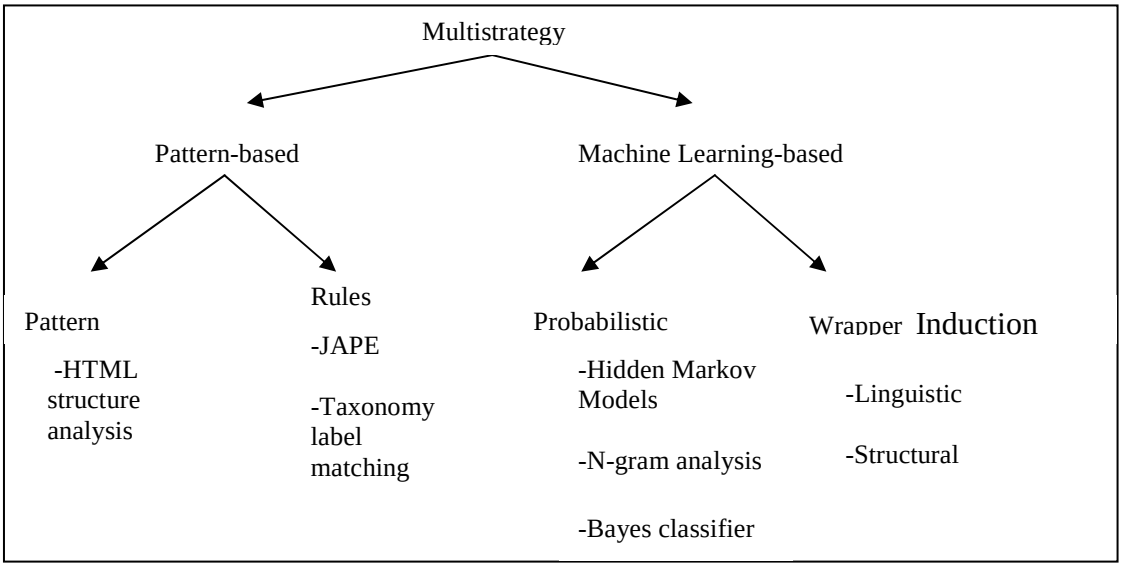

Figure 24: Classification of Semantic Annotation Platforms

6.1.1.1 Pattern Discovery

Patterns are also a widely-used technique in semantic annotation platforms. Pattern discovery works by taking a few seed samples, finding entities based on the patterns, expanding the seed samples with patterns from the new entities found, and repeating the process until no more instances are found, or the user stops the iterative process (Brin, 1998). Patterns can exploit known linguistic patterns, such as Hearst patterns (Hearst, 1992), to find entities, as is done in the Ont-O-Mat using PANKOW platform (Cimiano et al., 2004). The Ont-O-Mat platform has been updated to replace the Amilcare component with a pattern-based component, called PANKOW. The Ont-O-Mat platform (S. Handschuh et al., 2002) demonstrates the usefulness of an extensible semantic annotation platform, where components can be replaced without losing or duplicating the features already available (Cimiano et al., 2004). The Armadillo platform is also example of a platform that uses a small set of initial seeds to begin a pattern discovery process.

6.1.1.2 Rules

Rules can be manually-generated, as MUSE (Maynard, 2003) does with the JAPE grammar (Cunningham, Maynard, & Tablan, 2000), or they can be generated with machine-learning techniques. In this case, rules are considered the rules that are a subset of the pattern classification in Figure 24. The difference is the rules are initially manually specified by the user. Rules can take many forms. In the Seeker platform, the rules are labels (Dill et al., 2003). SemTag, the semantic tagging component of Seeker, uses the labels stored in the knowledgebase to find instances of ontology concepts, and then uses statistical likelihood to determine where in the ontology tree an instance is most likely to

be contained (Dill et al., 2003). The process is not completely automatic, however, as an initial set of training data must be provided. The authors report 700 entries as the initial size of the training set (Dill et al., 2003). In the MUSE platform, rules are written using the JAPE grammar (Maynard, 2003). The MUSE platform is an interesting rule-based system because it can adapt the rules used in entity identification depending on text attributes, such as language, document type, document source, and so forth (Maynard, 2003). The result is a rule-based platform that performs competitively with machine-learning based platforms (Maynard, 2003).

6.1.1.3 Machine Learning

Platforms based on machine learning are divided using the machine-learning method, which are currently two approaches: probabilistic and wrapper induction. The more common of the two approaches is wrapper induction. It is anticipated that future research in semantic annotation will take advantage of machine-learning based approaches, because they help to relieve the manual effort required in building rules, which is the primary drawback of pattern-based approaches. For example, the Rainbow system is a domain-specific annotation system developed for aggregating product information from multiple web sites, and then providing search over the aggregated information (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004).

6.1.1.3.1 Probabilistic

Probabilistic methods use algorithms such as Hidden Markov Models (HMMs) to perform information extraction. For example, the DATAMOLD tool (Borkar, Deshmukhy, & Sarawagiz, 2001) uses HMMs for information extraction, but it has not

yet been integrated into a semantic annotation platform. The Seeker platform uses a probabilistic approach to help eliminate mis-classification caused by its simple tagging approach within the SemTag component (Dill et al., 2003). SemTag has a pool of approximately 72,000 labels that it uses to find entity instances within text. It is likely that some of the labels will be duplicated but contained in different parts of the ontology. SemTag incorporates an algorithm as part of its tagging process to determine the probability of a particular label belonging to a particular class in the ontology. In the Rainbow project, where product catalog information from a specific domain is extracted from multiple sites, the information extraction portion project uses Hidden Markov Models as its primary method. While the performance is competitive with other approaches, the resulting system developed from Rainbow has not yet been incorporated into a general purpose semantic annotation platform. The most likely reason is that the generated model is domain-specific (Svab Ondrej, Labsky Martin, Svatek Vojtech, 2004).

6.1.1.3.2 Wrapper Induction

A wrapper is a function from a page to the set of tuples it contains (Kushmerick, Weld, & Doorenbos, 1997). Wrappers are used when there exists a repeatable structure to extract information from. Many web sites have pages generated using from a back-end database, and the pages generated follow a common template. The purpose of wrappers is to reverse the page generation process in order to retrieve the original database tuples. Wrappers can be hand-crafted, or they can be learned. Manual wrappers require the user to mark areas of interest within a document. The machine can then extract entities from documents with a similar structural format as the manually marked-up document

(Vargas-Vera et al., 2002). Kushmerick (1997) defined a method for performing wrapper induction, where the wrappers are automatically learned from example query responses from a data source. Wrappers are most effective when the data is presented in a structured format, such as product catalogs (Dingli et al., 2003).

Wrappers can also be linguistic-based, where the wrapper induction process discovers linguistic rules for identifying entities (Vargas-Vera et al., 2002). Amilcare implements the LP^2 algorithm (Ciravegna, 2001), which performs rule induction using both linguistic and structural information (Ciravegna, 2001). It is intended to provide a combination of IE approaches such as wrapper induction (Kushmerick et al., 1997) and linguistic approaches from the natural language processing community. The Amilcare toolkit is used by several platforms, such as MnM (Vargas-Vera et al., 2002), and Ont-O-Mat (S. Handschuh et al., 2002).

6.1.1.4 Multi-strategy

A multi-strategy approach uses a combination of both machine-learning and pattern-based approaches. No surveyed SAP uses the multi-strategy approach. However, this approach could be used by a platform which adapts its processing based on the document genre or specific text features. Such adaptive processing would apply the most effective processing for each type of document. The closest SAP that performs adaptive processing is the MUSE system (Maynard, 2003). MUSE uses a pipeline approach to identify text features that can be used to perform conditional processing. MUSE uses the JAPE grammar (Cunningham et al., 2000) to define rules to perform semantic annotation, and the JAPE rules are fire conditionally, based on text features. The conditional processing approach used by MUSE makes it competitive with machine-learning based approaches (Maynard, 2003).

Table 8 shows the author-reported performance of various platforms, with the exception of AeroDAML, Ont-O-Mat using Amilcare and SemTag, whose authors did not provide complete performance information. The standard measures of Precision, Recall, and F-measure, taken from the information retrieval field, were used by the remaining SAP authors in determining annotation effectiveness. In the general definition of recall and precision shown below, *accurate* and *inaccurate* refer to annotations generated semi-automatically by a SAP, while *all* refers to all annotations generated by a human annotator.

$$Annotation\ Recall = \frac{accurate}{all}$$

$$Annotation\ Precision = \frac{accurate}{accurate + inaccurate}$$

F-measure is the harmonic mean of precision and recall. The highest performing machine learning-based platform is MnM. For pattern-based platforms, MUSE is best. The worst performing is Ont-O-Mat using PANKOW. PANKOW is a recent effort to use unsupervised learning in a pattern-based system, and performance improvements are expected as the system develops further (Cimiano et al., 2004).

