Online Biomedical Concept Annotation Using Word Coverage Filtering and Language Model Mapping

Lawrence H. Reeve, Hyoil Han

Drexel University, College of Information Science and Technology, Philadelphia, PA USA

lhr24@drexel.edu

hyoil.han@acm.org

ABSTRACT

In this paper, we describe an online biomedical concept annotator, CONANN using a language modeling text-to-concept mapper. CONANN takes a biomedical source phrase and finds the best-matching biomedical concept, which is defined in resources such as the United States National Library of Medicine's Unified Medical Language System Metathesaurus. We show that techniques from the information retrieval field, specifically inverse document frequency and language modeling, can be applied to filter and then select best-matching domain-specific concepts. An advantage of such an approach is to improve annotation speed, facilitating the use of concept annotation in online environments. Our main contributions are 1) the design of a phrase-level concept annotator which is more readily usable in online environments than existing systems, 2) the use of a word coverage filter utilizing various word weights and word coverage algorithms, and 3) the use of a language model to find the best matching domain-specific concept for a phrase. An intrinsic evaluation shows that for the exact match of text-to-concept mapping the use of language modeling has 13-20% higher precision than the use of phrase counting. In addition, an extrinsic evaluation using the generated concepts by our proposed annotator in a text summarization task shows improvement (from 1.5% to 6.7%) in text summarization performance.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing – *indexing methods, thesauruses*.

General Terms

Algorithms, Measurement, Performance, Design, Economics, Experimentation.

Keywords

Biomedical semantic annotation, biomedical concept mapping, concept annotation.

CIKM'07, November 6-November 8, 2007, Lisbon, Portugal.

1. INTRODUCTION

The task of a concept annotator is to map each text unit (typically a phrase) of a source text into one or more domain-specific concepts. Biomedical concepts are defined in resources such as the Unified Medical Language System (UMLS) [1] and National Cancer Institute (NCI) Thesaurus [2]. The biomedicine community maintains large and continuously-updated information sources. For example, United States National Library of Medicine's PubMed service contains in excess of 16 million citations from over 5,000 worldwide biomedicine-related journals (United States National Library of Medicine, 2006). The PubMed service consists of citations and abstracts in addition to linking with full-texts. For physicians and biomedical researchers, finding and using relevant texts within these large resources can be challenging. To address this challenge, annotation systems using domain-specific concepts, rather than terms, have been developed. Examples of such systems include MetaMap Transfer (MetaMap)[3], SAPHIRE[4], and KnowledgeMap[5]. Among the benefits of using concepts, rather than terms, is 1) synonym merging, where synonymous phrases are merged to a single concept, and 2) the use of a domain-specific language for querying. Biomedical concept annotations have been used in applications for indexing and retrieval, data mining, decision support, patient records, medical curriculum searching, and text summarization [3] [5] [6].

Our main contributions are 1) development of an online as opposed to offline concept annotator, 2) the use of a word coverage filter utilizing various word weights and word coverage algorithms, and 3) the use of a language model to perform final best-concept mapping using the filtered output. The use of a language model for each concept is particularly distinctive. Previous systems score each candidate phrase independently, even if there is more than one candidate phrase from a single concept. We combine all phrases for a single concept into a single language model. A UMLS concept is then evaluated as to how likely the source phrase was generated from it. This concept language modeling approach allows us to consider all synonymous phrases within a concept. In our previous work [7], phrase counting was used for text-to-concept mapping, but in the this paper language modeling is used for it. In addition, we evaluate the use of binary and inverse phrase frequency word weights in two word coverage algorithms.

In this paper, we describe our biomedical concept annotator, CONANN, which supports both online as well as offline concept

annotation. The current concept resource is UMLS, but can support other concept resources as well. Its design is intended to achieve faster annotation time per phrase while maintaining annotation accuracy competitive with existing biomedical annotation systems. Such an online biomedical annotation system has the advantages of supporting texts unknown to the system ahead of time, as well as providing for constantly changing concept resources, which is common in a field such as biomedicine. These advantages overcome the limitations of purely offline biomedical concept annotators, which form the majority of such systems today. CONANN is designed for use with general biomedical text where the best or best top-N concepts are identified at the phrase level. It is not designed for biological text, where identification of biological entities such as genes is performed. Our current focus is to use CONANN as a component of a biomedical text summarization system. However, like the MetaMap system produced by the United States National Library of Medicine [8], we envision CONANN will be useful in tasks such as question-answering, information extraction, and concept-based indexing and retrieval.

This paper is organized as follows. Section 2 provides background on the biomedical resource we use for concept annotation of texts and also describes the general concept mapping process. Section 3 discusses the architecture of our concept annotator system, as well as the algorithms for two important parts of the mapping process. Section 4 describes previous work in this area. Section 5 describes our evaluation methodology, Section 6 discusses the evaluation results and Section 7 concludes.

2. BACKGROUND

2.1 Biomedical Concept Resource

Finding meaning in biomedical documents which can be processed by machines is accomplished by first creating one or more ontologies, and then linking information within each document to specifications contained in each ontology using a markup language. Ontologies are conceptualizations of a domain that typically are represented using domain vocabulary. Automated semantic annotation is the process of mapping data instances to an ontology. The resulting annotations from the semantic annotation processing are what provide the link between information stored within a document and the ontology. 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) [1]. UMLS contains many sub-components, but only the Metathesaurus is used in this work.

The UMLS Metathesaurus contains concepts and real-world instances of the concepts, including a concept name and its synonyms, lexical variants, and translations [9]. 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 associated with the concept. A *concept instance* is a phrase belonging to a UMLS concept. Each UMLS concept is associated with one or more synonymous concept instances. Concept instances are derived from biomedical vocabulary sources. The key idea is that a single UMLS concept may have multiple ways of being expressed (instances). The UMLS Metathesaurus organizes concept instances into concepts. A *concept name* is the name given to a particular UMLS concept.

