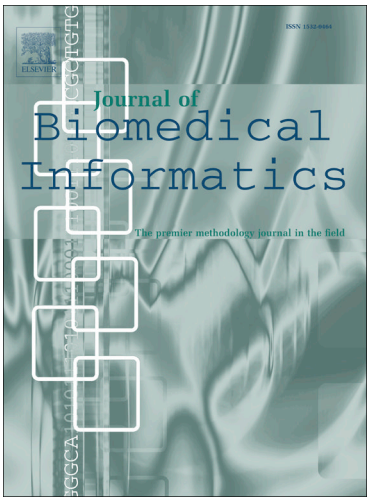


RysannMD: A Biomedical Semantic Annotator Balancing Speed and Accuracy

John Cuzzola, Jelena Jovanovic, Ebrahim Bagheri

PII: S1532-0464(17)30111-9
DOI: http://dx.doi.org/10.1016/j.jbi.2017.05.016
Reference: YJBIN 2788

To appear in: *Journal of Biomedical Informatics*

Received Date: 28 January 2017
Revised Date: 21 May 2017
Accepted Date: 22 May 2017

Please cite this article as: Cuzzola, J., Jovanovic, J., Bagheri, E., RysannMD: A Biomedical Semantic Annotator Balancing Speed and Accuracy, *Journal of Biomedical Informatics* (2017), doi: http://dx.doi.org/10.1016/j.jbi.2017.05.016

RysannMD: A Biomedical Semantic Annotator Balancing Speed and Accuracy

John Cuzzola[1]* jcuzzola@ryerson.ca
Jelena Jovanovic[2] jeljov@gmail.com
Ebrahim Bagheri[1] bagheri@ryerson.ca

* Corresponding author

1. Laboratory for Systems, Software and Semantics (LS3), Ryerson University, Ontario, Canada, http://ls3.rnet.ryerson.ca
2. Faculty of Organizational Sciences (FOS), University of Belgrade, Belgrade, Serbia, http://www.fon.bg.ac.rs/eng/

Abstract

Recently, both researchers and practitioners have explored the possibility of semantically annotating large and continuously evolving collections of biomedical texts such as research papers, medical reports, and physician notes in order to enable their efficient and effective management and use in clinical practice or research laboratories. Such annotations can be automatically generated by biomedical semantic annotators - tools that are specifically designed for detecting and disambiguating biomedical concepts mentioned in text. The biomedical community has already presented several solid automated semantic annotators. However, the existing tools are either strong in their disambiguation capacity, i.e., the ability to identify the correct biomedical concept for a given piece of text among several candidate concepts, or they excel in their processing time, i.e., work very efficiently, but none of the semantic annotation tools reported in the literature has both of these qualities.

In this paper, we present RysannMD (*Ryerson Semantic Annotator for Medical Domain*), a biomedical semantic annotation tool that strikes a balance between processing time and performance while disambiguating biomedical terms. In other words, RysannMD provides reasonable disambiguation performance when choosing the right sense for a biomedical term in a given context, and does that in a reasonable time. To examine how RysannMD stands with respect to the state of the art biomedical semantic annotators, we have conducted a series of experiments using standard benchmarking corpora, including both gold and silver standards, and four modern biomedical semantic annotators, namely cTAKES, MetaMap, NOBLE Coder, and Neji. The annotators were compared with respect to the quality of the produced annotations measured against gold and silver standards using precision, recall, and F_1 measure and speed, i.e., processing time.

In the experiments, RysannMD achieved the best median F_1 measure across the benchmarking corpora, independent of the standard used (silver/gold), biomedical subdomain, and document size. In terms of the annotation speed, RysannMD scored the second best median processing time across all the experiments. The obtained results indicate that RysannMD offers the best performance among the examined semantic annotators when both quality of annotation and speed are considered simultaneously.

Keywords: Automated Semantic Annotation, Entity Linking, UMLS Metathesaurus, Biomedical Ontologies, Natural Language Processing, Medical Terminology

1. Background

Biomedical research and clinical practice are producing ever increasing quantities of unstructured textual content in the form of scientific papers, medical reports, and physician notes, just to name a few. As a result, the biomedical community has now access to massive and continuously growing collections of textual content that needs to be organized, curated, and managed in order to be effectively used for both clinical practice and research purposes. In addition, the availability of biomedical databases, tools, and similar resources is also continuously increasing. For example, according to the 2016 Molecular Biology Database Update, there are 1,685 large-scale biological

databases. The resulting difficulties faced by researchers and practitioners are nicely illustrated by Jonquet et al. (2009): "a researcher studying the allelic variations in a gene would want to know all the pathways that are affected by that gene, the drugs whose effects could be modulated by the allelic variations in the gene, and any disease that could be caused by the gene, and the clinical trials that have studied drugs or diseases related to that gene. The knowledge needed to address such questions is available in public biomedical resources; the problem is finding [and connecting] that information." (Jonquet et al., 2009; p.2).

To enable effective and efficient use of the available and continuously evolving textual information repositories, the biomedical community has long been experimenting with and making use of Natural Language Processing (NLP) methods and techniques (Meystre et al., 2008; Fleuren & Alkema, 2015). In this paper, our focus is on semantic annotation, an NLP task specifically designed for detecting and disambiguating biomedical concepts mentioned in the text. For instance, in the sentence "*Heart attack victims, sometimes caused by MS, often suffer brain damage*", the role of a semantic annotation process would be to automatically recognize that the term "*Heart attack*" refers to the concept Myocardial Infarction (C0027051) from the UMLS Metathesaurus and hence establish a connection between that concept and its textual representation in the given sentence. Likewise, the term "*brain damage*" would be connected to the concept Brain Injuries (C0270611). By connecting terms with machine understandable concepts, semantic annotation is setting the grounds for (full or partial) automation of various laborious and time-consuming tasks such as search, classification, and/or organization of biomedical resources.

Contemporary general-purpose semantic annotation tools primarily rely on a combined use of text processing techniques, large-scale machine-readable knowledge bases, semantic similarity measures, and/or machine learning (ML) methods (Jovanovic et al., 2014). Graph-based algorithms and metrics are frequently used instead of, or in addition to, ML techniques (Agirre et al., 2010; Hoffart et al., 2011). Specifically, in the biomedical domain, semantic annotators typically rely on one of the following two general approaches (Tseytlin et al., 2016): term-to-concept matching and/or machine learning-based methods.

Term-to-concept matching, also known as dictionary lookup, is based on matching terms from the parsed text to the entries in a structured vocabulary/dictionary derived from a knowledge base, e.g., UMLS Metathesaurus. Due to its intrinsic flexibility, this approach is typically found in general-purpose biomedical annotators whereby changing the dictionary and (optionally) customizing the text-preprocessing phase can lead to the adoption of the same semantic annotator for a different biomedical sub-domain(s).

On the other hand, ML-based approaches are often found in annotators developed for specific, well-defined application cases such as annotating drugs in medical discharge summaries (Tikk and Solt, 2010) or recognizing gene mentions in biomedical papers (Hsu et al., 2008). These annotators tend to excel on the specific tasks they were developed for, but would give poor performance on other tasks. In addition, as they are often based on supervised ML methods, their development, namely training of the ML model, requires large domain-specific corpora manually annotated by medical experts, which is very expensive. The high costs associated with this approach has led to a shift towards unsupervised or semi-supervised ML methods that, instead of manually labelled data, rely on human expertise encoded in expert-curated knowledge bases (Chasin et al., 2014).

Our focus in this paper is on general-purpose biomedical annotators. While adaptable to diverse biomedical subdomains, annotators in this category tend to have weak disambiguation capacity, i.e., capacity to choose the most appropriate meaning of a term for the given context. Notable exceptions are MetaMap (Aronson & Lang, 2010) and cTAKES (Savova et al., 2010) as they include a Word Sense Disambiguation (WSD) component. More precisely, the original version of cTAKES lacked disambiguation capacity; this deficiency was later resolved through the integration of the YTEX WSD component (Garla & Brandt, 2013). Other annotators avoid the disambiguation step, for example, by finding the longest match of a concept label in the text (Divita et al., 2014); thus, aiming to assure that terms in the text are annotated with the most specific concepts. As an illustration, if the input text mentions

"blood vessel", an annotator would associate this mention only with the UMLS concept having this exact label (C0005847), and would not produce annotation for "blood" though this term also has a corresponding concept in UMLS (C0005767). Another approach is to apply a series of heuristic syntactic rules, i.e., transformations, to the input text to disambiguate concept mentions (D'Souza & Ng, 2015). However, in these and similar cases, matching is done purely at the syntactic level, while semantics of terms in the text and concepts from the knowledge base are not considered; in other words, no attempt is made to find the best matching concept based on the semantics, i.e., contextualized meaning of the terms in the given text. On the other hand, Open Biomedical Annotator - OBA (Jonquet et al., 2009) is an annotation tool that stands out for its reliance on semantics, as its annotation process strongly relies on biomedical ontologies. However, ontologies are *not* used to find unique, unambiguous meaning of biomedical terms in the input text, but to expand a set of annotations, i.e., concepts, identified by a concept recognition service in the first phase of the annotation process. In particular, OBA uses the MGrep service (Dai et al., 2008) for the efficient matching of text against a set of dictionary terms. MGrep is configured to return all the matching concepts (for the above example of "blood vessel", MGrep would return both C0005847 and C0005767). The set of concepts obtained from MGrep is further expanded using the structure and semantics of biomedical ontologies relevant for the domain from the input text. For instance, semantic distance measures are used to extend the initial set of concepts with semantically related concepts; the computation of semantic distance is configurable, and can be based, for example, on the distance between the concepts in the ontology graph.

The biomedical semantic annotation literature and software tools have received more attention in the recent years where reliable system such as NOBLE Coder (Tseytlin et al., 2016), ConceptMapper (Tanenblatt et al., 2010) and Neji (Campos et al., 2013a), among others have been introduced. While such tools stand out for their very short processing times, it should be noted that these tools either completely lack a strong disambiguation capacity or use simple, often heuristic, rules to choose one among several candidates for a particular ambiguous entity mention recognized in the text. Other semantic annotation tools such as cTAKES and MetaMap that do provide strong disambiguation capabilities have been shown to have long processing times. Based on these two types of observations (i.e., lack of strong disambiguation capacity or slow processing times), our objective has been to design a biomedical semantic annotation tool, known as RysannMD, that would strike a balance between sense disambiguation and processing time. To that end, our work has been focused on developing a semantic annotation technology that can provide reasonable performance when choosing the right sense, i.e., UMLS concept, for a biomedical term in cases of overlapping or ambiguous phrases, while keeping reasonable processing time. We will demonstrate, through experiments based on gold and silver standard benchmarking corpora and in comparison with the state of the art biomedical semantic annotators that, by balancing disambiguation performance and processing time, RysannMD presents a suitable alternative to the existing semantic annotation technology in the biomedical domain.

2. Biomedical Semantic Annotation Tools

To further set the context for our work presented in this paper, this section provides more details about the state of the art biomedical semantic annotators that have been used in our benchmarking study, namely MetaMap, cTakes (with YTEX extension), Neji, and NOBLE Coder. These annotators were selected for the following reasons: i) they represent state-of-the-art in the domain of semantic annotation of biomedical entities; ii) like RysannMD, they belong to the category of general-purpose biomedical semantic annotators; iii) their source code is available allowing for benchmarking in the lab.

MetaMap (Aronson & Lang, 2010) is probably the most well known and most widely used biomedical annotator. It was developed by the U.S. National Library of Medicine. It relies on a terms-to-concept matching approach to map terms in the input text to concepts from the UMLS Metathesaurus. Each annotation includes a score that reflects how well the concept matches the corresponding term/phrase from the input text. The annotation process can be adapted to different annotation tasks by configuring several elements of the annotation process such as the

vocabulary used, the syntactic filters applied to the input text, and the permissible types of matches between text and concepts, just to name a few. Besides the flexibility enabled by these configuration options, another strong aspect of MetaMap is its thorough and linguistically principled approach to the lexical and syntactic analyses of input text. However, this thoroughness is also the cause of one of MetaMap's main weaknesses, namely its long processing time, and thus its inadequacy for real-time annotation of large corpora. Another weakness lies in its disambiguation approach that is not able to effectively deal with ambiguous terms (Aronson & Lang, 2010). In particular, to disambiguate biomedical terms, MetaMap combines two approaches: i) removal of word senses deemed problematic for (literature-centric) NLP usage, based on a manual study of UMLS ambiguity; ii) a WSD algorithm that chooses a concept with the most likely UMLS Semantic Type (ST) for a given context (Humphrey et al., 2006). The WSD algorithm relies on ST indexing of sentences that contain ambiguous word(s). To this end, an UMLS ST index is created, where each word from a training set is associated with a vector of alphabetically ordered ST rankings; such a vector can be thought of as a description of the semantic context in which the corresponding word occurs. The training set consists of titles and abstracts of over 910K MEDLINE citations, with over 232K unique words (after removing words that are not expected to bear meaning). The unique words from the training set are assigned their ST rankings through Journal Descriptors, i.e., terms representing biomedical specialties (National Library of Medicine, 2002), of the journals the citations originate from (details are given in (Humphrey et al., 2006)). After a sentence with ambiguous term(s) has been ST indexed, the next step is to compute the centroid of the ST vectors associated with all the indexed words in the sentence (by taking the average of ST rankings of individual words from the sentence). The ST with the highest rank in the centroid vector is considered to reflect the true meaning of the sentence, and the ambiguous word is disambiguated using the candidate concept with the identified highest-ranked ST.

Clinical Text Analysis and Knowledge Extraction System (cTAKES) is another open-source toolkit for semantic annotation of biomedical documents in general, and clinical research texts in particular (Savova et al., 2010). It is built on top of two well-known and widely used open-source NLP frameworks: Unstructured Information Management Architecture - UIMA (UIMA, 2016) and OpenNLP (OpenNLP, 2016). cTAKES is developed in a modular manner, as a pipeline consisting of several text processing components that rely on either rule-based or ML techniques. Recognition of concept mentions and annotation with the corresponding concept identifiers is done by a component that implements a dictionary look-up algorithm. For building the dictionary cTAKES relies on UMLS. The concept recognition component does not resolve ambiguities that result from identifying multiple concepts for the same text span. Disambiguation is enabled through the integration of YTEX (Garla & Brandt, 2013) in the cTAKES framework and its pipelines. YTEX is a knowledge-based WSD component that relies on the knowledge encoded in UMLS. In particular, YTEX implements an adaptation of the Lesk method (Banerjee & Pedersen, 2003), which scores candidate concepts for an ambiguous term by summing the semantic relatedness between each candidate concept and the concepts in its context window; the candidate with the highest score is chosen as the correct sense for the ambiguous term. By default[1], YTEX uses the Intrinsic Path semantic similarity measure (Sánchez & Batet, 2011), but can be configured to work with several different semantic similarity measures based on path finding, corpus and/or intrinsic information content (Garla & Brandt, 2012).