Table 8: Semantic annotation platform information extraction methods and performance measurements.

Platform	IE Method	Precision	Recall	F-Measure
Armadillo	Pattern Discovery	91	74	87
KIM	Manual Rules	86	82	84
MnM	Wrapper Induction	95	90	n/a
MUSE	Manual Rules	93	92	93
Ont-O-Mat using PANKOW	Pattern Discovery	65	28	25
SemTag	Semi-automatic Rules	82	n/a	n/a

Semantic Annotation Platforms were developed to provide a level of automation to the semantic identification of text within documents, and to also overcome the limitations of manual annotation, such as annotator motivation and domain knowledge (Bayerl et al., 2003), changing and multiple ontologies, and providing multiple perspectives (Dill et al., 2003).

Several semantic annotation platforms currently exist, distinguished primarily by their annotation method, as that component has the largest impact on the effectiveness of semantic annotation. The two primary approaches are pattern-based and machine learning-based. Machine learning algorithms often perform more effectively than pattern-based methods, but the MUSE system shows that a rule-based system using conditional processing can perform as well as a machine learning system (Maynard, 2003).

Applications making use of annotations generated by semantic annotation platforms are beginning to appear. These applications are information retrieval by semantic entity rather than by keyword, custom web-page generation based on different user needs and perspectives, question-answering systems, and visualization of a domain.

6.1.2 Semantic Annotation for Biomedical Text-based Documents

Most work in semantic annotation for biomedical text is performed to support semantic indexing/retrieval and data mining of biomedical texts (Aronson, 2001). Concepts are identified in the text and then indexed. Users can then retrieve the biomedical texts using the biomedical concept rather than searching by terms. Indexing biomedical text can be done in an offline mode. There is no requirement to index a text while the user is waiting for a response. Indexing text is also primarily interested in finding all possible concepts within a text. For example, the phrase *lung cancer* can map to three concepts: {*lung*, *cancer, lung cancer*}. Some biomedical annotation systems, such as MetaMap (Aronson, 2001), find a single best match for phrases within a source text in order to understand the author's intentions. For example, *lung cancer* maps to the single concept {*lung cancer*} rather than the three distinct concepts listed above. The phrase is clearly talking about lung cancer, and not a lung and any type of cancer, as would be the case if the concepts {*lung*} and {*cancer*} were identified as unique concept instances. While most of the work identified here has been designed for semantic indexing, the methods used to find concepts in source text, as well as the methods used to evaluate the resulting system, are important to review.

6.1.2.1 General Approach

A common approach to most systems is to use the following method:

1. Construct a unit of analysis by generating subsets of words in the source text (e.g., phrase, sentence)

2. (Optional) Normalize the source text unit using UMLS (National Library of Medicine, United States, 2006b) by (a) removing possessives, (b) replacing punctuation with spaces, (c) removing stop words, (d) convert words to lower-case, (e) breaking a string into constituent words, and (f) sorting words into alphabetical order

3. For each word in the input phrase, build a set of all concepts containing the word;

4. Find the intersection of the concept sets;

5. (Optional) Find the best matching concept based on the common word membership between the source text and concept text

The unit of analysis is usually a phrase. Noun phrase are typically used because they have more content information (Elkin et al., 1988). Other unit of analysis include sentences (W. R. Hersh & Greenes, 1990), and unordered terms (Zou et al., 2003). Each unit of analysis has pros and cons. Phrases lose contextual value when a single concept is spread across multiple phrases (P. Nadkarni, Chen, & Brandt, 2001), (Zou et al., 2003). Sentences overcome the loss of context when using phrases, but suffer from several new problems (P. Nadkarni et al., 2001): 1) finding invalid concepts when using all permutations of words in the sentence, and 2) since more words are in a sentence than in a phrase, concept identification becomes more computationally complex. It is also possible to eliminate natural language processing and treat the entire text as a set of independent terms. The IndexFinder system (Zou et al., 2003) is an example of this approach, which overcomes the computational limitations of the sentence approach. A problem with this approach is that concept cannot be linked back to their source (e.g.,

phrase or sentence), since the words are permuted throughout the text. Another problem is the over-generation of concepts, which the authors control with various filters, such as semantic types and term ranges. The unit of analysis for each of the reviewed systems, as well as the method used to identify the unit of analysis, is shown in Table 9.

Four types of matches between the terms of a source text and UMLS concept phrase have been defined in the literature (Aronson, 1996):

- *None*: there is no match of terms;

- *Simple*: there is an exact match between the terms in the text of the source and the UMLS concept text;

- *Partial*: one or more terms do not match exactly;

- *Complex*: two or more sets of terms map to distinct concepts.

The most common types of matches are Simple and Partial. The last column in Table 9 lists for each system whether the system supports Simple, Partial or both. Typically the match types form a simple hierarchy. All systems support None and Simple. Systems which support Partial also support None and Simple, and systems supporting Complex support None, Simple, and Partial as well.

The identification of a phrase is typically done using natural language processing (NLP), where a part-of-speech tagger is first executed over a sentence, and then using parts of speech to build a noun phrase. Other approaches include using NLP Heuristics which do not use strict NLP processing (P. M. Nadkarni, 1997), moving window where all combinations of a sentence words are generated to build a phrase (Wollersheim et al., 2002), and using pre-defined block text sizes (W. R. Hersh & Greenes, 1990).

Table 9: Attributes of Several Biomedical Annotation Systems

System	Unit of Analysis	Identification Method	Mapping
Concept Locator	Phrase	NLP Heuristics	Simple, Partial
Dynamic Taxonomy	Phrase	Moving Window	Simple
KnowledgeMap	Phrase	NLP	Simple, Partial
IndexFinder	Unordered Terms	All words, excluding stop words	Simple, Partial
MetaMap Transfer	Phrase	NLP	Simple, Partial, Complex
PhraseX	Phrase	NLP	Simple
SAPHIRE	Sentence	Text block	Simple, Partial
SENSE	Phrase	User-specified Query	Simple

6.1.2.2 Biomedical Semantic Annotation Platforms

Multiple systems have been previously built to implement various ways to perform concept annotation of biomedical texts. In the following sub-sections, several existing systems which perform mapping from a source text to UMLS concepts are reviewed. The focus is on the algorithmic approach, evaluation method, and failure analysis, when provided, for each system.

6.1.2.2.1 Concept Locator

The Concept Locator (P. Nadkarni et al., 2001) system uses a phrase-based approach to identify concepts for indexing and retrieval.

Algorithm: Input phrases from the source text are identified using the IBM Intelligent Text Miner's Feature Extraction Tool. Concept Locator uses a subset of UMLS. The subset removes redundant concepts, concepts unrelated to clinical medicine, concepts having eight or more terms, and suppressible synonyms (synonyms which result in ambiguity, such as *complications* being a synonym of *Complications Specific to*

Antepartum or Postpartum (Aronson, 2001)). To match concepts to the input phrase, the algorithm steps are as follows: 1) The input phrase is first stripped of stop words and words which do not appear in the UMLS. Any resulting phrase over five words is rejected. 2) An exact match of concept words to input phrase words is then attempted by using all words in the input phrase to match all concepts having the same set of words, regardless of word order. If the match results in mapping the input phrase to a distinct concept, the concept is output and the mapping algorithm stops. 3) If no exact match is found, two cases are handled. In the first case, a phrase consists of a single word. If the word is defined as ambiguous by UMLS, no further matching is attempted. In the second case, a phrase consists of two or more words. Subsets of words in the phrase matching is performed by finding complete concept word matches for all combinations of words {N-1, N-2,…2} where the concept match results in a single distinct concept. Words which are not matched are sent back to algorithm step #2. If the subset matching results in no concept matches, individual words in the input phrase are matched to single-word concepts. 4) If an input phrase from step 2 or 3 matches several concepts, concept disambiguation is performed by stemming both the words in the concept and the input phrase, and then comparing each for an exact match of the stemmed words. The disambiguation step is on a best-effort basis; that is, it is possible for disambiguation to fail and still result in mapping an input phrase to multiple concepts.