Table 1: A UMLS concept and its concept instances

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

2.2 Text-To-Concept Mapping Process

The task of a biomedical concept annotator is to map small text units in the source text to concept instances which in turn, determine the concept name the text unit should have. For example, assuming our ontology consists of only the single concept and the five concept instances shown in Table 1, then in the sentence "The patient has multiple myeloma," the phrase "multiple myeloma" is mapped to the concept Multiple Myeloma. In reality, however, such simple mappings are rare.

Most biomedical concept annotators use the following generalized method to map a source text unit to a domain-specific concept:

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 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 [10];

3. For each word in the input phrase, build a set of all concepts based on their concepts instances 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.

3. Concept Annotator - CONANN

In this section, we discuss the general design of our CONANN concept annotator and describe how the UMLS domain resource is pre-processed for use within CONANN. In addition, we detail the algorithms for the coverage filter and the final concept mapping.

3.1 Architectural Overview

CONANN finds the best matching concept for a source phrase and uses a staged approach, as shown in Figure 1, where all possible candidate phrases are first generated, filtered and then used to determine the final concept mapping.

There are several phrase types used by CONANN. A *source phrase* is a phrase from the source text which the system will attempt to annotate with a biomedical concept. A *concept instance* is phrase belonging to a UMLS concept (each UMLS concept is associated with one or more synonymous phrases as shown in Table 1). *Candidate phrases* are concept instances 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.

A list of candidate phrases is generated based on the overlap of words from the source phrase and all concept instances. A candidate phrase is required to have only one word in common with the source phrase. If only a single candidate phrase exists, its associated concept name is returned. If there is more than one candidate phrase generated, the coverage filter is applied to remove unlikely candidate phrases. The coverage filter is based on a variant of inverse document frequency (see Section 3.4) The staged approach of filtering candidate phrases and then performing final mapping is different than existing approaches, which typically score a candidate phrase completely in one pass and then rank the set of resulting concepts [11], [3]. The idea of the staged approach is to filter out candidate concepts using basic techniques, and then find the best concept match for a source phrase using a small subset of possible candidate phrase matches.

The filter and mapping components are dependent on one another. While in this work only coverage filtering is used, it is possible to have more than one candidate phrase filter. We have done some evaluation of multiple filters and their individual contribution to performance using a Phrase Concept Counting Mapper (PCCM) (rather than a Language Model Concept Mapper(LMCM)) [7].

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

1. *Candidate Phrase Generation*: Construct a list of candidate phrases based on the common words between all concept instances and the source phrase. If only one candidate phrase remains, return its associated concept name.

2. *Candidate Phrase Filtering*: The list of candidate phrases is trimmed through using one or more filters (see Section 3.2.2). After each filter is applied, if only one candidate phrase remains, its associated concept name is returned. If more than one candidate phrase remains, they are passed to the next candidate phrase filter, if one exists, or to the final concept mapper. We present several variations of a single filter, *coverage*, which measures word overlap between a source phrase and a concept instance. In previous work, we applied two filters: a) a different coverage algorithm which used inverse phrase frequency (see Section 3.2.2) as well as several heuristics, and b) a *coherence* filter, which measures word order rather than overlap between a source phrase and a concept instance [7].

3. *Final Mapping*: If more than one candidate phrase remains after all filters have been applied, the candidate phrases are passed to a final stage to perform concept mapping. The final concept mapping is responsible for finding the best matching candidate phrase among the remaining candidate phrases. Many possible approaches to final concept mapping are possible. In this paper, we describe a language model approach to final concept mapping (see Section 3.4). In previous work we used a phrase counting method [7]. The final concept mapping returns the best-matching concept name (or concept names in cases of scoring ties). The difference between incremental filters and the final concept mapper is that the final concept mapper is expected to find the best-matching concept, while the filter is designed to narrow the choices down based on a feature, such as word coverage or word order.

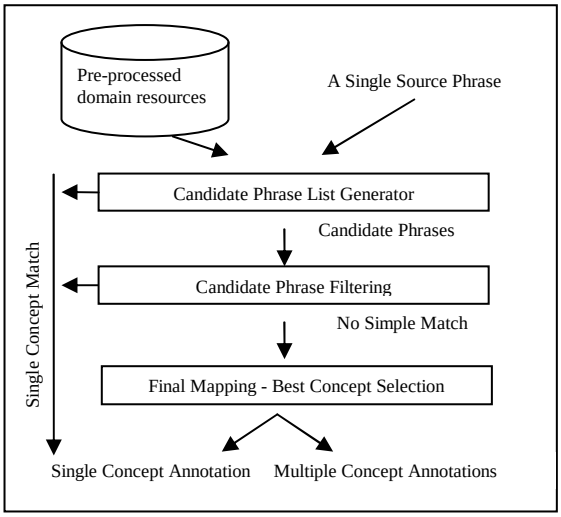

Figure 1: Architecture of CONANN

3.2 Domain Resource Pre-processing

Domain resource processing is done once with each new domain resource and then stored externally. Domain resource processing consists of converting UMLS text-based resources into a format usable for fast in-memory access. CONANN uses a set of pre-processed hash tables to allow fast in-memory lookup of words and candidate phrases. The tables are built based on the UMLS text-based files, specifically the ones included in the latest MetaMap release, version 2.4.b [8]. The tables are loaded from external storage at CONANN startup as part of CONANN initialization, and remain in main memory until the annotator is shutdown. In addition, calculations which can be performed ahead of time, such as inverse phrase frequency and concept language models (described in Sections 3.2.1 and 3.2.2, respectively) are completed.