NOBLE Coder (Tseytlin et al., 2016) is yet another open-source, general-purpose biomedical annotator. It is based on a term-to-concept matching algorithm, and can be configured to work with arbitrary vocabularies. Its flexibility also lies in the variety of supported concept matching strategies, aimed at meeting the needs of different kinds of NLP tasks. For example, the 'best match' strategy aims at high precision, and thus returns few candidates (at most); as such, it is suitable for concept coding and information extraction NLP tasks. The supported matching strategies allow for the annotation of terms consisting of single words, multiple words, and abbreviations. Thanks to its greedy algorithm, NOBLE Coder is scalable, that is, it can work efficiently with large textual corpora. The tool does not perform WSD in a traditional sense, i.e., using the semantics of the candidate concepts to choose the candidate that

[1] https://code.google.com/archive/p/ytex/wikis/WordSenseDisambiguation_V08.wiki

is the best match for the contextualized meaning of a term in the given text. However, if multiple candidate terms are recognized in a chunk from the input text, it can employ a set of heuristics to reduce the chance of spurious matches. For instance, in case of two or more candidate terms, it chooses candidates that are un-stemmed, thus avoiding potentially incorrect matches due to word stemming. NOBLE Coder was evaluated in a comprehensive benchmarking empirical study that included five state-of-the-art annotators that were compared based on the execution time speed and standard performance metrics (precision, recall, F_1-measure). The comparison was done on two publicly available, human-annotated (gold standard) corpora: ShARe (ShARe, 2016) consisting of annotated clinical notes, and CRAFT (Bada et al., 2012) based on annotated biomedical literature. The study showed that NOBLE Coder performance was comparable to that of the leading biomedical annotators (cTAKES and MetaMap). Error analysis, conducted as part of the evaluation study, identified the following two as the most frequent sources of errors: i) the lack of context/background knowledge, and ii) low level of importance of the recognized concepts, that is, entity mentions and the associated concepts were not present in the gold standard.

Neji (Campos et al., 2013a) is also an open source framework for the annotation of biomedical texts. One of its distinguishing features is high modularity achieved by encapsulating each text processing task in an independent module, with its own input and output specifications. These modules are then combined in a pipeline based on the requirements of a specific annotation task. Another important feature is multi-threaded data processing that assures high speed in the annotation process. Neji makes use of existing software tools and libraries for standard text processing, e.g., tokenization, sentence splitting, lemmatization, with some adjustments to meet the lexical specificities of biomedical texts. For concept recognition, Neji supports both dictionary-lookup matching and ML-based approaches by customizing existing libraries that implement these approaches. For instance, it uses Gimli (Campos et al., 2013b), an open source biomedical recognition engine based on Conditional Random Fields (CRFs). Hence, various CRF models trained for Gimli can be used in Neji, each model targeting a specific biomedical concept type such as gene or protein. Since Gimli performs only Named Entity Recognition, but does not match the recognized entities to concept identifiers from the underlying biomedical knowledge base, Neji introduced a simple algorithm to associate each recognized entity mention with a unique concept identifier. The algorithm is based on a set of prioritized dictionaries: an entity mention is matched against the concepts in a dictionary, starting from the dictionary with the highest priority and moving to dictionaries of lower priority until a match is found. When the first match is found the procedure stops and the entity mention is associated with, i.e. disambiguated by, the matched concept. However, this algorithm does not consider the semantics of an entity mention in the given context, and thus might not qualify as WSD. To improve the quality of its output, Neji can be configured to filter out some of the generated annotations. For instance, a user may require the removal of annotations from the resulting concept tree that are deeper than a specified depth. Neji was evaluated, in terms of its speed and the quality of the produced annotations, in a comprehensive empirical study (Campos et al., 2013a). The study assessed and compared Neji's performance against four contemporary annotators (Whatizit, Cocoa, MetaMap, BANNER) on three manually annotated corpora of biomedical publications: CRAFT, AnEM (Ohta et al., 2012) and NCBI disease corpus (Dogan et al., 2014). Benchmarking was done for several types of biomedical concepts separately, e.g., Species, Cell, Cellular Component, and Neji demonstrated the leading performance for almost all the examined concept types. In terms of speed, it significantly outpaced the contenders.

3. Unified Medical Language System (UMLS)

The knowledge base for RysannMD is the *Unified Medical Language System* (UMLS)[2]. UMLS is a thesaurus of amalgamated medical vocabularies from approximately 200 sources producing a very large lexicon of health domain terms. A major component of UMLS, referred to as the *Metathesaurus*, contains concepts from all the source vocabularies, identifies useful relationships between the concepts, and associates each concept with corresponding

[2] https://www.nlm.nih.gov/research/umls/about_umls.html

names (terms) from all the source vocabularies. Relationships between the concepts are maintained in the MRREL file (Table 1).

RysannMD uses the 2016AA full release of UMLS which consumes approximately 11GB of storage space and contains 2,397,167 unique concepts. This count swells to 6,154,963 when aliases for concepts are included. The database is distributed through a series of flat delimited text files that contain concepts, attributes, relationships, and types. Table 1 describes the specific flat files that RysannMD indexes.

Table 1: Core database files from UMLS indexed by RysannMD for its knowledge base.

Table name	Description	Use in RysannMD
MRCONSO	Main file containing concepts and aliases for each concept.	RysannMD indexes the concept label and alias labels for use as lexical triggers to look for when reading natural language text.
MRSTY	Semantic types of concepts.	RysannMD uses this information to produce output that shows how concepts are related to each other through the categories they share.
MRREL	Contains hierarchy information between concepts such as: immediate parents, immediate child, and immediate sibling relationships.	RysannMD constructs a tree structure to create a vocabulary for each concept. Used in disambiguation of terms.
MRDEF	Contains 215,429 definitions for a subset of the concepts.	RysannMD uses these for model training described in Section 5.5.
MRSAT	Attribute information for each concept. This file maps concepts from the "*Chemical Entities of Biological Interest* (ChEBI)" to UMLS.	Used to benchmark RysannMD's performance against the CRAFT corpus (see Section 7).

4. The Lottery Metaphor

Semantic annotators often rely on some measure of probability in order to disambiguate terms. Consider the term "discharge" and the two possible meanings of "*Body Fluid Discharge*" (C0012621) and "*Patient Discharge*" (C0030685) in UMLS. An annotator would compute a full joint probability model for the likelihood of each of these candidates such that Prob(C0012621) + Prob(C0030685) = 1. However, consider the sentence "*the gun accidentally discharged*". In this context, it is clear that neither of the UMLS choices is appropriate. This demonstrates the challenge with probability models in that the set of choices is not a closed set of possibilities, thus making the calculation of a full joint distribution impossible. Specifically, an annotator must consider the none-of-the-above option, sometimes referred to as the NIL case. RysannMD uses a hypergeometric mixture model to tackle the NIL dilemma. Hypergeometric models are often used in calculating lottery "odds"; given the general familiarity with how lottery functions, we use a lottery metaphor to describe the use of a hypergeometric model in RysannMD.

The Hypergeometric Distribution Model. Suppose we have a container that can hold N lottery balls, and assume that there are 42 balls (i.e. N=42). Now, suppose we fill the container with 10 balls labelled C0012621 and 13 balls labelled C0030685. The counts of 10 and 13 are determined based on the words surrounding "discharge" within the example sentence "*the gun accidentally discharged*". Our lottery container is still not at its capacity, so we top it up with 19 additional balls labelled NIL. Figure 1 illustrates this scenario.

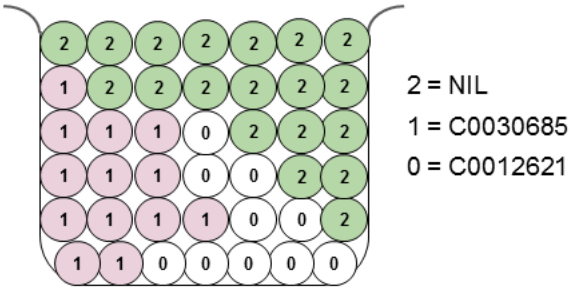


Figure 1: A lottery container holding C0012621(10), C0030685(13) and NIL(19) labelled balls.

This container now holds a mixture of candidate concepts for a recognized spot, i.e., *discharge*, determined by some algorithm that estimates the proportion of lottery balls. In order to select the correct concept for the given context through disambiguation, RysannMD estimates the uncertainty associated with the selection of each of the candidate concepts, including NIL, and chooses the one with the lowest uncertainty score. The computation of uncertainty scores for the candidates is based on the hypergeometric distribution and framed as the task of answering the following questions:

- If 10 lottery balls are drawn at random, what is the probability that none of them are C0012621?
- If 13 lottery balls are drawn at random, what is the probability that none of them are C0030685?
- If 19 lottery balls are drawn at random, what is the probability that none of them are NIL?

Intuitively, we can see that the above questions can be phrased as the conditional probability of not drawing a particularly labelled ball after "x" number of successive draws. We can see that the odds of not drawing any of the NIL balls is the lowest because they have the highest number of balls in the container and we draw more of them (x=19) than we do for C0012621 (x=10) and C0030685 (x=13). Consequently, the answer to "*the gun accidentally discharged*" should be NIL for the word "discharge". Furthermore, because we use a hypergeometric model as a full joint probability distribution, we can calculate the exact probabilities [0-1] for each case of x=10, 13, 19. This allows us to do more than simply pick the answer with the highest number of labelled lottery balls; we can compute confidence scores to assert how certain we are of the concept that was chosen and discard it if it does not meet some minimum confidence threshold. From this example we can generalize the following:

If B_A is the set of all balls and B_b is the set of labelled balls of interest, then $P_{Bb}(none|B_b \rightarrow \infty)=0$ as the count of B_b increases, i.e., $|B_b| \rightarrow \infty$, This translates to the case that as the number of particular labelled balls increases so does the likelihood of randomly selecting that label at least once. Also, $P_{Bb}(none|B_b \rightarrow 0)=1$ or $P_{Bb}(none|B_A \backslash B_b \rightarrow \infty)=1$ as either the count of B_b decreases $|B_b| \rightarrow 0$ or the count of all balls other than B_b increases $|B_A \backslash B_b| \rightarrow \infty$. This states that the odds that randomly choosing no balls labelled B_b increases when there is a small number of B_b balls or when there is a large number of balls other than B_b. From this observation, it is also clear that $P_{NIL}(none|N \rightarrow \infty)=0$ as the size of the container increases $N \rightarrow \infty$. In other words, as the container size gets larger, the odds of choosing NIL increases.

The Rationale for Hypergeometric. Our goal with RysannMD was to develop a biomedical semantic annotator that performs well while still achieving a level of throughput that would allow for reasonable processing times. Modern annotators are either fast but suffer in their accuracy, or are accurate but overly slow to be practical. The challenge lies in constructing a model that can both encompass a large word lexicon *and* capture word co-occurrences within the same document, paragraph or sentence. For example, the word "*discharge*" likely takes on a specific meaning when it is situated near the term "*gun*". When dealing with likelihoods, a probability model is a logical choice. However, this choice is not without its complexities. Consider the question: does *discharge* in the sentence '*after three days of tests I was discharged'* refer to *Patient Discharge* (C0030685)*?* A probability model would reformulate the query as P(C0030685 | w_1="after", w_2="three", w_3="days", w_4="tests", w_5="discharged") which would require prior knowledge of any dependencies (co-occurrences) of w_1 appearing with w_5, w_2 appearing with w_5, w_3 appearing with w_5 and so forth. Other word dependencies not involving the spotted lexical term (w_5) should also be considered such as w_1 with w_3. A *Bayesian Network* could encapsulate these relationships but would be difficult to construct for a large corpus such as the UMLS. This dependency discovery is a challenge in probability modelling. Popular models such as *Naive Bayes* avoid it by assuming complete independence, i.e.,

$P(C0030685|w_1)$, $P(C0030685|w_2)$,..., which is computationally fast but suffers in accuracy. Solutions to this could include *Principal Component Analysis*, which would require a very large sparse two-dimensional matrix in the case of UMLS. Vector solutions such as *Support Vector Machines* and *Neural Network* variants can learn these dependencies but need long vectors to represent UMLS concepts, consequently resulting in excessive training times.

A further speed obstacle exists pertaining to the annotator's datastore holding UMLS. Relational databases can search through petabytes of stored information. However, simple queries such as "*how many times does the word discharge appear in UMLS?*" is faster than queries such as "*how many documents both contain the words discharge and blood together?*". Consequently, even if the co-dependencies are discovered, a semantic annotator would need to query its database on-the-fly using the slower queries to obtain the frequency counts necessary to calculate the probabilities. Now consider a hypergeometric calculation defined as:

$$P(x = k) = \frac{\binom{K}{k}\binom{N-K}{n-k}}{\binom{N}{n}} \quad (1)$$

where N is the size of the container, K is the number of lottery balls labelled with the concept of interest (C0030685 in our example), n is the number of successive draws without replacement (n=K in our model), and k is the exact number of lottery balls of the interested concept we would like to appear after the n draws (k=0 in our model). This equation is fast to compute thanks to its simplicity. Furthermore, the calculations describing how the size of the lottery container (N) and the mixture of labelled balls within it (K) are determined do not depend on knowledge of word co-dependencies and require relatively lightweight queries to the database.

It is important to note that we position our proposed hypergeometric solution not as a replacement for the aforementioned methods, but rather as another method of representation. For perspective, Table 2 offers advantages and disadvantages of each considered method.

Table 2: Summary table of advantages/disadvantages for some commonly used methods.

MODEL	PROS	CONS
Bayesian Network	(1) Can model with "incomplete" data or gaps in the data. (2) Can discover cause and effect relationships. (3) Can tolerate overfitting and bad data (outliers) well.	(1) Generating the dependency graph may be difficult. (2) Causal relationships within the graph may require human insight.
Naive Bayes	(1) Easy to understand and compute. Fast calculation. (2) Good classification performance even when the independence assumption does not hold.	(1) Independence of variables assumption is often incorrect. (2) Gaps in data (unseen cases) can result in incorrect answer of zero probability.
Principal Component Analysis	(1) Dimension reduction can focus on the most important features. (2) Can discover orthogonal (independent) features.	(1) The covariance matrix difficult to construct. Large and sparse. (2) Most important features are the ones with the highest variance.
Support Vector Machines	(1) Can build a model with relatively little data. (2) Handles overfitting well. (3) Can model non-linear data.	(1) Long training time. (2) Slow performance.
Neural Network	(1) Can model (non-)linear and (non-)independent variables. (2) Large selection of Neural Net variants. (3) Incremental learning.	(1) Susceptible to overfitting. (2) Requires large amount of training data. (3) Long training time.
Hypergeometric Distribution	(1) Easy and fast to compute (Equation 1). (2) Lottery metaphor easy to understand. (3) Simple "mixture" model.	(1) Limited to only 1-dimension or feature, i.e., lottery ball labels. Must encapsulate multiple dimensions into a single "super" variable that might be hard to accurately model. (2) May be overly simplistic.

5. The Algorithm

The function of RysannMD is to determine the contextual meaning of natural language text through the linking of biomedical entity mentions recognized in the text to the corresponding semantic concepts within the UMLS knowledge base. This is accomplished through a four stage pipeline of *spotting*, *disambiguation*, *filtering*, and *overlap resolution*. To ease the comprehension of this pipeline, we will use the sample text of "*Heart attack victims, sometimes caused by MS, often suffer brain damage*" as a walkthrough example of the four stages from spotting to overlap resolution.

5.1 Spotting

The objective of this first stage of the RysannMD pipeline is to recognize biomedical n-grams within the input text and provide entity linking candidates from the UMLS knowledge base (KB) for each identified n-gram. The recognized n-grams, i.e., word sequences, are known as "*spots*" or "*mentions*".

