A SNOMED annotator for UIMA Framework

By

Deepthi VL Yellapragada

Thesis submitted to the College of Engineering and Mineral Resources at
West Virginia University in partial fulfillment of the requirements
for the degree of

Master of Science

in

Electrical Engineering

Committee Members
Dr.V.Jagannathan, Ph.D., (Committee Chair)
Dr.Ramana Reddy, Ph.D.
Dr. Arun Ross, Ph.D.

Lane Department of Computer Science and Electrical Engineering
Morgantown, West Virginia
2007

ABSTRACT

A SNOMED annotator for UIMA Framework

Deepthi VL Yellapragada

Paper based health records has proved to be not only costly but also has many disadvantages. Such records are difficult to organize and are hard to manage. This has led to the adoption of computerized electronic health records (EHR). To facilitate interoperability and consistency in exchange of information between various computerized clinical applications standard medical terminologies plays a vital role. These standard terminologies are based on controlled medical vocabularies like SNOMED, ICD9, and LOINC etc. Coding clinical documents with controlled medical vocabularies benefits the health care system in a wide range of applications like laboratory reporting, clinical research, computerized patient entry etc. This document deals with using the SNOMED medical vocabulary subset from UMLS for coding clinical documents.

Coding of clinical documents would be more effective if we can avail the available natural language processing techniques. This work utilizes the UIMA platform which has a collection of natural language processing (NLP) tools. It also provides a framework on how these open NLP tools can be plugged into the UIMA architecture and how they can be used in developing any application. This report discusses the use of a statistical n-gram natural language processing technique that proved effective in assigning SNOMED codes to composite medical terms. With the component based architecture of UIMA which allows sharing and using of components between projects the thesis discusses how two components developed in separate projects are combined to form a SNOMED annotator.

Thus this document describes the process of developing a SNOMED annotator for coding clinical documents which is scalable, flexible, UIMA framework compatible and extendible.

ACKNOWLEDGEMENT

I would like to express my gratitude to my advisor Dr.V.Jagannathan for his support and guidance through my thesis. I am also grateful to him for introducing me to new and interesting technologies in software development.

I am also grateful to my other committee members, Dr. Ramana Reddy and Dr.Arun Ross for their valuable advice and support. I would also like to thank all the professors and staff of the Lane Department of Computer Science and Electrical Engineering.

I also extend my sincere thanks to Charles J. Mullett, MD, PhD, James G. Arbogast, MD,KevinA. Halbritter, MD,Linda Crossely, Tad Davis, Chen He, David Taylor, Timothy Owensby and Christian Richards for their kindly co-operation.

Finally I would like to thank my family, my husband and friends Nikhil and Jyotsna for their constant help and support.

Table Of Contents

1. INTRODUCTION

1.1 Problem Description

Currently, enormous amount of medical data are available in electronic health records. An electronic health record consists of a collection of information related to the care provided to a patient. This information is used in a variety of clinical applications, such as clinical decision support, drug-drug interaction alerts. However, the encoding of the information, in such records is not standardized. With the result, clinical information cannot be shared consistently between computerized applications. Semantic interoperability is one of the major issue to be addressed for making effective use of electronic health records in clinical applications. One of the efforts to meet the interoperability needs resulted in the generation of HL7 Clinical Document Architecture that specifies the encoding, structure and semantics of clinical documents for exchange**.**

The CDA allows the medical data that is being exchanged to be mapped to a standard terminology. The need for a standard terminology is because in the medical field there exists different clinical terms that would mean the same thing, and medical information needs to be exchanged consistently between health care providers, researchers and others. Terminologies help in defining a standardized language for health i.e. a computerized language for global use. The various coding systems available are ICD, CPT, SNOMED, UMLS etc. Use of standard terminologies for describing medical terms enables health professionals to send and receive data in an understandable manner. It helps in electronic data collection at the point of care, extraction of medical information, and various other purposes like patient safety reporting, clinical decision support etc.

1.2 Objective:

The effort here describes a way to automatically generate SNOMED terms included in a dictated clinical report. The approach used is to use the UIMA framework and the open NLP tool kits including using the sliding window NLP technique. In this method the biomedical text documents are passed through an aggregate analysis engine that is a combination of Open NLP Parts Of Speech (POS) Tagger and Open NLP Parser. The medical term annotator iterates through the indexes of the Common Analysis System (CAS) object generated by the above analysis engine. For identifying single words it uses the Open NLP POS Tagger annotations and for identifying medical terms that exist as group of words it uses

the annotations produced by the Open NLP Parser. While iterating through these annotations it assigns SNOMED codes for medical terms that are identified in the My SQL database. The SNOMED codes of the medical terms that have multiple entries in the database are concatenated together as a single string. The SNOMED database itself, was obtained from UMLS Metathesaurus. Also another method has been implemented to achieve enhanced results. This involves adapting the Statistical Natural Language Processing Technique called the n-gram windowing technique. This method has been implemented based on the assumption that a medical term will contain maximum of four terms. Based on this the tokens which are annotated from the Tokenizer and grouped together as groups of words containing 2,3 and 4 words and the corresponding terms are looked for in the database for the SNOMED code. These codes are assigned as feature values for all the token annotations. This method proves effective for medical terms that are made of more than one word and those that cannot be identified by a Phrase detector.

1.3 Related Work

There are many efforts made in mapping terminological codes to clinical text. An effort has been made by Cooper and Miller in developing and evaluating three different methods (i.e. PostDoc (a lexical indexing system) and Pindex (a statistical indexing system) and a hybrid system of the two).This helps in MEDLINE searches that include the concepts in the text. The PostDoc method which was based on lexical indexing and string matching resulted in a recall of 40%, the Pindex method which was based on statistical indexing that assigns phrases in text to the corresponding MeSH terms based on frequency gave a recall of 45%. And a hybrid system of the above two methods gave a recall of 66%. Also Nadkarni, Chen and Brandt of Yale university school of medicine explored the feasibility in using NLM's Unified Medical Language System (UMLS) in computational indexing of clinical terms. They have identified that concept indexing cannot be used as the only means of preprocessing medical text. Some of the causes of the problems are the existence of redundant concepts, abbreviations, spelling mistakes; missing concepts in UMLS etc.An effort by Lowe reported using the Saphire indexing system for mapping of important biomedical concepts in radiological reports to the concepts in UMLS Metathesurus. This helps in retrieving radiological report images from patient records. This method produced good recall but poor precision and for improving the precision context based mapping is adapted. Also various phrase based methods were used in indexing clinical information have been reported

1.4 Previous Work Related to Application of UIMA in life sciences

BioTeKS is the major IBM application that makes use of the UIMA framework for analyzing biomedical text. Several IBM research efforts are reported for developing a system for analyzing and

searching of biomedical text that helps in problem solving in life science. Biological Text Knowledge Service (BioTeKS) is result of such a effort from various IBM research laboratories. The major goal of this system is information extraction from biological text. This includes identifying biomedical entities like genes, proteins, drugs etc and the various relations between them. The text Analysis components of BioTeKS systems consist of a text analysis engine which consists of a sequence of text annotators that process each of the input documents. The use of UIMA helps in improving the process of text analysis by providing a standard way of developing, deploying and integration of various text analysis methods that can operate at different levels of analysis. UIMA provides the best way of orchestration and integration of different analysis methods that needs to be applied at different levels of the linguistic structure. BioTeKS makes use of various NLP text annotators like Parts Of Speech Tagger, Finite state transducer (FST) with shallow parsing syntax rules and the Language Ware linguistic engine. The text analyzed data is stored in a database and a text search engine. This data can be accessed by the application-enabling engines and can apply application specific text mining methods. The various application prototypes built with the BioTeKS system are Semantic search, Document Clustering, Analyzing mutual co-occurrences among terms using lexical networks, Analyzing gene clustering using Med Mesh Summarizer etc... BioTeKS system includes software components for loading UMLS and MeSH ontology information into the relational database.

2. BACKGROUND

This chapter discusses about medical terminologies with main emphasis on SNOMED terminology. It also explains about the Unstructured Information Management Architecture (UIMA) and its usage in medical field.

2.1 CONTROLLED MEDICAL VOCABULARIES

Medical informatics is the combination of computer science, information science and medical care. It concentrates on the devices, resources and methods which are needed for the optimizing storage, retrieval and use of medical information in health. The requirement for a standard terminology is widely recognized with the increasing growth of medical informatics. Use of a standard terminology enables efficient indexing and processing of patient data. It becomes an essential element for adapting knowledge based clinical decision-support, data aggregation and retrieval. In this process of adapting standard medical terminologies the usage of controlled vocabularies has played a vital role. Creation of applications related to medical informatics requires controlled medical vocabularies to support their systems. Also systems that deal with free text and images seemed to have improved capabilities with the coding of the data using controlled medical vocabularies. During the earlier times of medical informatics, systems developers used ad hoc sets of medical terms which were sufficient for the development of their own application. Developing applications with the small sets of vocabularies is not a difficult task. But with the increasing complexity and function of the application the need for developing a controlled medical vocabulary has gained importance. Introduction of every new system requires creation of a new vocabulary as the previous vocabularies proved to be not useful. Also interoperability between two systems has been a major issue because of the differences between the medical vocabularies used by the two systems. The use of standard, controlled medical vocabularies for coding patient information is a well established procedure for U.S. health care providers. Some of the standard medical vocabularies available are SNOMED, ICD9, LOINC, UMLS and READ.

The following are the requirements that should be accomplished by standard controlled medical vocabularies[2]

Synonymy:

As the users cannot remember all the terms in the vocabulary, there should be a provision for synonyms which enables relating medical terms with their synonyms when the medical term does not exist in the vocabulary directly. There should also be provision for relating abbreviations to the corresponding terms.

Domain Completeness:

There is no vocabulary that has complete coverage of medical terms. Though there might be vocabularies that have complete coverage for specific domains. Domain completeness is allowed in theory, because there is no inherent limitation on the size of the network with regard to number of nodes in the networks as a whole, number of nodes in a class, depth of a node in the hierarchy, number of relations in the network, or number of relations involving any one node.

Nonredundancy:

It is required that a concept can be represented in only one way in a vocabulary. Relating two terms in a vocabulary relate to the same concept will result in decreased sensitivity while querying the vocabulary.

Unambiguous:

All the terms in a vocabulary should be unambiguous. It means that the concepts included in a vocabulary should be complete in meaning. Also the vocabulary terms must not be referring to more than one concept. If a term turns ambiguous then it has two types of data stored under it and that would affect the specificity of the query.

Hierarchy:

A medical vocabulary cannot be just a collection of medical terms as it leads to complexity in retrieval and maintenance. Hence a hierarchical way of representing concepts is required though it reduces the flexibility of allowing a term to belong to more than one class.

2.1.1 SNOMED (Systematized Nomenclature Of Medicine)

There is a high degree of variation in expressing medical terms by various clinicians around the world. For example the terms myocardial infraction, heart attack and MI means the same to a cardiologist, but they cannot be treated the same by a computer without telling it they are the same. Each clinician has their own way of expressing the term and this is not a problem for other humans to understand, however, computer applications such as clinical decision support cannot function without being told exactly in unambiguous terms. Thus in order to achieve clear and

unambiguous communication between various health care applications there is a requirement for a standard medical terminology. SNOMED is a comprehensive medical terminology that covers diseases, clinical findings, procedures, micro organisms, pharmaceuticals etc. The terms are systematically organized so that they are computer processable allowing a consistent way to index, store, retrieve and aggregate clinical data across all sites of clinical care. This terminology helps in structuring and computerizing medical data, and thus avoiding variability in the way data is captured, encoded and used in patient care and research. SNOMED originally developed by College Of American Pathologists, is now being managed and maintained by International Health Terminology Standards Organization, for improving and safe guarding patient care with the usage of an agreed terminology.

SNOMED is a compositional concept system which means that the concepts in the vocabulary are specialized by combining them with other concepts. The main components of a SNOMED system are:

Concepts: These are the basic units of the vocabulary which contains the meanings. They are designated by a unique code, unique name and description. They might also be assigned a preferred term of usage and any synonyms if exists.

Descriptions: These include the terms or names that are assigned to a concept.

Hierarchies: SNOMED vocabulary consists of 19 higher level hierarchies, with each having a sub hierarchy.

Relationships: These would link concepts that exist within a hierarchy or across hierarchies.