3.2.1 Inverse Phrase Frequency Calculation

As part of pre-processing, each word in the UMLS is given a weight based on its usage in all concept instances within UMLS. In information retrieval, inverse document frequency value (IDF) uses the frequency of a word across all documents as a way to identify semantically-focused words [12]. Semantically-focused words do not frequently occur across all documents within a collection, and thus are more likely to have more discrimination. To apply the ideas of TF*IDF [13] to CONANN, each UMLS concept instance is substituted for document, resulting in a weight called Inverse Phrase Frequency (IPF), shown in Figure 2. Each unique UMLS word is assigned its semantic importance based on

its inverse phrase frequency value. More semantically important words will be given a higher weight than semantically unimportant words. The idea is to give some indication of the importance of a word in UMLS based on its usage within all UMLS concept instances. Term frequency, typically combined with IDF in information retrieval, is not considered here since it is highly likely the frequency for each word within a phrase will be one, due to the short length of phrases. Like inverse document frequency, IPF assumes term independence. This assumption is false since phrase words are combined in particular order to have meaning. For example, the terms in the phrase *lung cancer* are not independent since it is not possible to consider each individually and retain the original meaning. However, the term independence assumption is accepted so that we can focus in the most semantically meaningful words in identifying candidate phrases. In addition, we can apply a separate filter to score candidate phrases higher based on their common word order with the source phrase. This will overcome the limitations of the word independence assumption of the coverage filter.

$$InversePhr\ aseFrequen\ cy = \log \frac{N}{n_i}$$

Figure 2: Inverse Phrase Frequency. *N* is the number of UMLS phrases, *i* is a UMLS word, and n_i is the number of UMLS phrases word *i* appears in.

3.2.2 Concept Language Models

A unigram language model is built for each of the concepts in UMLS. Each UMLS concept contains one or more synonymous phrases, called concept instances, as previously shown in Table 1. The language model contains the word identifier, frequency of the word in the concept, and the probability of the word occurring across all concept instances within the concept, that is, $P(w) = \frac{|w|}{N}$, where *w* is a word in a particular concept's concept instances, |*w*| is a count of the number of times the word appears in all of the concept instances of a concept, and *N* is the total number of words in all of the particular concept's concept instances. Since there are 797,152 concepts defined in the UMLS resource used, 797,152 unigram language models were generated.

All of the UMLS concepts are assumed to be independent, even though the UMLS Metathesaurus concepts are organized into a hierarchical form using the UMLS Semantic Network. The Semantic Network categorizes and provides relationships between UMLS concepts. The concept independence assumption allows us to evaluate the effectiveness of language models in the concept mapping task without considering other factors which may influence performance. For example, it may be possible that considering semantic relations between candidate concepts (e.g., hypernymy, hyponymy, meronymy) would allow better selection among related concepts and therefore increase performance. In our current language model, we assume each concept (or concept name) is independent of other concepts (or other concept names) in UMLS.However, if we consider semantic relations, then in our evalution of concept language models, it would not be clear whether the performance of the final concept mapper was due to concept language models or semantic relations or some combination of the two.

3.3 Candidate Phrase List Generation

The first step in annotating a source phrase is to find a list of all possible candidate phrases in UMLS. This pool of phrases represents possible matches. Finding candidate phrases is done by first removing all words from the source phrase which do not appear in UMLS. Stop words in the PubMed stop list are also removed [14]. The words in the source phrase are mapped to their UMLS base form, which removes inflection. For example, 'cancers' is mapped to 'cancer.' This is done to eliminate word variation, and to allow exact matching of concept instance 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 instances 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. It is not required that a candidate phrase have all words in common, since exact mappings between a source phrase and concept instances are expected to be rare. In addition to finding all candidate phrases having one or more of the base words, the same process is repeated for all word variants of the base word. For example, *pulmonary* is a variant of *lung*, so phrases such as *pulmonary carcinoma* will be added to the list of candidate phrases.

3.4 Coverage Filter

Coverage measures the overlap of words in common between a source phrase and a candidate phrase. Each candidate phrase is given a score based on the word overlap between it and the source phrase. The coverage filter removes candidate phrases which do not have as many words in common with the source phrase. In contrast to existing systems such as MetaMap [3], SAPHIRE [11], and IndexFinder [15], which consider the count of common words between a source phrase and a candidate phrase (i.e., binary weighting), we also consider the information contribution of each word in the source phrase, as measured by each word's IPF value (see Section 3.2.2). The idea behind IPF word weighting as opposed to binary word weighting is that candidate phrases which have the most semantically meaningful words in common with the source phrase are favored.

Both the binary and IPF weighting of common source and candidate phrase words are used in two different candidate phrase scoring approaches. The first approach, naïve, simply sums the word weights for words in common between the source and candidate phrase. Figure 3 shows the how the IPF and binary values of words in common are summed to form the naïve candidate phrase score. Words in the candidate phrase which are not in the source phrase are ignored.

A second method for scoring candidate phrases comes from the Metamap Transfer system and is called Involvement [16]. Involvement first computes the normalized word weights of words in common between a source phrase and a candidate phrase in both directions (phrase involvement), and then averages the two phrase involvement values to determine the candidate phrase score. For example, consider the source phrase words {A, B, C} and candidate phrase words {A, B} and assume binary word weights. The candidate phrase involvement is 2/2, since candidate phrase words {A,B} are contained in the source phrase. The

source phrase involvement is 2/3 since the source phrase words {A,B} are also in the candidate phrase words, but source phrase word {C} is not. The candidate phrase score is then the average of (2/2 + 2/3)/2, or 0.83. As in the naïve scoring method, the involvement score is calculated using binary values (as in the example), and also using IPF values.

$$NaiveCandidateScore = \sum_{i=1}^{N} WordWeight_i$$

Figure 3: Naive candidate scoring method. *N* is the number of words in common between a source and candidate phrase. *WordWeight* is either 1 for binary weighting or the inverse phrase frequency value for IPF weighting.