The process involves a sliding window of expanding size followed by a search of the KB. The KB is created from the UMLS file MRCONSO:STR (Table 1). A spot and the corresponding entity candidate are discovered when there is a match between the text within the sliding window and the title of a concept within the KB. This procedure is formalized in Algorithm 1.

Input (TEXT): Natural language input text TEXT.
Output(H_s,T): A set H_s of spots with entity candidates for each spot. A set T of all n-grams identified in TEXT.
Method:

1. Let TOK be a sequence of tokenized words of size n from the input text TEXT. TOK = {w_1,w_2,w_3,..w_n}
2. Let k be the maximum size of the sliding window. Default k = 4
3. Initialize set H_s = {}
4. Initialize set T = {}
5. Initialize start of sliding window index i = 1
6. Initialize end of sliding window index j = i
7. Create sliding window $Q_{i,j}$ = {w_i ... w_j} from TOK.
8. Add n-gram $Q_{i,j}$ to set T: T = T + $Q_{i,j}$.
9. Search KB for concepts where MRCONSO:STR(title) = $Q_{i,j}$. Let M be the set of matching concepts.
10. IF M not empty THEN
11. Create spot-to-possible-concepts map ∂=Q_{ij}→M and add it to set H_s: H_s = H_s + ∂.
12. IF j < i+k THEN
13. Expand sliding window: j = j + 1
14. Goto step 7
15. ELSE IF i<n THEN
16. Advance to next token: i = i +1
17. Goto step 6
18. Return set H_s and set T.

Algorithm 1: Method for identifying spots and entity link candidates.

To demonstrate the spotting process defined in Algorithm 1, let us consider our walkthrough example in a tokenized form: "Heart(w_1) attack(w_2) victims(w_3), sometimes(w_4) caused(w_5) by(w_6) MS(w_7) often(w_8) suffer(w_9) brain(w_{10}) damage(w_{11})" (line 1). RysannMD uses a sliding window of maximum size four (line 2). This value was chosen as a length that would give good spot coverage while avoiding the performance penalty of looping through large windows. We begin with the token at position one and a sliding window of size one, which results in the n-gram {heart} (line 5-7). We add this n-gram as the first n-gram to set T (line 8). Then, we search UMLS to find a single entry of "*heart*" (line 9-10), construct a mapping from spot-to-concept-identifier (heart→{C0018787}), and add this mapping to the set H_s (line 11). The window now expands to a size of two (line 12-14) and a new n-gram of {heart

attack} is generated and added to the n-gram set (line 7-8). Again, a UMLS search yields a single candidate of "*Myocardial Infarction*" (C0027051) to be added to the spot and entity candidates set (line 9-11). The 3-gram {Heart attack victims} and 4-gram {Heart attack victims sometimes} yield no UMLS matches but are still added to the n-gram set T. Since we have now reached our maximum n-gram size of four, we reset our windows size back to one and slide the window over by one position (line 15-17) and repeat the process until the entire input text is searched.

At the conclusion of the spotting stage, we obtain a set of n-grams (T) and a set of recognized spots mapped to a set of concept candidates (H_s). Table 3 shows the output of Algorithm 1 when applied to our example sentence; it provides the input for the disambiguation phase of the pipeline.

Table 3: Output from Algorithm 1 for the input text: "*Heart attack victims, sometimes caused by MS, often suffer brain damage*".

Set of spot-to-UMLS-identifier mappings (set H_s)	**Set of n-grams (set T)**
"heart"→ {C0018787} "heart attack" → {C0027051}	*"heart"*, "*heart attack"*, "*heart attack victims", "heart attack victims sometimes"*
"attack"→{C1261512, C1304680}	*"attack", "attack victims", "attack victims sometimes", "attack victims sometimes caused"*
"victims"→ {C0680681}	*"victims", "victims sometimes", "victims sometimes caused", "victims sometimes caused by"*
"sometimes"→{C1998882}	*"sometimes", "sometime caused", "sometime caused by", "sometime caused by MS"*
"caused by"→{C0015127}	*"caused", "caused by", "caused by MS", "caused by MS often"*
	"by", "by MS", "by MS often", "by MS often suffer"
"MS"→{C0025867, C0026221, C0026269 ,C0026514, C0026769, C0037813, C0039676, C0183396, C0439223, C1417325, C1417453, C1513009, C1868685, C1881819, C3539704}	*"MS", "MS often", "MS often suffer", "MS often suffer brain"*
"often""→{C0332183}	*"often", "often suffer", "often suffer brain", "often suffer brain damage"*
"suffer"→ {C0683278}	*"suffer", "suffer brain", "suffer brain damage"*
"brain"→{C0006104} "brain damage"→{C0270611}	*"brain", "brain damage"*
"damage"→{C0010957}	*"damage"*

5.2 Disambiguation

The most difficult task faced by an annotator is determining the correct meaning of a spot. Our working example contains a very ambiguous term "MS" that the spotter mapped to 15 possible UMLS concepts ranging from "*Master of Science*" to "*Multiple Sclerosis*" (Table 3). Adding to this difficulty is the consideration that none of the 15 candidates may be right (the NIL case). To tackle this, RysannMD first constructs a concept relationship tree for each of the 15 candidates by following the relationships listed in the MRREL flat file (Table 1). Figure 2 gives such a relationship tree generated for the concept candidate "*Multiple Sclerosis*".

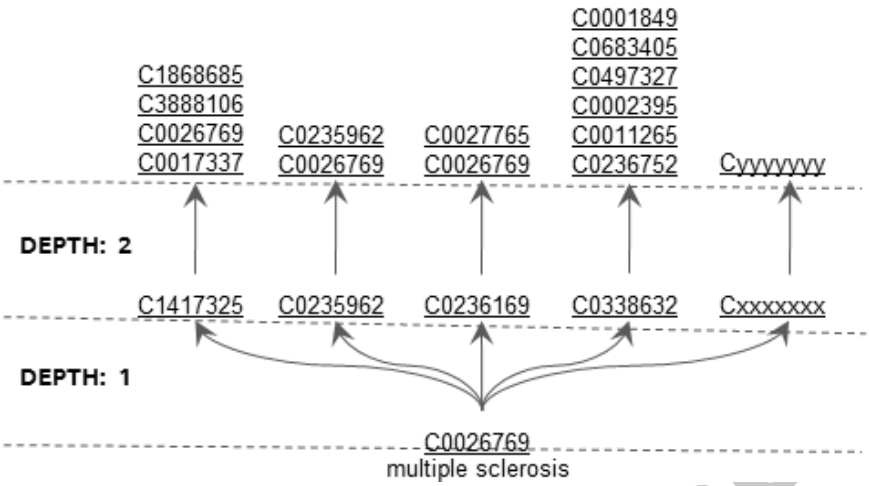


Figure 2: (Partial) Concept relationship tree for *Multiple Sclerosis* (C0026769)

Directly related to multiple sclerosis are *MS gene* (C1417325), *Lupoid Sclerosis* (C0281901), *nervous system disorder* (C0027765) and 20 other concepts. These concepts are considered to be at the first level or at depth 1 of the candidate *multiple sclerosis* relationship tree. This process repeats by identifying directly related concepts to the depth 1 concepts. For example, *MS gene* is related to four concepts: C1868685, C0026769, C0017337, C3888106 that become second level or depth 2 concepts of *multiple sclerosis*. Titles of the first and the second depth concepts are collected to generate n-grams that form a lexicon for *multiple sclerosis* (Table 4).

We limit the depth of a concept relationship tree to two levels for the following reasons: (1) At one level of relatedness we found the lexicons to be overly specific, which was detrimental towards accuracy; the accuracy improved noticeably with a depth of two levels. (2) Depth of three and higher negatively affected accuracy substantially as the lexicons became overly general to the concepts being represented. Note that the concept relationship trees are independent of any input text and thus are constructed and stored *offline*. Consequently, with respect to speed, the depth of the tree adversely affects the one-time construction of these trees and not the runtime of RysannMD (see Section 6 for implementation details).

Table 4: (Partial) Lexicon of n-grams for *multiple sclerosis* (C0026769) derived from the concept relationship tree (Figure 2).

CUI	**Generated n-grams**
C0026769	*1-gram:* [multiple], [sclerosis], *2-gram:* [multiple sclerosis]
C1417325	*1-gram:* [MS], [gene], *2-gram:* [MS gene]
C0281901	*1-gram:* [Lupoid], [Sclerosis], *2-gram:* [Lupoid Sclerosis]
C0027765	*1-gram:* [nervous], [system], [disorder], *2-gram:* [nervous system], [system disorder], *3-gram:* [nervous system disorder]
C1868685 C3888106	*1-gram:* [MULTIPLE], [SCLEROSIS], [SUSCEPTIBILITY], [TO], [1] *2-gram:* [MULTIPLE SCLEROSIS], [SCLEROSIS SUSCEPTIBILITY], [SUSCEPTIBILITY TO], [TO 1] *3-gram:* [MULTIPLE SCLEROSIS SUSCEPTIBILITY], [SCLEROSIS SUSCEPTIBILITY TO], [SUSCEPTIBILITY TO 1] *4-gram:* [MULTIPLE SCLEROSIS SUSCEPTIBILITY TO], [SCLEROSIS SUSCEPTIBILITY TO 1]
C0026769	*1-gram:* [Multiple], [Sclerosis] *2-gram:* [Multiple Sclerosis]
C0017337	*1-gram:* [Genes]

With the concept lexicons available, when RysannMD is faced with a disambiguation challenge for the spot "MS", it looks up the lexicon for *multiple sclerosis* (Table 4) and counts the matching n-grams from this lexicon to the n-grams generated from the input text (set T of Table 3). This count is also done for the other 14 candidate concepts and the candidate with the highest number of matched n-grams is selected to disambiguate the ambiguous "MS" spot.

Broadly speaking, this method is similar to the Lesk algorithm (Lesk, 1986) in which the words surrounding an ambiguous term in the input text and the words in a dictionary definition of the same term, are sampled. The chosen sense of the term is the one which has the most matching words between these two sets. However, we do not simply count the number of matching n-grams, but instead *weigh* each n-gram giving some matches more importance than others. Formally, suppose T is the set of n-grams from the input text, H_s the set of possible concepts for spot s (Table 3), and L_c is the lexicon for concept c (Table 4) . Furthermore, let $\omega_{Lc}(t)$ be a positive real-valued function for computing the weight of n-gram t using lexicon L_c or zero weight if t is not in L_c. Specifically:

$$\omega_{L_c}(t) = \frac{1}{|Q|_t}\left[\frac{|G_{1,L_c}(t)|}{1} + \frac{|G_{2,L_c}(t)|}{2}\right] if\ t \in L_c, 0\ otherwise \quad (2)$$

We define $|G_{1,Lc}(t)|$ and $|G_{2,Lc}(t)|$ as the count of occurrences of n-gram t within lexicon L_c at depth 1 and depth 2 of the relationship tree for the concept *c* (Figure 2). We define $|Q|_t$ as the count of t within all lexicons thus making $\frac{1}{|Q|_t}$ a normalizing constant for the equation. We weigh the occurrence of n-gram t at depth 1 twice more than its occurrence at depth 2 to convey that the related concepts at depth 1 might be more closely associated to concept c than those at depth 2. RysannMD can now calculate the weight of an n-gram as it relates to a particular concept; for example, the different weights of the uni-gram "*heart*" for each of the 15 candidate concepts of the spot "*MS*". Next, RysannMD inputs these weights into an *area-under-the-bell-curve* calculation, to determine how significantly each weight is above or below the average. Formally:

Let H_s be the set of possible concepts for spot s and let $W_t(H_s)$ be the set of weights (Equation 2) for n-gram t for all concept lexicons from set H_s, e.g., $W_{heart}(H_{MS})=\{\omega(heart)_{C0026769}, ..., \omega(heart)_{CC1513009}\}$. Let $\bar{X}(W)$ and $\sigma(W)$ be the mean and standard deviation for $W_t(H_s)$. Assuming a standard normal distribution, we define $AUBC(x, \bar{x}, \sigma)$ as a function of a bell-curve with mean $\bar{x}$ and standard deviation σ, having as its output the area-under-the-bell-curve for some value x. Then, the distribution weight for n-gram t of set T as it relates to concept c and triggered by spot s is:

$$\kappa_{c,s}(t) = AUBC(\ \omega_{L_c}(t), \bar{X}\big(W_t(H_s)\big), \sigma(W_t(H_s)))\ c \epsilon H_s, t \epsilon T \quad (3)$$

From Equation 3, we calculate a *support score* for a concept from the n-grams (T) within the input text.

Definition 1 (support score): Let V_0 and V_t be real-valued scaling constants where V_0 is static and V_t is a scalar based on the size of some n-gram *t*. Let T be the set of n-grams generated from the input text using a sliding window (Table 3). Then, the support score for concept c triggered by spot s is the sum of the distribution weights of T scaled by the constants V_0 and V_t.

$$Z(T, c, s) = \sum_{t \epsilon T} \kappa_{c,s} \times V_0 \times V_t \qquad (4)$$

The scalars V_0 and V_t are determined during the training phase (Section 5.5) and are included to provide some model flexibility during training. This is used in a hypergeometric distribution, defined as the *uncertainty score*, which brings this explanation full circle back to the lottery metaphor of Section 4.

Definition 2 (uncertainty score): Let $HG(e,p_1,p_2,p_3)$ be a hypergeometric distribution with parameters p_1 as the population size, p_2 as the number of successes within the population, p_3 as the number of samples to draw at random without replacement, and *e* as the number of successes desired from the drawn sample p_3. If H_s is the set of candidate concepts for spot s, T is the set of n-grams from the input text (Algorithm 1), and c is a concept candidate from H_s, then we define the uncertainty score U(T,c,s) for a candidate concept c as the likelihood that c is *not* the correct concept for spot s as:

$$U(T,c,s) = HG\left(0, V_p \times \sum_{h \epsilon H_s} Z(T,h,s), Z(T,c,s), Z(T,c,s)\right) \quad (5)$$

We can rationalize Equation 5 in terms of the lottery metaphor of Section 4. Parameter p_1 is the population size, which corresponds to filling the container with lottery balls associated with all the candidate concepts (H_s). The number of balls assigned to each label, i.e., concept, is determined by the concept's support score (Equation 4). This total population is multiplied by a scalar constant V_p whose exact value comes from model training (Section 5.5). This multiplier increases the size of the container so that there is room for additional lottery balls that are not part of the H_s set. This extra space accounts for the NIL case. Parameter p_2 determines the number of balls, say *j*, within the container labelled as target concept c. Parameters *e* and p_3 ask if we randomly draw *j* balls from the container, what is the probability that none of these drawn balls are labelled with concept c (i.e. *e=0*)? Intuitively, a large number of balls labelled as target concept c in the container reduces the chances of drawing no balls of that label. A small container size with no extra space for the NIL case (V_p=1) also reduces the likelihood of zero concept c balls drawn. Conversely, a container with lots of extra space (V_p >1) will increase the likelihood of drawing no labelled balls of concept c given fixed values for p_2 and p_3.

Table 5 gives uncertainty scores for the 15 candidate concepts for the spot "MS" in our walkthrough example "*Heart attack victims, sometimes caused by MS, often suffer brain damage*". The calculation shows that the concept with the lowest uncertainty score, and consequently the most likely concept for this context, is "*multiple sclerosis*".