Evaluation: A manual evaluation of the concept mapping out was performed using a corpus of 24 biomedical documents (12 each of discharge summaries and surgery notes). A domain expert manually identified UMLS concepts in the corpus texts, and then compared the output of Concept Locator concept mappings to the manually-annotated

corpus. It was found Concept Locator matched concepts correctly for 76.3% of the phrases.

Failure Analysis: The authors identify three categories of concept-mapping failures: (a) use of noun phrases, (b) UMLS content, and (c) matching algorithm. The use of noun phrases cannot locate concepts spread across two or more noun phrases, resulting in matching two or more concepts rather than a specific single concept. Also, spelling, grammatical errors, and proper names can confuse natural language parsers. UMLS content influences performance because it is incomplete. UMLS does not list all possible term variations, may have missing biomedical concepts, and has redundant concepts. The matching algorithm has built-in limitations, such as five-word maximum phrase length, which causes some phrases not to be mapped.

6.1.2.2.2 Dynamic Taxonomy

The Dynamic Taxonomy system (Wollersheim et al., 2002) was designed to automate the construction of indexes to be combined with ontologies for biomedical text retrieval.

Algorithm: The Dynamic Taxonomy (DT) uses two different methods for extracting phrase: NLP and moving window. The NLP approach uses part-of-speech taggers to construct phrases. DT studied three different part-of-speech taggers. The moving window approach is an algorithm the authors propose which produces input phrases by taking a combination of N consecutive words. A word is considered to be a text string with more than three characters. The authors studied window sizes of N=1 to 6. UMLS concept matching was performed by normalizing the input phrase text using the

UMLS lexical tool LVG (National Library of Medicine, United States, 2006b). The normalized input phrase is then compared to the UMLS concept phrase list looking for exact matches.

Evaluation: The DT system was evaluated by first having a domain expert identify all possible concepts found in a six-page biomedical text having 2,982 words. The domain expert's annotations were then compared against the output of the DT matching process. Nine variations of the system were evaluated based on phrase selection method: three variations using part-of-speech taggers, and six variations using sliding window sizes of one to six words. Precision and recall were measured. The best-performing variation using part-of-speech tagging had a precision of 28% and recall of 19.5%. The recall for the six variations using sliding windows was approximately 11% while precision ranged from approximately 47% to 51%.

Failure Analysis: The authors believe that UMLS contains words which do not contain much medical content, and therefore if eliminated would result in better precision/recall scores by eliminating false positive mappings. In addition, they found that the domain expert maximized index size, and so suggested multiple concepts for a single phrase. The multiple concepts identified by the domain expert went from highly-general concepts to highly-specific, and often did not include words from the original source text. Finally, the authors suggest using meaning from paragraph to identify meaning in subsequent paragraphs.

6.1.2.2.3 KnowledgeMap

The KnowledgeMap system (J. C. Denny, Irani, Wehbe, Smithers, & Spickard, 2003) is designed to identify concepts in biomedical educational texts for indexing.

Algorithm: KnowledgeMap's concept identification component is known as KnowledgeMap Concept Identifier (KMCI) and consists of three phases: sentence identification, concept identification, and concept disambiguation. Sentence and noun phrase identification is done by using a natural language parser. Concept identification is done by taking a noun phrase and finding a set of UMLS concepts which match the noun phrase. If no concept match is found, then variants of the terms in the noun phrase are generated and the matching process retried. Nearby noun phrases linked by grammar (such as conjunctions and prepositions) are attempted to match concepts with the current noun phrase. This overcomes the phrase-based problem of identifying concepts occurring over two or more noun phrases. In addition, KMCI will distribute modifying adjectives. For example, "large and small intestine" is converted to "large intestine and small intestine" (J. C. Denny et al., 2003). If multiple concepts for a phrase are located, disambiguation is attempted. Two resources are used to perform disambiguation. The first resource is an externally-maintained list of concept co-occurrences which occur in MedLine abstracts. The second resource is a dynamic list of concepts with an exact match to a source text noun phrase. Disambiguation is performed by discarding concepts which are not similar to either the text's dynamic list of exact-match concepts or which do not co-occur with concepts in MedLine abstracts.

Evaluation: Evaluation of KnowledgeMap was done by first having two domain experts manually annotate five biomedical educational texts with important terms and phrases. Each text was then split into its component sentences, and each sentence was

then submitted to KMCI and MetaMap, a state-of-the-art concept matching system produced by the National Library of Medicine (see section 4.1.2.2.3). The domain experts then determined if the concepts for the identified terms and phrases were accurate or not. Precision and recall are the evaluation measures. Recall is defined as the number of important terms and phrases identified. Precision is defined as the number of correctly identified concepts. The recall is measured at 86% and precision at 92%, which outperforms MetaMap, which has a recall of 81% and a precision of 89% (J. C. Denny, Smithers, Miller, & Spickard, 2003).

Failure Analysis: The authors identified nine primary reasons for failure. The biggest failure of concept matching (62% of failures) in KMCI is the lack of a corresponding concept in UMLS. Other sources of failure include acronym/abbreviation/hyphen handling, and overmatching. Overmatching occurs when no exact match exists between the input phrase and UMLS phrases, and additional words in UMLS concept phrases are used to find a match with the input phrase. The disambiguation stage was responsible for only 2% of failures, while accounting for 18% of successful matches.

6.1.2.2.4 IndexFinder

IndexFinder (Zou et al., 2003) is a concept matching system designed for real-time indexing applications.

Algorithm: IndexFinder uses a series of in-memory table structures to find all possible concepts by using all combinations of words in the text. IndexFinder treats the terms within a text independently regardless of order. The terms in a text are first

normalized to lowercase them, remove unknown acronyms/abbreviations, stop terms, and terms unknown to UMLS, and map remaining terms to their base form. Next, for each unique term, all UMLS concept phrases containing the term are retrieved. Each retrieved phrase maintains its length and the count of terms matched so far. After all terms have been evaluated, the retrieved phrases are then evaluated based on their counts. Phrases having an exact count of terms matched have their associated concepts extracted for indexing.

Evaluation: The authors used a corpus of 5,783 patient reports totaling 10.8MB in size and report a text processing speed of 42.7KB/second. No evaluation was reported on the accuracy of the concepts extracted, although one was planned to evaluate the number of false positives and false negatives.

Failure Analysis: No failure analysis was reported. However, six filters are available to restrict the output. The filters focus on two aspects: 1) restricting the location of text, and 2) restricting the number of concepts generated. In order to generate concepts based on smaller units (i.e., not the full text), the term length can be restricted to less than six terms or less than 11 terms, as well as all terms. In addition, the range filter restricts terms to a certain distance, such as ten terms. Terms outside of the range are not counted in concept matching. The effect is to index only concepts occurring with terms a certain distance from each other. To restrict the generation of concepts, concept subsets can be removed if they are contained within a larger concept. In addition, concepts can be included only if they appear within certain UMLS semantic types.

6.1.2.2.5 MetaMap

MetaMap (Aronson, 2001) is a concept matching system produced by the National Library of Medicine. MetaMap is considered a state-of-the-art system for concept matching (J. C. Denny et al., 2003). MetaMap was originally developed to support indexing applications, but has also been used in data mining, decision support, and patient record applications.

Algorithm: The MetaMap algorithm consists of five steps. Parsing of the source text by a natural language parser is first done to form noun phrases. Variant generation on each noun phrase is done to find a term's acronyms, abbreviations, synonyms, inflectional and spelling variants. UMLS concept phrases containing the term or its variants are then identified. Each concept phrase is then evaluated to find the concept phrase which best matches the noun phrase from the source text. Evaluation is done using a four metrics: centrality, variation, coverage, and cohesiveness (Aronson, 1996). Centrality measures if the phrase head term is used in the concept phrase. Variation is a distance value that determines how far the term variant is from the term, based on the variant type. Coverage measures how much the terms between the concept phrase and source noun phrase overlap, ignoring term gaps. Cohesiveness measures similarly to coverage, but factors in sequences of terms which co-occur in the concept phrase and the source noun phrase. The four scores are computed into a weighted average. The coverage and cohesiveness scores are weighted twice as heavily as centrality and variation. The final stage forms a final mapping of the source text, which may result in mapping a source text into one or more concepts. The final mappings are shown on a scale of 0 to

1000, with 1000 being a complete mapping. No explicit disambiguation step is performed.