Once all candidate phrases have been scored using one of the word weighting approaches and one of the scoring approaches, the standard deviation of the candidate phrase scores for the set of candidate phrases is calculated. A threshold value is chosen as the mean candidate phrase score plus two standard deviations. All candidate phrases having candidate phrase scores greater than or equal to the threshold value are passed to the final concept mapper. By using two standard deviations, we capture the top 5% [17] of candidate phrases. If no candidate phrases have a candidate phrase score value greater than or equal to the threshold value, the candidate phrases with the highest candidate phrase score value are passed to the final concept mapper.

3.5 Concept Mapping Using Language Model

CONANN uses a multinomial unigram language model to find the concept most likely to have generated the source phrase. A list of candidate phrases is first retrieved from the output of the coverage filter (Section 3.4). Each candidate phrase belongs to one or more concepts. A list of concepts is generated from the candidate phrases to form a set of candidate concepts.

Each candidate concept is assigned a score based on its probability of generating the source phrase. The probability score is calculated as shown in Figure 4, which is a standard unigram language mixture model [18] which combines a source phrase word (w) probability within the concept language model ($M_{concept}$- see Section 3.2.2) with the source phrase word probability of the entire UMLS phrase collection ($M_{conceptCollection}$). We initially set λ=0.5 to balance the concept language model with the collection model. We also evaluated the extreme values of λ=0.1 and λ=0.9, but did not notice any change in the final concept annotation output. Also, to allow for more word variation, we also expand the source phrase words to include all source phrase word variants (provided by the UMLS resources) of each source phrase word.

$M_{conceptCollection}$ is calculated as part of pre-processing (see Section 3.2.2) and contains the probability of UMLS words occurring across all UMLS phrases. $M_{concept}$ is the language model for a particular concept based on the concept instances for the particular concept. For example, the concept *Lung Cancer* might contain the concept instances {*lung cancer*, *pulmonary carcinoma*}. $M_{concept}$ for the concept *Lung Cancer* is the language model constructed from these two concept instances.

Each candidate concept is assigned a score by retrieving the $M_{concept}$ and $M_{conceptCollection}$ probabilities of each source phrase word and applying the retrieved probability values as shown in Figure 4. The idea is to get the probability that the concept generated the source phrase, using each concept's language model which is composed of one or more concept instances defined by domain experts. The highest-probability candidate concept is then output as the best-matching concept for the source phrase. In the case of ties, all of the highest-scoring concepts are output.

$$P(concept) = \prod_{w \in SrcPhrase} ((1-\lambda)P(w \mid M_{concept} + \lambda P(w \mid M_{conceptCollection}))$$

Figure 4: Multinomial Unigram Language Mixture Model

3.6 Concept Mapping Using Phrase Counting

The Phrase Counting final concept mapping method was described in our previous work [7] and we include it here for comparison to the language modeling final concept mapping method. In the phrase counting method, the candidate phrases are clustered by the concepts they belong to. Each candidate concept is then scored based on the number of candidate phrases it contains. The highest scoring candidate concept is then output as the best-matching concept for the source phrase. In the event of tie scores, multiple candidate concepts are output. The idea is that the number of candidate phrases per concept after filtering gives an indication of the matching likelihood of a source phrase to a concept.

4. PREVIOUS WORK

We are not aware of any concept annotators using a language model approach. CONANN uses a unigram language model, which is widely used in information retrieval research [18] [19]. Other language model applications, such as speech recognition, use more than one n-gram to capture the order of the text [20]. The unigram model is more advantageous in concept annotation applications because it allows for gaps in word order when matching a source phrase to a concept instance, where exact matches usually do not occur. In addition, the unigram language model has proven an effective approach in information retrieval, outperforming the vector space model [19].

Most work in semantic annotation for biomedical text is performed to support semantic indexing/retrieval and data mining of biomedical texts [3]. Our work is most closely related to MetaMap [3], KnowledgeMap [5], and SAPHIRE [4]. We focus on scoring candidate phrases, since that is one of the primary differences between systems, SAPHIRE uses simple and partial mapping, and for candidate phrase scoring combines measures of term overlap, term proximity, and length of term matches. KnowledgeMap uses simple and partial matching, and for candidate phrase scoring uses an exact match approach and if no matches are found, performs iterative variant-word-generation and re-matching. KnowledgeMap also offers a disambiguation stage which uses concept co-occurrence information derived from existing medical texts to find a best-matching concept. MetaMap uses simple, partial and complex mapping. MetaMap scores candidate phrases using a mixture of four different scores: a)

Centrality where a source phrase head term used in concept instance; b) *Variation* how far a source phrase term variant is from concept instance term; c) *Coverage* which measures the overlap between source phrase and concept instance terms, ignoring gaps; and d) *Coherence* which finds term sequence overlaps between source phrase and concept instance. Compared to SAPHIRE, our CONANN uses simple and partial matching, but does not score every candidate phrase for final mapping. Like KnowledgeMap and MetaMap, we incorporate word variants of the source phrase, but we do not incorporate disambiguation or exact matching as KnowledgeMap does or extensive word variants generation as MetaMap does. Our system reduces computational complexity by deferring complex scoring until after most candidate phrases have been eliminated. In addition, we build a language model of each concept's phrases, whereas existing systems consider each candidate phrase as independent of one another, even from the same concept.

Other related systems include SENSE [21], which translates source and concept instance to low-level semantic factors, then performs exact matching of the semantic factors; Concept Locator [22] which simply sub-divides a phrase & looks for exact matches; PhraseX [23] which focuses on phrase identification and performs an exact match with candidate phrases; and IndexFinder [15] which treats the source text as a bag of words and finds all matching words, regardless of their location. In some systems, such as MetaMap [3], efforts are made to find a best-matching concept, while in other systems, such as IndexFinder [15], all possible concepts are found. The difference is usually determined by the application in which the annotations will be used. For example, finding all concepts within a source text is useful in search and retrieval indexing, while best-matching annotations are useful in applications such as text summarization where the meaning of small text units (e.g. phrases), is needed.