Table 5: Uncertainty scores for 15 concept candidates for the "MS" spot in the context of "*Heart attack victims, sometimes caused by MS, often suffer brain damage".*

UMLS Concept	**Uncertainty Score U(T,c,s)**
Multiple Sclerosis (C0026769)	0.0466
Mitral Valve Stenosis (C0026269)	0.5225
MULTIPLE SCLEROSIS, SUSCEPTIBILITY TO (C1868685)	0.7038
MS gene (C1417325)	0.7674
Master of Science (C1513009)	0.7885
MTR wt Allele (C3539704)	0.7885
Mississippi (geographic location) (C0026221)	0.9189
Metric System (C0025867)	0.9189
Millisecond (C0439223)	0.9189
MTR gene (C1417453)	0.9189
Microbiology Susceptibility Domain (C1881819)	0.9189
Mass Spectrometers (device) (C0183396)	0.9189
Mass Spectrometry (C0037813)	0.9189
Montserrat (C0026514)	0.9846
5-Methyltetrahydrofolate-Homocysteine (C0039676)	0.9846

Finally, we complete our adapted version of Lesk's Algorithm with our weighted alternative that states the best concept c for spot s given a set of generated n-grams T from the input text (Algorithm 1), and a set of candidate concepts H_s (Table 3), is the concept c that minimizes the uncertainty score:

$$f(s,T) = \arg\min_{c \epsilon H_s} U(T,c,s) \quad (6)$$

5.3 Filtering

In the spotting phase, we identify multiple UMLS concepts as possible semantically correct interpretations of the recognized spots. In the disambiguation phase, RysannMD chooses the correct entity by selecting the candidate concept with the lowest uncertainty score. In the filtering phrase, we apply four thresholds in an attempt to identify and disregard potentially weak candidates.

Minimum support (support$^-$) [0,∞): The support score (Definition 1) must exceed this value. Formally, concept c is valid for spot s if:

$$Z(T, c, s) > support^- \quad (7)$$

where T is the set of n-grams from the input text (Algorithm 1). Since this benchmark is unbounded, this value is susceptible to the size of the input text with larger verbose text producing higher scores than shorter terse submissions.

Minimum normalized support (norm$^-$) [0,1]: The support score is first normalized then tested. Let c* be some candidate concept that maximized the support score. Then, candidate concept c is valid if:

$$\frac{Z(T,c,s)}{Z(T,c^*,s)} > norm^- \quad (8)$$

This adaptation of the minimum support score handles the variability of the length of the input text by forcing a value between zero and one.

Maximum uncertainty (uncertain$^+$) [0,1]: The uncertainty score (Definition 2) gives confidence that a spot has been disambiguated correctly. Larger scores indicate a higher likelihood of a mistake. Formally, candidate concept c is a valid consideration for spot s given the n-grams from the input text T if:

$$U(T, c, s) < uncertain^+ \quad (9)$$

This metric can be judged independently of the other candidate concepts; thus allowing for a measure of statistical assurance.

Minimum strength (strength$^-$) [0,1]: The strength of a spot-to-concept assertion is the product of normalized support and uncertainty scores. Namely, let c* be the concept candidate that maximizes the support score Z for n-grams T from the input text. Then, candidate c is valid for spot s if:

$$\frac{Z(T,c,s)}{Z(T,c^*,s)}\left(1 - U(T, c, s)\right) > strength^- \quad (10)$$

This condition is best satisfied when the uncertainty score is low and/or the normalized support is high.

These thresholds (*support*$^-$, *norm*$^-$, *uncertain*$^+$, *strength*$^-$) are fixed real-valued tunable parameters. They are determined by supervised training described in Section 5.5.

5.4 Overlap Resolution

The method of using a sliding window for finding spots (Algorithm 1) returns n-grams that may overlap into neighboring spot candidates; this problem is illustrated in our example (Figure 3).

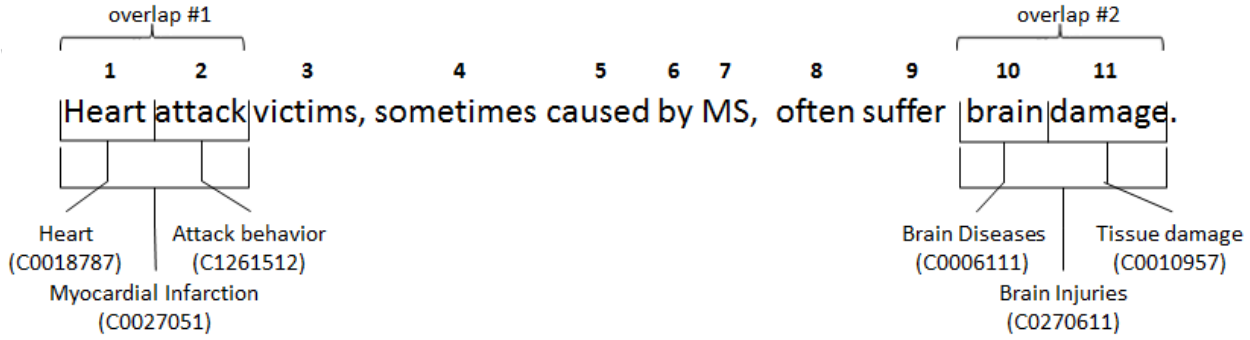


Figure 3: Overlapping spots of {*Heart, attack, Heart attack*} and {*brain, damage, brain damage*} as determined by RysannMD.

The spots of *Heart* at position 1 and *attack* at ordinal position 2 are index disjoint until the recognized bigram of *Heart attack* causes an overlap conflict between all three. A similar issue appears with spot *brain damage* at positions 10-11. Some annotators use a simple technique of choosing the longest n-gram as the solution (Divita et al., 2014). However, this method would be of little use in the example text "*genetic hypertension risk*" where *genetic hypertension* (C0598428) and *hypertension risk* (C0559060) are both overlapping bigrams.

RysannMD handles overlap situations by grouping overlapping n-grams together and selecting the concept candidate with the lowest uncertainty score within the group (Definition 2). Table 6 demonstrates how the concepts of *Myocardial Infarction* and *Brain Injuries* would be chosen over UMLS:{*Heart, Attack behavior*} and UMLS:{*Brain Diseases, Tissue damage*}, respectively.

Table 6: An example of overlap resolution using smallest uncertainty score.

Overlap #1		**Overlap #2**	
UMLS concept	**Uncertainty**	**UMLS concept**	**Uncertainty**
Myocardial Infarction	2.063E-8	Brain Injuries	2.063E-8
Heart	3.635E-7	Brain Diseases	0.00149
Attack behavior	0.05128	Tissue damage	0.01057

5.5 Training

RysannMD requires default parameter settings for disambiguation and filtering. Namely, default values need to be determined for parameters V_0, V_t, V_p (Section 5.2) and thresholds: support$^-$, norm$^-$, uncertain$^+$, strength$^-$ (Section 5.3). This requirement is not unique to RysannMD. Indeed, all semantic annotators offer adjustable parameters based on the characteristics of the input text (e.g. length, verbosity, and domain) to tune performance. An annotator must settle on some default parameter values that in general provide good precision versus recall tradeoffs.

RysannMD relies on a Genetic Algorithm (GA) and supervised training to find a near-optimal solution for its parameters. We use the UMLS flat file MRDEF, containing 215,429 concepts with descriptions, as the ground truth for the training.

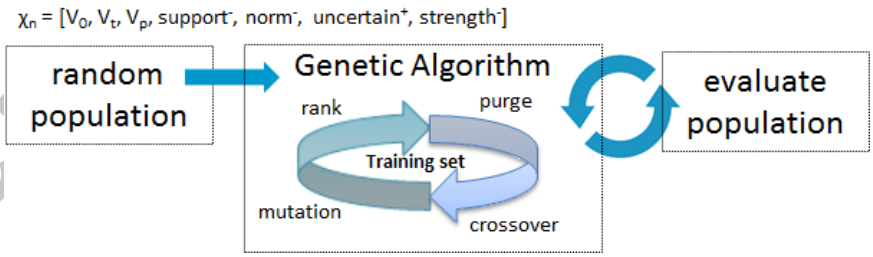


Figure 4: RysannMD training starting with a random population of solutions until Genetic algorithm completion.

Figure 4 outlines the process for a conventional GA starting with an initial random population of possible solutions. In particular, we begin with an arbitrary number of configurations in the form of 7-tuple vectors [V_0, V_t, V_p, *support*$^-$, *norm*$^-$, *uncertain*$^+$, *strength*$^-$] initialized at random; we refer to this set of configurations as set X. Next, we randomly select an arbitrary number of concepts from the MRDEF file as the gold standard. However, the MRDEF file does not contain explicitly defined spans of text with links to UMLS concepts, i.e., spots, which are necessary for supervised training. Instead, for each UMLS concept, it contains the concept's unique identifier (CUI) and a description. From these two fields we construct a gold standard that includes the necessary spots, spans, and CUIs as follows. First, we look-up the name of the concept (title) associated with a particular CUI via the MRCONSO UMLS file (Table 1). This will become the known spot. Second, we concatenate the spot with the

concept description separated by a space to create the input text for an annotator: [concept title] + [space] + [concept description]. We refer to this modification of the MRDEF file in our algorithm as set Y. This reformulation now creates an explicit span for the spot, i.e., concept title, starting at ordinal position 1 and ending at ordinal position equal to the length of the concept title. RysannMD can now annotate this input text and discover if the text from position with index {1} to position with index {*length of concept title*} was recognized as a spot and whether that span is linked to the known correct CUI from the MRDEF.

We configure RysannMD with a vector v from X, and then annotate the set Y. If the size of the set Y is y, then there are y-known correct concepts with known index positions (spans). If K_Y represents this known set, and if $A_Y(v)$ represents the set of all annotations produced by RysannMD including spans for set Y using configuration v, then a fitness function to rank the effectiveness of RysannMD is defined as:

$$FIT(v, Y) = |A_Y(v) \cap K_Y|^2 - |K_Y \setminus A_Y(v)|^2 \quad (11)$$

Equation 11 computes the fitness of v by counting the number of correctly identified known annotations less the number of those that were incorrectly disambiguated or missed (NIL). We take the difference of the square of these two terms to emphasize larger values over smaller values, and thus reward more correct answers and fewer mistakes. We must stress that GAs find local maxima solutions and not necessarily the global optimal answer. Furthermore, our fitness function is one of many where others may or may not achieve solutions closer to the global maximum. We refer the interested reader to our work on parameter tuning of semantic annotators for variations of this fitness function (Jcuzzola et al., 2015).

Each vector v from X is ranked using Equation 11 and those low scoring vectors from X are purged, i.e., we keep the top-n vectors. The survivors produce offspring through crossover with random variations (mutation). We select a new random set of gold standard concepts Y and repeat the cycle of rank, purge, crossover, and mutation for an arbitrary number of iterations. Upon the final iteration, the top scoring remaining vector v becomes the default configuration for RysannMD. Algorithm 2 formalizes this process.

Input: MRDEF UMLS file.
Output (v): Default configuration for RysannMD.
Method:

1. Initialize maximum number of configurations n. Default n = 10.
2. Initialize size of gold standard concepts y. Default y = 10.
3. Initialize number of iterations before terminating. Default loops = 10000.
4. Generate n-random configurations as set X.
5. FOR i = 1 TO loops
6. Generate set Y. Randomly select y concepts from MRDEF UMLS file.
7. FOR each $v$in X
8. Annotate set Y using configuration v. $i.e., A_Y(v)$.
9. Score v using Equation 11.
10. NEXT v
11. Keep top-n ranking configurations $v_1, v_2, \ldots, v_n$ in X.
12. IF i < loops THEN
13. Perform crossover. $v_1 \leftrightarrow v_2, v_2 \leftrightarrow v_3, \ldots, v_{n-1} \leftrightarrow v_n$. Add offspring to X.
14. Perform mutation. $\forall\, v \epsilon X$, Randomly perturb configuration of v.
15. NEXT i
16. Return best RysannMD configuration. ie: v_1 of X.

Algorithm 2: Genetic Algorithm for finding V_0, V_t, V_p, *support*$^-$, *norm*$^-$, *uncertain*$^+$, and *strength*$^-$ parameters.

Table 7 lists the default RysannMD parameter values discovered using the proposed GA. The scalar values of $V[1..4]_t$ are conditional multipliers based on whether a spot is a unigram, bigram, trigram, or a 4-gram (Equation 4).

Since unigrams, bigrams and trigrams are more frequent than 4-grams, an occurrence of a recognized 4-gram spot in the input text should be a better determiner of disambiguation. This rationale is reflected in the increasing weight values assigned by the GA. The V_0 scalar exists to help the GA traverse the solution space while training and can be absorbed (multiplied into) the $V[1..4]_t$ constants without further mention. A more interesting constant is V_p which provides the *extra space* in our lottery container metaphor and determines the strength of the NIL case (Equation 5). The computed default value indicates substantial extension of the container - 16 times the total value of the other candidate concepts; thus giving the NIL consideration a high probability.

In terms of the thresholds, we see that the GA gave a low minimum value of six for support$^-$ showing that even candidate concepts with a small number of *lottery balls* in the container will be considered. Similarly, the norm$^-$ and strength$^-$ parameters were assigned low thresholds thus preventing removal of concepts for all but the proportionally lowest candidates. This leniency continues with a high uncertain$^+$ maximum threshold allowing RysannMD to keep candidates with uncertainties as high as 75%. This reluctance to filter candidates is explained by the high V_p multiplier; the choice of these forgiving threshold values acts as counterbalance to the NIL likelihood. An online web demo of RysannMD is available which allows one to experiment with different threshold values (see Section 9 for the web address).

Table 7: The default tunable parameters for RysannMD as determined by a Genetic Algorithm.

PARAMETER	VALUE	TYPE
$support^{-}$	6	Minimum Threshold
$norm^{-}$	0.03879	Minimum Threshold
$uncertain^{+}$	0.7585	Maximum Threshold
$strength^{-}$	0.02907	Minimum Threshold
V_p	15.92	Constant Scalar
V_0	0.7516	Constant Scalar
$V1_t$	0.5919	Constant Scalar
$V2_t$	0.7492	Constant Scalar
$V3_t$	1.0431	Constant Scalar
$V4_t$	1.2295	Constant Scalar

6. RysannMD Implementation

In this section, we describe our current implementation of RysannMD and how our system design coincides with the spotting, disambiguation, filtering, and overlap resolution stages of the algorithm. The RysannMD platform resides on a 40 core Intel Xeon CPU E5-2690 v2 at 3.00GHz and 128GB of RAM. From text input to annotation output the processing goes through six phases.

