11-792 Software Engineering
EMR Project Report

Team Members

Phani Gadde
Anika Gupta
Ting-Hao (Kenneth) Huang
Chetan Thayur
Suyoun Kim

Vision

Our aim is to build an intelligent system which is capable of identifying disease codes from the **E**lectronic **M**edical **R**ecord (EMR) document with little or no human supervision.

Motivation

An EMR can be described as a well recorded log of all the procedure that is followed when a patient is diagnosed with a disease. These documents are later used by insurance companies to come up with a list of diseases (codes) they need to pay the hospital for the patient. Currently the EMR coding is performed by humans. Besides being time consuming, it is likely to have errors in the task. The task of mapping EMR Documents to disease codes might vary from human to human, and might be inconsistent. Our aim is to automate the task of EMR coding so that it may assist the humans to accurately and efficiently extract disease codes from the document.

Requirements

- A system that takes a raw EMR as input and gives disease codes as output
- A system design for the above task that's simple and generic enough to be extended when more resources are at hand

Proposed Solution

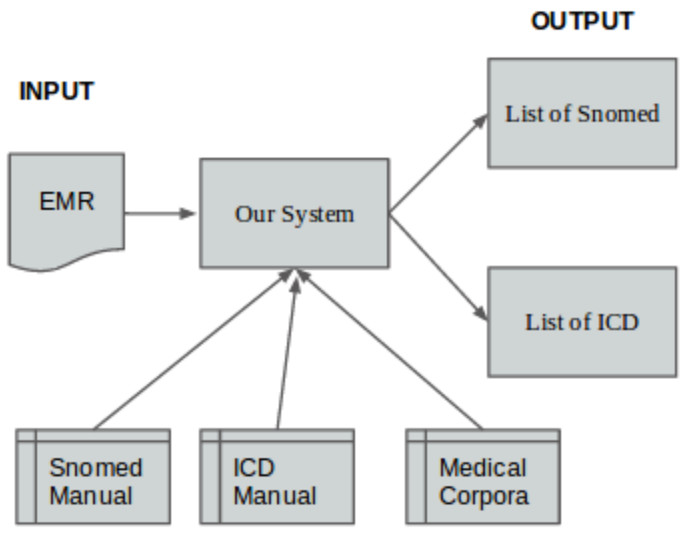
1. **Goal**
   We have tried to tackle the problem by working on two subproblems.
   - Text to Snomed[1] Mapping
   - Snomed to ICD[2] Mapping
2. **Target Market**
   Our system is aimed at assisting the health industry by providing them a means to select the list of relevant ICD Codes or automatically generate the ICD Codes.
3. **Research Question**
   Our hypothesis is whether automation of EMR Coding task can be improved to make it comparable to the task performed by humans.
4. **Risk**
   - Domain Knowledge: We, as a team did not have a background with health industry and did not know about the current procedure of converting from EMR Documents to disease codes which is employed by other health care centres.
   - Lack of Gold Standard Data: Annotating EMRs with gold-standard codes requires expert knowledge of the diseases and might be time consuming
   - Data is private and not available - Our analysis is based on very few documents due to the fact that the EMR documents are private, and mostly it is not allowed to be shared. Since our results are based on a very small dataset, they are not statistically significant, although the motivation for our approaches has partly has come from the sample documents.

System Overview

Design

[1] **S**ystematized **No**menclature Of **Med**icine **C**linical **T**erms
[2] **I**nternational **C**lassification of **D**iseases

The following diagram gives an overview of our system. Our system takes as input an EMR Document. By using the Snomed Manual, the ICD Manual and the Medical Corpora as resources it performs the task of identifying and extracting Snomed and ICD Codes. The system outputs a list of Snomed and ICD Codes.