Evaluation: Several evaluations of MetaMap have been performed to date. The National Library of Medicine (NLM) performed a failure analysis of MetaMap by using a short evaluation (Divita, Tse, & Roth, 2004). Five annotators without domain expertise annotated two documents from genetic information Web site. Two documents were chosen due to the labor intensive activity of annotating documents, and then having the annotators all agree on a final set of concepts. The two documents were processed by MetaMap and then compared to the manual annotations. The recall score for the two documents was measured at 53%; precision was not calculated.

The University of Washington (UW) also performed an evaluation of MetaMap (Pratt & Yetisgen-Yildiz, 2003). They used six domain experts to identify concepts within a corpus of 60 texts. The study used precision and recall as measures, reporting a recall of 53% for exact matches and 93% for partial matches. Precision is reported as 28% for exact matches and 55% for partial matches.

Failure Analysis: The NLM and UIW evaluations also concluded with a failure analysis. The NLM study identified thirteen sources of failure. The most common failures resulted from needing implicit knowledge to map a term, the use of broader concepts by annotators because the UMLS Metathesaurus is incomplete, and co-reference resolution. The UW study identified four types of failures: incorrect splitting of a noun phrase, concept ranking and identification failed, and noun phrase breaking which changed the meaning of the phrase.

6.1.2.2.6 PhraseX

The PhraseX program performs noun extraction (National Library of Medicine, United States, 2004). A study of UMLS phrases in MEDLINE abstracts used the output of PhraseX to perform simple concept matching (Srinivasan, Rindflesch, Hole, Aronson, & Mork, 2002). A newer application uses PhraseX as a component of a larger biomedical text indexing application (National Library of Medicine, United States, 2006a).

Algorithm: Noun phrase identification is done by first tagging a sentence with its parts-of-speech, and then using the barrier word method (Tersmette et al., 1988) to delimit a phrase. In this case, tagger output delimits a phrase based on its part of speech; for example, a verb ends one phrase and begins another. PhraseX defines three types of successively complex phrases: (a) simp – phrases with a head noun; (b) macro – phrases with a preposition to the right; and (c) mega – a phrase with words to the left and right of a finite verb. Once a phrase is identified, the mapping is done in one of three ways: (a) exact match, (b) exact match with lower casing done to all terms in the phrase; and (c) exact matching done after UMLS normalization (lower casing, possessive removal, inflectional variation, and term sorting, among others).

Evaluation: The PhraseX evaluation was performed by downloading all MEDLINE citations from PubMed as of Fall 2001. The noun phrases were extracted resulting in approximately 175 million unique phrases. The authors report that there 63% of the phrases were simp, 16% were macro, and 21% were mega. Each unique phrase was then attempted to be mapped with one of the three matching methods. The result is 6.5% for exact match, 22.5% for lower case match, and 30% for normalized match.

Failure Analysis: Five types of failures were identified.

1. The UMLS Metathesaurus contains strings which are not useful for mapping. Examples are long descriptive strings and strings containing codes for what are known as Logical Observations Identifiers, Names and Codes (LOINC).

2. Syntactic analysis of text cannot address ambiguity due to grammar and writing style.

3. Trade names are not always included in the UMLS Metathesaurus.

4. Conservative constraints on the definition of a macro phrase.

5. There were exact matches only; partial matches were excluded.

6.1.2.2.7 SAPHIRE

The SAPHIRE system (W. R. Hersh & Greenes, 1990) was originally developed to support biomedical text indexing, but was later used to support other applications, such as extraction of concepts from patient medical records (W. R. Hersh & Donohoe, 1998).

Algorithm: The original SAPHIRE algorithm is substantially different from the current version. The original algorithm is based on strict pattern matching. All terms in the source text phrase must be present and in the same order as the UMLS phrase for a successful match (W. R. Hersh & Greenes, 1990). The latest algorithm (W. Hersh & Leone, 1995) relaxes the strict requirements and allows for partial matches and out of sequence terms. The algorithm takes a source text in the form of a phrase or sentence as input and extracts individual terms using the barrier word method (Tersmette et al., 1988). Barrier words are high-frequency words considered to be common. For each term found, a list of UMLS concepts containing the word is retrieved. UMLS concepts having high-frequency terms must also have low frequency terms as well or the concepts are

excluded. This is to eliminate concepts containing low content terms. The individual term concept lists are merged, and any concept having less than one-half the number of terms from the input text is excluded. The resulting set of UMLS concepts is then scored. The highest score occurs if all terms in the concept appear in the source text. Term order is ignored. If there is no exact term match, a weighting formula scores the concept based on proportion of words between source text and concept, term proximity, and length of term matches between source text and concept (W. R. Hersh et al., 2001).

Evaluation: SAPHIRE has been evaluated in the context of effectiveness as an information retrieval system (W. Hersh, Hickam, Haynes, & McKibbon, 1991), but a more recent evaluation focusing on finding best concepts rather than all concepts was completed using radiology reports (W. R. Hersh et al., 2001). Fifty radiology reports were processed using SAPHIRE to extract UMLS concepts. Precision and Recall were the evaluation measures. Precision was defined as number of correctly mapped concepts divided by number of total concepts. Recall was defined as the number of correctly mapped concepts divided by total number of correct concepts. Precision was measured at 30% and Recall at 63%.

Failure Analysis: A failure analysis of the radiology report concept matching was performed. The failures affecting recall were due to scoring errors, where the correct concept was scored lower than other concepts, or with issues regarding the barrier method of phrase identification. Failures affecting precision include negation in the phrase which was not identified, and disambiguating competing concepts. A key problem with SAPHIRE's algorithm is that it may return multiple matching concepts for a text segment which, due to partial matching, are incorrect concept mappings (P. Nadkarni et

al., 2001). The disambiguation problem was handled by adding a semantic type filter, which filters out concepts belonging to a particular semantic type.

6.1.2.2.8 SENSE

The SENSE (SEarch with New Semantics) system (Zieman & Bleich, 1997) is designed to map user queries to the National Library of Medicine's Medical Subject Heading (MeSH) terms. The system addresses the mismatch between user's natural language queries and MeSH index terms. The idea is to map user queries to MeSH terms, which are indexed by biomedical information retrieval systems.

Algorithm: In order to translate source text into concepts, SENSE translates a user query (a short phrase) into what the authors call *semantic factors*. Semantic factors are base concepts which cannot be decomposed further. SENSE defines 3,400 semantic factors. Semantic factors are constructed from an input phrase by having the Semantic Analyzer component look up phrase terms in a knowledge base, which handles variants, such as spelling and plural forms, and produce identical semantic factors for all input phrases having the same meaning. The output is a suggested list of MeSH terms which can be used to perform a search.

Evaluation: No evaluation was performed.

Failure Analysis: No failure analysis was performed because no evaluation was done. However, it should be noted that the purpose of SENSE is build a list of suggested MeSH terms. No effort is made to identify the best matching concept. Therefore, SENSE needs a disambiguation stage to filter out concepts which are not the best match for a phrase.

6.2 Text Summarization

Text summarization is a data reduction method and has roots dating back to the 1950s with the work initially done using statistical analysis of terms (Luhn, 1958). Text summarization has been defined as "the process of distilling the most important information from a source (or sources) to produce an abridged version for a particular user (or users) and task (or tasks)" (Mani & Maybury, 1999), and "a reductive transformation of source text to summary text through content reduction by selection and/or generalization on what is important in the source" (Sparck Jones, 1999). Text summarization is generally a three-phrase model: Interpretation, Transformation and Generation (Sparck Jones, 1999). Interpretation provides some analysis of the source text (such as removing insignificant words) and puts it into an intermediate form. Transformation takes the interpreted form and builds a summary form through methods such as content selection and concept generalization. Generation then takes the summary form and generates output appropriate for the user. Generation includes, for example, how to order the highest-ranking sentences when using an extractive approach.

Summaries are designed to serve one of two purposes: 1) indicative summaries give some idea of what the source text is about, so that the user can determine whether the source text should be read completely, and 2) informative summaries are intended as surrogates for the source text, where the main ideas are captured and presented (Mani & Maybury, 1999).