Some of the computer applications that use SNOMED CT are: Electronic Health Record, Computerized Patient Order Entry, Cancer Reporting Emergency Room Charting, Surgical Procedure Masters, Laboratory Reporting, Problem Lists Literature Encoding, Autopsy Databases, and Global Benchmarking etc. Also usage of SNOMED codes reap benefits like: consistent communication of patient clinical records, simplified data entry and retrieval, provision of a single and comprehensive system of terms that is centrally maintained and updated to be used by all organizations, reliable research and analysis as it is based on a common understanding of health terms and concepts that are stored in coded form etc..

2.2 Unstructured Information Management Architecture (UIMA)

UIMA stands for Unstructured Information Management Architecture. It is a Java based framework developed by IBM. It is an open, industrial-strength, extensible and scaleable platform used for creation, integration and deployment of unstructured information management solutions. It uses various combinations of semantic analysis and search components. It is component software for the discovery, development, composition and deployment of multi-modal analytics for the analysis of unstructured information. It integrates them with the search technologies developed by IBM. UIMA is made available as a free SDK; also the core Java Framework is available as open source software. And it thus provides a common foundation for industry and academia to collaborate and accelerate the world wide development of technologies that are important for discovering the vital knowledge present in the fastest growing sources of information.

2.2.1 What is Unstructured Information?

Unstructured Information implies the large amount of fast growing data that is available all over the world to businesses and governments agencies. It also includes documents found on internet, information generated by enterprises around the world. This data exists in forms such as natural language speech, text or video that lacks the organization of traditional sources like database records. In structured information, meaning is expressed by the structure or format of the data; where as the meaning of unstructured information cannot be inferred in like manner.

2.2.2 What is Structured Information?

Structured information is characterized as the kind of information whose meaning is unambiguously defined and is explicitly represented in the structure or format of the data. Standard example for structured information is relational database.

2.2.3 Importance of Extracting structured Information from Unstructured data:

The impact of extracting information from unstructured data will not only change the commercial industry but also change the society as well in the coming decades. The usage of database technologies have enabled end users in managing every aspect of their operations as it involves accessing of structured data. Unstructured Information Management technology extends

this usage of database technologies to the data that exists in unstructured form like text documents, audio files and video content. Most of the business documents of an enterprise, the large knowledge corpus available on the World Wide Web are all in the form of unstructured data and emerging technologies in mining these unstructured data would be of great help in solving major part of the societal and business problems.

2.2.4 UIMA A Platform for Integrating All Natural Language Processing Technologies:

With all the above problems mentioned for applying natural language processing techniques an alternative approach that capitalizes on the computational power of distributed systems has been developed within IBM research. This is an architecture that enables integration of all the above mentioned technologies with search and retrieval. This architecture developed by IBM is named as Unstructured Information Management Architecture. It is a software architecture that supports development, integration and deployment of these Unstructured Information Management technologies.

2.2.5 Developing A UIMA Application:

2.2.5.1 What is an UIMA Application?

An Unstructured Information Management application may be thought of a software application that analyses large amounts of unstructured data and retrieves information in a structured format that is useful to the end user. An example of UIMA applications is a software system that analyses huge volumes of medical abstracts and extracts useful end user information and extract the drugs mentioned in each abstract. Such data can later be used to analyze drug-drug interactions.

2.2.5.2 Phases in developing a UIMA application:

1. **Analysis Phase**: In this phase of the application the collection of documents are acquired and are analyzed. Analysis of the document includes functions like tokenization and semantic search. It also includes annotating certain entities like organizations, person names, drug names by referring to dictionaries and ontologies. Also relationships among these classes like working in and citizen of might also be annotated. The results of the analysis phase are used as a search engine index containing the annotations and their relationships.

2. **Delivery Phase**: This phase uses the indexed data delivered in the analysis phase for the query interface. This forms the user interface where the user queries for the required documents by mentioning a combination of tokens. The appropriate results are selected through a semantic search engine using a Boolean combination of these tokens, entities and relationships.

2.2.6 The Feedback Loop Concept:

One of the important features of the UIMA architecture is the information Feedback loop concept. In this the structured resources which are produced in the collection level analysis are subsequently used in analyzing the incoming unstructured information. An example for this can be development of ontologies. In this the analysis engine is first trained with a set of documents for identifying concepts and also their possible mentions in the documents at various instances that would add weightage to the concept annotated. This might result in creation of a structured data base with all the concepts and their corresponding links. These results are used in analyzing new documents coming in.

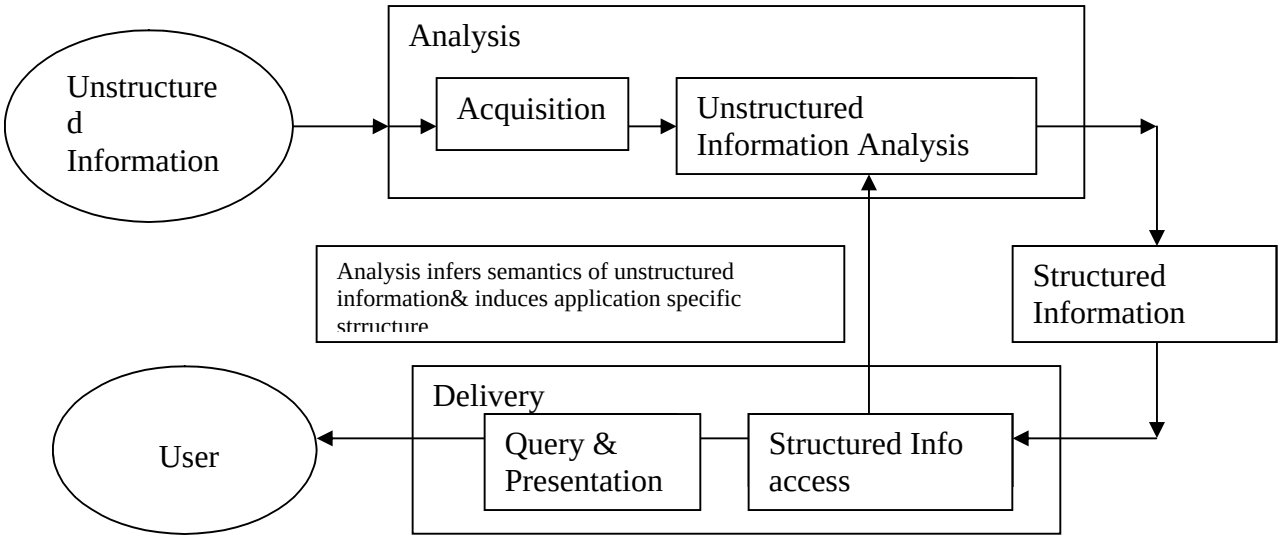


Fig 1 : Feedback loop concept of UIMA

2.2.7 UIMA Architecture Overview:

The two main components of UIMA architecture are Text Analysis Engine (TAE) for processing a single document and a Collection Processing Engine (CPE) for collection level

analysis of the documents. The main properties that a component of UIMA architecture should posses are:

1. Data driven (i.e. each component's function should depend only on its input data which ensures that the only requirement for reusing a component in an integrated system is that the components input requirements are met. The components should not rely on factors like presence of other components because this does not allow reusability).
2. Self descriptive (i.e. each component should publish its metadata in the form of descriptors. The main metadata are the component's capabilities which specify the components input requirements and output format. This would be useful for the developers for choosing components that meet their need).

The main components in UIMA architecture are:

1. **Text Analysis Engine (TAE)**: The text analysis engine handles document level analysis. The main component of a TAE is an annotator. Annotator interface basically implements the required analysis algorithm e.g. identifying person names, tokenization etc. This annotator along with its descriptor forms the total TAE component. A group of TAEs form an aggregate TAE and this is defined by an aggregate descriptor.
2. **Collection Processing Engine**: This consists of collection of TAEs, Common Analysis System (CAS) reader, CAS initializes and CAS consumers. CAS readers forms the interface with the documents, they fetch each document and pass it to the CAS initializers. These initializers would format the document into a form usable by the TAE components.TAE applies the appropriate algorithm on the input CAS and pass the results to the CAS consumers. A CAS consumer can either be a database or a search index based on the application. These are mainly concerned with delivering the data to the end user.

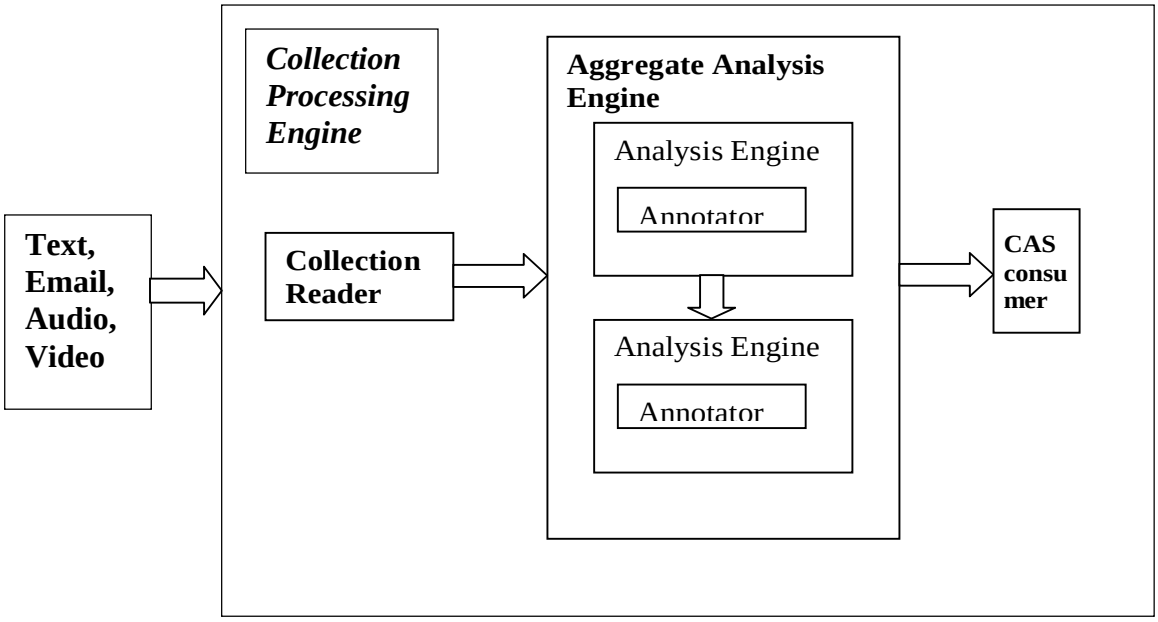


Fig2: UIMA framework architecture

2.2.8 The Common Analysis System:

In addition to the above mentioned components the CAS (Common Analysis System) forms the primary subsystem in the UIMA architecture. It handles data exchanges between different components and also different UIMA applications. The CAS data model for the UIMA framework is a tradeoff between the following two approaches:

1. Flexibility of just passing the data itself between the components without any knowledge of its structure and that the kind of data that can be transferred is component specific. Here the framework does not extend any support in handling the data. Also the APIs and tools for handling data should be provided by the components itself. Thus this approach has the disadvantage of lack of support from the framework.
2. In the second approach the data services are totally supported by the framework. The data that is exchanged between components has a fixed format and the framework defines the API s for handling data. This approach is not flexible as the data model is fixed and cannot be extended.

As a tradeoff between the above two approaches UIMA has come up with a new kind of data model that is structured but can be extended as well according to the requirements of the application. The main disadvantage here is that the data model is complicated.

Thus CAS is a data model and a data container that monitors the data produced and consumed by different components and uses this knowledge in controlling the work flow. It also controls the data produced by the analysis engines so that it is generated in a format compatible for network transport and storing.

The CAS entities come under three major groups:

1. A type system that has a dual role to play: one to create a data model from the TAE's specifiers at the beginning. It also provides information about the data model during the run time of the system.
2. The second group of CAS is concerned with defining the data according to the type system. Thus they create and support feature structures.
3. The third group defines a set of APIs that specifies indexing information. Here indexing defines the way data can be accessed by different components.

The CAS implementation that specializes for text analysis is called TCAS. It provides several types, features, indexes for processing texts. Though it does not add any additional functionality it just forms a compatible layer over the basic component.

2.2.9 ROLES DEFINED FOR DEVELOPING A UIMA APPLICATION:

In order to combine their work with the other researchers work and to come up with coherent products and taking into account the wide range of skill sets required for each of these tasks, IBM has identified 5 different roles for a UIMA application developer. The roles are:

1. **Annotator Developer**: Following are the functions of an annotator developer:
   Code the core analysis algorithm.
   Define data model for representing the annotations created.
   Declare metadata in its descriptor about their components.
2. **Analysis Engine Assembler**: Analysis engine assemblers mainly concentrate on aggregating the available annotators in a particular sequence required to perform a specific analysis task. It also finds if there are any more annotators to be designed to complete the task and is any thing found will inform the annotator developer about it.