Other work in biomedical text extraction includes the MEDLEE system which has been primarily used for biomedical reports, such as the subdomains of radiology and mammography [24], and GENIEs, which is an extension of MEDLEE for biological entity and relation identification.

5. EVALUATION

Evaluation of the annotation system is done using both an intrinsic and an extrinsic method. The intrinsic evaluation is intended to evaluate the speed and accuracy of CONANN against an existing biomedical concept annotator. The extrinsic evaluation is designed to measure the effect of annotation output on a task. We chose text summarization using concepts as the task. Eight variations of CONANN were used. The four filtering methods: a) Naïve IPF, b) Naïve Binary, c) Involvement Binary, and d) Involvement IPF were combined with the two final concept mapping approaches: a) Phrase Counting and b) Language Model.

5.1 Evaluation Corpus

For the intrinsic evaluation we required a corpus of biomedical noun phrases. For the extrinsic evaluation, we required biomedical texts and several model summaries, which are used for comparison to system-generated summaries. In both cases our corpus was culled from a citation database of approximately 1,200 oncology clinical trial papers physicians feel are important to the field [25]. Of the 1,200 papers cited, 24 were randomly selected based on the minimum requirements of the ROUGE summary evaluation tool [26]. The PDF versions of these 24 papers were then converted to plain-text format and manually processed to remove the abstract, graphics, tables, figures, captions, citation references, and the bibliography section.

For the noun phrase corpus, the resulting texts were processed by MetaMap to find all noun phrases and their corresponding concept annotations in the 24 papers, resulting in a corpus of 4,410 unique phrases. The corpus was pruned to retain only those phrases which MetaMap annotated with a single concept, allowing for meaningful mapping comparisons between the two systems. There were 1,628 phrases with a single MetaMap concept annotation. This set of phrases was used to perform the intrinsic evaluation. Concepts with multiple MetaMap mappings were excluded because they are usually caused by a) the inability of MetaMap to disambiguate among multiple concepts and b) breaking a single phrase into two or more constituent phrases and then mapping each phrase individually. In such cases, it is not clear what the correct mapping should be. We chose to use only phrases where MetaMap could map a single phrase to a single concept. This does not affect the testing of the language model mapping, as the final mapping process is required to choose the best concept among multiple candidate concepts.

For the summary corpus, each of the 24 texts was summarized by three different domain experts, resulting in three summaries for each of the 24 texts. The task presented to each human summarizer was to select 20% of the sentences within each text to form a summary. The human summaries are called model summaries to indicate they are what people consider good summaries. In effect, the human summarizers are performing the same extractive task as the system summarizer. The human summarizers are medical students in their final year of study.

5.2 Intrinsic Evaluation

The intrinsic evaluation is intended to evaluate the speed and accuracy of CONANN against an existing biomedical concept annotator. We use the MetaMap system [3] provided by the United States National Library of Medicine as the baseline system.

To measure the amount of time it takes for MetaMap to annotate the test corpus of phrases, MetaMap was executed using the 1,628 corpus phrases as input. CONANN was then executed against the same set of 1,628 phrases and its annotation time measured. CONANN also produced concept annotations for the list of phrases. These mappings were then compared to MetaMap, producing the annotation precision metric shown in Figure 5.

Accuracy is measured by comparing CONANN's annotation of each phrase to the MetaMap's annotation output for each phrase. There are two measures for the intrinsic evaluation: (a) precision, and (b) phrase annotation time. The first measure looks at the accuracy of the concept annotation, and the second measure looks at the speed of the concept annotation. As shown in Figure 5, the Annotation Precision measure uses the same idea as in information retrieval, but adapted to fit concept mapping [27]. Annotation Precision is defined as the fraction of mapped concepts which are correct. Recall is not considered because the source phrase corpus that is correctly annotated by MetaMap is

only provided to CONANN to annotate, and so recall is not meaningful for this evaluation. For measuring speed, the average time to annotate a phrase is used, which is calculated by taking the total annotation time divided by the total number of phrases annotated, as shown in Figure 6.

$$Annotation\ \Pr ecision = \frac{\#\ of\ Correct\ Concepts}{Total\ \#\ of\ Concepts\ Mapped}$$

Figure 5: Annotation precision metric

$$AveragePhraseTime = \frac{Total\ AnnotationTime}{Total\ \#\ of\ Phrases\ Mapped}$$

Figure 6: Average phrase time metric

5.3 Extrinsic Evaluation

The output of a concept annotator is a list of phrases and their associated domain-specific concepts. This output is an intermediate format, not directly useable by an end-user. The extrinsic evaluation is a complimentary evaluation to the intrinsic evaluation, designed to show the usefulness of the concept output in some task. We selected text summarization as the end-user task. We used two probabilistic summarizers, FreqDist [28] and a version of SumBasic [29] modified to use concepts rather than terms. Both summarizers only use concept frequency as the sole feature to select salient sentences. Both summarizers' performance is entirely reliant on the frequency of 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. Conversely, if the concept identification is not accurate, text summarization performance will degrade.

The corpus of 24 texts (see Section 5.1) is annotated using both CONANN and MetaMap. 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 both annotators. The system-generated summaries are then evaluated using ROUGE (Recall-Oriented Understudy for Gisting Evaluation) [30], which is an automated evaluation tool which compares a system-generated summary from an automated system with one or more ideal summaries produced by people. ROUGE uses n-gram co-occurrence to determine the overlap between a summary and the models. An n-gram can be considered as one or more consecutive words. The ROUGE parameters from the DUC 2005 conference [31] are used to evaluate system summarizer performance. The ROUGE-2 and ROUGE-SU4 recall scores are used. 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 ROUGE scores indicate the n-gram overlap between the source text and the model summaries. The summary output from both CONANN and MetaMap annotators is evaluated against manual summaries generated from domain experts.