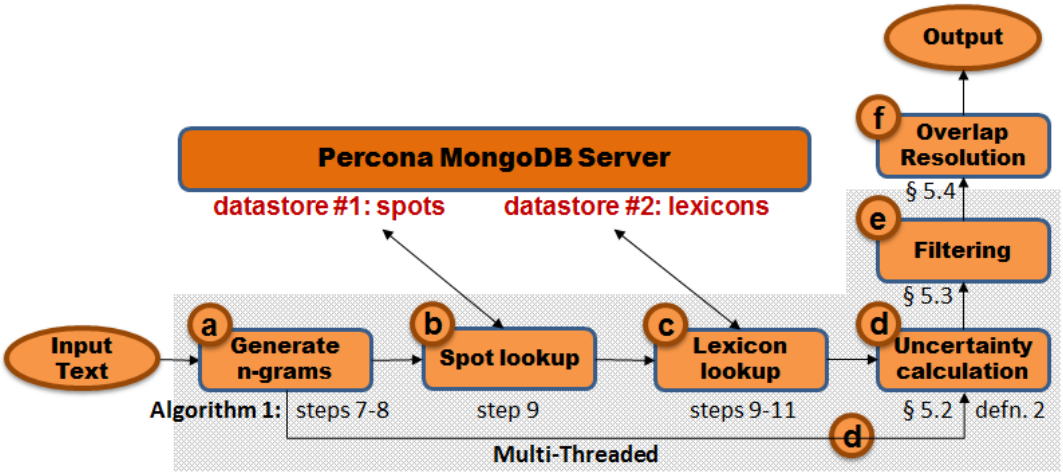


Figure 5: The six phases (A-F) of text processing for RysannMD

These phases are supported by a UMLS-based Knowledge Base containing two critical lookup tables: *spots* and *lexicons* stored on the Percona MongoDB NoSQL database management system[3]. The *spots* datastore contains a collection of UMLS concept labels and aliases grouped together by matching labels/aliases that become a primary key for a list of candidate concepts that the label/alias may refer to. This datastore is constructed by indexing the MRCONSO flat file of the UMLS. Table 8 shows a Percona NoSQL database object for the term "MS". When the unigram "MS" appears within the input text, a query to this datastore quickly returns the 15 possible candidates for this ambiguous term. The current spots datastore contains 4,479,375 of these objects.

[3] https://www.percona.com/software/mongo-database/percona-server-for-mongodb

Table 8: The NoSQL object for the term "MS" in the spots datastore.


```
{   "key": "MS",
    "CUIs" : [
       "C0025867", "C0026221", "C0026269", "C0026514", "C0026769",
       "C0037813", "C0039676", "C0183396", "C0439223", "C1417325",
       "C1417453", "C1513009", "C1868685", "C1881819", "C3539704"
            ] }
```


The *lexicon* datastore holds the lexicon for every concept c (L_c) of UMLS (see Table 4) and also stores the weight of every word t associated to concept c ($\omega_{L_c}(t)$ equation 2). This datastore is indexed by UMLS concept unique identifier (CUI) and currently contains 2,397,167 records. To illustrate, consider Table 9 that shows the weight of unigram "*heart*" and bigram "*heart attack*" for concepts Heart(C0018787) and Myocardial Infarction(C0027051). The lexicon for the Heart concept stores 9,799 weighted n-grams compared to 8,659 for Myocardial Infarction.

Table 9: Weight of n-grams "heart" and "heart attack" on concepts C0018787 and C0027051 as stored in the lexicon datastore.

Heart(C0018787) (primary key)	**Myocardial Infarction(C0027051) (primary key)**
$\omega_{L_{C0018787}}("heart") = 0.009698$ $\omega_{L_{C0018787}}("heart\ attack") = 0.009411$ *⇒ total of 9,799 n-grams in this lexicon.*	$\omega_{L_{C0027051}}("heart") = 0.005143$ $\omega_{L_{C0027051}}("heart\ attack") = 0.01050$ *⇒ total of 8,659 n-grams in this lexicon.*

Note that the construction of the spots and lexicon datastores is not dependent on the input text to be annotated. Consequently, this calculation is done offline. The indexing of these two datastores required approximately three days of computation on our aforementioned hardware.

RysannMD takes full advantage of the 40-core hardware; utilizing multithreading wherever it can to complete the six phases of processing efficiently. To illustrate, consider Figure 6 which illustrates the phases and parallel threads of computation for our walkthrough example.

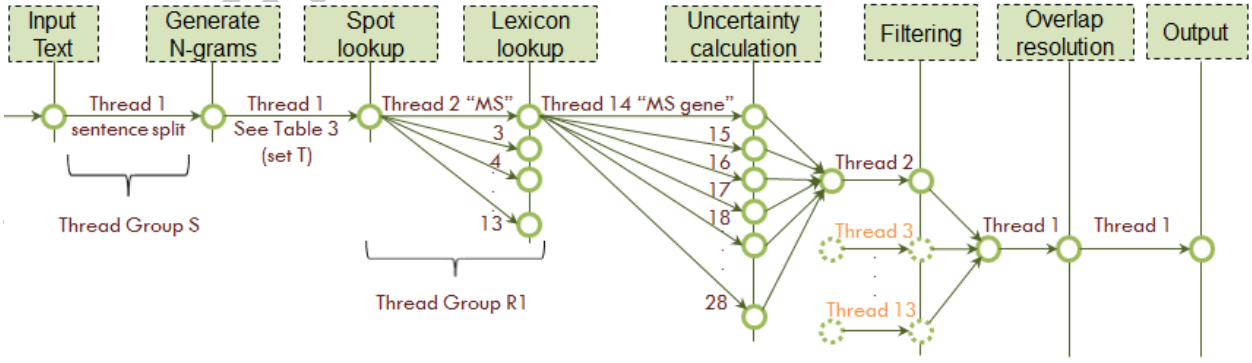


Figure 6: Thread creation for the six phases of computation for the input text "*Heart attack victims, sometimes caused by MS, often suffer brain damage*".

Upon receiving an input text, RysannMD uses sentence detection to split the text into individual sentences. Each sentence is processed concurrently in its own thread of execution. These sentence threads are collected into a thread

group called Thread Group S (TGS). Thread 1 is created and given the sentence "*Heart attack victims, sometimes caused by MS, often suffer brain damage*". Thread 1 is the only member of TGS since our walkthrough example is only a single sentence.

Within Thread 1 we create 46 n-grams: "*heart", "heart attack", "heart attack victims", ..., "brain damage", "damage"* and so forth using Algorithm 1 (Table 3). Using these n-grams as the primary keys, the *spots* datastore is queried for an exact match in which spots "*heart*", "*heart attack*", "*MS*" and 9 others are found and their corresponding concept candidate list is discovered (Table 8). A separate and parallel-running thread is created and assigned to each of these recognized spots (Thread 2-13); we group these threads into a recognized spots group called Thread Group R for Sentence Thread 1 (TGR1). If there were more than one sentence in the input text, we would generate TGR2, TGR3 and so forth in the same manner.

Suppose Thread 2 is assigned the spot "MS". From the previous query to the *spots* datastore (Table 8), we learn the term "MS" is associated with 15 possible concepts: *Multiple Sclerosis* (C0026769), *MS gene* (C1417325), *Montserrat* (C0026514), etc. RysannMD now spawns another 15 simultaneous threads, one for each of the possible concepts for the "MS" spot (Thread 14-28). These threads now concurrently calculate the uncertainty score (Equation 5) for each "MS" candidate concept separately by matching each of the 46 n-grams generated (from the input sentence) in Thread 1 (Table 3) with the weights stored in the lexicon datastore for that candidate. Once Threads 14 to 28 complete their calculation, and rejoin Thread 2, the concept with the lowest uncertainty (Equation 6) moves on to the filtering phase (Section 5.3).

Overlap resolution (Section 5.4) must wait for all other recognized spots threads (those in TGR1, re:Thread 2-13) to finish since this stage requires the uncertainty scores for all spots whose spans may intersect each other. Finally, RysannMD pauses until all sentence threads (those in TGS; re: Thread 1) complete before concatenating the results into the desired output of annotated text.

7. Experimentation

Our experimentation involves comparing RysannMD against four modern biomedical annotation systems using four large corpora for benchmarking. In Section 7.1, we describe customizations made to the default installation or configuration of the competing annotators. In Section 7.2, we give an overview of the corpora used for the performance evaluation and describe the difference between silver and gold standard corpora. In Section 7.3, we examine RysannMD relative to the other four annotators with respect to both effectiveness as measured by precision, recall, and F_1 measure and speed. The comparison is done on different benchmarking corpora – different in terms of both the used biomedical vocabularies and the construction approach (silver versus gold). All tests were performed on the same 40 core Intel Xeon CPU E5-2690 v2 @ 3.00GHz with 128GB of RAM.

7.1 Benchmarking Annotators

To benchmark RysannMD for both annotation effectiveness and speed, we make use of four modern biomedical semantic annotators, introduced in Section 2, as baselines for comparison. For an accurate comparison of performance, based upon both accuracy and speed, each annotator must have the same opportunity for achieving high F_1 as quickly as it is capable. The challenge with balancing speed and accuracy is that giving priority to one dimension is often to the detriment of the other. For example, consider Algorithm 3 as a specific implementation of a semantic annotator named *TrivialAnnotator.*

Input (t): Some natural language text (t) for annotating.

Output (Θ): The set of annotations Θ for text t.
Method:

1. Return empty set $\Theta = \{\}$.

Algorithm 3: The pseudocode for *TrivialAnnotator*: A very fast but extremely poor semantic annotator

Algorithm 3 illustrates a rhetorical implementation of a semantic annotator that always returns an empty set of annotations for any input given. This annotator will outperform any of its competitors for speed but will always rank worst with respect to F_1. If we weigh speed and accuracy equally, then TrivialAnnotator would be evaluated incorrectly as an average performing (median) annotator. Therefore, we must first emphasize accuracy, then decide how much of it we are willing to sacrifice for speed. The challenge is how to fairly adjust the default configuration parameters of the competing annotators so that comparative tests are fair. We accomplish this by first configuring the annotators so that each has access to relatively the same number of UMLS concepts. The datastore for RysannMD and MetaMap contains 2,397,167 unique concepts and 2,902,947 aliases. NOBLE Coder was configured to use its largest available UMLS lexicon with 2,335,627 concepts. The out-of-the-box configuration for cTAKES only accounted for 300,099 concepts while Neji's default covered a miniscule 3% of what RysannMD and MetaMap must consider. Consequently, we configured each annotator with a relatively equal number of concepts in its knowledge base thus (1) allowing for an equal opportunity F_1 score and (2) offering the same speed challenge of contending with millions of concepts. Once these concept counts were equitable, then we optimized for speed. Each annotator was installed on the same 40-core server. We deviated from their default configuration as follows:

MetaMap. We installed MetaMap using the default configuration with the exception of enabling the "*use word sense disambiguation*" option (-y). This change was made to improve F_1.

Neji. The default installation of Neji includes a restrictive set of UMLS concepts limited to UMLS types T023 and T047, totaling in only 81,785 concepts. However, Neji offers instructions for building custom dictionaries[4], which we followed boosting our installation of Neji to 2,652,560 UMLS concepts. This change was made to improve F_1. To improve speed, we configured Neji to use the maximum allowed number of threads (32) to take full advantage of the 40-core hardware.

NOBLE Coder. Default configuration settings were used along with the NCI Metathesaurus prebuilt terminology with 2,335,627 concepts. This terminology was selected to achieve best F_1.

cTAKES+ytex. By default in this tool, UMLS annotation is accomplished through the *DictionaryLookup* pipeline. However, we replaced this pipeline with the YTEX UIMA module which improves annotation results by incorporating semantic similarity and word sense disambiguation capabilities into cTAKES[5]. The default installation of YTEX limits the UMLS concepts to SNOMED and RXNORM vocabularies exposing only 300,099 recognized concepts of specific types[6]. However, this vocabulary/type filter can be switched off which upon disabling gave YTEX access to 5,286,317 UMLS concepts. This change was made to improve F_1.

7.2 Silver and Gold Corpora

Traditionally, evaluation corpora are constructed by human domain experts to be used as the *gold* standard for benchmarking. Although this approach often results in a high quality evaluation corpus, the size of the generated corpus is typically too small to be useful for training machine learning models. Alternatively, a *silver* standard

[4] https://github.com/bioinformatics-ua/neji/wiki/request:-more-dictionaries
[5] https://cwiki.apache.org/confluence/display/CTAKES/YTEX+Installation
[6] https://code.google.com/archive/p/ytex/wikis/DictionaryLookup_V07.wiki

corpus can be created by combining annotations produced by two or more automated annotators over a large textual corpus. To mitigate the possibility of machine-introduced errors, a small sample of automatically produced annotations may be examined by human experts to assess the quality in a semi-supervised approach. Another method of error correction involves multiple computer oracles coming to a consensus in their produced answers. Our experiments on RysannMD and the four baseline annotators included both gold and silver standard evaluation corpora.

CALBC (*silver*). The "*Collaborative Annotation of a Large Biomedical Corpus*" is a community joint research project, initiated by the *European Bioinformatics Institute*, whose goal is to create a large annotated corpus from 118,438 Medline abstracts. However, the difficulty in acquiring domain experts and the enormity of the volume of abstracts made manual annotation impractical. To overcome this impasse, four automated annotators[7] jointly annotated each abstract. Their results were combined through a voting mechanism, and those annotations of majority agreement ultimately became part of the CALBC corpus.

SemRep (*silver and gold)*. *Semantic Knowledge Representation* is a program designed by the *U.S. National Institutes of Health* that extracts semantic relationships from biomedical texts in the form of n-triples <subject, predicate, object>[8]. For example, the sentence "*We used hemofiltration to treat a patient*" would produce the triple <Hemofiltration, TREATS, Patients> along with various metadata which include the UMLS Concept Unique Identifier (CUI) for both the subject and the object of a triple, as well as their ordinal index positions within the original text. This allows for the creation of a silver standard evaluation set by ignoring the predicate component of the triple. We downloaded the latest version of the SemRep dataset (version 1.7) which contained 85,777,203 n-triples generated from 158,910,856 text sentences, and used it as our silver standard. Also included in our evaluations is the SemRep gold standard dataset which is a human annotated set of only 500 sentences whose main function was to judge the accuracy of the SemRep software.

CRAFT (*gold*). *The Colorado Richly Annotated Full Text Corpus*. This dataset is a manually annotated corpus of 67 published medical articles containing approximately 100,000 annotations in seven biomedical terminologies. Of these seven, we focus on three vocabularies: *Chemical Entities of Biological Interest* (CHEBI), *National Center for Biotechnology Information* (NCBI), and *Gene Ontology* (GO). We further sub-categorize the GO ontology into two taxonomies: *Cellular Components* (GO_CC) and *Biological Processes and Molecular Functions* (GO_BPMF). CHEBI, NCBI, and GO were chosen because the UMLS Metathesaurus contains explicit identifier mappings from these vocabularies to the UMLS concept identifiers (CUIs) via the flat files MRSAT and MRCONSO (Table 1).

MeSH WSD Test Collection *(silver). The MeSH Word Sense Disambiguation Test Collection*[9], from the *U.S. National Library of Medicine*, was created specifically to benchmark the disambiguation capability of biomedical annotators. It centers around 203 ambiguous terms (e.g. "ALS") that can only be resolved to a UMLS concept by context (e.g. C0003372 / C0002736 correspond to "ALS"). These 203 ambiguous terms correspond to 409 unique UMLS concepts reoccurring uniformly within 37,388 paragraphs retrieved from MEDLINE citations. The challenge for an automated annotator is to recognize (spot) each of these 203 terms throughout the 37,388 paragraphs and link it to one of the 409 UMLS concepts.

7.3 Results

[7] EBI (Rebholz-Schuhmann et al.; 2007), EMC (Schuemie et al., 2007), FSU Jena (JCoRe; Hahn et al., 2008), and LGM LINGUAMATICS.

[8] https://semrep.nlm.nih.gov/

[9] https://wsd.nlm.nih.gov/

We compared RysannMD with the four benchmarking annotators (Section 7.1) with respect to the precision, recall and F_1 scores, as well as the speed, using a variety of silver and gold corpora (Section 7.2). All annotators were run in non-overlapping mode to test their performance with overlap conflict resolution (Section 5.4). All reported F_1 scores are within +/- 0.01 interval with a confidence of 95%.