There are several methods for producing text summaries: Surface-level, Entity-level, and Discourse-level as well as hybrid combinations of these approaches (Mani & Maybury, 1999). Surface-level approaches use statistical information about term

occurrences (Luhn, 1958), (A. Nenkova & Vanderwende, 2005) , (Vanderwende & Suzuki, 2005), term locations (Edmundson, 1999) , (Teufel & Moens, 1999), and important, known phrases (Kupiec, Pedersen, & Chen, 1999) Entity-level approaches typically rely on graph-based structures to identify important information. Such structures are built from and typically rely on external information sources, such as thesaural relationships (Barzilay & Elhadad, 1997). Discourse-level approaches derive important information based on the structure of the text (Strzalkowski, Stein, & Wang,). In terms of performance, Hovy reports that frequency measures perform between 15%-35% precision and recall, while cohesive approaches typically range from 30%-60% (E. H. Hovy, 2005), where precision and recall scores are generated by comparing sentences extracted by a machine vs. sentences extracted by a person.

In text summarization, a key goal is identifying important text which should be presented to the user. This subset of the original source text is considered to contain the main ideas of a text. The first research work done in summarization used term frequency to identify important words in a text (Luhn, 1958). The sentences containing one or more high-frequency words are considered more important than other sentences which do not contain high-frequency words, or fewer of them. The idea is that frequently repeated terms are due to the author reiterating the main idea. Further research in text summarization confirms frequency (reiteration) is a strong feature (Rath, Resnick, & Savage, 1961), (Pollock & Zamora, 1975), (Edmundson, 1999), (A. Nenkova & Vanderwende, 2005). Another method for finding important text is recent work done lexical chaining (Morris & Hirst, 1991). Lexical chaining finds common links between words. For example, the words *lung* and *pulmonary* are related by a common concept

lung. A text is analyzed to find all chains of words, and the strongest chains of words are then assumed to represent the main idea of a text. This research applies both approaches to biomedical text summarization, using concept chaining rather than term chaining, and using concept frequency rather than term frequency. The planned research covers all three summarization approaches: surface (concept frequency summarizer), entity (concept chaining summarizer), and discourse (to be integrated with the concept frequency summarizer). The sub-sections below review the literature for the frequency and lexical chaining methods.

6.2.1 Text Summarization Using Lexical Chaining

Lexical chaining has been used for many years for text summarization. Lexical chaining is a method for determining lexical cohesion among terms in a text (Barzilay & Elhadad, 1997). Lexical cohesion is a property of text that causes a discourse segment to "hang together" as a unit (Morris & Hirst, 1991). Lexical cohesion is important in computational text understanding for two major reasons: 1) providing term ambiguity resolution, and 2) providing information for determining the meaning of text (Morris & Hirst, 1991). Lexical chaining is useful for determining the *aboutness* of a discourse segment, without fully understanding the discourse. As a basic assumption, the text must explicitly contain semantically related terms identifying the main concept. For example, if a text is about a political candidate and does not contain terms signifying the person is a candidate, lexical chaining cannot identify the fact the person is a political candidate.

Lexical chains for text summarization were first introduced by Morris and Hirst (Morris & Hirst, 1991). Their initial work described the approach, but did not implement

it because electronic versions of a thesaurus were not available at the time. A thesaurus is used to relate words semantically; for example, through synonymy and hypernym/hyponym semantic relationships. A hypernym is the word with the more general meaning among a set of related words. A hypernym relationship moves from a specific concept to a general concept. For example, given two related words {flower,plant}, *plant* is a hypernym of *flower* because *plant* is more general than *flower*. A hyponym relationship is the opposite of hypernymy, and moves from a general concept to a specific concept. For example, *flower* is a hyponym of *plant* because *flower* is more specific than *plant.* A machine implementation of lexical chaining by Barzilay and Elhadad (Barzilay & Elhadad, 1997) showed the theoretical work by Morris/Hirst could be practically realized for document summarization. While Barzilay/Elhadad proved the feasibility of computing lexical chains, their implementation ran in exponential time, making its mainstream use unlikely. A linear time algorithm was later defined and implemented by (Silber & McCoy, 2002). A more recent implementation by Galley and McKeown (Galley & McKeown, 2003) focused on improving word sense disambiguation based on the idea of one sense per discourse. All of these implementations use WordNet (Fellbaum, 1998) as the knowledge source for identifying semantic relationships between terms. WordNet is a freely-available lexical database for the English language which organizes nouns, verbs, adjectives and adverbs into synonym sets (known as a *synset*). These synsets are then linked to one another via different relations to form semantic relations between lexical concepts. WordNet is an ongoing research project developed at the Cognitive Science Laboratory of Princeton University.

The SUMMARIST system (E. Hovy & Lin, 1999) uses WordNet (Fellbaum, 1998) concept counting not for identifying salient sentences, but for topic interpretation. In topic interpretation, concept frequency counting is used to find a node in the concept hierarchy which sufficiently generalizes more specific concepts (e.g., {pear, apple} → fruit). The SUMMARIST authors cite the lack of domain-specific resources as a serious drawback to this approach. Our work uses domain-specific resources exclusively, but we have not used these resources for topic interpretation, only with sentence identification.

Lexical chains are an intermediate representation of source text, and are not used directly by an end-user. Instead, lexical chains are applied internally in some application; in this case, the application is document summarization. In document summarization, either single document or multiple documents, the top-n-sentences approach is the most common. In this approach, the summarization system identifies the sentences that most likely capture the main idea of a document or set of documents. The top sentences are then output, up to some limit either on the number of sentences or the number of characters, in order to produce a summary.

6.2.2 Text Summarization Using Frequency

Term frequency was first used in extractive text summarization in the late 1950's (Luhn, 1958). A follow-up study of an analysis of five term frequency methods showed high agreement in sentence selection among the methods (Rath et al., 1961). Subsequent research using frequency methods focused on the use of frequency as one feature among many for identifying important sentences, such as cue phrases (Pollock & Zamora, 1975) (Edmundson, 1999). Summarization using larger units of text has also been researched.

The LAKE system uses keyphrases for summarization (D'Avanzo, Magnini, & Vallin, 2004). The SUMMARIST system (E. Hovy & Lin, 1999) uses WordNet (Fellbaum, 1998) concept counting not for identifying salient sentences, but for generalizing concepts for topic interpretation (e.g., {pear, apple} → fruit). Most recently, the SumBasic algorithm uses term frequency as part of a context-sensitive approach to identifying important sentences while reducing information redundancy (A. Nenkova & Vanderwende, 2005). The use of frequency as a feature in locating important areas of a text has been proven useful in the literature (Luhn, 1958) (Rath et al., 1961) (Pollock & Zamora, 1975) (Edmundson, 1999). This is most likely due to reiteration, where authors state important information in several different ways, in order to reinforce main points (Sparck Jones, 1999).

6.2.3 Document Understanding Conferences

There has been much work done in the Document Understanding Conferences (DUC) (National Institute of Standards and Technology (NIST), 2005). DUC provides an annual forum for researchers to extend text summarization technology. In recent DUC conferences, several approaches for identifying sentences for extraction were used, and several of the approaches are surveyed here. The News Story system uses a pattern extraction algorithm (C5.0) to generate a decision tree to predict the words in the source text that should be part of summary (W. Doran et al., 2004). They use eight text features: (a) term frequency (TF) of words in the document, (b) inverse document frequency (IDF) of terms in external news corpus, (c) position of words from start of document, (d) lexical cohesion score between words and the document, (e) a Boolean noun flag for each word,

(f) a Boolean verb flag for each word, (g) a Boolean adjective flag for each word, and (h) a noun or proper noun phrase flag. Their findings are that TF, word position and IDF have greatest impact on summary quality. In addition, they concluded that lexical cohesion adds little as a feature in decision tree classification.

The LAKE system uses a keyphrase extraction approach that is used to identify candidate sentences (D'Avanzo et al., 2004). LAKE relies on word N-grams, which is a way to group sequences of words together. For example, the phrase *lung cancer* has unigrams {lung, cancer}, a single bigram {lung cancer} and no further N-grams where N has value greater than two (that is, three or more terms). LAKE begins by extracting all unigrams, bigrams, trigrams, and four-grams and filters them with part-of-speech patterns. A Naïve Bayes classifier trained using manual keyphrases is then used to identify relevant keyphrases. The resulting keyphrases are scored using two features: (a) keyphrase TF*IDF, and (b) distance of keyphrase from the start of document. Their results scored in the middle of all 2004 DUC submissions. The authors feel their system can be improved by finding additional features that capture the semantic properties of keyphrases. One possibility they mention is to compute lexical chains and using keyphrase membership in a chain as a feature.

The KMS system describes a system where a text is decomposed into a parse tree format (Litkowski, 2004). The parse tree is then used to identify noun phrases and score them based on a frequency analysis of terms in the noun phrases in addition to the occurrence of words in a DUC topic specification. Their performance fell in an acceptable range, and the authors observe that in general their frequency-based approach performs better than systems based on other approaches.