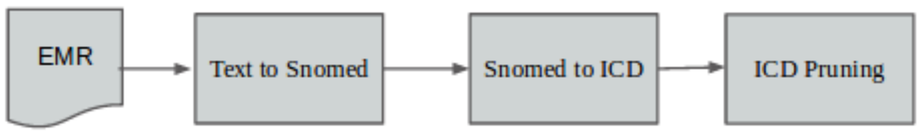


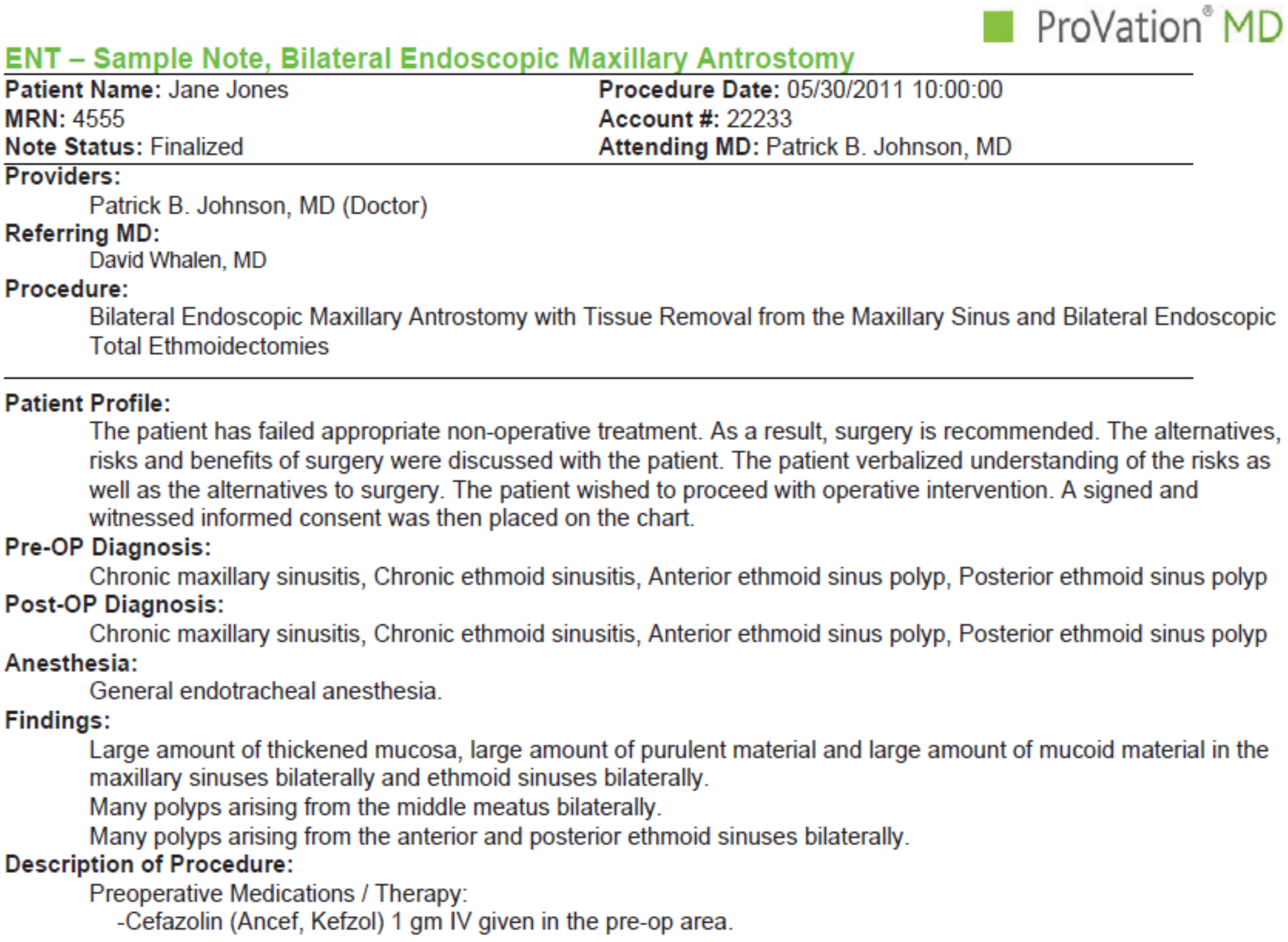
Component

The major components of our system are as follows

1. Processing Unstructured Documents
2. Text to Snomed Mapping
3. Snomed to ICD Mapping
4. ICD Pruning

The following diagram shows the pipeline of our system. First, we process the unstructured EMR Document into pieces of information which is important to our task. We then extract the Snomed Codes from the text. Using the rules we map the Snomed codes to ICD Codes. The ICD Codes are further pruned to improve the performance of our system.