It should be noted that ROUGE scores are designed to be a ranking mechanism among summarizers working on the same data set [32]. It is not possible to compare ROUGE results outside of the corpus and model summaries used in a particular evaluation. For example, it would be inappropriate to compare the ROUGE scores presented here with ROUGE scores generated in other evaluations, such as the Document Understanding Conferences, since the source texts and model summaries (as well as domain) are different.

The two frequency-based summarization algorithms used are SumBasic and FreqDist. SumBasic [29] is. SumBasic first finds the probability distribution of all concepts within a source text. It then iteratively finds the highest-scoring sentence by summing the probabilities of all concepts within a sentence. After the highest-scoring sentence is selected, the probability of each concept in the selected sentence is reduced so that weaker concepts have a chance to be incorporated into a summary. The FreqDist algorithm [28] builds a frequency distribution model of the text's concepts. It then selects sentences which, when added to the summary constructed so far, best match the frequency distribution of the source text. The goal is to get the frequency distribution of the summary and the source text to match as closely as possible.

6. RESULTS

The intrinsic evaluation compares the speed and accuracy of our CONANN annotation system versus the MetaMap baseline system. It also shows that the Language Model Concept Mapper (LMCM) outperforms the Phrase Counting Concept Mapper (PCCM). The extrinsic evaluation compares the use of the MetaMap and CONANN generated concept annotations on a text summarization task.

6.1 Intrinsic Evaluation Results

As shown in Figure 7, MetaMap initialization time was 88 seconds, while for our CONANN, initialization time ranged from a low of 20 seconds using the Phrase Counting Concept Mapper (PCCM) to 40 seconds for the Language Model Concept Mapper (LMCM). The LMCM took longer to initialize than the PCCM because it required loading the approximately 700,000 language models (one for each UMLS concept).

Figure 8 presents the total time to annotate all 1,628 phrases in the evaluation corpus. MetaMap total annotation time was 5.7 minutes, while CONANN PCCM ran with a high of 22 seconds using the Coverage-Involvement filter while CONANN LMCM ran with a high of 1.76 minutes using the Coverage-NaiveBinary filter. Figure 9 shows the average time to annotate each phrase. Average Phrase Annotation Time is calculated by taking the total annotation time and dividing it by 1,628, which is the total number of phrases annotated. MetaMap average time to annotate a phrase was 208 milliseconds, while CONANN ranged from a high of 14 milliseconds for the PCCM to 64 milliseconds for the LMCM.

Table 2 shows the CONANN annotator precision for each coverage filter and concept mapper combination. Two precision scores are presented. The first is the precision when matching CONANN's output exactly with MetaMap's output. The second is the precision when matching any of the top five concepts produced by CONANN for a single phrase with the single

MetaMap concept. The best performing PCCM (0.75) uses the binary involvement filter. Precision increases to 0.90 when using the top five CONANN concepts, indicating that an improved final concept mapper method could increase precision. The best performing LMCM (0.85) also uses the binary involvement filter, and its precision increases to 0.93 when using the top five concepts. LMCM outperforms PCCM in exact matching from 13%-20% depending on the coverage filter. LMCM performance can be attributed to the fact LMCM considers all concept instances of a concept, while PCCM considers only those concept instances which have passed through the coverage filter.

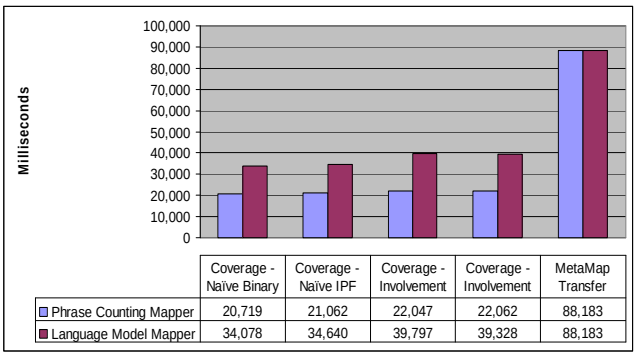

Figure 7: Annotator initialization time

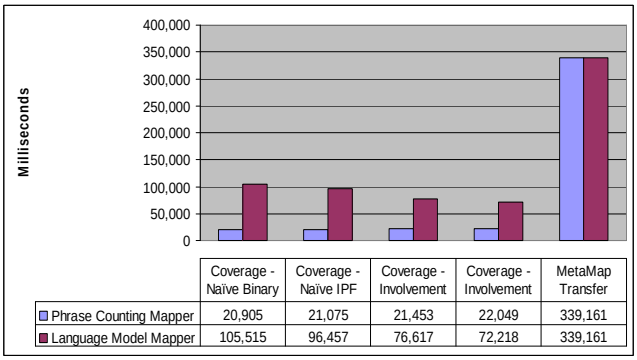

Figure 8: Total annotation time

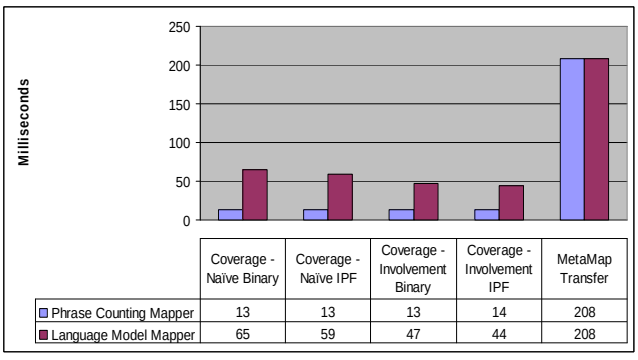

Figure 9: Average phrase annotation time

The intrinsic evaluation shows the best-performing word coverage filter for both phrase counting and language model mapping is binary involvement. The highest precision final concept mapper is language model, but has the tradeoff of longer average annotation phrase annotation time at over three times longer than PCCM. The precision of LMCM is 13-20% higher than the PCCM for the exact match. For average annotation time, LMCM is over four times faster than MetaMap while PCCM is nearly 15 times faster. The tradeoff CONANN makes for higher annotation performance is less precision when compared to MetaMap. The impact of this precision tradeoff is evaluated in the extrinsic evaluation, discussed in the next section.