3. **Collection-processing-component developer**: A collection processing component developer mainly deals with developing an application that deals with a collection of documents. This includes developing a list of components like Collection readers, CAS initializers, and CAS consumers. The job of a collection processing component developer is similar to that of a annotator developer except that a annotator developer implements the algorithm class on a simple interface where as a CPE developer implements it on a consumer interface.
4. **CPE assembler**: Similar to an Analysis Engine Assembler a CPE assembler assembles the available components to achieve a particular task. It mainly deals with specifying required analysis engines and collection-processing components for achieving a particular task. The descriptor builder tool can be used to identify the components and their work flow.
5. **Component deployer**: A component deployer deploys all the required TAEs, CPEs and any other resources on specific hardware and system middleware. He chooses appropriate middleware and also generates a deployment descriptor. This includes the kind of communication protocol used, and the number of simultaneous request that can be supported. UIMA extends its support for component deployer through its adapter framework.

2.2.10 OTHER UIM Frameworks:

There are other UIM frameworks like GATE and ATLAS that preceded the UIMA architecture by IBM. In this section a brief review of each of these architectures and their comparison with UIMA is made.

GATE (General Architecture for Text Engineering):

Similar to UIMA, GATE is a software infrastructure on which different NLP processing systems can be integrated, analyzed and refined individually if required. The GATE architecture is only considered with text analysis, where as UIMA framework can be used for analyzing unstructured data that include audio and video content. GATE architecture does not specify any specific data layer unlike the CAS in UIMA framework. It uses the native data structures for data exchange and management. GATE enables development of many end user applications like information extraction, document generation and machine translation. UIMA extends support for running many of the GATE analytic models.

UIMA and GATE share many common features like representing documents as text plus annotations and allowing users to aggregate multiple analysis engines together to performing a set of functionalities on the document. Hence efforts are being made to include UIMA components in GATE applications so as to allow GATE users to take advantage of the flexible UIMA framework and also that the UIMA users can take advantage of accessing pattern matching language like JAPE supported by GATE and also can use the already existing useful plug-in of GATE. This is achieved by defining an interoperability layer between GATE and UIMA architectures.

ATLAS (Architecture and Tools for Linguistic Analysis Systems):

The ATLAS framework provides an architecture that facilitates the development of linguistic annotation applications. It makes use of a generic data model. Unlike UIMA ATLAS has a rich built in data model that is suited only for specific tasks and all the data appear to be in annotation graphs. In UIMA these structures need to be developed out of the basic building blocks. . The ATLAS architecture mainly comprises of 4 main components: a data model, an Application Program Interface, the ATLAS interchange format and the ATLAS Meta Annotation Infrastructure. The ATLAS framework can be used for modeling both simple and most complex annotations and with the introduction of the MAIA concept it provides a way of modeling the semantic dimension also. UIMA mainly concentrates on providing flexibility for the convenience of application developers at the price of reduced generality. Also more specific data models and support layers can always be added on top of the existing CAS layer of the UIMA architecture.

2.2.11 CURRENT UIMA APPLICATIONS:

1. **BioTeKS(Biological Text Knowledge Services):**

   It is the first major application developed in the IBM research. It is an integration of various research technologies from IBM research labs. This project mainly targets on knowledge discovery in the field of life science which aids in improving drug development, medical practice and health care and would be of great use to areas like medical informatics, clinical genomics etc.. It concentrates on extracting useful information entities like genes, drugs, proteins etc from resources like biomedical

abstracts from MEDLINE, patents, medical reports etc. It also extracts the concepts or facts related to all the biomedical entities extracted.

2. Information Portal For Market Intelligent Management Systems:

Integration of market intelligence is very important for enterprise executives who require market intelligence information from various sources. As the amount of data available is huge and as their rate of growth is high there is a high demand for systems that automatically mange market intelligence information. To meet this IBM has developed a knowledge portal that provides a single access point for the user to all the market information. This uses entity level knowledge technologies in spite of document level by forming a knowledge network called EntityNet. The framework enables the network to store entity level information and it provides semantic services for the user. Most significant features of the EntityNet are the customizing of document- processing flow and personalized view of data.

3. Processing Dialogue Based Data in UIMA Framework:

Efforts are being made in analyzing dialogue based data which includes audio, video and text using the UIMA framework. Currently the UIMA framework is used for merging and analyzing the linguistic annotations in the given transcription. A statistical analysis of these annotations and their probable combinations help better understanding of the linguistic and behavioral cues in human machine translation. This data along with the time stamp included is used in analyzing the audio and video content of the transcription being analyzed.

4. The DKPro project:

There is an ongoing project which aims at creating a intuitive information access in Web and E-learning undertaken by the Ubiquitous Knowledge Group at Darmstadt University of Technology. The mid term goal for this project is to provide a collection of software components for semantic information processing base on UIMA. This is named as the Darmstadt Knowledge Processing Software Repository. This project aims to do semantic information processing on all media types (i.e. text, audio and video) on all domains (e.g. Semantic web services and e-learning) and on all languages.

2.3 Unified Medical Language System (UMLS)

The Unified Medical Language System is a controlled compendium of various available vocabularies. It also provides mapping between the included vocabularies. It was designed by Donald Lindberg, M.D., Director of the US National Library of Medicine in 1986. UMLS enables the computer systems to understand biomedical language and results in higher performance in creating, processing, retrieving and integrating health care data. One of the primary goals of medial informatics is to develop a Universal Medical Records for all patients. There is no problem in understanding a patient file by any hospital system as the record is written in some kind of universally understood language. But the problem here is knowledge representation. It is found that current computerized vocabularies use totally incompatible ways of classifying disease. While one vocabulary classifies a disease as acute or chronic, other vocabulary classifies it based on the location. Hence there is no uniform way of classifying such things. This created a need for a universal vocabulary which resulted in the development of UMLS. The main aspects of UMLS are the knowledge sources (databases) and the software tools required to implement them. Developers use the knowledge resources required for their application and the UMLS software tools helps them in customizing the knowledge sources for a particular purpose. The three Knowledge resources available through UMLS are: the Metathesaurus, the Semantic Network, and the SPECIALIST Lexicon. The tools available for distributing these knowledge resources is the MetamorphoSys install.

Metathesaurus:

It is like super thesauri of the entire thesaurus currently available. It is the core of the UMLS. It is a large, multi-purpose and multi-lingual database that contains information from various medical concepts, their names and the relationships between them. The Metathesaurus has been generated from the source vocabularies using a combination of automated processing software and also using extensive hand editing by human editors. In the automatic processing of source vocabularies for generating metathesaurus, if a particular term is not found in any of the other vocabularies, human intervention is made to check if the term exists in different forms in other vocabularies or if it is a completely new term. If it is a new term then a new concept is created in the UMLS with a unique concept id (CUI).Otherwise if the term is different only in the textual sense then it is declared a separate string of the same concept and is assigned a unique string id (SUI). This includes examples where the terms differ in their case (upper or lower). Apart from all these each term irrespective of its existence or non existence in any source

vocabulary is treated as an atom and is assigned a unique atom id (AUI).i.e. each appearance of a vocabulary term is assigned an AUI. Following table retrieved from the UMLS standard site:

Concept (CUI)	Terms (LUIs)	Strings (SUIs)	Atoms (AUIs) * RRF Only
C0004238 Atrial Fibrillation (preferred) Atrial Fibrillations Auricular Fibrillation Auricular Fibrillations	**L0004238** Atrial Fibrillation (preferred) Atrial Fibrillations	**S0016668** Atrial Fibrillation (preferred)	**A0027665** Atrial Fibrillation (from MSH)
			A0027667 Atrial Fibrillation (from PSY)
		S0016669 Atrial Fibrillations	**A0027668** Atrial Fibrillations (from MSH)
	L0004327 (synonym) Auricular Fibrillation	**S0016899** Auricular Fibrillation (preferred)	**A0027930** Auricular Fibrillation (from PSY)
		S0016900 (plural variant) Auricular Fibrillations	**A0027932** Auricular Fibrillations (from MSH)

The Metathesaurus is linked with other Knowledge Sources. All terms in the Metathesaurus are linked to at least one Semantic Type in the Semantic Network and many of the words or multi-word terms that appear in the concept names or strings in the Metathesaurus find an entry in the Specialist Lexicon.

Semantic Network:

This mainly deals with the relationships between the various concepts that are mentioned in the Metathesaurus. It provides a consistent categorization of concepts mentioned in the Metathesaurus. The knowledge representation scheme involves nodes and links. Nodes determine the various concepts and the links define the relationships between these concepts. The network mainly consists of Semantic types and Semantic relationships. There are 135 semantic types and 54 relationships. The major existing semantic types are organisms, anatomical structures, biologic function, chemicals, events, physical objects, and concepts or ideas. And the different kinds of semantic relationships include "is a" link, "physically related to”

link, "spatially related to" link, temporally related to" link, "functionally related to" link and "conceptually related to" link.

Specialist Lexicon:

The Specialist Lexicon covers both commonly used English terms and most of the biomedical terms. For each concept the lexicon entry consists of the syntactic, morphological and orthographic information which is required by the Specialist Natural Language Processing System. The entries in the Specialist Lexicon can be one word or multiple words. For each entry the record consists of the basic form of the word, the parts of speech of the word, a unique identifier for the word and any available spelling variants that may exist. For multi word concepts the lexicon provides word order variants for the concepts. There are two formats of Specialist lexicon available: unit word format which consists of slots (this includes the elements like spelling variant, base form etc) and fillers are the values assigned for the slot entries.

3. SYSTEM OVERVIEW

This chapter gives a brief overview the application developed for automatically assigning SNOMED codes to medical terms present in a free text. It explains how the open NLP technologies for tokenization and phrase detection have been utilized.

3.1 Goals

The main objective of this application is for automatic assignment of SNOMED codes for medical terms in free text are summarized below:

- *Framework compatibility*

  This is one of the primary goals of the system developed. The application should be capable of using the existing resources of the UIMA framework. It should be made compatible to the framework to make use of these resources. The current application is developed making use of the existing open nlp tools available through UIMA. It works as any other annotator on the UIMA framework. This helps in easier extension of the application in future just by adding its descriptor to the analysis engine.

- *Handling multi word medical terms*

  The application should be able to identify medical terms that are made of more than one word. In this application this is accomplished using the following ways:

  - Using an open nlp phrase detector
  - Using the statistical n-gram model

- *Scalability*

The application created should be able to be extended further in any manner desired. This can be achieved just by adding requisite annotators to the aggregate analysis engine. This functionality is achieved because of the implementation of the application on UIMA architecture.

- *Flexibility*

The application should be flexible and be able to work with other coding systems. This is achieved with the usage of UMLS which has all the current medical vocabularies and the application can make use of any required vocabulary just by selecting the required database using the Metamorphosis installation of UMLS.