ProVation® MD

ENT – Sample Note, Bilateral Endoscopic Maxillary Antrostomy

Patient Name: Jane Jones	**Procedure Date:** 05/30/2011 10:00:00
MRN: 4555	**Account #:** 22233
Note Status: Finalized	**Attending MD:** Patrick B. Johnson, MD

Providers:
Patrick B. Johnson, MD (Doctor)
Referring MD:
David Whalen, MD
Procedure:
Bilateral Endoscopic Maxillary Antrostomy with Tissue Removal from the Maxillary Sinus and Bilateral Endoscopic Total Ethmoidectomies

Patient Profile:
The patient has failed appropriate non-operative treatment. As a result, surgery is recommended. The alternatives, risks and benefits of surgery were discussed with the patient. The patient verbalized understanding of the risks as well as the alternatives to surgery. The patient wished to proceed with operative intervention. A signed and witnessed informed consent was then placed on the chart.
Pre-OP Diagnosis:
Chronic maxillary sinusitis, Chronic ethmoid sinusitis, Anterior ethmoid sinus polyp, Posterior ethmoid sinus polyp
Post-OP Diagnosis:
Chronic maxillary sinusitis, Chronic ethmoid sinusitis, Anterior ethmoid sinus polyp, Posterior ethmoid sinus polyp
Anesthesia:
General endotracheal anesthesia.
Findings:
Large amount of thickened mucosa, large amount of purulent material and large amount of mucoid material in the maxillary sinuses bilaterally and ethmoid sinuses bilaterally.
Many polyps arising from the middle meatus bilaterally.
Many polyps arising from the anterior and posterior ethmoid sinuses bilaterally.
Description of Procedure:
Preoperative Medications / Therapy:
-Cefazolin (Ancef, Kefzol) 1 gm IV given in the pre-op area.

Processing unstructured EMR document:
The above diagram shows a typical EMR document.

In general, each EMR document describes the administrative details of the patient, details of the doctors supervising the patient and an elaborate procedure of the patient's profile, the symptoms, the methodologies used, the description of the procedure and the findings of the procedure.
It is interesting to note that the layout and structure of an EMR document varies with each medical representative. Even the style of the sentences and the words used to describe the condition of a patient is different. Therefore, the first step we take is to extract the relevant information from this highly unstructured text.
Broadly, we carry out two main tasks.

- Extracting Context from Sentence
  In order to achieve our task of detecting disease codes from the text it is important to provide context for identifying codes. By limiting the context to each sentence for finding disease codes

and matching, makes the SNOMED codes more coherent.

- Extracting Information
  From the original unstructured text, our aim is to extract pieces of information which would make the system generic across EMR document and would also let the model focus on important information. We identify sentences, word tokens annotated with their POS Tag and a 0-1 value indicating whether it is a Named Entity or not. We also extract and add the personal attributes of a patient such as the age and gender which would be helpful in identifying the disease codes.
- Expanding Noun Phrases
  Different medical experts use different style to indicate the same condition of a patient. In order to efficiently perform the extraction of Snomed from Text, we try to expand our search by finding the synonyms of the noun phrases. We use the Expanded Noun Phrases to identify more Snomed codes and provide additional context besides that provided by the original sentence. The following are some examples of Noun Phrase Expansion.

Noun Phrase	Expanded Noun Phrase
Cavity	Dental Caries
Wheal	Urticaria
Pus	Suppuration
Clips	Surgical Instruments

The following are the technical details along with the models used for converting the unstructured EMR Document into structured pieces of information .

We used the Medline Sentence Model provided by LingPipe[3] which is a heuristic sentence model designed for operating over biomedical research abstracts. We used the tokenizer and parser of Stanford CoreNLP[4] to break the sentences into tokens and annotate the tokens with their POS Tags. We used the Named Entity Recognizer of Lingpipe and Abner[5] to identify the Named Entities on the sentence level. A simple regular expression ([0-9]+?)[|-]year[|-]old (.*)ale is used to identify the age and the gender of the patient. This works well on the 14 examples provided to us.

In order to find the synonyms of medical terms in the context of medical data, we use the Resource Wrappers[6] for the NCBI corpus[7], the Gene Ontology Database[87] and the Medical Subject Heading Dataset[9] to find the technical terms associated with the EMR document.