The CALBC Silver Corpus. We annotated this machine generated corpus of medical abstracts with each of the five examined annotators, and report on the average F_1 and speed per 1,000 abstracts. The results are shown in Figure 7.

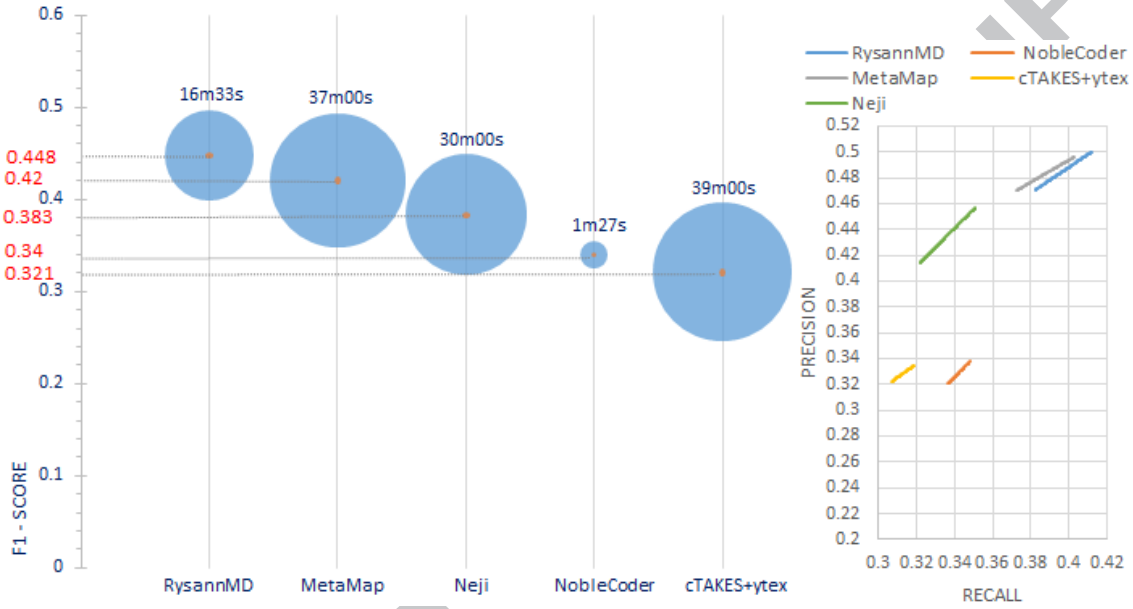


Figure 7: ***left***: F_1 measure using the silver CALBC corpus including average processing time (mm:ss) per 1,000 abstracts. ***right***: Precision versus Recall plot; the plot shows the range of precision / recall values obtained on the examined sample of 1,000 abstracts.

RysannMD scored the highest F_1 of 0.448 closely followed by MetaMap with F_1=0.420. cTAKES and NOBLE Coder had the lowest F_1 values on the CALBC corpus (0.321 and 0.340, respectively), while Neji positioned itself in the middle with F_1=0.383. With respect to speed, NOBLE Coder significantly outperformed others taking only 1.5 minutes to annotate 1000 abstracts, but struggled with the fourth place when it comes to the examined performance measure, i.e., F_1. RysannMD took the 2nd place in terms of speed while achieving the best F_1. MetaMap, although performing close to RysannMD in terms of F_1, scored 4th at 37 minutes besting only cTAKES at 39 minutes. Neji positioned itself in the middle with 30 minutes processing time. Recall versus precision results indicate NOBLE Coder and cTAKES perform equally with respect to precision (Figure 7, *right*); however, cTAKES is surpassed by NOBLE Coder in recall. Neji achieved precision in the range of 0.42 to 0.45, and a recall in the range of 0.32 to 0.35. MetaMap and RysannMD tracked similar precision and recall from 0.47 to 0.51 and 0.37 to 0.41, respectively.

The SemRep Silver and Gold Corpus. We repeated our speed and performance tests with the SemRep silver corpus. Results in Figure 8 show F_1 scores, computational time, and precision/recall per 1,000 sentences from the SemRep silver corpus.

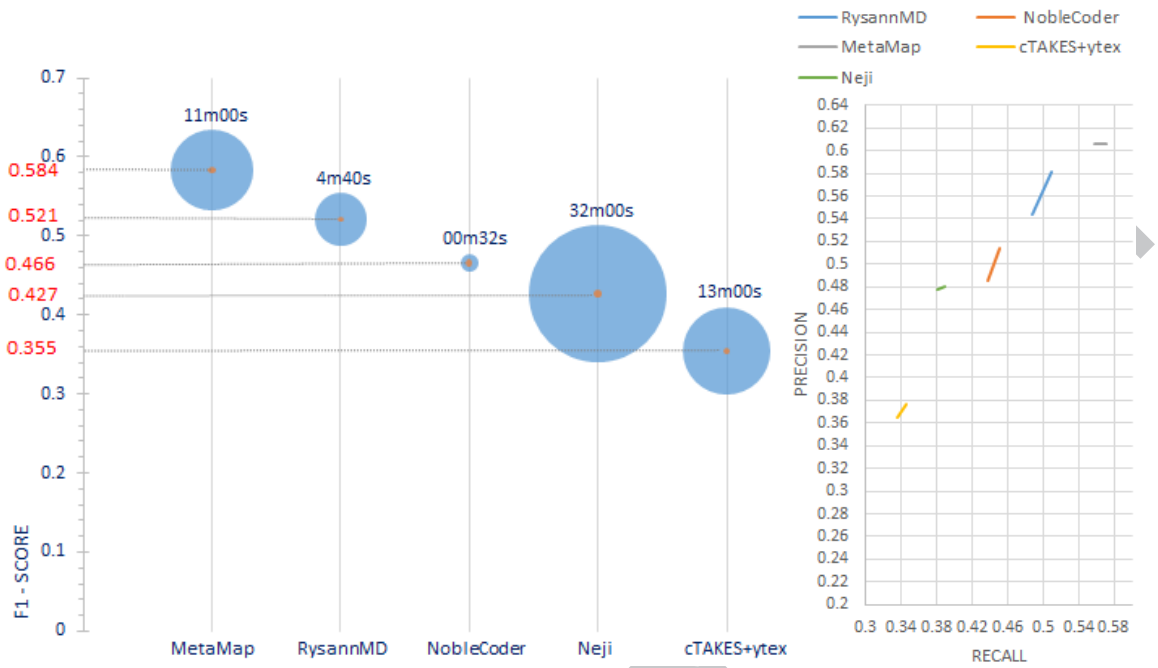


Figure 8: ***left***: F_1 measure using the silver SemRep corpus including average processing time (mm:ss) per 1,000 sentences. ***right***: Precision versus Recall plot.

MetaMap scored the highest F_1 score of 0.584, but did so with a speed of 11 minutes per 1,000 SemRep silver corpus sentences. Furthermore, this top F_1-score is somewhat misleading as the SemRep silver corpus was constructed using the *word sense disambiguation (WSD) engine* of MetaMap itself[10]. Considering that in the SemRep silver corpus the ground truths were generated by computer rather than human experts, the use of MetaMap's WSD engine for the corpus construction gave MetaMap a distinct advantage. RysannMD came second in F_1 (0.521) and speed, taking just under five minutes of processing time. Nobel Coder bested all annotators in speed, requiring only 30 seconds to achieve the 3rd place ranking in F_1 (0.466). Neji took 4th ranking in F_1 and 5th in speed, while cTAKES performed 5th in F_1 and 4th in speed.

Next, we tested the annotators' effectiveness (precision, recall, and F_1) on the gold standard SemRep corpus; the results are given in Table 10. We forego the speed evaluation due to the small size of this corpus of only 500 sentences.

Table 10: Precision, Recall, and F_1 for the gold standard SemRep corpus.

	P	R	F_1
NOBLE Coder	0.528	0.406	0.459
Neji	0.504	0.396	0.443
RysannMD	0.493	0.399	0.441
MetaMap	0.451	0.292	0.354
cTAKES + ytex	0.358	0.313	0.334

NOBLE Coder, Neji, and RysannMD ranked 1st, 2nd, and 3rd in F_1, respectively, scoring with a difference from 1st to 3rd of only 0.018. Because this difference is seemingly negligible and the corpus size small, we surmised that the differences in the F_1 scores of the three first ranked annotators were statistically insignificant. To test this hypothesis, we used the paired two-tailed t-test to compare F1 scores of each pair of annotators. Our null hypothesis

[10] https://semrep.nlm.nih.gov/SemRep.v1.7_Installation.html

was that the F_1 scores of any two annotators were not significantly different. For a 95% confidence, we reject the null hypothesis, if the p-value is less than 0.025. Table 11 gives the p-value for any pair of annotators.

Table 11: Paired two-tail t-test for any two annotators. * indicates difference in F_1 is statistically significant (i.e. p-value<0.025).

	MetaMap	NOBLE Coder	Neji	cTAKES + ytex
RysannMD	*1E-9	**0.415**	**0.663**	*0.003
MetaMap		*1.7E-12	*2.2E-8	*0.006
NOBLE Coder			**0.954**	*0.002
Neji				*0.0004

RysannMD paired with NOBLE Coder gave a p-value of 0.415, indicating that we cannot reject the null hypothesis and conclude that the F_1 scores of RysannMD and NOBLE Coder are statistically equivalent. We make the same conclusion for RysannMD paired with Neji (p=0.663), and Neji paired with NOBLE Coder (p=0.954).

The CRAFT Gold Corpus. The CRAFT corpus comprises annotations from a long body of text sourced from a relatively small number of full medical publications; a total of 67. These annotations are linked to specific vocabularies: CHEBI, NCBI, and GO; where GO is further sub-divided into two main taxonomies: GO_CC and GO_BPMF. This separation produces four distinct gold standard datasets for testing. All 67 publications were processed by each of the examined annotators with the combined results presented in Figure 9 and with results separated by vocabulary in Figure 10.

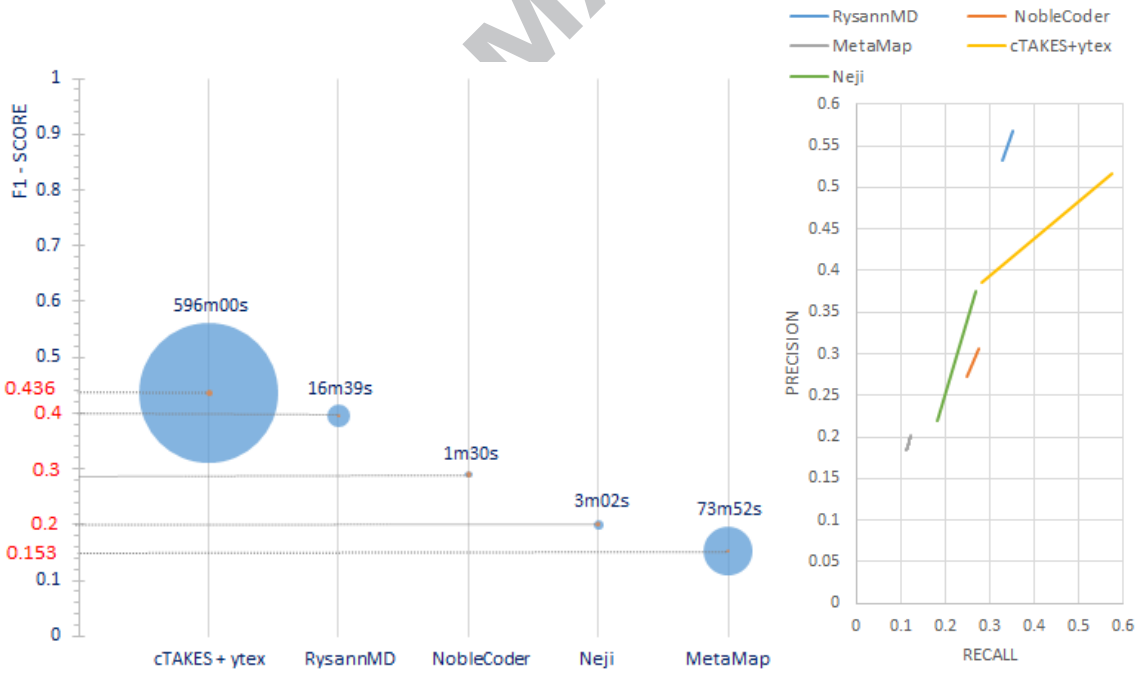


Figure 9: ***left***: F_1 measure using the gold CRAFT corpus including processing time (mm:ss) for 67 medical publications. ***right***: Precision versus Recall plot.

cTAKES performed best with F_1=0.436 followed closely by RysannMD with F_1=0.4. However, cTAKES required nearly 6 hours to process the CRAFT corpus compared to RysannMD's time of 17 minutes. The precision versus recall plot shows RysannMD obtained most of its F_1 from the highest precision while cTAKES benefited from the

top-most recall. When we separate F_1 by vocabulary (Figure 10), we find RysannMD ranks first with CHEBI (F_1=0.7) and GO_BPMF (F_1=0.244); while cTAKES tops with NCBI (F_1=0.859) and GO_CC (F_1=0.541).

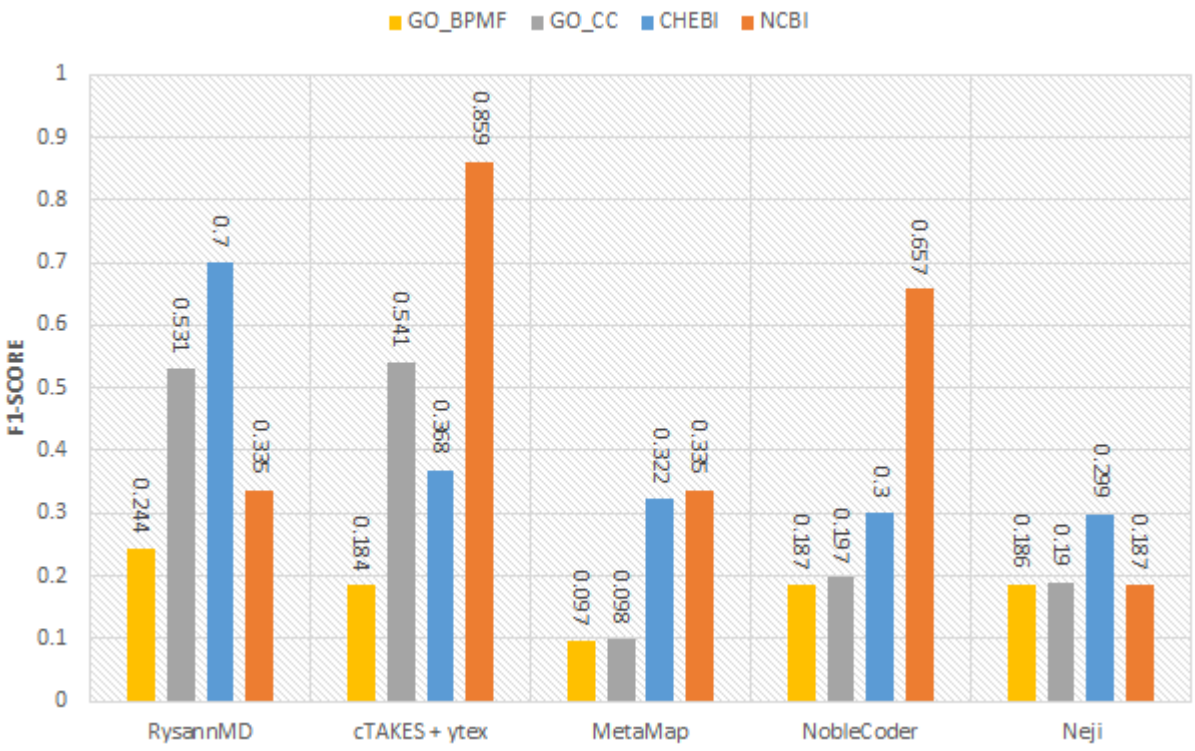


Figure 10: Individual F_1 for the vocabularies GO (BMF and CC), CHEBI, and NCBI of the CRAFT gold dataset.