System Architecture

The following diagram depicts the architecture of the system:

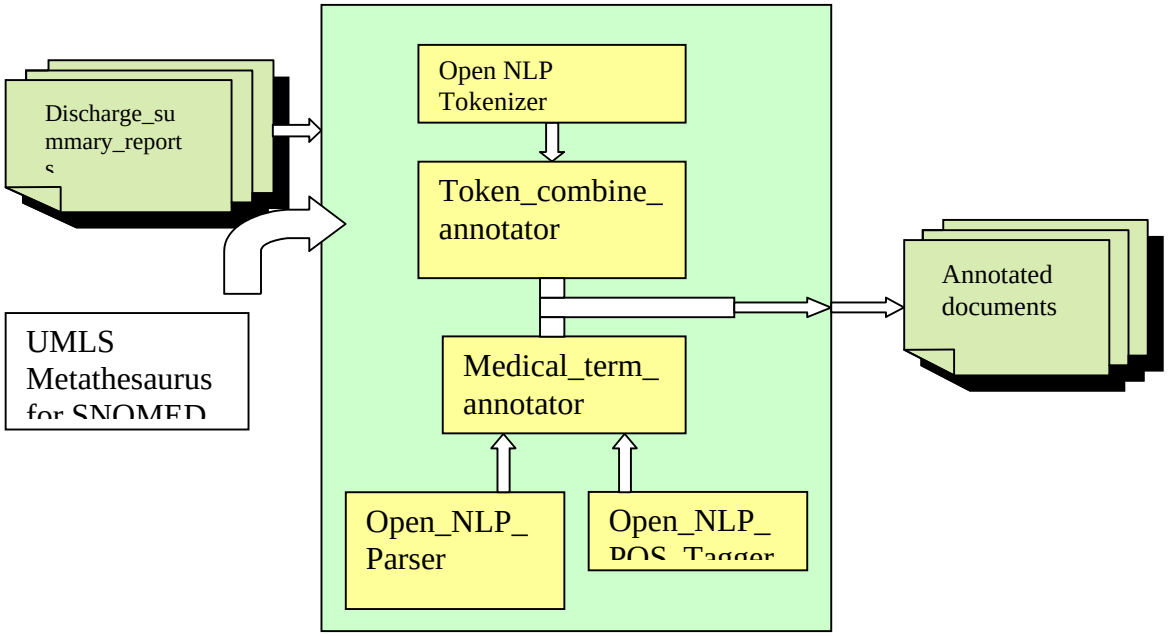


Fig 3: System Architecture of SNOMED Annotator for UIMA framework

Following is the description of each of the components:

3.2.1 Discharge Summary Reports

These form the input corpus that is fed to the system. The data corpus used is the 1000 discharge summary reports from WVU hospitals. These are text documents that have the patient information de-identified.

3.2.2 UMLS Metathesaurus for SNOMED

The medical ontology used in our method is the NLM's Unified Medical Language Systems coding. The three Knowledge Sources available with the UMLS are the Metathesaurus, the Semantic Network and the Specialist Lexicon. Our method is restricted to the usage of

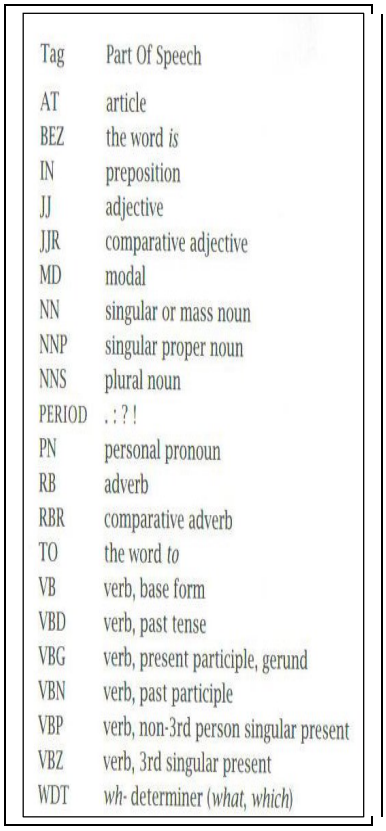
Metathesaurus knowledge Source. The UMLS Metathesaurus is a very large, multi-lingual and multi-purpose database that consists of terms from various source vocabularies (i.e. various thesauri, classifications and code sets). The Metathesaurus consists of information about biomedical and health related concepts with their corresponding names and relationships between various concepts. The installation of the UMLS Metathesaurus is done using the UMLS Metamorphosis wizard. The scope of the Metathesaurus that was installed is restricted to the SNOMED source vocabulary. MySQL database was used for storing the SNOMED database.

3.2.3 Open NLP Parser from IBM

This NLP parser is similar to the POS tagger, and is available as a open source solution from source forge.net. It parses a given document and creates phrasal and clausal annotations. It uses the opennlp.tools.lang.english.TreebankParser. The Open NLP parser analysis engine takes the Parse Tag mapping parameter which has the mapping for each parse tag identified by the Tree Bank Parser to the Java Common Analysis Structure s for the corresponding annotation. It requires the input from the Tokenizer and Sentence detector and also requires the model files for English parser stored in the given directory. Thus the parser creates a phrase for each phrase tag identified by the Tree Bank Parser. It also creates clauses in the similar way. But in our application we just used the phrase annotations.

3.2.4 Open NLP POS Tagger from IBM

This is open source nlp tool available from source forge.net. It detects the parts of speech in the given sentences. It assigns parts of speech to tokens using the model opennlp.tools.lang.english.PosTagger. For annotating documents with parts of speech it takes input from the open nlp tokenizer and the open nlp sentence detector annotators and assigns parts of speech to the posTag field of each Token annotation. The various parts of speech identified by this annotator are:



Tag	Part Of Speech
AT	article
BEZ	the word *is*
IN	preposition
JJ	adjective
JJR	comparative adjective
MD	modal
NN	singular or mass noun
NNP	singular proper noun
NNS	plural noun
PERIOD	. : ? !
PN	personal pronoun
RB	adverb
RBR	comparative adverb
TO	the word *to*
VB	verb, base form
VBD	verb, past tense
VBG	verb, present participle, gerund
VBN	verb, past participle
VBP	verb, non-3rd person singular present
VBZ	verb, 3rd singular present
WDT	*wh*- determiner (*what, which*)

Fig 4: Parts Of Speech identified by the SNOMED Annotator

(Ref: Foundations of Statistical Natural Language Processing By: Christopher D. Manning And Hinrich Schutze)

3.2.5 Medical Term annotator

This is the annotator whose analysis engine takes input from the open nlp POS tagger for detecting medical terms that just have single word and the open nlp Phrase detector for detecting medical terms that are made of more than one word. The medical term annotation has a feature named SNOMED whose value is populated with the corresponding SNOMED code for the token or the phrase detected by the open nlp tokenizer and open nlp parser. The SNOMED codes are searched for in the database populated from UMLS Metathesaurus. It has also an aggregate analysis engine which outputs the medical term annotations with the corresponding SNOMED codes assigned.

3.2.6 Token Combiner

The performance of the medical term annotator depends on the performance of the open nlp parser and open nlp pos tagger. This limits the number of medical terms that are annotated. To overcome this, the statistical n-gram model has been implemented. This primitive analysis engine of this annotator takes the tokens from the Tokenizer as input. These tokens are grouped in the following way:

The sentence:

Congestive heart failure, probably not quite compensated is split in the following way:

Congestive

Congestive heart

Congestive heart failure

Congestive heart failure probably

Heart

Heart failure

Heart failure probably

Heart failure probably not

The word splitting is done based on the assumption that most of the medical terms are made of maximum of 4 words. This avoids missing of any terms in the free text.

3.2.7 Annotated Documents

The output documents consist of medical term annotations that are annotated in two ways. The medical-term annotation has each identified medical term annotation assigned a SNOMED code if it exists to the SNOMED feature of the annotation. The token-combine annotation has the list of combined tokens and the corresponding SNOMED codes if it exists, concatenated together. The output documents are in xml format with each annotation forming separate tags.

3.3 Constraints

The number of medical terms identified by SNOMED annotator is limited by the following factors:

- The performance of the open nlp phrase detector which might not identify all the multi-word medical terms as they might not be identified as phrases.
- As the SNOMED annotator does not make use of neither Specialist Lexicon nor Semantic Network it misses some of the medical terms as they might not occur directly in the UMLS database and it s required to look for their synonyms or related terms.

3.4 Benefits

The facilities provided by the SNOMED annotator are summarized below:

- It allows automatic assigning of SNOMED codes for the identified medical terms.
- As it is developed on UIMA framework it entails all the advantages of UIMA framework like flexibility, scalability etc.
- Because of the usage of statistical natural language processing n-gram windowing technique it avoids missing of any multi-word medical terms that has SNOMED codes in the UMLS database.
- The SNOMED annotator can work even with other medical vocabularies like ICD9, CPT just by changing the database during the UMLS Metamorphosis installation.

4. IMPLEMENTATION

The application developed focuses on mapping SNOMED codes to clinical text using the UIMA framework. Following are the steps involved:

Step 1: Installing the UIMA SDK for eclipse.

Step 2: Installing the UMLS Metathesaurus for SNOMED using MySQL database.

Step 3: Using the Open NLP tool components as UIMA annotators:

Step 4: Developing the medical_term_annotator that takes the annotations from the opennlp annotator and maps them to the SNOMED codes that are stored in the My SQL database.

Step 5: Developing an aggregate analysis engine that gives the final annotated text documents which has the medical terms and the corresponding SNOMED codes.

Step 6: A comparative analysis of the results of the method developed with the results obtained from Language and Computing NLP engine.

4.1 UIMA SDK setup for Eclipse

The standard UIMA SDK v2.0 from alpha works has been used for this method. The Eclipse Modeling Framework (EMF) has been installed to make the UIMA Eclipse tooling work. This can be done by using the Software Updates feature of the eclipse SDK and selecting the EMF update. Also the Eclipse plug-ins that comes with the UIMA SDK are stored in the ECLIPSE_HOME directory. This completes the basic installation of the UIMA SDK for Eclipse and creates a platform for developing our application.

4.2 UMLS Metathesaurus setup for SNOMED codes into MySQL database:

The medical ontology used in our method is the NLM's Unified Medical Language Systems coding. The three Knowledge Sources available with the UMLS are the Metathesaurus, the Semantic Network and the Specialist Lexicon. Our method is restricted to the usage of Metathesaurus knowledge Source. The UMLS Metathesaurus is a very large, multi-lingual and multi purpose database that consists of terms from various source vocabularies (i.e. various thesauri, classifications, code sets etc.). The Metathesaurus consists of information about biomedical and health related concepts with their corresponding names and relationships between various concepts. The installation of the UMLS Metathesaurus is done using the UMLS Metamorphosis wizard. Following are the steps required during the UMLS Metathesaurus installation using the Metamorphosis installation wizard:

Metamorphosis helps in customizing the Metathesaurus for the current application. Customization of Metathesaurus is required for two main purposes:

- To exclude medical vocabularies that are not required in the output for the current application.
- To have a customized subset using a variety of data output filters and output options.

In the SNOMED annotator application the Metathesaurus is customized to include only the SNOMED medical vocabulary and also include only the Metathesaurus Knowledge Source of UMLS. Following are the steps for customizing the Metathesaurus as required by the local application:

- Installing UMLS: This include creation of a destination directory whose name is equal to version name of the UMLS (eg: 2007AC) with the following folders: LEX, NET and META. When the Specialist Lexicon is installed the files are stored under the LEX folder. When the Semantic Network is installed the files are stored under the NET folder. In the SNOMED annotator application as only the Metathesaurus is installed the only the folder META is populated with corresponding files.
- Configuring the Metathesaurus subset: In the SNOMED annotator application following options are taken care while configuring the Metathesaurus subset:

  Input options: This defines the directory where the configuration file and input and output directories are specified. As this is the initial installation the NLM data file format has been selected.

  Output Options: It defines the following

  Output Format: We can ether choose the Original Release Format or the Rich Release Format. As this is the initial installation the Rich release format is chosen.

  Subset Folder: It defines the folder where the new subset files should be stored.

  Write Database Load scripts: This is the provision for saving the entire code set in a database. Metathesaurus has provision for both Oracle and My SQL. For our current application we have chosen the My SQL load scripts as we have used the My SQL database for storing the SNOMED code set.

  Source List: This tab allows us to select the medical vocabularies required for the current application. There are two levels of selection Level 0: Consists of all the vocabularies whose license is free for use and Level 0+ SNOMED CT which has all the license free vocabularies and the SNOMED codes. For the SNOMED annotator application we have just chosen the SNOMED medical vocabulary.

Now as the Metathesaurus has been configured and as we have chosen the MySQL load scripts, the SQL files related to the SNOMED vocabulary are stored under the 2006AA directory. Now there SQL

scripts are run on the My SQL command window to create corresponding tables. Following are the tables that are populated for the SNOMED medical vocabulary:

Following tables signifies Concepts, Concept Names, and their sources:

MRCONSO.RRF: This table consists of exact one row for each atom of the database (i.e. each unique string or concept name within each source vocabulary).It has entries for unique identifier for the concept (CUI),language used, Term status, Unique identifier for term (LUI), type of string used, unique identifier for string, ISPREF which signifies if the atom status is preferred or not within this concept, unique identifier for atom, source asserted atom, concept or descriptor identifiers(these are optional),abbreviated source name, abbreviations for term types in source vocabulary, source identifier(this signifies the SNOMED code), String description of the code value etc.

Following tables signifies Attributes:

MRSAT.RRF: This table contains entries for the concepts, atom and relationship attributes theta do not have a sub element structure. This contains entries for those attributes that does not find entry in any other tables.