[3] http://alias-i.com/lingpipe/
[4] http://nlp.stanford.edu/software/corenlp.shtml
[5] http://pages.cs.wisc.edu/~bsettles/abner/
[6] http://mu.lti.cs.cmu.edu:8081/nexus/content/repositories/releases/edu/cmu/lti/oaqa/bio/annotate/resource-wrapper/
[7] http://www.ncbi.nlm.nih.gov/
[8] http://www.geneontology.org/
[9] http://www.nlm.nih.gov/mesh/

The following Table shows the importance of introducing Expanded Noun Phrases into the System by evaluating the contribution of Expanded Noun Phrases on the ICD to Description document. We see that Expanded Noun Phrases matches more with the ICD description as compared to the Noun Phrases of the original EMR document. However, the precision decreases due to the overgeneration caused by the number of Expanded NP.

Method	Average # of Matched per EMR Document	Average Precision
Noun Phrases	53.28	0.5
Expanded Noun Phrases	149.5	0.26

Our system generate two types of sentences. The first ones Sentence are the original sentence segmented from the original EMR document, and the second ones ExpandedSentence are the sentences formed by replacing the original noun phrase in the sentence by the expanded noun phrase. Both types of sentences are given to the ContextRuleAnnotator which takes in the context as a sentence. It is ensured that the correct start and end of sentence position in the original text are added to the CAS. The Expanded Noun Phrases are further used to match with the description of the Snomed to generate more Snomed Codes.

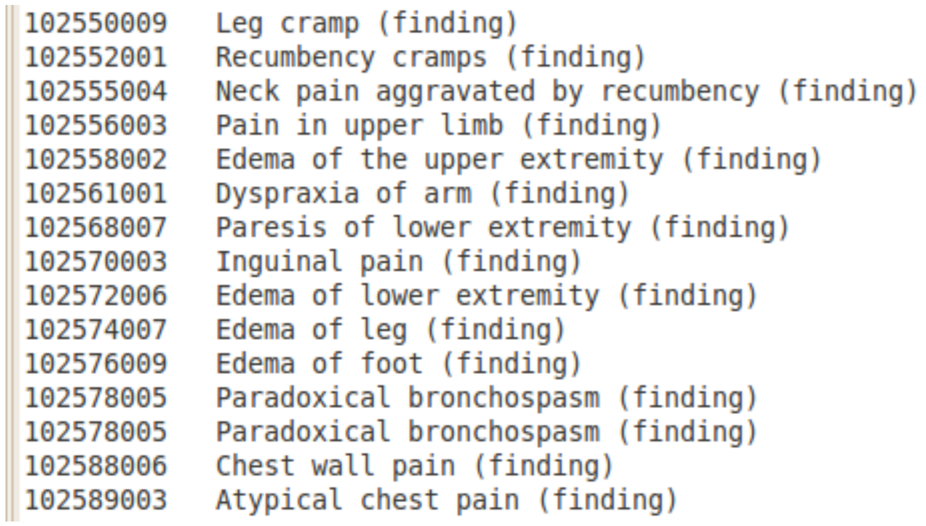
Text to SNOMED conversion

Given the EMR document with the basic text processing done to identify sentences, useful phrases and synonyms, the task is to map the text to **S**ystematized **No**menclature Of **Med**icine Clinical Terms (SNOMED) codes. SNOMED codes were designed for effective clinical recording and to improve patient care. It is a system that standardized the disease names so that the same terms are used to describe events in medical histories. The Text-to-SNOMED module takes text as input and produces SNOMED codes as output.

From a Natural Language Processing (NLP) perspective, this is a typical classification task with text as features. Since we don’t have enough resources to train a statistical model, we resort to a rule based model in our work. Moreover, since we are still exploring various possibilities of assigning ICD codes to EMR documents, it makes sense to start with a simple setup without any assumptions.

Any rule based technique needs a source from which we can mine the rules. We are able to get such a resource from the US National library of medicine[10]. The resource contains SNOMED codes, their descriptions, the corresponding ICD codes and the ICD descriptions. For the Text-to-SNOMED module, we used the SNOMED codes and the corresponding descriptions as our resource to come up with the rule based technique. The figure below shows a snapshot of the resource.