Finally, the GISTexter system uses a frequency-based method to identify sentences to extract (Lacatusu, Hickl, Harabagiu, & Nezda, 2004). GISTexter computes a weight for each term in a collection based on term frequency in a relevant set of documents. This weight is then used to score each sentence. The top scoring sentences are then extracted. DUC generally imposes limits on the lengths of extracted summaries. GISTexter used the following method to generate a summary:

1. Put highest ranked sentence as summary

2 .If summary not long enough, then add next sentence that has more information than the summary has in common with it

3 .Iterate until summary length exceeded or no more sentences

4. Perform summary compression, using several different approaches: (a) if (summary length - last sentence length) > 600 characters, remove last sentence; and (b) if summary length > 665 characters, truncate at 665 characters

The system performed among the top systems. The authors found the best approach for summary compression is truncation – 4b (listed above). This approach rated the second highest score of all systems.

7. RESEARCH PLAN

Fall 2006

- Implement semantic annotator for biomedical text
- Defend dissertation proposal

Winter 2007

- Evaluate semantic annotator
- Implement additional summarizers for summarization evaluation
- Participate in DUC 2007

Spring 2007

- Continue work on semantic annotation and evaluation
- Determine characteristics of biomedical text, optimal summary size, sections drawn from
- Participate in DUC 2007
- Begin writing final dissertation

Summer 2007

- Based on sections drawn from, modify concept chaining and frequency distribution summarizers to modify scoring and evaluate
- Continue final dissertation writing
- Defend dissertation

8. PUBLICATIONS

***2007*:**

Lawrence Reeve, Hyoil Han and Ari D. Brooks (2007). *Biomedical Text Summarization Using Concept Chains*, International Journal of Data Mining and Bioinformatics. *Refereed. Accepted*.

Lawrence Reeve, Hyoil Han and Ari D. Brooks. *The Use of Domain-Specific Concepts in Biomedical Text Summarization*, Journal of Information Processing and Management. *Refereed. Accepted*.

***2006*:**

Lawrence H. Reeve, Hyoil Han, Saya V. Nagori, Jonathan C. Yang, Tamara A. Schwimmer, and Ari D. Brooks (2006). *Concept Frequency Distribution in Biomedical Text Summarization*. Proceedings of the 15th Conference on Information and Knowledge Management. *Refereed* - 15% acceptance.

Lawrence Reeve, Hyoil Han, and Ari D. Brooks (2006). *BioChain: Using Lexical Chaining Methods for Biomedical Text Summarization*. Proceedings of the 21st Annual ACM Symposium on Applied Computing, Bioinformatics track. *Refereed* - 32% acceptance.

Lawrence Reeve and Hyoil Han (2006). *A Comparison of Semantic Annotation Systems for Text-based Web Documents.* Web Semantics and Ontology, David Taniar and J. Wenny Rahayu (Eds.), Idea Group Publishing. *Refereed*.

***2005*:**

Lawrence Reeve and Hyoil Han (2005). *Survey of Semantic Annotation Platforms*. Proceedings of the 20th Annual ACM Symposium on Applied Computing, Web Technologies and Applications track. Presentation. *Refereed* - 37% acceptance.

Lawrence Reeve and Hyoil Han (2005). *Semantic Annotation for Semantic Social Networks Using Community Resources*. AIS SIGSEMIS Bulletin, Vol. 2, Issue (3&4), pp: 52-56.

Lawrence Reeve, Hyoil Han, and Chaomei Chen (2005). *Information Visualization and the Semantic Web.* Visualizing the Semantic Web, Vladimir Geroimenko and Chaomei Chen (Eds.), Springer. *Refereed*.

***2004*:**

Lawrence Reeve (2004). *Adapting the TileBar Interface for Visualizing Resource Usage*.

Proceedings of the 30th International Conference for the Resource Management and Performance Evaluation of Enterprise Computing Systems. Presentation**.** *Refereed*.

BIBLIOGRAPHY

Afantenos, S. D., Karkaletsis, V., & Stamatopoulos, P. (2005). Summarization from medical documents: A survey. *Journal of Artificial Intelligence in Medicine, 33*(2), 157-177.

Aronson, A. R. (2001). Effective mapping of biomedical text to the UMLS metathesaurus: The MetaMap program. *Proceedings of the AMIA Symposium 2001,* 17-21.

Aronson, A. R. (1996). *MetaMap: Mapping text to the UMLS metathesaurus.* Unpublished manuscript.

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (1st ed.). Harlow, England: Addison-Wesley.

Barzilay, R., & Elhadad, M. (1997). Using lexical chains for text summarization. *In Proceedings of the Intelligent Scalable Text Summarization Workshop (ISTS'97), ACL,* Madrid, Spain. 10-18.

Bayerl, P. S., Lüngen, H., Gut, U., & Paul, K. I. (2003). Methodology for reliable schema development and evaluation of manual annotations. *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003),* Florida, USA.

Berners-Lee, T., Hendler, J., & Lassila, O. (2001). The semantic web. *Scientific American, 284*(5), 34-43.

Borkar, V., Deshmukhy, K., & Sarawagiz, S. (2001). Automatic segmentation of text into structured records. *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data,* Santa Barbara, California, USA. , *30*(2) 175-186.

Brin, S. (1998). Extracting patterns and relations from the world wide web. *Proceedings of the WebDB Workshop at 6th International Conference on Extending Database Technology,* Valencia, Spain. , *1590* 172-183.

Brooks, A. D., & Sulimanoff, I. (2002). Evidence-based oncology project. *Surgical oncology clinics of north america* (pp. 3-10)

Carbonell, J., & Goldstein, J. (1998). The use of MMR, diversity-based reranking for reordering documents and producing summaries. *SIGIR '98: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Melbourne, Australia. 335-336. from http://doi.acm.org/10.1145/290941.291025

Chandrasekaran, B., Josephson, J. R., & Benjamins, V. R. (1999). What are ontologies and why do we need them? *IEEE Intelligent Systems, 14*(1), 20-26.

Cimiano, P., Handschuh, S., & Staab, S. (2004). Towards the self-annotating web. *Thirteenth International Conference on World Wide Web,* New York, NY, USA. 462-471.

Ciravegna, F. (2001). Adaptive information extraction from text by rule induction and generalisation. *Proceedings of 17th International Joint Conference on Artificial Intelligence (IJCAI 2001),* Seattle, Washington, USA. 1251-1256.

Cohen, A. M., & Hersh, W. R. (2005). A survey of current work in biomedical text mining. *Briefings in Bioinformatics, 6*(1), 57-71.

Cunningham, H., Maynard, D. & Tablan, V. (2000). *JAPE: A java annotation patterns engine (second edition). technical report CS--00--10, university of sheffield, department of computer science.* Retrieved April 24, 2005 from ftp://ftp.dcs.shef.ac.uk/home/hamish/auto_papers/Cun00e.ps.gz

D'Avanzo, E., Magnini, B., & Vallin, A. (2004). Keyphrase extraction for summarization purposes: The LAKE system at DUC-2004. *Proceedings of the 2004 Document Understanding Conference,* Boston, USA.

Dalianis, H. (2000). *SweSum - A text summarizer for swedish* No. TRITA-NA-P0015). Stockholm, Sweden: NADA, KTH.

Denny, J. C., Smithers, J. D., Miller, R. A., & Spickard, A. (2003). "Understanding" medical school curriculum content using KnowledgeMap. *Journal of the American Medical Informatics Association, 10*(4), 351-362.

Denny, J. C., Irani, P. R., Wehbe, F. H., Smithers, J. D., & Spickard, A.,3rd. (2003). The KnowledgeMap project: Development of a concept-based medical school curriculum database. *Proceedings of the Annual AMIA Symposium, ,* 195-199.

Dice, L. R. (1945). Measures of the amount of ecologic association between species. *Ecology, 26,* 297-302.

Dill, S., Eiron, N., Gibson, D., Gruhl, D., Guha, R., & Jhingran, A., et al. (2003). SemTag and seeker: Bootstrapping the semantic web via automated semantic annotation. *Twelfth International World Wide Web Conference,* Budapest, Hungary. 178-186.

Dingli, A., Ciravegna, F., & Wilks, Y. (2003). Automatic semantic annotation using unsupervised information extraction and integration. *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the Second International Conference on Knowledge Capture (K-CAP 2003),* Florida, USA.