MRDEF.RRF: This table finds an entry for each definition in the Metathesaurus. It contains entries for concept, atom and attribute identifiers, source asserted attribute identifier, abbreviated source name, definition of the attribute, and flags for content viewing and suppressing the attribute status.

MRSTY.RRF: This table contains one row for each concept specifying the Semantic Type assigned to it. It find entries for concept identifier, semantic type identifier, semantic type tree number, the specific semantic type, attribute identifier, and the content view flag.

MRHIST.RRF: This file contains the history related to a particular source. The Metathesaurus currently include only the history of SNOMED-CT.

Following tables signifies Relationships:

MRREL.RRF: This table contains an entry for each relationship between concepts or atoms that are defined in the Metathesaurus. If there exists an asymmetrical relationship for a particular concept there exists one row for direction of the relationships. It finds entries for the unique identifiers of first concept and first atom and the second concept and second atom, unique identifier for the relationship, additional relationship label etc.

MRCOC.RRF: This file includes a list of cooccurences of meanings n external resources. There exist two rows for each cooccurrence. This is for each direction of cooccurrence.It finds entries for unique

identifiers of the first and second concepts and first and second atoms, type of co-occurrence, frequency and attribute of co-occurrence.

Following tables signifies data about the Metathesaurus

MRFILES.RRF: This table contains details about the files that are present in the database. It describes the filename, its description, comma separated column names, number of columns, number of rows, size of the file in bytes.

MRCOLS.RRF: This table describes about the columns in the database. Columns that exist in multiple tables (eg: CUI, AUI, LUI etc) will have multiple rows in this table. It contains the column name, its descriptive name, section number where the related documentation exists, minimum length of characters in the column, average length of characters, maximum length of characters, file name in which this column exist and the data type of the column.

MRDOC.RRF: This table has entries for each data element in the database (eg: CUI, AUI, etc).It also has attributes that have finite number of abbreviations allowed.(eg: ATN,TTY,TS etc).It contains the data element or attribute, abbreviation of one of the allowed values, type of information stored in the data element , explanation of the allowed values.

4.3 Using the Open NLP tool components as UIMA annotators:

The SNOMED Annotator application makes use of the open nlp tools package which is the open source project related to natural language processing. This package includes a sentence detector, tokenizer, parts of speech tagger, noun phrase chunker, shallow parser, named entity detector, and co-reference resolver. Together they prove to be a rich source for text analysis capabilities. We have used the Apache UIMA Example Wrappers that provides UIMA annotators for the Open Nlp Tool components.

Compilation of Open Nlp Wrappers for UIMA: Following are the steps for compiling open nlp to work with UIMA:

Downloading and Compiling Open nlp tools: This is downloaded from open nlp home page. The package consists of source code for Open NLP Tools, jar files required by Open NLP and an Ant build script. The compilation of the open nlp tools is done by running the Ant scripts.

Downloading Model files: The model files which are required for running the Open NLP Tool components are downloaded from the Models link of the Open NLP home page.

Compiling the UIMA wrappers for Open NLP: In order to add the open nlp wrappers to the UIMA SDK following procedure is followed:

- Import the uima_examples project from UIMASDK.

- Add the open nlp wrappers source directory to he uima_examples project.
- Add the open nlp jar files to the build path of uima_examples project. The jar files that should be imported from open nlp home page are: maxent-2.4.0.jar, trove. jar and opennlp-tools-1.3.0.jar

The two annotators used from this open nlp tool package are: Open NLP POS Tagger and Open NLP Parser. The models required for these annotators are downloaded from the open nlp site. An aggregate analysis engine of the above 2 annotators is formed and the output annotations of this engine are given as the input to the annotator that identifies the medical terms in the free text.

4.4 Annotating medical terms and assigning SNOMED codes

Annotation of medical terms and assigning corresponding SNOMED codes is done in three processes.

4.4.1 Process 1

Annotating medical terms and assigning corresponding SNOMED codes using Open NLP tools package. Following flow chart shows the work flow for the process:

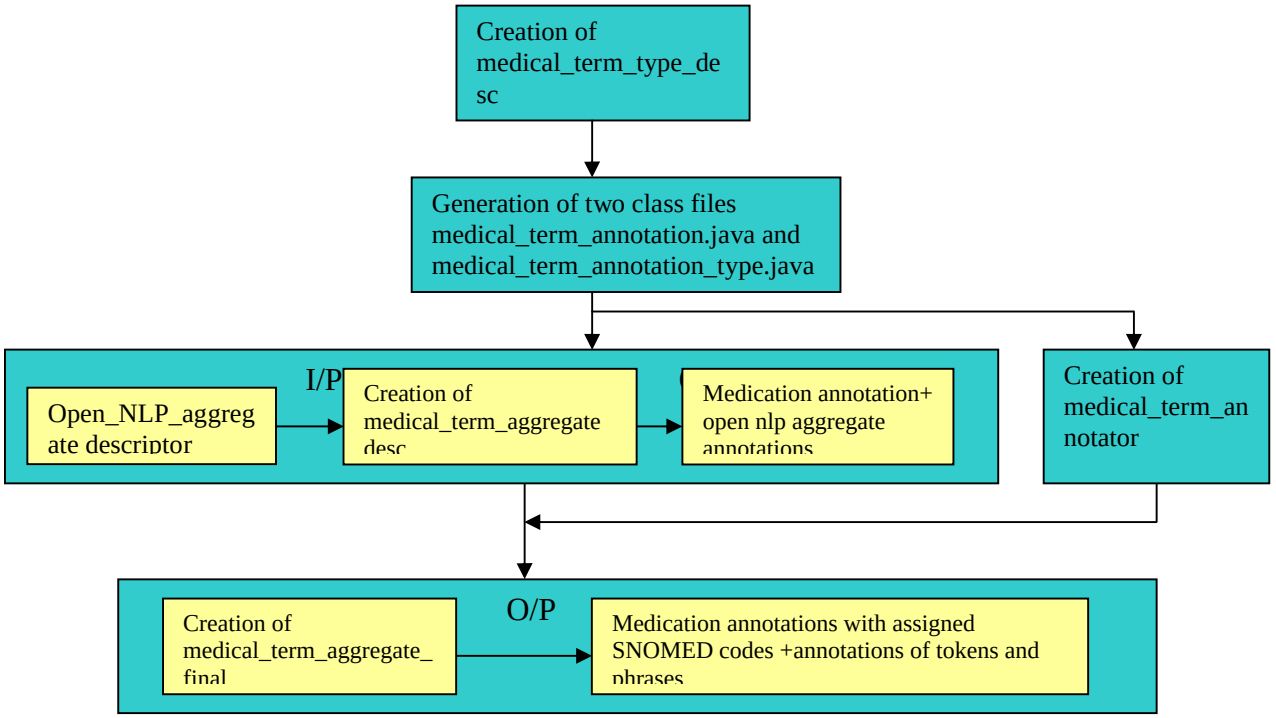
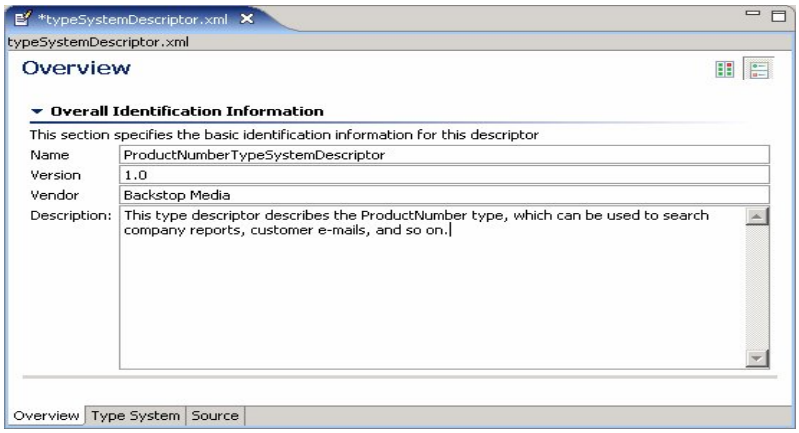


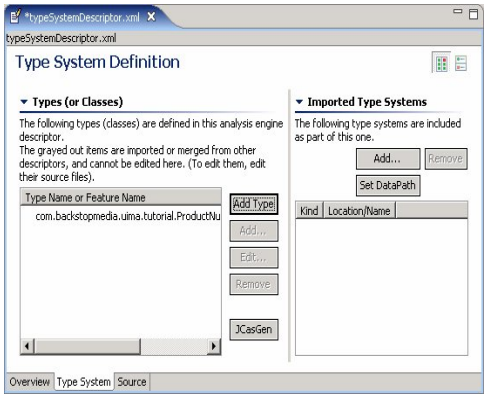
Fig 5: Workflow for Process1

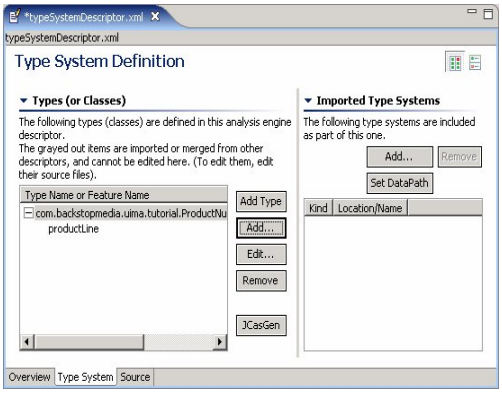
Step 1: Creation of the Type _descriptor medical_term_type_desc:

A type descriptor file defines the kind of data the application being developed is looking for. It allows us to gives a short description of the application. The screen shot is of this is shown below:



Now after the initial details about the application are updated, the type of data that is being annotated is given under the Type system tab. Also if for a particular annotation we need to have features assigned we can even add them under the Type system tab.





With this the creation of the type system descriptor for the given application is done. Here in our application the Type System added is medical_term which is of String type and feature added to this is SNOMED which is also a String Type. Hence our application annotates medical_term with the assigned SNOMED code as its feature.

Step2:

Generation of java classes medical_term_annoataion.java and medical_term_annoatation_type.java:

These two classes are automatically generated after creating type descriptor file. The medical_term_annotation.java class defines the basic methods ad indicates the beginning and

ending position of the annotation. The medical_term_annoatation_type.java calls methods that are internal to the UIMA framework.

Step3: Creation of the annotator medical_term_annotator.java:

In this annotator the CAS object of each of the documents after passing through the open nlp POS Tagger and the Open NLP phrase detector is given as input. Once the JCas is obtained an integrator is created over the indexes. For each iteration, following checks are made:

For each of the Token annotated it is checked if the Parts Of Speech tag feature value is either 'JJ'(Adjective),'NN'(Noun) ,'VB'(verb) ,'NNP' (Proper noun) etc. If it is so that word is searched for the SNOMED database. If the SNOMED code for a particular word is found a medical annotation is created and the corresponding SNOMED code is attached to the SNOMED_code feature of that annotation. If there exist more than one SNOMED code for a medical term all the SNOMED codes are concatenated.