[10] http://www.nlm.nih.gov/research/umls/mapping_projects/snomedct_to_icd10cm.html



102550009	Leg cramp (finding)
102552001	Recumbency cramps (finding)
102555004	Neck pain aggravated by recumbency (finding)
102556003	Pain in upper limb (finding)
102558002	Edema of the upper extremity (finding)
102561001	Dyspraxia of arm (finding)
102568007	Paresis of lower extremity (finding)
102570003	Inguinal pain (finding)
102572006	Edema of lower extremity (finding)
102574007	Edema of leg (finding)
102576009	Edema of foot (finding)
102578005	Paradoxical bronchospasm (finding)
102578005	Paradoxical bronchospasm (finding)
102588006	Chest wall pain (finding)
102589003	Atypical chest pain (finding)

Figure: A snapshot of the resource with SNOMED codes and descriptions

Rule-based Techniques for Text-to-SNOMED conversion:

We started with a basic technique of match the text with the SNOMED description on the right and fire the SNOMED code when there is a match. In this method, we briefly take the following steps:

- Tokenize the EMR document
- Iterate through each snomed description
- Create a pattern from the description and match the entire text with the pattern

Fst based skip-match Technique:

However, we realized that the exact string match technique explained in the previous section highly constraints the matches, as the diseases are not written in the same way by everyone. One way of relaxing exact match is to do a skip-match, where we match phrases of words even though some words in between are missing. For efficiency, we implement this idea using a finite state transducer with words on the edges that is built from the SNOMED descriptions and used to query a sequence of words. The sequence of words in our case is the EMR document.

A general pattern we see with these two techniques is that the recall for SNOMED identification increases compared to the exact match when using skip-match, but the precision drops, as we might give more noisy matches.

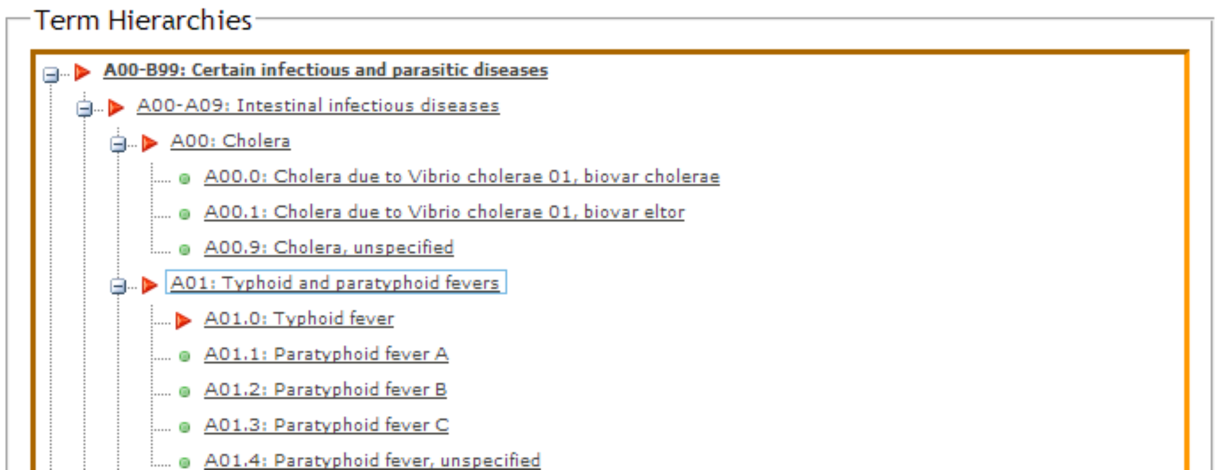
SNOMED to ICD conversion

The second step of the pipeline is trying to identify ICD 10 codes from the already generated SNOMED codes.

What are ICD codes?

- ICD 10 : International Classification of Diseases
    - international standard for diagnostic classification
    - over 141,000 codes available

ICD 10 Code Hierarchy:


Term Hierarchies

- A00-B99: Certain infectious and parasitic diseases
    - A00-A09: Intestinal infectious diseases
        - A00: Cholera
            - A00.0: Cholera due to Vibrio cholerae 01, biovar cholerae
            - A00.1: Cholera due to Vibrio cholerae 01, biovar eltor
            - A00.9: Cholera, unspecified
        - A01: Typhoid and paratyphoid fevers
            - A01.0: Typhoid fever
            - A01.1: Paratyphoid fever A
            - A01.2: Paratyphoid fever B
            - A01.3: Paratyphoid fever C
            - A01.4: Paratyphoid fever, unspecified

As we can see in the above image, ICD 10 codes have organized in a hierarchy. Unlike SNOMED codes, which are numeric, ICD codes are alphanumeric. They start with a letter followed by numbers, a decimal and more numbers. An ICD code without a decimal represents a general description of a disease or a condition. An ICD code containing a decimal corresponds to a specific disease based on the generalization.
For e.g. ICD code ‘A00’ refers to a generalization of the disease ‘Cholera’ and the ICD code ‘A00.0’ corresponds to the specific disease ‘Cholera due to Vibrio choleae 01, biovar cholerae’.