Table 2: CONANN Precision using MetaMap as baseline

Map	*Filter*	*Exact Match*	*Top 5*
PCCM	Coverage - Naïve Binary	0.66	0.78
	Coverage - Naïve IPF	0.63	0.75
	Coverage - Involvement Binary	0.75	0.90
	Coverage - Involvement IPF	0.67	0.87
LMCM	Coverage - Naïve Binary	0.76	0.86
	Coverage - Naïve IPF	0.74	0.83
	Coverage - Involvement Binary	0.85	0.93
	Coverage - Involvement IPF	0.81	0.91

(Note: PCCM represents Phrase Counting Concept Mapper and LMCM represents Language Model Concept Mapper)

6.2 Extrinsic Evaluation Results

Table 3 shows the ROUGE-2 and ROUGE-SU4 scores using the FreqDist summarizer with both CONANN and MetaMap annotation output. For the ROUGE-2 and ROUGE-SU4 metrics, both the FreqDist and SumBasic summarizers using any of the CONANN variations outperform FreqDist and SumBasic using MetaMap. Tables 3 and 4 show CONNAN with IPF word weights in word coverage filtering outperforms MetaMap annotations from 1.5% to 6.7% in the extrinsic text summarization task (marked with * in tables 3 and 4).

For the Phrase Counting Concept Mapper (PCCM), ROUGE-2 and ROUGE-SU4 metrics indicate the best performing FreqDist and SumBasic summarizers use the binary involvement filter. For the Language Model Concept Mapper (LMCM), the ROUGE-2 and ROUGE-SU4 metrics show for FreqDist that the coverage filter used made no difference. However, for SumBasic, the best performing filter is Naïve Binary.

We conclude from the extrinsic evaluation results shown in Table 3 that even with the lower intrinsic evaluation precision of the CONANN variations shown in Table 2, the extrinsic summarization task is able to use the CONANN concept output to identify important areas of text and improve system-generated text summarization performance as compared to human model summaries.

Table 3: ROUGE scores using Phrase Counting Concept Mapper (PCCM)

Summarizer	*ROUGE-2 Score*	*ROUGE-SU4 Score*
FreqDist - MetaMap	0.12080	0.21864
FreqDist - Coverage - Naïve Binary	0.12872	0.22199
FreqDist - Coverage - Naïve IPF*	0.12897	0.22252
FreqDist - Coverage – Involve. Binary	0.13018	0.22361
FreqDist - Coverage – Involve. IPF*	0.12872	0.22199
SumBasic - MetaMap	0.11412	0.19868
SumBasic - Coverage - Naïve Binary	0.11210	0.20191
SumBasic - Coverage - Naïve IPF*	0.11702	0.20670
SumBasic - Coverage - Involve. Binary	0.11834	0.21039
SumBasic - Coverage – Involve. IPF*	0.11827	0.20770

Table 4: ROUGE scores using Language Model Concept Mapper (LMCM)

Summarizer	*ROUGE-2 Score*	*ROUGE-SU4 Score*
FreqDist - MetaMap	0.12080	0.21864
FreqDist - Coverage - Naïve Binary	0.12897	0.22252
FreqDist - Coverage - Naïve IPF*	0.12897	0.22252
FreqDist - Coverage – Involve. Binary	0.12897	0.22292
FreqDist - Coverage – Involve. IPF*	0.12897	0.22292
SumBasic - MetaMap	0.10920	0.19868
SumBasic - Coverage - Naïve Binary	0.12028	0.21212
SumBasic - Coverage - Naïve IPF*	0.11614	0.20794
SumBasic - Coverage – Involve. Binary	0.11839	0.21053
SumBasic - Coverage - Involve. IPF*	0.11839	0.21053

7. CONCLUSION

We presented an online biomedical concept annotator, CONANN, which takes a source phrase, identifies potential matching concepts and phrases in a domain-specific thesaurus, uses a coverage (word overlap) filtering method to remove unlikely candidate phrases, and maps the source phrase to best-matching concepts. A language model approach that maps concepts based on the probability of the language model of each candidate concept generating the source phrase was contrasted with a simple phrase counting approach. An intrinsic evaluation was performed to compare the precision of CONANN's concept output to the output of MetaMap, a state-of-the-art concept annotator. In addition, an extrinsic evaluation was performed to measure the usefulness of the concept output of each annotator in a text summarization task. CONANN is designed for use with general biomedical text where the best or best top-N concepts are identified at the phrase level.

The intrinsic and extrinsic evaluations show the use of filtering candidate phrases using word coverage and then mapping source phrases to biomedical concepts using language models is effective and faster than current systems. The average annotation time per phrase can be significantly reduced over existing annotation systems (by four to 15 times) for applications which require annotation at the phrase level. The precision of LMCM is 13-20% higher than the PCCM for the exact match. Finally, in the extrinsic text summarization evaluation task, we showed the use of IPF word weights in word coverage filtering outperforms MetaMap annotations.

Future work includes integrating synonymous word variants into the concept language models, finding methods to reduce the size of the initial candidate list, incorporating concept disambiguation, and evaluating IPF word weights in other string similarity matching algorithms. We would also like to consider including semantic relations into concept language models to see whether the performance of the final concept mapper improves. Our eventual goal is to provide a biomedical concept annotator operating at the phrase level which has high accuracy compared to existing systems, and which can operate in an online environment. Such a system would be useful for ad-hoc physician and biomedical research tasks such as summarizing texts, question-answering, information extraction, data mining and concept-based indexing and retrieval.

8. REFERENCES

[1] United States National Library of Medicine, "Unified Medical Language System (UMLS)," 2005.