In order to identify medical terms that might include more than one word the opennlp POS Tagger will not be able to identify the medical term. In order to overcome we have used the opennlp parser that identifies various kinds of phrases in a text document. An iterator is created over the index of each type of phrase (e.g. Noun Phrase, Adjective Phrase, Verb phrase, Adverb Phrase etc.) If a particular phrase is found in the SNOMED database the corresponding SNOMED code is assigned to the group of words that exist as a medical term. Thus this method enables identifying medical terms that exist as phrases. Similar as the above method if a medical term contains more than one SNOMED code, all the SNOMED codes are concatenated and assigned to the SNOMED code feature of the annotation.
Following figure shows the logic flow in medical_term_annotator:

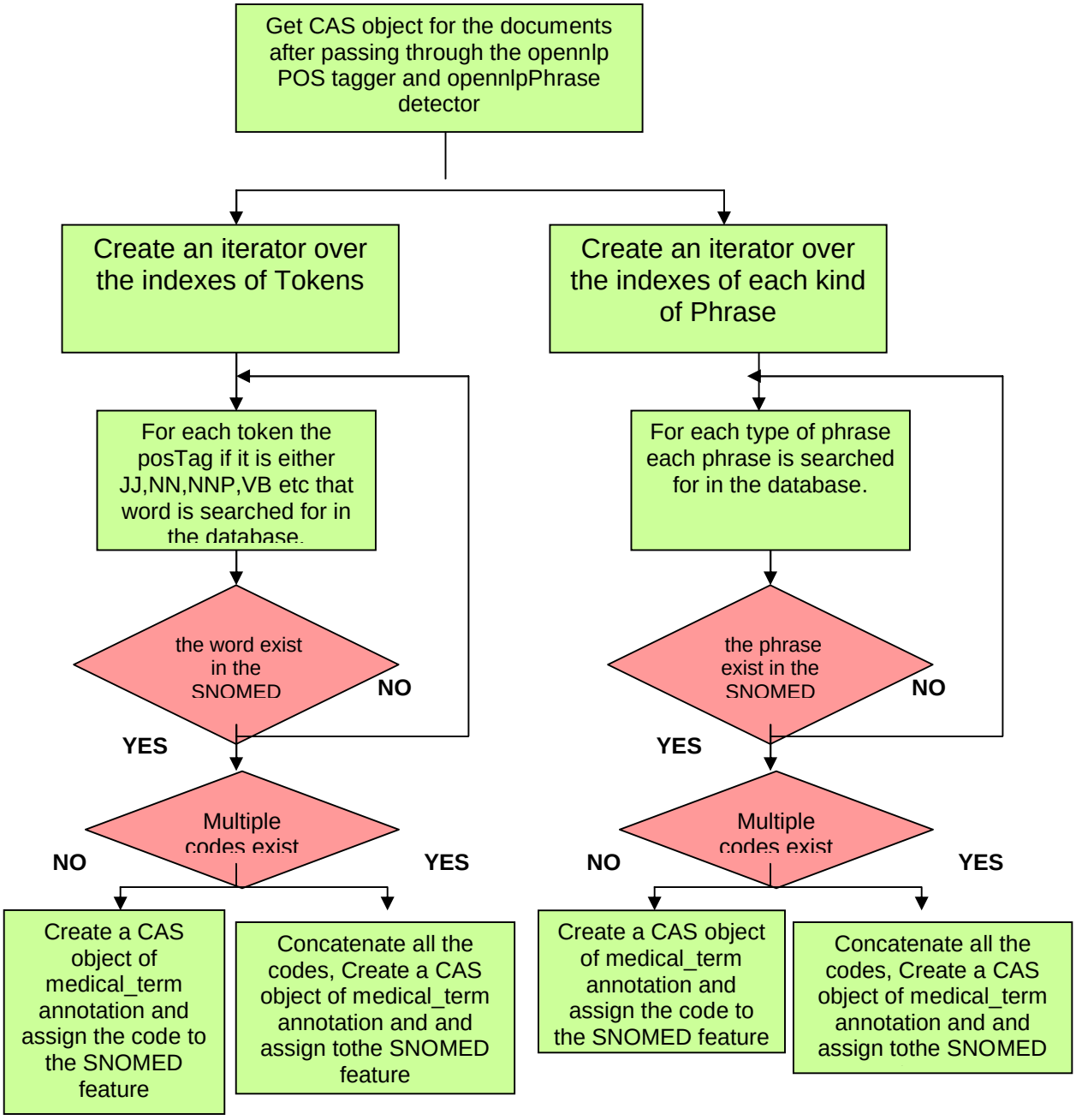
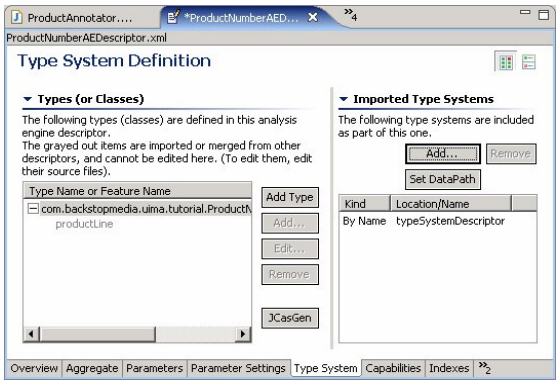


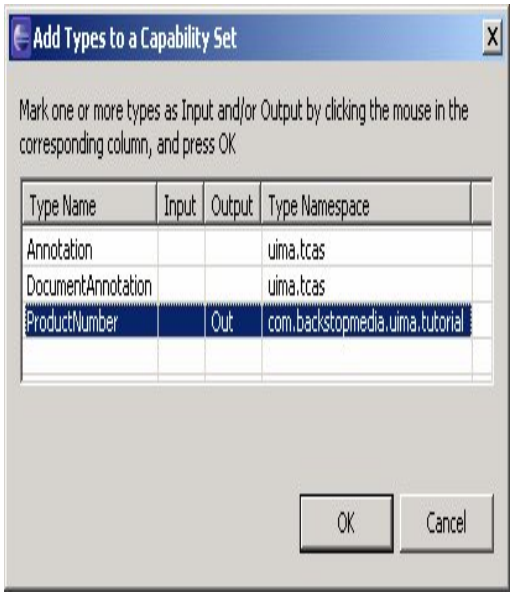
Fig 6: Workflow for creating medical_term annotator

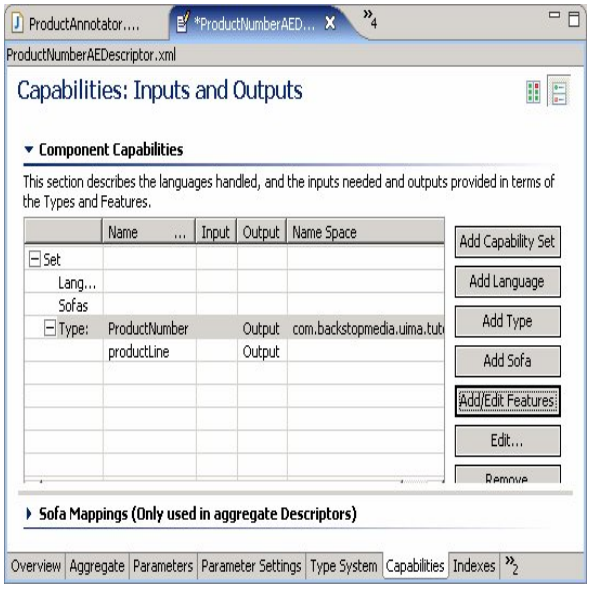
Step4: Creation of the primitive analysis engine medical_term_aggregate_desc.xml:

The major aspects in this step are selecting the analysis engine as primitive and assigning the corresponding annotator to the analysis engine. Here the annotator medical_term_annotator.java is assigned to the primitive analysis engine

medical_term_aggregate_desc.xml. The next aspect is assigning the Type system. It defines the type systems that should be included in this analysis engine. In our application the Open _NLP_Example_Types type descriptor and the medical_term_type.xml type descriptor are added to the Type system. The next aspect is assigning capabilities. Capabilities of an analysis engine define the inputs and outputs for the particular analysis engine. Our application takes the annotations from the open nlp POS Tagger and Open NLP Parser as input and outputs the medical annotations with the SNOMED codes assigned. The Type system assignment and the capabilities assignment are as shown below:





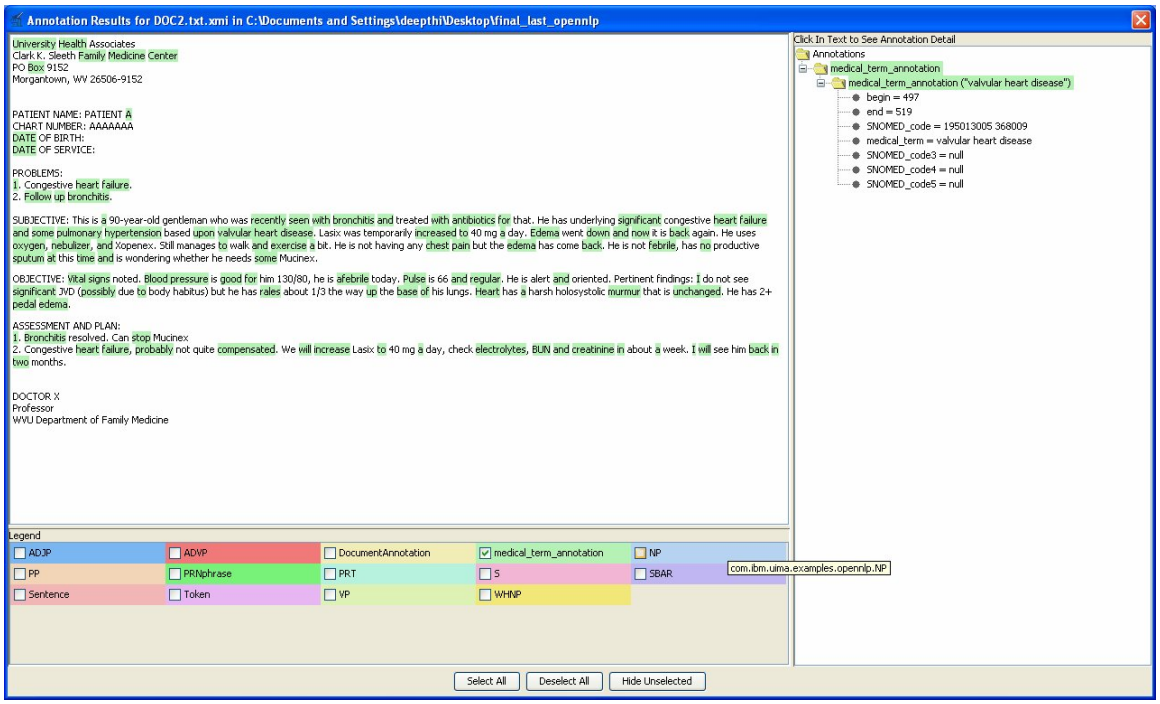


Step 5: Creation of the aggregate analysis engine medical_term_aggregate_final.xml:

Now as the medical_term_annotator identifies the medical terms and assigns the corresponding SNOMED codes in order to show the output an aggregate analysis engine is created which consists of the

opennlp_aggregate descriptor that is formed earlier and the medical_term descriptor. Here the capabilities include outputs of the medical_term annotation and the annotations of tokens and phrases annotated by the Open_NLP_aggregate descriptor.

Following figure shows the output of the medical_term annotator. The results are displayed using Java Viewer Style of Document Analyzer (a brief description of this is given below):



A typical output xml files is as shown below. As shown each annotation forms a separate tag and features of annotations becomes the attributes of the corresponding tags:

```
- <com.ibm.uima.examples.opennlp.NP sofa="Sofa" begin="218" end="245" componentId="OpenNLP
    Parser">
- <thesis_final.medical_term_annotation sofa="Sofa" begin="218" end="219"
    SNOMED_code="421379005" medical_term="1" SNOMED_code3="null" SNOMED_code4="null"
    SNOMED_code5="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="218" end="219" posTag="CD"
    componentId="OpenNLP Tokenizer">1</com.ibm.uima.examples.opennlp.Token>
   </thesis_final.medical_term_annotation>
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="219" end="220" posTag="."
    componentId="OpenNLP Tokenizer">.</com.ibm.uima.examples.opennlp.Token>
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="221" end="231" posTag="NNP"
    componentId="OpenNLP Tokenizer">Congestive</com.ibm.uima.examples.opennlp.Token>
- <thesis_final.medical_term_annotation sofa="Sofa" begin="232" end="237" SNOMED_code="U000438
    T-32000 80891009 302509004" medical_term="heart" SNOMED_code3="null" SNOMED_code4="null"
    SNOMED_code5="null">
```

```
<com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="232" end="237" posTag="NN"
    componentId="OpenNLP Tokenizer">heart</com.ibm.uima.examples.opennlp.Token>
  </thesis_final.medical_term_annotation>
- <thesis_final.medical_term_annotation sofa="Sofa" begin="238" end="245" SNOMED_code="F-04400
    76797004" medical_term="failure" SNOMED_code3="null" SNOMED_code4="null"
    SNOMED_code5="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="238" end="245" posTag="NN"
    componentId="OpenNLP Tokenizer">failure</com.ibm.uima.examples.opennlp.Token>
```

4.4.2 Process 2: Annotating medical terms and assigning SNOMED codes using Statistical n-gram windowing technique:

Following flow chart shows the work flow for process2:

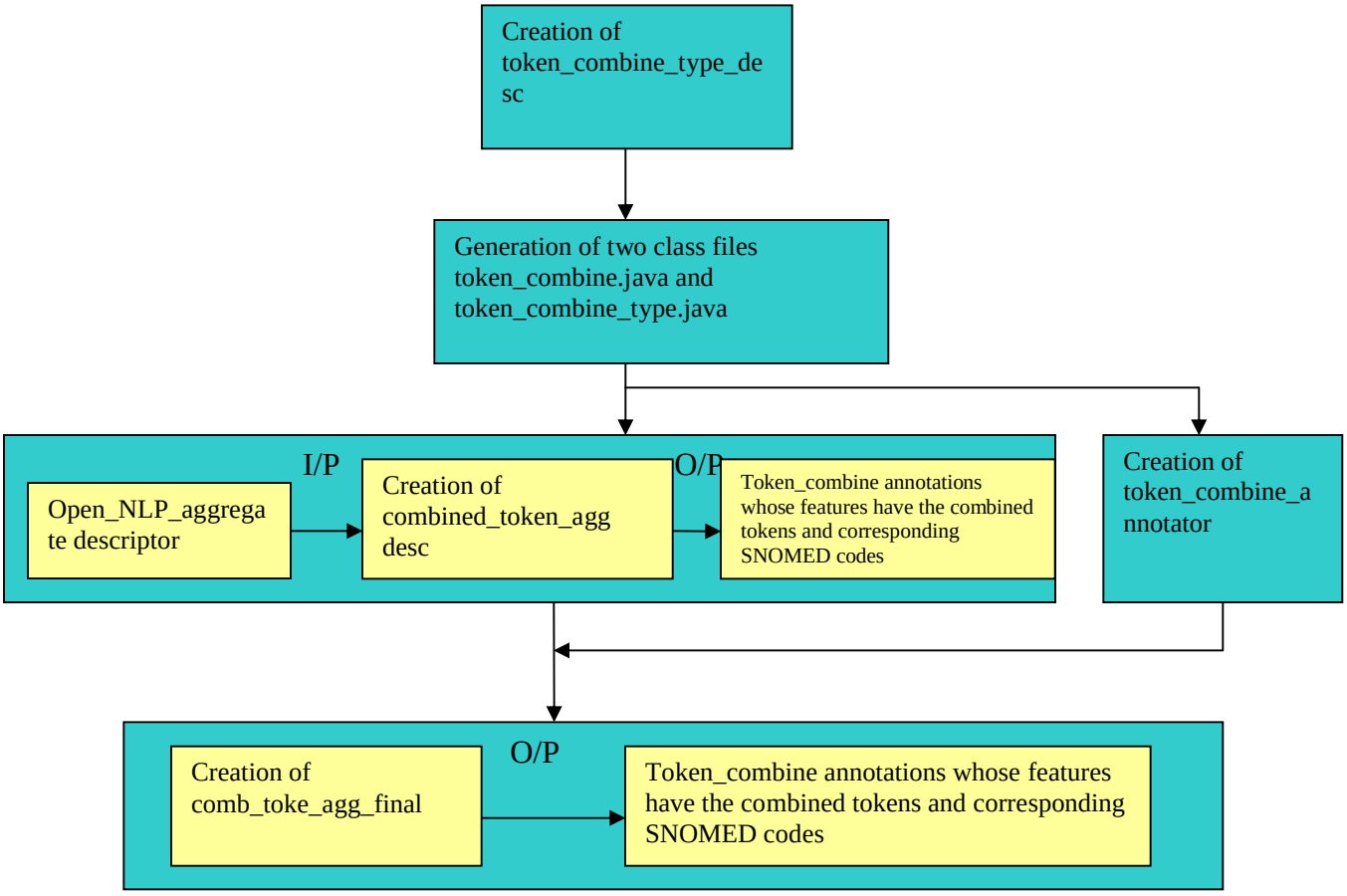


Fig 7: Workflow for Process2

Step1: Creation of token_combine_type_descriptor:

This type descriptor defines the token that is being annotated as String Type. It also adds four features for the annotation which are values of String data type.

Step2: Generation of Java classes token_combine.java and token_combine_type.java:

These two classes are automatically generated after creating type descriptor file. The token_combine.java class defines the basic methods ad indicates the beginning and ending position of the annotation. The token_combine_type.java calls methods that are internal to the UIMA framework.

Step3: Creation of token_combine_annoatator.java:

The technique used in this annotator is based on the statistical natural language processing n-gram windowing technique. This annotator takes the token CAS from the Open _NLP_ Tokenizer. For each of these token annotations there are assigned four features. The featureconcat1 contains the token itself. The feature concat2 contains the token concatenated with the immediate next token. Similarly the feature concat3 and concat4 contains the token + token immediate next two tokens and token+ token immediate next three tokens respectively. Also for each of these features the corresponding SNOMED code is searched for in the SNOMED database. If there exists a SNOMED code then this value is concatenated to the corresponding feature of the token. Following figure shows the logic flow in a combine token annotator:

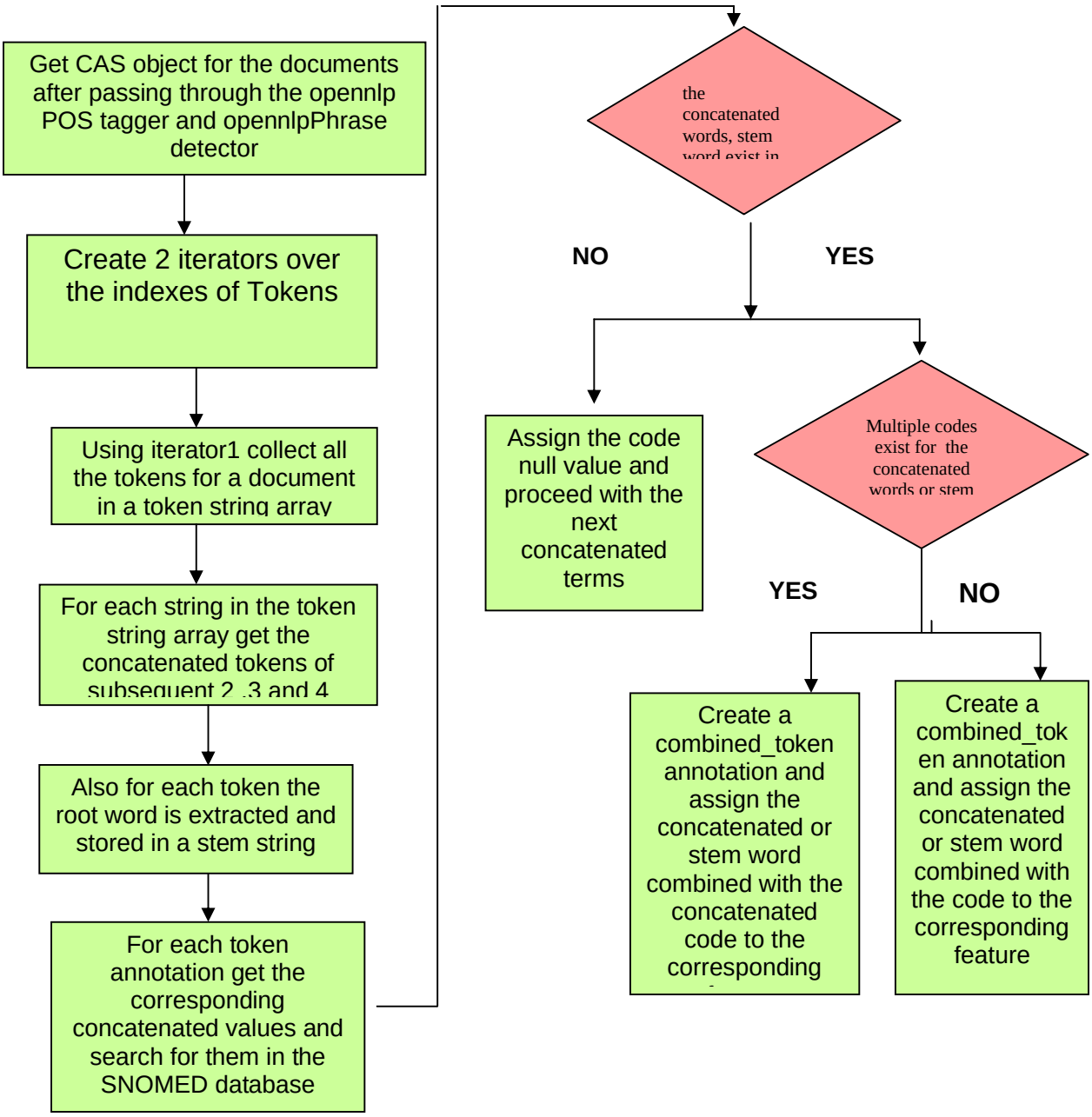


Fig 8: Workflow for creating token_combine annotator

Step4: Creation of the primitive analysis engine combined_token_agg.xml:

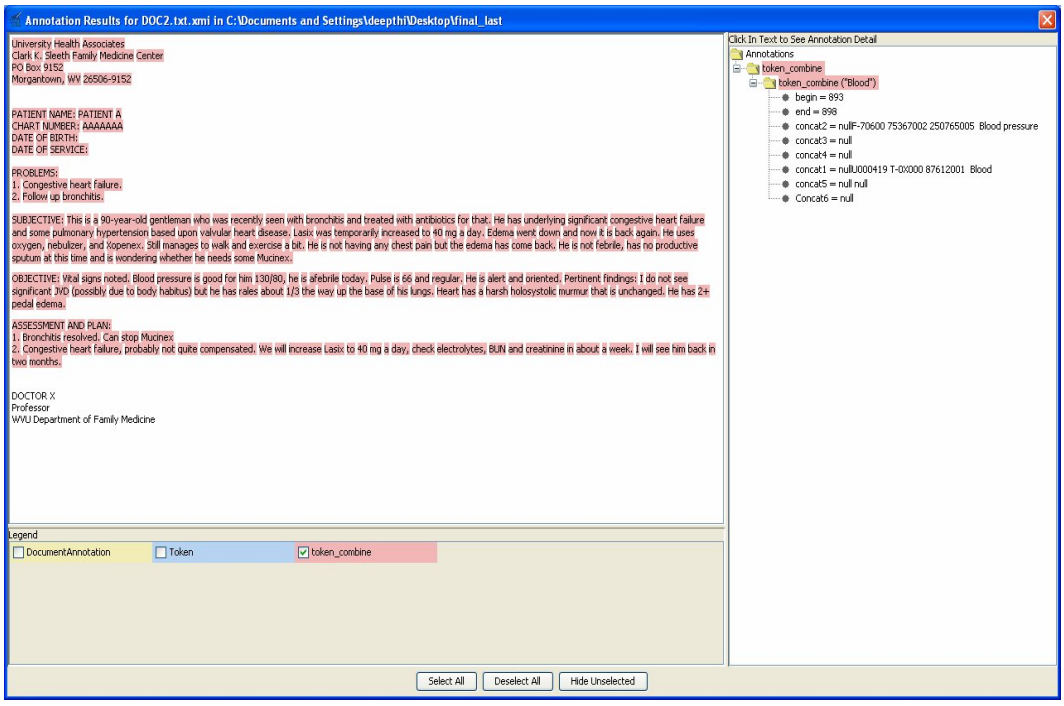
The token_combine_annoator.java is assigned to the primitive analysis engine combined_token_agg.xml. The type systems that are added to this analysis engine are the Open_NLP_Example_Types.xml and the token_combine_type_desc.xml.Under the capabilities the inputs are the token annotations and the output consists of the token_combine annotations

with the features containing the combined tokens concatenated with the corresponding SNOMED codes.

Step5: Creation of aggregate analysis engine comb_toke_agg_final.xml:

With the token_combine_annotator identifying the combine tokens and the corresponding SNOMED codes assigned in order to show the output an aggregate analysis engine is created which consists of the opennlp_aggregate descriptor that is formed earlier and the token_combine_type descriptor. Here the capabilities include outputs of the token_combine annotation and the annotations of tokens annotated by the Open_NLP_aggregate descriptor.