SNOMED to ICD conversion approach:

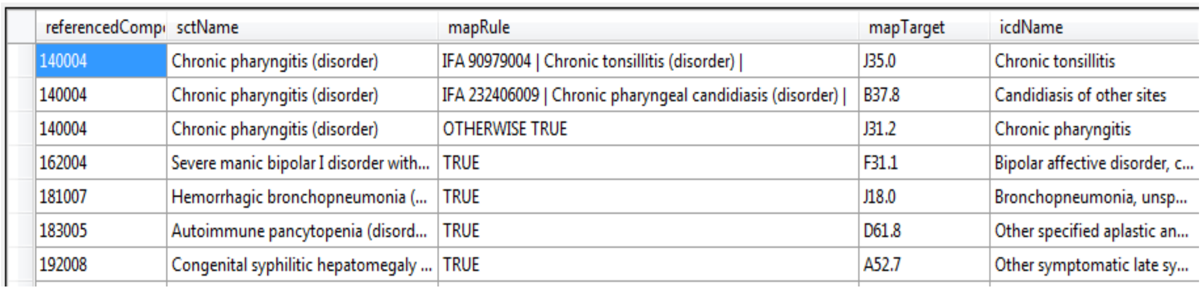
An initial design decision to make was how to map SNOMED to ICD’s. The decision was taken to employ a ‘rule based technique’ instead of using machine learning..

Rule Based SNOMED to ICD Mapping:

In order to map SNOMED to ICD’s we had to obtain a dataset that contained these accurate relations. Hence we used a map provided by the US National library of medicine[11].

This data set contained mappings for over 27,000 ICD 10 codes. The dataset was obtained in the form of a tab separated file. After removing all the unnecessary columns, the dataset that we worked with resembled the image below:

[11] http://www.nlm.nih.gov/research/umls/mapping_projects/snomedct_to_icd10cm.html



referencedComp	sctName	mapRule	mapTarget	icdName
140004	Chronic pharyngitis (disorder)	IFA 90979004 | Chronic tonsillitis (disorder) |	J35.0	Chronic tonsillitis
140004	Chronic pharyngitis (disorder)	IFA 232406009 | Chronic pharyngeal candidiasis (disorder) |	B37.8	Candidiasis of other sites
140004	Chronic pharyngitis (disorder)	OTHERWISE TRUE	J31.2	Chronic pharyngitis
162004	Severe manic bipolar I disorder with...	TRUE	F31.1	Bipolar affective disorder, c...
181007	Hemorrhagic bronchopneumonia (...	TRUE	J18.0	Bronchopneumonia, unsp...
183005	Autoimmune pancytopenia (disord...	TRUE	D61.8	Other specified aplastic an...
192008	Congenital syphilitic hepatomegaly ...	TRUE	A52.7	Other symptomatic late sy...

The first two columns correspond to the SNOMED codes and their corresponding description. Similarly, the the last two columns correspond to ICD 10 codes and their descriptions.
However, the third column helps define how exactly one SNOMED maps to an ICD code.

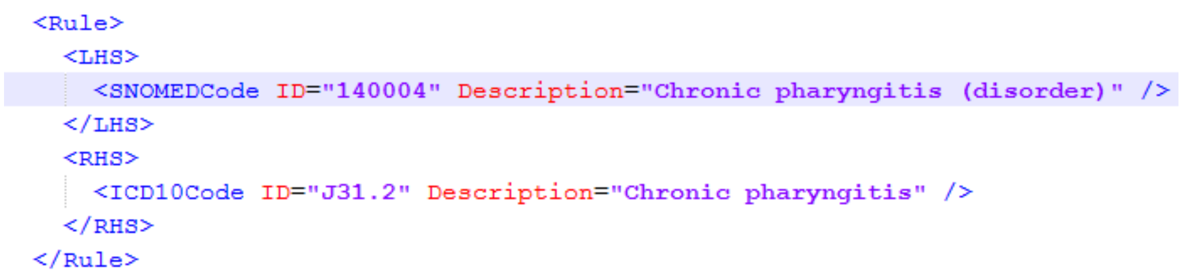
By observing the dataset we can infer that the mapping is both a one to one and two to one depending on the circumstance.
For e.g. by looking at the first record in the image we can understand that if in the emr text we have identified the SNOMED code '140004' and if we can also find the SNOMED code '90979004', only then can we allocate the ICD code as 'J35.0'.