Divita, G., Tse, T., & Roth, L. (2004). Failure analysis of MetaMap transfer (MMTx). *Medinfo, 11*(Pt 2), 763-767.

Doran, W. P., Stokes, N. S., Dunnion, J., & Carthy, J. (2004). Assessing the impact of lexical chain scoring methods and sentence extraction schemes on summarization. *Proceedings of the 5th International Conference on Intelligent Text Processing and Computational Linguistics CICLing-2004*,

Doran, W., Stokes, N., Newman, E., Dunnion, J., Carthy, J., & Toolan, F. (2004). News story gisting at university college dublin. *Proceedings of the Document Understanding Conference (DUC-2004)*,

Edmundson, H. P. (1999). New methods in automatic extracting. In I. Mani, & M. T. Maybury (Eds.), (pp. 23-42). Cambridge, MA: MIT Press.

Elkin, P., Cimino, J., Lowe, H., Aronow, D., Payne, T., & Pincetl, P., et al. (1988). Mapping to MeSH(the art of trapping MeSH equivalence from within narrative text). 185-190.

Fellbaum, C. (1998). *WORDNET: An electronic lexical database*The MIT Press.

Galley, M., & McKeown, K. (2003). Improving word sense disambiguation in lexical chaining. *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence,* Acapulco,Mexico. 1486-1488.

Goldstein, J., Kantrowitz, M., Mittal, V., & Carbonell, J. (1999). Summarizing text documents: Sentence selection and evaluation metrics. *SIGIR '99: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* Berkeley, California, United States. 121-128. from http://doi.acm.org/10.1145/312624.312665

Handschuh, S., Staab, S., & Volz, R. (2003). On deep annotation. *International WWW Conference,* 431-438.

Handschuh, S., Staab, S., & Ciravegna, F. (2002). S-CREAM -- semi-automatic CREAtion of metadata. *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*,

Harnly, A., Nenkova, A., Passonneau, R., & Rambow, O. (2005). Automation of summary evaluation by the pyramid method. *Proceedings of the Recent Advances in Natural Language Processing,* Borovets, Bulgaria.

Hearst, M. (1992). Automatic acquisition of hyponyms from large text corpora. *Proceedings of the Fourteenth International Conference on Computational Linguistics,* Nantes, France. *, 2* 539-545.

Hersh, W. R., Mailhot, M., Arnott-Smith, C., & Lowe, H. J. (2001). Selective automated indexing of findings and diagnoses in radiology reports. *Journal of Biomedical Informatics, 34*(4), 262-273.

Hersh, W., Hickam, D. H., Haynes, R. B., & McKibbon, K. A. (1991). Evaluation of SAPHIRE: An automated approach to indexing and retrieving medical literature. *Proceedings / the ... Annual Symposium on Computer Application [Sic] in Medical Care. Symposium on Computer Applications in Medical Care, ,* 808-812.

Hersh, W., & Leone, T. J. (1995). The SAPHIRE server: A new algorithm and implementation. *Proceedings / the ... Annual Symposium on Computer Application [Sic] in Medical Care. Symposium on Computer Applications in Medical Care, ,* 858-862.

Hersh, W. R., & Donohoe, L. C. (1998). SAPHIRE international: A tool for cross-language information retrieval. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium, ,* 673-677.

Hersh, W. R., & Greenes, R. A. (1990). SAPHIRE--an information retrieval system featuring concept matching, automatic indexing, probabilistic retrieval, and hierarchical relationships. *Computers and Biomedical Research, an International Journal, 23*(5), 410-425.

Hovy, E. H. (2005). Automated text summarization. In R. Mitkov (Ed.), *The oxford handbook of computational linguistics* (pp. 583-598). Oxford: Oxford University Press.

Hovy, E., & Lin, C. (1999). Automated text summarization in SUMMARIST. In I. Mani, & M. T. Maybury (Eds.), *Advances in automatic text summarization* (pp. 81-94). Cambridge, MA: MIT Press.

Jaques, D. P. (Ed.). (2002). *Surgical oncology clinics of north america: Prospective randomized clinical trials in oncology*W.B. Saunders Company.

Kiess, H. O. (2002). *Statistical concepts for the behavioral sciences* (Third ed.). Boston, MA: Allyn and Bacon.

Kogut, P., & Holmes, W. (2001). AeroDAML: Applying information extraction to generate DAML annotations from web pages. *Proceedings of the Workshop on Knowledge Markup and Semantic Annotation at the First International Conference on Knowledge Capture (K-CAP 2001),* Victoria, BC.

Kosala, R., & Blockeel, H. (2000). Web mining research: A survey. *SIGKDD Explorations, 2*(1), 1-15.

Kupiec, J., Pedersen, J., & Chen, F. (1999). A trainable document summarizer. In I. Mani, & M. T. Maybury (Eds.), (pp. 55-60). Cambridge, MA: MIT Press.

Kushmerick, N., Weld, D. S., & Doorenbos, R. (1997). Wrapper induction for information extraction. *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI '97),* Nagoya, Japan. 729-737.

Lacatusu, F., Hickl, A., Harabagiu, S., & Nezda, L. (2004). Lite-GISTexter at DUC 2004. *Proceedings of the 2004 Document Understanding Conference,*

Lavie, A., Sagae, K., & Jayaraman, S. (2004). The significance of recall in automatic metrics for MT evaluation.

Lee, D. L., Chuang, H., & Seamons, K. (1997). Document ranking and the vector-space model. *Software, IEEE, 14*(2), 67-75.

Lin, C. Y., & Och, F. J. (2004a). Automatic evaluation of machine translation quality using longest com-mon subsequence and skip-bigram statistics. *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics,* Barcelona, Spain. 605-612.

Lin, C. Y., & Och, F. J. (2004b). Orange: A method for evaluating automatic evaluation metrics for machine translation. *Proceedings of the 20Th International Conference on Computational Linguistics,* Geneva, Switzerland. 501-507.

Lin, C. (2004). Looking for a few good metrics: Automatic summarization evaluation - how many samples are enough? *Proceedings of the NTCIR Workshop 4,* Tokyo, Japan.

Lin, C., & Hovy, E. H. (2003). Automatic evaluation of summaries using N-gram co-occurrence statistics. *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003),* Edmonton, Canada. , *1*(1) 71-78.

Lin, C. (2005). *Recall-oriented understudy for gisting evaluation (ROUGE).* Retrieved August 20, 2005 from http://www.isi.edu/~cyl/ROUGE/

Litkowski, K. C. (2004). Summarization experiments in DUC 2004. *Proceedings of the 2004 Document Understanding Conference,* Boston, USA.

Luhn, H. P. (1958). The automatic creation of literature abstracts. *IBM Journal of Research and Development, 2*(2), 159-165.

Maedche, A., & Staab, S. (2001). Ontology learning for the semantic web. *IEEE Intelligent Systems, 16*(2), 72-79.

Mani, I., & Bloedorn, E. (1998). Machine learning of generic and user-focused summarization. *Proceedings of the Fifteenth national/tenth Conference on Artificial intelligence/Innovative Applications of Artificial Intelligence,* Madison, Wisconsin, United States. 820-826.

Mani, I., & Maybury, M. T. (1999). In Mani I., Maybury M. T. (Eds.), *Advances in automatic text summarization*. Cambridge, MA: MIT Press.

Manning, C. D., & Schutze, H. (1999). *Foundations of statistical natural language processing* (1st ed.). Cambridge, Massachusetts: The MIT Press.

Maynard, D. (2003). Multi-source and multilingual information extraction. *Expert Update,*

Microsoft Coporation. (2002). *Microsoft word 2002*. Redmond, Washington, USA:

Missikoff, M., Navigli, R., & Velardi, P. (2002). The usable ontology: An environment for building and assessing a domain ontology. *1st International Semantic Web Conference (ISWC2002),* 39-53.

Morris, J., & Hirst, G. (1991). Lexical cohesion computed by thesaural relations as an indicator of the structure of text. *Computational Linguistics, 17*(1), 21-43.

Nadkarni, P., Chen, R., & Brandt, C. (2001). UMLS concept indexing for production databases. *Journal of the American Medical Informatics Association, 8,* 80-91.

Nadkarni, P. M. (1997). Concept locator: A client-server application for retrieval of UMLS metathesaurus concepts through complex boolean query. *Computers and Biomedical Research, an International Journal, 30*(4), 323-336.