Following figure shows the output of the token_combine annotator. The results are displayed using Java Viewer Style of Document Analyzer(a brief description of this is given below):



A typical output xml files is as shown below. As shown here each token annotation form a separate tag while the combined tokens and their corresponding SNOMED codes becomes the attributes:

```
- <thesis_final.token_combine sofa="Sofa" begin="219" end="220" concat2="null" concat3="null"
    concat4="null" concat1="null" concat5="null null" Concat6="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="219" end="220" posTag="."
    componentId="OpenNLP Tokenizer">.</com.ibm.uima.examples.opennlp.Token>
  </thesis_final.token_combine>
```

```
- <thesis_final.token_combine sofa="Sofa" begin="221" end="231" concat2="null" concat3="nullD-7050
    D3-16010 195108009 42343007 Congestive heart failure" concat4="null" concat1="null"
    concat5="null null" Concat6="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="221" end="231" posTag="NNP"
    componentId="OpenNLP Tokenizer">Congestive</com.ibm.uima.examples.opennlp.Token>
  </thesis_final.token_combine>
- <thesis_final.token_combine sofa="Sofa" begin="232" end="237" concat2="nullD-7050 155374007
    155374007 84114007 Heart failure" concat3="null" concat4="null" concat1="nullU000438 T-32000
    80891009 302509004 Heart" concat5="null null" Concat6="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="232" end="237" posTag="NN"
    componentId="OpenNLP Tokenizer">heart</com.ibm.uima.examples.opennlp.Token>
  </thesis_final.token_combine>
- <thesis_final.token_combine sofa="Sofa" begin="238" end="245" concat2="null" concat3="null"
    concat4="null" concat1="nullF-04400 76797004 Failure" concat5="null null" Concat6="null">
  <com.ibm.uima.examples.opennlp.Token sofa="Sofa" begin="238" end="245" posTag="NN"
    componentId="OpenNLP Tokenizer">failure</com.ibm.uima.examples.opennlp.Token>
```


4.4.3 Process 3: Combining the analysis engines created in process 1 and process2 to create the final analysis engine:

Following figure gives a pictorial view of process3:

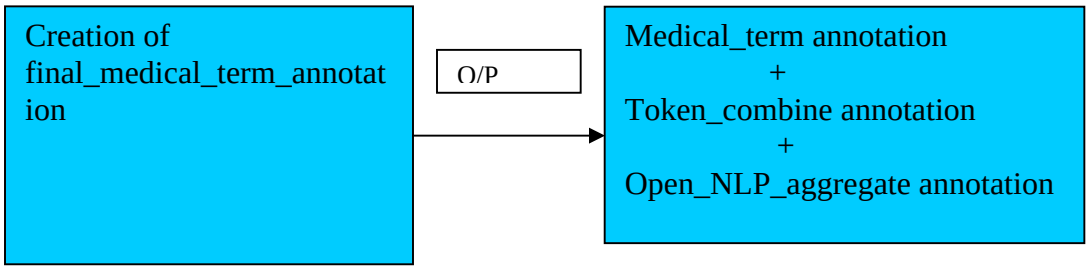


Fig 9: Workflow for Process3

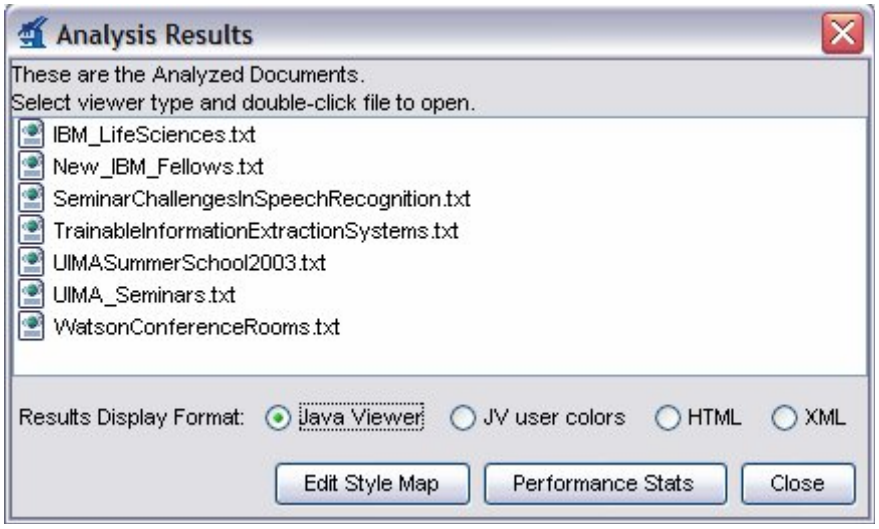
In this process an aggregate analysis engine is created. The type system of this analysis engine consists of the Open_NLP_Example_Types.xml, token_combine_type_desc.xml and medical_term_type_desc.xml. The capabilities consist of the outputs from the Open_NLP_aggregate annotations, medical_term annotations and token_combine annotations.

4.5 TOOLS FOR TESTING ANNOTATORS AND ANALYSIS ENGINES

4.5.1 DOCUMENT ANALYZER

This is the tool I have used in viewing the results on my SNOMED annotator. This has been provided by UIMA SDK. It reads the text files which are present in the folder specified as input process them using the specified analysis engine and enables viewing the results. This is designed to work only with text files as input typical interface of a document analyzer is as shown below:

As it can be observed it requires a input folder where the text files are located, and output folder where the output xml files are stored and the name of the analysis engine or the annotator which is to be run on the input files. It is also required to specify the language and the character encoding. Below figure shows a typical output viewing interface after running the Document Analyzer:



As observed the results can be viewed in 4 different formats: Java Viewer, JV user colors, HTML and XML. The JV user colors and HTML views are customizable views where the user can specify the colors they want to assign for the annotations. The Edit Style Map button enables this customization. All the result views that are shown below as outputs of process1, process2 and process 3 are Java Viewer views. There are other 2 modes of viewing the results:

Interactive Mode: In this mode the user can directly enter text and analyze it.

View Mode: This can be used to view any previously run results just by specifying the output folder where these files are saved.

5. ANALYSIS

This chapter discusses about the analysis of the annotation results produced by the SNOMED annotator. It also discusses the ways in which its performance can be enhanced in future.

5.1 Method for Analysis of Results

The results obtained by our method are compared with the output of the L&C NLP engine. The SNOMED code generated for each medical term identified in both L&C and our method is compared and precision and recall are calculated. Here the L&C results are considered as the gold standard. Following diagram illustrates the procedure:

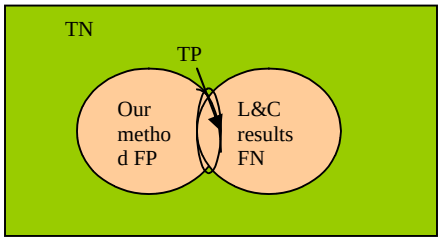


Fig 10: Venn diagram for results analysis

In this case False Positives (FP) are the number of terms that are not coded or not identified as a medical term in the L&C results but are coded by our method. False Negatives (FN) are the number of medical terms that are coded by the L&C but are either not coded or have different SNOMED code assigned. True Positives (TP) are the medical terms that are similarly coded by both out method and L&C.

Then the calculations for the following measures are done:

Precision: It gives the percentage of relevant terms that are retrieved to the total number of retrieved terms.

Precision= TP/(TP+FP)

Recall: It gives the percentage of relevant terms retrieved to the total number of relevant terms.

Recall/Sensitivity = TP/(TP+FN)

5.2 Extracting of medical terms and SNOMED codes from L&C data files:

The output files of L&C NLP engine are in the form of xml. In these files each term is not only assigned a SNOMED code but is also assigned other vocabulary codes like ICD9. For a given input xml file the required term which is the value under the <String> tag and the corresponding SNOMED code value which is the attribute value for the <code> tag are extracted using a XML Dom Parser. A typical L&C output xml file is as shown below:

```
<code coding-system="domain" id="4816500" kn="UNIVERSITIES" />
<code coding-system="domain" id="16143008" kn="UNIVERSITY" />
<code coding-system="SNCT CATEGORY" coding-system-version="none" id="23567354"
  kn="SNCT : 224871002 : UNIVERSITY (ENVIRONMENT)" level="0" mapping-
  linktype="HAS-CCC" />
 </token>
- <token begin="11" end="17" id="tok13898" number="1" type="word">
<string begin="11" end="17">Health</string>
- <id features="classified;common;mass-noun;no-temporal-qualifier;nonadjectival-
  noun;nonjussativised;nonpossessive-np;nontemporal-nouns;normal-noun;not-
  determined;not-formula-post-qualified;not-ordered;not-post-qualified;not-
  prequalified;not-quantified;not-relative-determined;not-slash-
  postqualified;noun;participant-nominal;singular-noun;singular-
  nouns;unbraced;unit;without-reflexive-pronoun;word" root="health">
<infl_form features="common;mass-noun;normal-noun;noun;plural-
  common">healths</infl_form>
 </id>
```

5.3 Results

The results of L&C and SNOMED annotator are stored in an Access database and the following values are calculated:

True positives: There are the number of terms annotated by both L&C and SNOMED annotator applications.

Total number of true positives (TP)= 392

False Positives: These are the number of terms annotated by SNOMED annotator but are not annotated by L&C NLP engine.

Total number of False positives (FP)= 143

False negatives: These are the number of terms annotated by L&C NLP engine but not by SNOMED annotator.

Total number of False negatives (FN): 246

The final values are:

Precision = 81.1% 81 percent of the medical terms are relevant among the total number of retrieved medical terms..

Recall = 65% 65 percent of the medical terms are relevant among the total number of relevant medical terms.

5.4 Result Analysis

Below figure shows reasons with examples for medical terms not identified by SNOMED annotator but found by L&C:

medical_term	code	Corresponding string assigned in SNOMED database	Analysis
a little bit	129264002	Action , Clinicalaction,Clinicalactions,Action(qualifier value)	
	255507004	Small,Little,Small(qualifier value)	the related term (synonymous) is analyzed and searched for in the database and the code is assigned,
	255606000	Minor, Minor(qualifier value)	
Associates	20909006	Combine ,CombineNOS,Combine,device(physical object)	
	285231000	Mental function ,Mentalfunctions,Mentalfunction(observable entity),	the related term (synonymous) is analyzed and searched for in the database and the code is assigned,
	89780004	Combined,Combination,Combined(qualifier value)	
Bacitracin ointment	312413002	Substancecategorizedstructurally, Substancecategorisedstructurally,	

		Substance categorized	
	385101003	Ointment, Ointment(product)	For the combination of terms, if the term is directly not found in the database, the single terms are searched for in the database and assigned corresponding SNOMED codes.
	417519007	Ointment, Ointment(substance)	
	5220000	Bacitracin,Bacitracin,NOS,Bacitracin(substance)	
be	256871003	Ilealsegment,IS-Ilealsegment,Ilealsegment	Unknown assignment, these terms does not exist directly in the database
became	385652002	Started, Started(qualifier value)	Unknown assignment, these terms does not exist directly in the database
basic	57195005	Basal, Basal(qualifier value)	Unknown assignment, these terms does not exist directly in the database
basilar	57195005	Basal, Basal(qualifier value)	
BUN	105011006	Bloodureanitrogenmeasurement,BUNmeasurement,Bloodurea	abbreviations are not handled

5.5 Improvements:

- Usage of Specialist Lexicon and Semantic Network helps in identifying synonyms of given terms if the term does not exist directly in the database.
- Improved stemming process helps in annotating more number of medical terms as most of the terms in the database exist in the root form, or in any other form of the root word.
- Application of more NLP techniques would enable assignment of the exact SNOMED code that is true to the context instead of assigning all the related SNOMED codes.

5.6 Conclusion:

Compatibility with the UIMA framework which is the main goal of the SNOMED annotator application has been achieved. The application is extendible to any other controlled medical vocabulary available through UMLS. It is scalable and can be applied to any number of input documents.

Identifying medical terms made of more than one word which is the major issue to be addressed have been solved using the opennlp phrase detector and using the statistical n-gram nlp technique. For words which does not find an entry in the database but rather exist in their root form the stemming technique has been applied. Because of the system capacity constraints the SNOMED annotator could not include the UMLS Specialist Lexicon which would have enables improved performance by identifying synonyms, related terms, abbreviations etc for the medical terms that do not find a direct entry in the database.

References

1. **Desiderata for Controlled Medical Vocabularies in the Twenty-First Century**
James J. Cimino *Department of Medical Informatics, Columbia University, New York, USA*

2.http://med.dmi.columbia.edu/vocab.htm

3. www.snomed.org

4. http://en.wikipedia.org/wiki/SNOMED_CT

5. http://www.connectingforhealth.nhs.uk/systemsandservices/data/snomed

6. http://www.lisperati.com/tellstuff/umls.html

7. http://www.nlm.nih.gov

8. http://uima.lti.cs.cmu.edu:8080/UCR/pages/static/osnlp/OpenNLPReadme.html

9. http://en.wikipedia.org/wiki/Medical_informatics

10. An Applied Evaluation of SNOMED CT as a Clinical Vocabulary for the Computerized Diagnosis and Problem List Henry Wasserman, MS and Jerome Wang, MD, FAAP