The complementary condition can be seen on line 3, which is simply interpreted as a one to one mapping between SNOMED '140004' and ICD 'J31.2'. But this is the rule that we can only use if the first two lines (rules) can't be satisfied. We refer to this as a conditional rule.

Adapting rules in our pipeline:

The rules that are described above have been utilized in our system and are being consumed as an XML that accurately replicates the mapping in the tsv file.

Simple Rules:



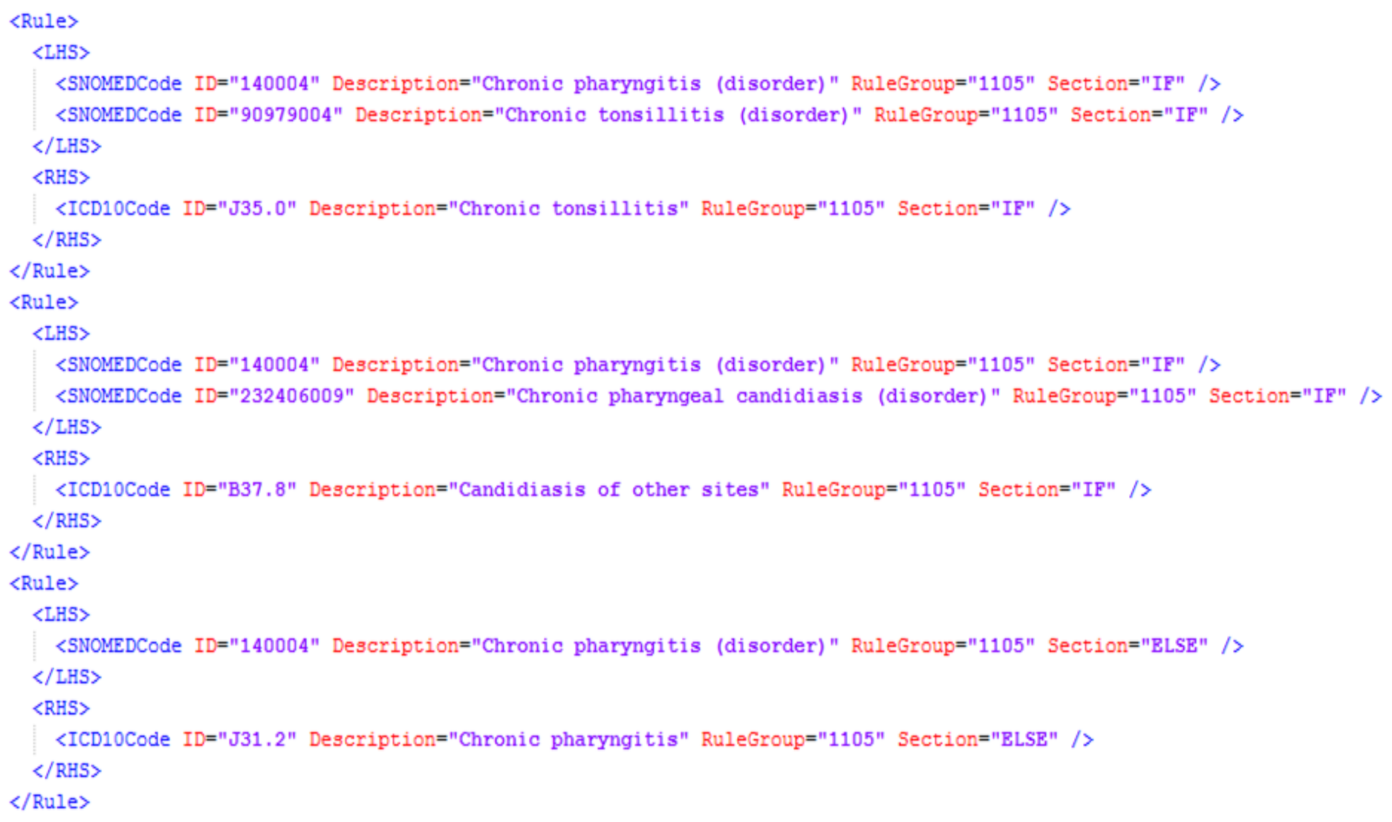
```
<Rule>
  <LHS>
    <SNOMEDCode ID="140004" Description="Chronic pharyngitis (disorder)" />
  </LHS>
  <RHS>
    <ICD10Code ID="J31.2" Description="Chronic pharyngitis" />
  </RHS>
</Rule>
```

The above image is a section from the XML rules file that we generated initially.
These rules only represented all the 'Otherwise True' cases from the tsv file. I.e. this initial file was a simple one to one map from SNOMED to ICD.