The MeSH Word Sense Disambiguation (WSD) Silver Corpus. Last, we compare the annotators against the MeSH WSD Collection with results reported in Figure 11. RysannMD performed best in F_1 with 0.533; outscoring second placed NOBLE Coder by a significant 0.153 margin. RysannMD top rank was attributed to comparatively high precision and recall, respectively. The second best achieving F_1 by NOBLE Coder was hindered by a recall that was on average below that of RysannMD (0.3 to 0.5 versus 0.5 to 0.6) and by a precision whose range of values (0.4 to 0.6) varied more than that of RysannMD (0.53 to 0.6). NOBLE Coder excelled in speed taking on average just under two minutes to annotate 1,000 paragraphs while RysannMD required 15 minutes for the same set. MetaMap and Neji needed 75 and 87 minutes with cTAKES positioned as median at 40 minutes.

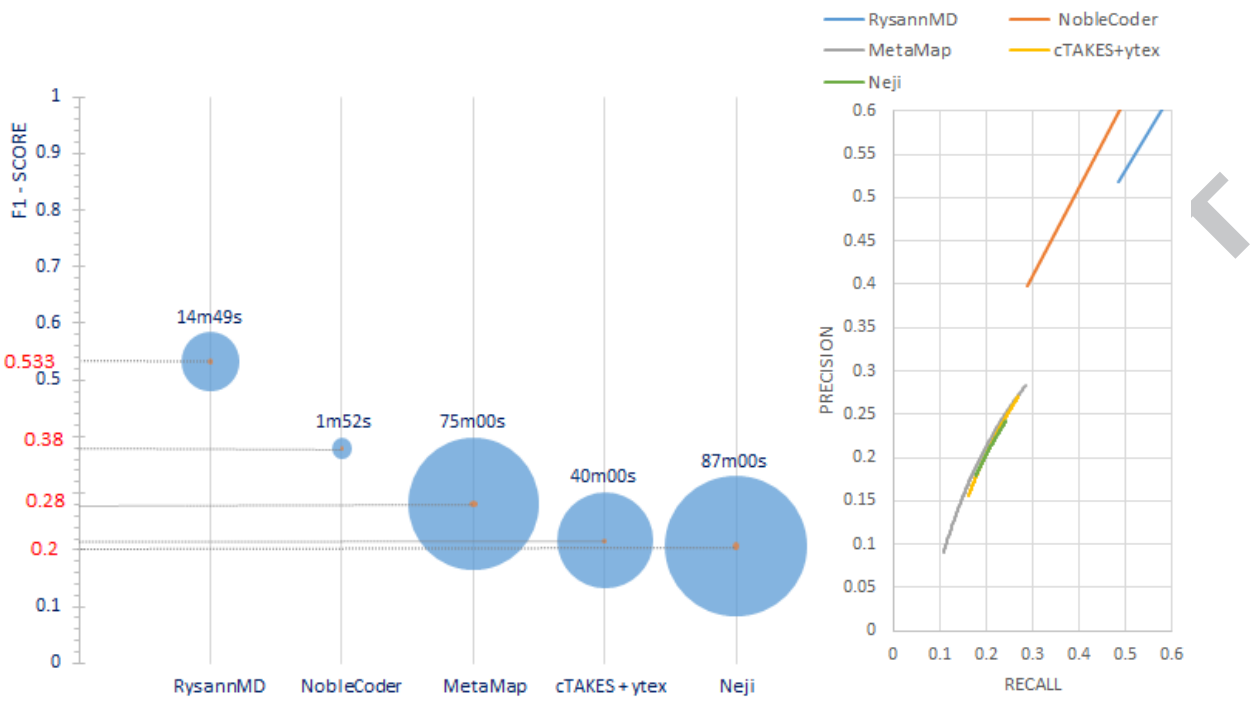


Figure 11: ***left***: F_1 measure using the silver MeSH WSD Collection corpus including average processing time (mm:ss) per 1,000 paragraphs. ***right***: Precision versus Recall plot.

8. Discussion and Summary

In this section, we discuss and summarize the results of Section 7. Specifically, we interpret our findings of RysannMD's effectiveness, errors and speed with consideration for multiple corpora, different taxonomies, and silver versus gold benchmarking corpora.

Effectiveness. Each of the five examined annotators performed best in terms of F_1 scoring for at least one corpus. Neji and NOBLE Coder ranked first with the SemRep gold standard corpus, while MetaMap arguably lead with SemRep silver corpus (note should be taken that the SemRep silver corpus was constructed using MetaMap's word sense disambiguation engine). The CRAFT annotations with the NCBI and GO_CC vocabularies were best matched by cTAKES, while RysannMD topped on CALBC, SemRep(gold), CRAFT:CHEBI, CRAFT:GO_BPMF, and MesH WSD. The full rankings are given in Table 12 with median average across all corpora tested.

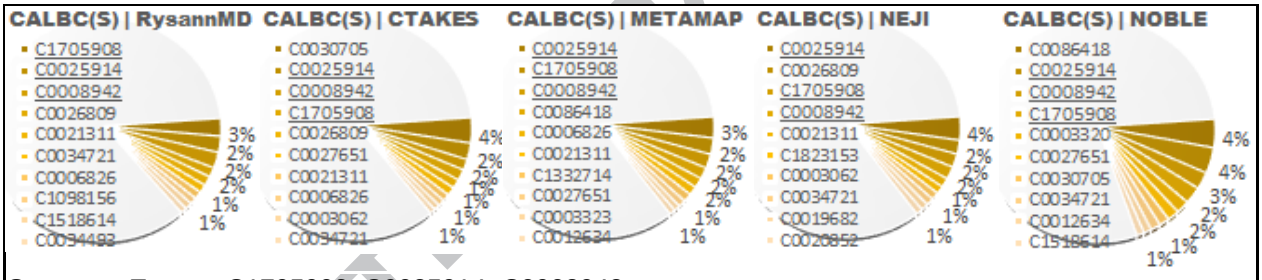
Table 12: F_1 ranking amongst the five annotators against the benchmarking corpora. ** indicates statistical tie (i.e. F_1 is within +/- of 95% confidence interval).

	RysannMD	MetaMap	NOBLE Coder	Neji	cTAKES+ytex
CALBC(silver)	1st	2nd	4th	3rd	5th
SemRep(silver)	2nd	1st	3rd	4th	5th
SemRep(gold)	1st**	2nd	1st**	1st**	3rd
CRAFT(gold):CHEBI	1st	3rd	4th**	4th**	2nd
CRAFT(gold):NCBI	3rd	4th	2nd	5th	1st
CRAFT(gold):GO_CC	2nd	4th	3rd**	3rd**	1st
CRAFT(gold):GO_BPMF	1st	3rd	2nd**	2nd**	2nd**
MeSH WSD(silver)	1st	3rd	2nd	4th**	4th**
Median:	1	4	2	5	3

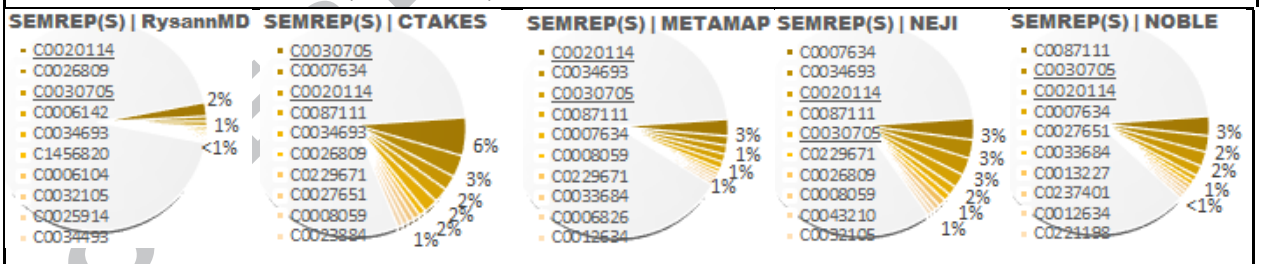
Error Analysis. Further insights into annotator effectiveness can be obtained by investigating the frequency and type of errors made. Figure 12 lists the top ten annotator mistakes, i.e., erroneously chosen UMLS concepts, by corpus including error percentages. We excluded the silver MeSH WSD corpus from this analysis due to its limited number of ambiguous terms (203). Similarly, we excluded the SemRep gold corpus due to its small size of 500 sentences.

The charts provide an ordered list of concept unique identifiers (CUIs) associated with the most frequent errors, with gradient scale illustrating the error percentage of the top ten mistakes, from the most to the least frequent ones. The proportional percentages are from the upper part of the pie chart where most mistakes are shown by dark gradients, to bottom of the chart where least mistakes are denoted by light gradient and assigned to the corresponding ordered list of CUIs in a clockwise direction. For example, for CRAFT(G):NEJI, the errors (with percentages) are associated with concepts: C3282337 (23%), C0026809 (16%), C0021027 (4%) and so forth; accounting for 57% of all mistakes leaving the remaining 43% "other" errors with less than 1% of occurrences (gray area). The underlined CUIs highlight common errors made by all (most) of the five annotators on the given corpus.

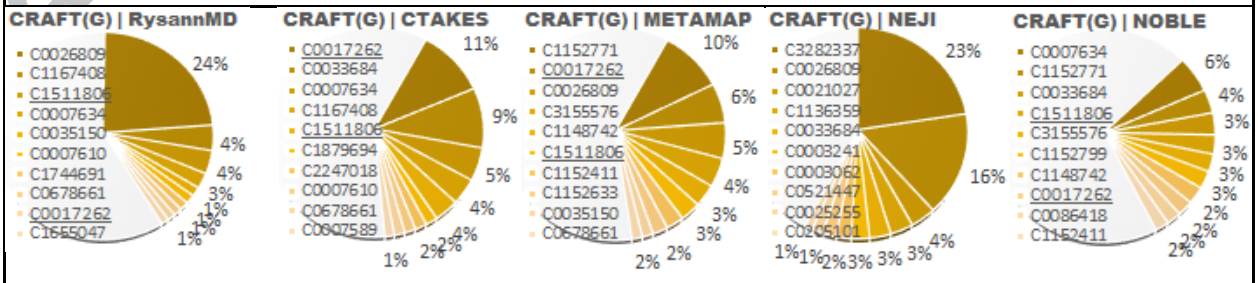
In general, the CRAFT corpus provided the most variance in the errors encountered with noticeably larger percentages than CALBC or SemRep. This variance might be attributed to a gold versus silver standard corpus construction or possibly due to the multiple taxonomies of CHEBI, NCBI, GO_CC, and GO_BPMF that comprise CALBC.



Common Errors: C1705908, C0025914, C0008942



Common Errors: C0020114, C0030705



Common Errors: C1511806, C0017262 except NEJI.

Figure 12: Top ten annotation errors for combination of corpus and annotator.

A matrix of the most frequent errors for any annotator-corpus pair, with the most frequent alternative answers, i.e., annotations assigned by the annotator is provided in Table 13. RysannMD had difficulty with the concept *Veterinary Patient* (C1705908) accounting for 3% of the total errors made on the CALBC corpus. In response, RysannMD offered *Patients* (C0030705), which is a broader concept error, though semantically acceptable as all veterinary patients are patients. The exact opposite is a narrower concept error, which occurred, for example, with cTAKES that assigned the *Veterinary Patient* concept, whereas the SemRep ground truth expected *Patients*. Similarly, MetaMap and Neji suggested an annotation of *Mus* (C0026809) and *Mus sp* (C1331371) instead of the more precise *House mice* (C0025914) defined in CALBC ground truth. A challenge for a general-purpose biomedical annotator is to satisfy benchmarking corpora where some favor broader general answers (SemRep, CRAFT) while others may require narrower specific annotations (CALBC).

CRAFT's ground truth of *Mus* (C0026809) defined as "*any of numerous species of small rodents belonging to the genus Mus*" gave RysannMD the most difficulty with 24% of all mistakes and was the reason for its 3rd place in F_1 ranking with the NCBI taxonomy (Figure 10). However, in the given context, the alternatives of *Laboratory mice* (C0025929) or *House mice* (C0025914) could arguably be better substitutions. The error of *Homo sapiens* on the SemRep corpus is a result of an older version of the UMLS based on which the SemRep corpus was constructed. The Concept Unique Identifier (CUI) was changed from C0020114 to C0086418 in later releases of the UMLS. This non-error adversely affected MetaMap, as well. NOBLE Coder also had difficulty with this concept but on the CALBC corpus. Specifically, the input text containing the spot "human" was not linked to *Homo sapiens,* but to *Persons* (C0027361). On its own, the word "human" could be associated with either homo sapiens or person but given the context of biomedical text, the concept of homo sapien (a scientific classification of a genus) is better suited than person (a human being regarded as an individual). Finally, a straightforward disambiguation error is illustrated with NOBLE Coder's mistaken annotation of *Cellular Phone* (C1136359) for the spot "cell" of the CRAFT corpus.

Table 13: Matrix presenting the most common annotation errors, also including most frequently given alternative answers (i.e., concepts chosen by the annotators).

	CALBC(S)	**SemRep(S)**	**CRAFT(G)**
RysannMD	truth:Veterinary Patient (C1705908)	truth: Homo sapiens (C0020114)	truth: Mus (C0026809)
	ans:Patients (C0030705)	ans:Homo sapiens (C0086418)	ans: Laboratory mice (C0025929), House mice (C0025914)
CTAKES+ytex	truth:Patients (C0030705)	truth:Patients (C0030705)	truth:Gene Expression (C0017262)
	ans: Report source - Patient (C1578483), Relationship modifier - Patient (C1578484)	ans: Veterinary Patient (C1705908), Disabled Person Code - Patient (C1578486)	ans: Expression Procedure (C0185117)
METAMAP	truth:House mice (C0025914)	truth:Homo sapiens (C0020114)	truth:olfactory receptor activity (C1152771)
	ans: Mus (C0026809)	ans: Homo sapiens (C0086418)	ans: Term - temporal (C1515273)
NEJI	truth: House mice (C0025914)	truth:Cells (C0007634)	truth:Cells [Chemical/ Ingredient] (C3282337)

	ans: Mus sp. (C1331371)	ans:Cells [Chemical/ Ingredient] (C3282337)	ans: Cells (C0007634)
NOBLE CODER	truth:Homo sapiens (C0086418)	truth:Therapeutic procedure (C0087111)	truth:Cells (C0007634)
	ans: Persons (C0027361)	ans:Treatment (CL370626)	ans: Cellular Phone (C1136359)

Speed. Table 14 ranks the computational time for different annotator and corpus combinations. Clearly, NOBLE Coder tops this measure with the 1st place ranking on all evaluated corpora. Conversely, cTAKES required the most time with a median ranking of 5th, followed by MetaMap (4th) and finally Neji (3rd). RysannMD performed well with the second fastest processing times averaging 12m37s. Admittedly, this pales to NOBLE Coder's 1m10s average, but is substantially better than Neji (21m41s), MetaMap (40m37), and cTAKES (216m00).