National Institute of Standards and Technology. (2005a). *DUC 2005 - responsiveness task.* Retrieved September 28, 2006 from http://duc.nist.gov/duc2005/responsiveness.assessment.instructions

National Institute of Standards and Technology. (2005b). *DUC 2005 quality questions.* Retrieved September 28, 2006 from http://duc.nist.gov/duc2005/quality-questions.txt

National Institute of Standards and Technology (NIST). (2005). *Document undertanding conferences.* Retrieved August 20, 2005 from http://www-nlpir.nist.gov/projects/duc/

National Library of Medicine, United States. (2006a). *Medical text indexer.* Retrieved September 3, 2006 from http://ind.nlm.nih.gov/mti.shtml

National Library of Medicine, United States. (2006b). *Specialist lexicon and lexical tools.* Retrieved August 31, 2006 from http://www.nlm.nih.gov/research/umls/meta4.html

National Library of Medicine, United States. (2004). *PhraseX and the SPECIALIST minimal commitment parser.* Retrieved September 3, 2006 from http://ind.nlm.nih.gov/MTI/phrasex.shtml

Nenadic, G., Mima, H., Spasic, I., Ananiadou, S., & Tsujii, J. (2002). Terminology-driven literature mining and knowledge acquisition in biomedicine. *International Journal of Medical Informatics, 67*(1), 33-48.

Nenkova, A., & Passonneau, R. (2004). Evaluating content selection in summarization: The pyramid method. Paper presented at the *Proceedings of HLT/NAACL 2004,* Boston, MA.

Nenkova, A., & Vanderwende, L. (2005). *The impact of frequency on summarization* No. MSR-TR-2005-101). Redmond, Washington: Microsoft Research.

Pollock, J. J., & Zamora, A. (1975). Automatic abstracting research at chemical abstracts service. *Journal of Chemical Information and Computer Sciences, 15*(4), 226-232.

Popov, B., Kiryakov, A., Kirilov, A., Manov, D., Ognyanoff, D., & Goranov, M. (2003). KIM - semantic annotation platform. *2nd International Semantic Web Conference (ISWC2003), , 2870* 834-849.

Pratt, W., & Yetisgen-Yildiz, M. (2003). A study of biomedical concept identification: MetaMap vs. people. *AMIA.Annu.Symp.Proc.*, , 529-533.

Radev, D., Allison, T., Blair-Goldensohn, S., Blitzer, J., Celebi, A., & Drabek, E., et al. (2004). MEAD - a platform for multidocument multilingual text summarization. *LREC 2004,* Lisbon, Portugal.

Rath, G. J., Resnick, A., & Savage, R. (1961). The formation of abstracts by the selection of sentences. [Electronic version]. *American Documentation, 2*(12), 139-208.

Reeve, L., & Han, H. (2005). Survey of semantic annotation platforms. *Proceedings of the 20th Annual ACM Symposium on Applied Computing, Web Technologies and Applications Track,* Santa Fe, New Mexico.

Reeve, L., Han, H., & Brooks, A. D. (2006a). BioChain: Using lexical chaining methods for biomedical text summarization. *Proceedings of the 21st Annual ACM Symposium on Applied Computing, Bioinformatics Track,* Dijon, France. 180-184.

Reeve, L., Han, H., & Brooks, A. D. (2006b). Biomedical text summarization using concept chains. [Electronic version]. *International Journal of Data Mining and Bioinformatics, To Appear.*

Reeve, L., & Han, H. (2006). A comparison of semantic annotation systems for text-based web documents. In D. Taniar, & J. W. Rahayu (Eds.), *Web semantics and ontology* (1st ed.,). Hershey, PA USA: Idea Group.

Reeve, L., Han, H., Nagori, S. V., Yang, J., Schwimmer, T., & Brooks, A. D. (2006). Concept frequency distribution in biomedical text summarization. *Proceedings of the ACM Fifteenth Conference on Information and Knowledge Management (CIKM'06),* Arlington, VA USA.

Rotem, N. (2003). *Open text summarizer (OTS).* Retrieved July 3, 2006, 2006 from http://libots.sourceforge.net

Silber, G. H., & McCoy, K. F. (2002). Efficiently computed lexical chains as an intermediate representation for automatic text summarization. *Computational Linguistics, 28*(4), 487-496.

Sparck Jones, K. (1999). Automatic summarizing: Factors and directions. In I. Mani, & M. T. Maybury (Eds.), *Advances in automatic text summarization* (pp. 2-12). Cambridge, MA: MIT Press.

Sparck Jones, K. (1972). A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation, 28*(1), 11-21.

Srinivasan, S., Rindflesch, T. C., Hole, W. T., Aronson, A. R., & Mork, J. G. (2002). Finding UMLS metathesaurus concepts in MEDLINE. *Proceedings / AMIA ... Annual Symposium. AMIA Symposium,* , 727-731.

Strzalkowski, T., Stein, G., & Wang, J. B, wise. A robust practical text summarizer. 1999. *Advances in Automatic Text Summarization,* , 137–154.

Subhash, S. (1996). *Applied multivariate techniques* (1st ed.). USA: John Wiley and Sons.

Svab Ondrej, Labsky Martin, Svatek Vojtech. (2004). RDF-based retrieval of information extracted from web product catalogues. *Proceedings of the 27th Annual ACM SIGIR Conference on Research and Development in Information Retrieval, Semantic Web Workshop,* Sheffield, UK.

Tallis, M. (2003). Semantic word processing for content authors. *Second International Conference on Knowledge Capture,* Sanibel, Florida, USA.

Tersmette, K. W. F., Scott, A. F., Moore, G. W., Matheson, N. W., & Miller, R. E. (1988). Barrier word method for detecting molecular biology multiple word terms. 207-211.

Teufel, S., & Moens, M. (1999). Argumentative classificatin of extracted sentences as a first step towards flexible abstracting. In I. Mani, & M. T. Maybury (Eds.), (pp. 155-171). Cambridge, MA: MIT Press.

The Lemur Project. (2006). *Lemur language modeling toolkit.* Retrieved July 3, 2006, 2006 from http://www.lemurproject.org/

United States National Library of Medicine. (2005a). *ClinicalTrials.gov*.http://www.clinicaltrials.gov/

United States National Library of Medicine. (2005b). *MetaMap transfer*.http://mmtx.nlm.nih.gov/

United States National Library of Medicine. (2005c). *PubMed*.http://www.ncbi.nlm.nih.gov/entrez/query.fcgi

United States National Library of Medicine. (2005d). *Unified medical language system (UMLS)*.http://www.nlm.nih.gov/research/umls/

United States National Library of Medicine. (2004a). *UMLS metathesaurus fact sheet.*http://www.nlm.nih.gov/pubs/factsheets/umlsmeta.html

United States National Library of Medicine. (2004b). *UMLS semantic network fact sheet.*http://www.nlm.nih.gov/pubs/factsheets/umlssemn.html

University of Sheffield. (2002). *Amilcare.* Retrieved December 28, 2004 from http://nlp.shef.ac.uk/amilcare/amilcare.html

Vanderwende, L., & Suzuki, H. (2005). Frequency-based summarizer and a language modeling extension. *Proceedings of the MultiLingual Summarization 2005 Workshop,*

Vargas-Vera, M., Motta, E., Domingue, J., Lanzoni, M., Stutt, A., & Ciravegna, F. (2002). MnM: Ontology driven semi-automatic and automatic support for semantic markup. *The 13th International Conference on Knowledge Engineering and Management (EKAW 2002),* 379-391.

Wollersheim, D., Rahayu, W., & Reeve, J. (2002). Evaluation of index term discovery in medical reference text. Paper presented at the *Proceedings of the International Conference on Information Technology and Applications*, Bathhurst, NSW, Australia.

Yesilada, Y., Harper, S., Goble, C., & Stevens, R. (2004). Ontology based semantic annotation for visually impaired web travellers. *Proceedings of the 4th International Conference on Web Engineering (ICWE 2004),* Munich, Germany. *, 3140* 445-458.

Zieman, Y. L., & Bleich, H. L. (1997). Conceptual mapping of user's queries to medical subject headings. *Proceedings : A Conference of the American Medical Informatics Association / ... AMIA Annual Fall Symposium. AMIA Fall Symposium, ,* 519-522.

Zou, Q., Chu, W. W., Morioka, C., Leazer, G. H., & Kangarloo, H. (2003). IndexFinder: A method of extracting key concepts from clinical texts for indexing. *Proceedings of the AMIA Annual Symposium,* 763-767.