[2] Center for Bioinformatics, United States National Cancer Institute. (2006, National cancer institute thesaurus.

[3] A. R. Aronson, "Effective mapping of biomedical text to the UMLS metathesaurus: The MetaMap program," in *Proceedings of the AMIA Symposium 2001*, 2001, pp. 17-21.

[4] W. R. Hersh and R. A. Greenes, "SAPHIRE--an information retrieval system featuring concept matching, automatic indexing, probabilistic retrieval, and hierarchical relationships," *Comput. Biomed. Res.*, vol. 23, pp. 410-425, Oct. 1990.

[5] J. C. Denny, P. R. Irani, F. H. Wehbe, J. D. Smithers and A. Spickard 3rd, "The KnowledgeMap project: development of a concept-based medical school curriculum database," *Proceedings of the Annual AMIA Symposium*, pp. 195-199, 2003.

[6] L. Reeve, H. Han and A. D. Brooks, "BioChain: Using lexical chaining methods for biomedical text summarization," in *Proceedings of the 21st Annual ACM Symposium on Applied Computing, Bioinformatics Track*, 2006, pp. 180-184.

[7] L. H. Reeve and H. Han, "CONANN: An online biomedical concept annotator," in *Proceedings of the 2007 Data Integration in the Life Sciences Conference (DILS'07)*, 2007,

[8] United States National Library of Medicine, "MetaMap Transfer," 2005.

[9] United States National Library of Medicine. (2006, 28 March 2006). UMLS metathesaurus fact sheet.

[10] National Library of Medicine, United States. (2006, Specialist lexicon and lexical tools. *2006(August 31)*, pp. 1.

[11] W. Hersh and T. J. Leone, "The SAPHIRE server: a new algorithm and implementation," *Proc. Annu. Symp. Comput. Appl. Med. Care.*, pp. 858-862, 1995.

[12] C. D. Manning and H. Schutze, *Foundations of Statistical Natural Language Processing.* ,1st ed.Cambridge, Massachusetts: The MIT Press, 1999, pp. 620.

[13] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," *Journal of Documentation*, vol. 28, pp. 11-21, 1972.

[14] United States National Library of Medicine. (2006, December 14, 2006). PubMed stop word list. *2007(1/18)*,

[15] Q. Zou, W. W. Chu, C. Morioka, G. H. Leazer and H. Kangarloo, "IndexFinder: A method of extracting key concepts from clinical texts for indexing," in *Proceedings of the AMIA Annual Symposium*, 2003, pp. 763-767.

[16] A. R. Aronson, "MetaMap Evaluation," pp. 1-12, 2001.

[17] H. O. Kiess, *Statistical Concepts for the Behavioral Sciences.* Third ed., vol. 1, Boston, MA: Allyn and Bacon, 2002, pp. 568.

[18] C. D. Manning, P. Raghavan and H. Schütze, *Introduction to Information Retrieval*, 1st ed.Cambridge, England: Cambridge University Press, 2007,

[19] X. Liu and W. B. Croft, "Statistical language modeling for information retrieval," in *Annual Review of Information Science and Technology* , vol. 39, B. Cronin, Ed. American Society for Information Science and Technology, 2005, pp. 1-31.

[20] R. Rosenfeld, "Two decades of statistical language modeling: where do we go from here?", *Proceedings of the IEEE*, vol. 88, pp. 1270-1278, 2000.

[21] Y. L. Zieman and H. L. Bleich, "Conceptual mapping of user's queries to medical subject headings," *Proc. AMIA. Annu. Fall. Symp.*, pp. 519-522, 1997.

[22] P. M. Nadkarni, "Concept locator: a client-server application for retrieval of UMLS metathesaurus concepts through complex boolean query," *Comput. Biomed. Res.*, vol. 30, pp. 323-336, Aug. 1997.

[23] S. Srinivasan, T. C. Rindflesch, W. T. Hole, A. R. Aronson and J. G. Mork, "Finding UMLS Metathesaurus concepts in MEDLINE," *Proc. AMIA. Symp.*, pp. 727-731, 2002.

[24] C. Friedman, "A broad-coverage natural language processing system," in *Proceedings of the AMIA Annual Symposium*, 2000, pp. 270-274.

[25] A. D. Brooks and I. Sulimanoff, "Evidence-based oncology project," in *Surgical Oncology Clinics of North America* , vol. 11, Anonymous 2002, pp. 3-10.

[26] C. Lin, "Looking for a few good metrics: Automatic summarization evaluation - how many samples are enough?" in *Proceedings of the NTCIR Workshop 4*, 2004,

[27] W. R. Hersh, M. Mailhot, C. Arnott-Smith and H. J. Lowe, "Selective Automated Indexing of Findings and Diagnoses in Radiology Reports," *J. Biomed. Inform.*, vol. 34, pp. 262-273, 2001.

[28] L. Reeve, H. Han, S. V. Nagori, J. Yang, T. Schwimmer and A. D. Brooks, "Concept frequency distribution in biomedical text summarization," in *Proceedings of the ACM Fifteenth Conference on Information and Knowledge Management (CIKM'06)*, 2006, pp. 604-611.

[29] A. Nenkova and L. Vanderwende, "The impact of frequency on summarization," Microsoft Research, Redmond, Washington, Tech. Rep. MSR-TR-2005-101, 2005.

[30] C. Lin and E. H. Hovy, "Automatic evaluation of summaries using N-gram co-occurrence statistics," in *Proceedings of 2003 Language Technology Conference (HLT-NAACL 2003)*, 2003, pp. 71-78.

[31] National Institute of Standards and Technology (NIST), "Document Understanding Conferences," vol. 2005, 2005.

[32] D. K. Evans, "Identifying similarity in text: Multi-lingual analysis for summarization," *Columbia University*, vol. 1, pp. 1-168, 2005.