Table 14: Speed ranking of the five annotators against the benchmarking corpora.

	RysannMD	MetaMap	NOBLE Coder	Neji	cTAKES+ytex
CALBC(silver)	2nd	4th	1st	3rd	5th
SemRep(silver)	2nd	3rd	1st	5th	4th
CRAFT(gold)	3rd	4th	1st	2nd	5th
MeSH WSD(silver)	2nd	4th	1st	5th	3rd
Median:	**2nd**	**4th**	**1st**	**3rd**	**5th**

Generally the annotators performed proportionally to the size of the input text with some exceptions. That is, the SemRep corpus required the smallest amount of processing time; followed by MeSH WSD, CALBC, and lastly CRAFT, which required the longest processing time. Recall that the SemRep dataset consists of singleton sentences per input text. In CALBC and MeSH WSD input texts are abstracts and multiple sentences, while CRAFT includes full publications (multiple paragraphs). RysannMD required 17 (CALBC), 5 (SemRep), 17 (CRAFT), and 15 (WSD) minutes to process, respectively. Note that it took RysannMD the same amount of time to process CALBC and CRAFT (17 minutes), which indicates the size of 1,000 CALBC abstracts is about the same amount of text as 67 publications of CRAFT. Comparatively, MetaMap required 37 (CALBC), 11 (SemRep), 74 (CRAFT), and 75 (WSD) minutes. MetaMap took noticeably longer with WSD than CALBC even though the input text size of both of these corpora are similar. This may indicate that MetaMap may require additional time with texts that contain significant ambiguity. However, none of the annotators was more susceptible to long texts than cTAKES requiring 39 (CALBC), 13 (SemRep), 596 (CRAFT), and 40 (WSD) minutes with full document processing particularly difficult. Somewhat surprising was Neji with 30 (CALBC), 32 (SemRep), 3 (CRAFT), and 87 (WSD) minutes. Neji required approximately the same time to process input texts of full abstracts (CALBC) as it needed to process individual sentences (SemRep). However, it only took 3 minutes to process 67 full documents which indicates the speed of Neji is not adversely affected by the size of the input text. Furthermore, Neji required the most time with MeSH WSD even though the input text of this corpus is relatively similar in size to that of CALBC which may indicate Neji's disambiguation algorithm requires more time when presented with choices.

Effectiveness and Speed. Figure 13 summarizes the effectiveness versus speed findings in a magic quadrant plot. The quadrant plot shows that RysannMD and NOBLE Coder are more capable than cTAKES, MetaMap, and Neji. The choice between RysannMD or NOBLE Coder depends on the importance of speed in the particular annotation task at hand. If speed is crucial, then the selection of NOBLE Coder would relinquish effectiveness to RysannMD. If effectiveness is the top concern, then RysannMD sacrifices speed only to NOBLE Coder. Consequently, RysannMD is the most well rounded of the benchmark annotators balancing both speed and accuracy.

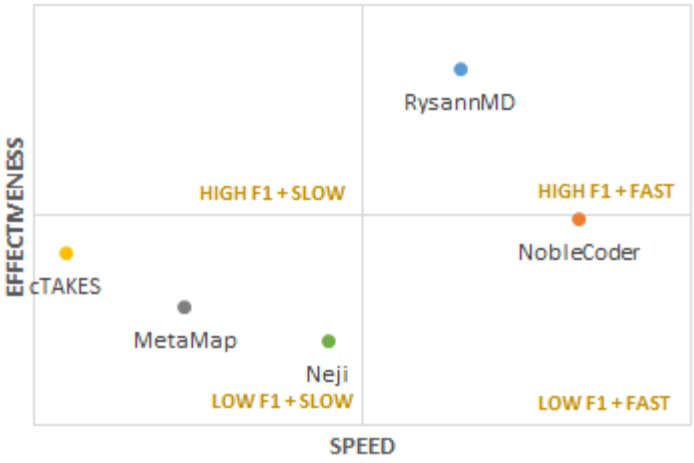


Figure 13: Magic quadrant plot of effectiveness versus speed.

9. Conclusion

In this paper, we have introduced RysannMD, a biomedical UMLS-based annotator that achieves accurate results across multiple corpora with comparatively low computational times. Our tests involved a diverse assortment of corpora: silver and gold standard, heterogenous domains and vocabularies, single sentence, paragraph, and full document-sized text. We benchmarked RysannMD against four modern annotators. Comparatively, RysannMD achieved the best median F_1 score across the benchmarking corpora independent of standard (silver/gold), domain, and document size. NOBLE Coder placed second in median F_1 followed by cTAKES, MetaMap, and Neji. We performed an analysis of annotation speed using the identical hardware platform for all tests. NOBLE Coder was clearly the fastest followed by RysannMD (2^{nd}), Neji (3^{rd}), MetaMap(4^{th}), and cTAKES(5^{th}). We conclude that RysannMD performed the best among the annotators when both accuracy (1^{st}) and speed (2^{nd}) are considered. Future work involves identifying and alleviating bottlenecks in our implementation to position RysannMD closer to NOBLE Coder's speed. For further study, the RysannMD source code is attainable by making a request to the third author. An online demo is also available[11].

Acknowledgements

Natural Sciences and Engineering Research Council of Canada
RGPIN-2015-06118

[11] http://denote.rnet.ryerson.ca/RysannMD

References

Jovanovic, J., Bagheri, E., Cuzzola, J., Gasevic, D., Jeremic, Z., Bashash, R. Automated Semantic Tagging of Textual Content. IT Professional 2014; 16(6):38-46.

Agirre, E., Soroa, A., & Stevenson, M. Graph-based word sense disambiguation of biomedical documents. Bioinformatics 2010;26(22):2889–2896. https://doi.org/10.1093/bioinformatics/btq555

Chasin, R., Rumshisky, A., Uzuner, O., & Szolovits, P. Word sense disambiguation in the clinical domain: a comparison of knowledge-rich and knowledge-poor unsupervised methods. Journal of the American Medical Informatics Association 2014;21(5):842–849. https://doi.org/10.1136/amiajnl-2013-002133

Hoffart, J., Yosef, M. A., Bordino, I., Fürstenau, H., Pinkal, M., et al. Robust disambiguation of named entities in text. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP '11). Association for Computational Linguistics, Stroudsburg, 2011, PA, USA, 782-792.

Tikk, D. & Solt, I. Improving textual medication extraction using combined conditional random fields and rule-based systems. J Am Med Inform Assoc 2010;17(5):540–544.

Hsu, C.-N., Chang, Y.-M., Kuo, C.-J., Lin, Y.-S., Huang, H.-S., Chung, I.-F. Integrating high dimensional bi-directional parsing models for gene mention tagging. Bioinformatics 2008;24(13):i286–i294.

Garla, V. N., & Brandt, C. Knowledge-based biomedical word sense disambiguation: an evaluation and application to clinical document classification. Journal of the American Medical Informatics Association 2013; 20(5):882–886. https://doi.org/10.1136/amiajnl-2012-001350

Divita, G., Zeng, Q. T., Gundlapalli, A. V., Duvall, S., Nebeker, J., & Samore, M. H. Sophia: A Expedient UMLS Concept Extraction Annotator. AMIA Annual Symposium Proceedings, *2014*; 467–476.

D'Souza, J., & Ng, V. Sieve-Based Entity Linking for the Biomedical Domain (pp. 297–302). Association for Computational Linguistics, 2015. https://doi.org/10.3115/v1/P15-2049

Savova, G. K., Masanz, J. J., Ogren, P. V., Zheng, J., Sohn, S., Kipper-Schuler, K. C., & Chute, C. G. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. Journal of the American Medical Informatics Association, 2010;17(5):507–513.

Jonquet, C., Shah, N., Musen, M. The Open Biomedical Annotator. *AMIA Summit on Translational Bioinformatics*, Mar 2009, San Francisco, CA, United States. pp.56-60, 2009.

Dai, M., Shah, N.H., Xuan, W., Musen, M.A.,Watson, S.J., Athey, B. Meng, F. An Efficient Solution for Mapping Free Text to Ontology Terms. AMIA Summit on Translational Bioinformatics, San Francisco (March 2008)

Tseytlin, E., Mitchell, K., Legowski, E., Corrigan, J., Chavan, G., & Jacobson, R. S. NOBLE – Flexible concept recognition for large-scale biomedical natural language processing. BMC Bioinformatics, 2016;17:32. https://doi.org/10.1186/s12859-015-0871-y

Aronson AR, Lang FM. An overview of MetaMap: historical perspective and recent advances. J Am Med Inform Assoc 2010;17(3):229–236.

Campos D, Matos S, Oliveira JL. A modular framework for biomedical concept recognition. BMC Bioinformatics, 2013;14:281. https://doi.org/10.1186/1471-2105-14-281

Campos D, Matos S, Oliveira JL. Gimli: open source and high-performance biomedical name recognition. BMC Bioinformatics 2013;14:54. https://doi.org/10.1186/1471-2105-14-54.

Cunningham H, Maynard D, Bontcheva K, Tablan V, Ursu C. The GATE User Guide 2002. http://gate.ac.uk/

ShARe/CLEF eHealth evaluation lab. SHARE-Sharing Annotated Resources. 2013. https://sites.google.com/site/shareclefehealth/home. Accessed 22 Nov 2016.

(Bada et al., 2012) Bada, M., Eckert, M., Evans, D., Garcia, K., Shipley, K., Sitnikov, D., Baumgartner Jr., W. A., Cohen, K. B., Verspoor, K., Blake, J. A., and Hunter, L. E. Concept Annotation in the CRAFT Corpus. BMC Bioinformatics. 2012 Jul 9;13:161

Kafkas S, Lewin I, Milward D, van Mulligen E, Kors J, Hahn U, Rebholz- Schuhmann D. Calbc: Releasing the final corpora. In: Proc. of the 8th International Conf. on Language Resources and Evaluation (LREC'12). Istanbul, Turkey: 2012.

Cuzzola J, Jovanovic J, Bagheri E, Gasevic D. Evolutionary Fine-Tuning of Automated Semantic Annotation Systems. Expert Systems with Applications. 2015; 42(20):6864–6877.

Lesk M. Automatic sense disambiguation using machine readable dictionaries: how to tell a pine cone from an ice cream cone. Proceedings of the 5th Annual International Conference on Systems Documentation; New York, NY, USA: 1986. P.24-26.

OpenNLP. https://opennlp.apache.org/ Accessed 30 November 2016.

Unstructured Information Management Architecture - UIMA. https://uima.apache.org/ Accessed 7 December 2016.

The CALBC Silver Standard Corpus for Biomedical Named Entities – A Study in Harmonizing the Contributions from Four Independent Named Entity Taggers

Rebholz-Schuhmann, D., et al. (2007) EBIMed – Text crunching to gather facts for proteins from Medline. Bioinformatics 23(2): e237-e244.

Schuemie, M., et al.. (2007) Peregrine: Lightweight gene name normalization by dictionary lookup, Proceedings of the Biocreative 2 workshop 2007 April 23-25, Madrid, 131-140

Hahn, U., et al., (2008) An overview of JCoRe, the JULIE Lab UIMA Component Repository, In Proceedings of the LREC'08 Workshop _Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP', Marrakech, Morocco, May 2008, pages 1-7.

Fleuren, W. W. M., & Alkema, W. (2015). Application of text mining in the biomedical domain. Methods, 74, 97–106. https://doi.org/10.1016/j.ymeth.2015.01.015

Meystre, S., Savova, G., Kipper-Schuler, K.C., & Hurdle, J. F. (2008). Extracting Information from Textual Documents in the Electronic Health Record: A Review of Recent Research In Geissbuhler, A. and Kulikowski, C., (eds.) *IMIA Yearbook of Medical Informatics 2008* (pp. 128- 144). Stuttgart, Germany: Schattauer.

Tanenblatt, M., Coden, A., & Sominsky, I.L. (2010). The ConceptMapper Approach to Named Entity Recognition. Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10). URL: http://www.lrec-conf.org/proceedings/lrec2010/pdf/448_Paper.pdf

Doğan R.I., Leaman R., Lu Z. (2014) NCBI disease corpus: a resource for disease name recognition and concept normalization. J. Biomed. Inform., 47, 1–10.

Ohta, T., Pyysalo, S., Tsuji, J., Ananiadou, S. Open-domain Anatomical Entity Mention Detection. In Proceedings of ACL 2012 Workshop on Detecting Structure in Scholarly Discourse (DSSD), 2012, pp. 27-36.

Humphrey, S. M., Rogers, W. J., Kilicoglu, H., Demner-Fushman, D., & Rindflesch, T. C. (2006). Word Sense Disambiguation by Selecting the Best Semantic Type Based on Journal Descriptor Indexing: Preliminary Experiment. *Journal of the American Society for Information Science and Technology*, *57*(1), 96–113. http://doi.org/10.1002/asi.20257

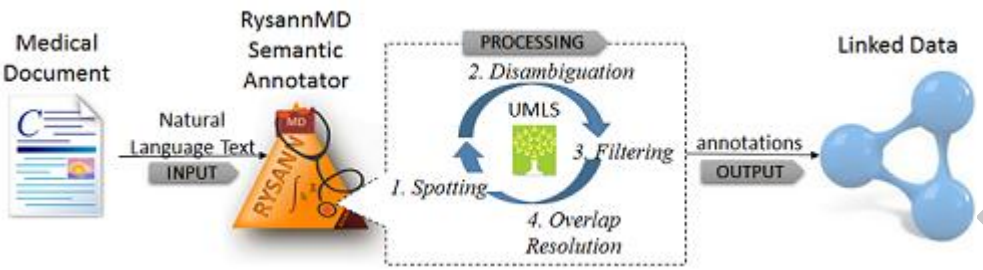
National Library of Medicine. List of journals indexed in Index Medicus 2002. NIH Publication No. 02-267; Bethesda, MD: National Library of Medicine; 2002.

Banerjee, S. & Pedersen, T. (2003). Extended gloss overlaps as a measure of semantic relatedness. *In Proceedings of the 18th international joint conference on Artificial intelligence (IJCAI'03)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 805-810.

Sánchez, D. & Batet, M. (2011). Semantic similarity estimation in the biomedical domain: An ontology-based information-theoretic perspective. *J Biomed Inform, 44*, 749–759. 10.1016/j.jbi.2011.03.013

Garla, V. N. & Brandt, C. (2012). Semantic similarity in the biomedical domain: an evaluation across knowledge sources. *BMC Bioinformatics, 13*, 261. https://doi.org/10.1186/1471-2105-13-261

Graphical abstract



HIGHLIGHTS:

- We propose a UMLS-based biomedical semantic annotator that is both accurate and fast.
- Our annotator was benchmarked against four state of the art biomedical annotators.
- The benchmarking was done on silver and gold standard biomedical corpora.
- Our method ranked 1st in median accuracy and 2nd in median speed across all tests.
- Our method performs well on documents of varying sizes and biomedical subdomains.