Due to this approach, a lot of ICD codes were not accounted for by the system. This file was incomplete and incorrect. Hence we decided to utilize the complete tsv file to generate the rules.
This meant programmatically factoring in the ‘conditional’ nature of the original dataset.

Conditional Rules:

The dataset conditional mapping can be viewed as follows:
If snomed A identified and if snomed B identified => Assign ICD code X
Else If snomed A identified and if snomed C identified => Assign ICD code Y
Else if only snomed A identified => Assign ICD code Z
We then translated this logic directly into the XML file for ease of identification.



```
<Rule>
  <LHS>
    <SNOMEDCode ID="140004" Description="Chronic pharyngitis (disorder)" RuleGroup="1105" Section="IF" />
    <SNOMEDCode ID="90979004" Description="Chronic tonsillitis (disorder)" RuleGroup="1105" Section="IF" />
  </LHS>
  <RHS>
    <ICD10Code ID="J35.0" Description="Chronic tonsillitis" RuleGroup="1105" Section="IF" />
  </RHS>
</Rule>
<Rule>
  <LHS>
    <SNOMEDCode ID="140004" Description="Chronic pharyngitis (disorder)" RuleGroup="1105" Section="IF" />
    <SNOMEDCode ID="232406009" Description="Chronic pharyngeal candidiasis (disorder)" RuleGroup="1105" Section="IF" />
  </LHS>
  <RHS>
    <ICD10Code ID="B37.8" Description="Candidiasis of other sites" RuleGroup="1105" Section="IF" />
  </RHS>
</Rule>
<Rule>
  <LHS>
    <SNOMEDCode ID="140004" Description="Chronic pharyngitis (disorder)" RuleGroup="1105" Section="ELSE" />
  </LHS>
  <RHS>
    <ICD10Code ID="J31.2" Description="Chronic pharyngitis" RuleGroup="1105" Section="ELSE" />
  </RHS>
</Rule>
```

In order to accommodate two SNOMED’s to be mapped to one ICD, we added two SNOMED elements to the LHS of a rule. Also, we can see from the image above that we added two new attributes to elements: Rule Group and Section.

In code, we identify if the rule being executed is coming from the if conditions by looking for a value of “IF’ in the section attribute. Similarly the ‘Otherwise True’ condition is identified by the attribute value of ‘ELSE’. Also, the RuleGroup attribute is used to help us identify if the rule being executed is coming from any else if blocks.

Since the sample input data (14 documents) is small, adapting the conditional rule processing did not make a difference to the output. None of the documents had any combinations of two snomed codes that would map to one icd code. But we have tested the functionality and this conditional processing will work when such a circumstance presents itself.

SNOMED to ICD Conversion Options:

Simple Bag Rule Annotator
We initially implemented a simple annotator that would consider the entire document. I.e. A snomed code identified at the top of the document can be paired with a snomed code identified at the bottom of the document giving one icd code.

Context Bag Rule Annotator
By looking at the sample input documents, we can infer that certain sections of the document don't actually correspond to the disease being diagnosed at this particular time. For e.g. the document may contain patient history where some disease may have been identified. The simple annotator is not concerned with the section of disease identification or the location of the snomed code. Hence, there is a good chance that two snomed codes can be identified in completely unrelated sections which can lead to a two to one rule being executed leading to incorrect results.

To counter this, we implemented an annotator that can consider a particular context to restrict the text to which the rules are being applied. I.e. the annotator will only look within contexts to apply the two to one mapping.

Currently, our pipelines only allows a sentence as a context to iterate over. However, in the future, this annotator can be very beneficial if we can provide paragraph segmentation. It is likely that two relevant and related snomed codes can be identified within a paragraph rather than in the scope of the entire document.

Pruning the ICD codes
Exact matching of snomed descriptions lead to very little snomed's being generated. But the use of the 'Skip Matching' with FST leads over generation of snomed codes. This in turn leads to over generation of icd codes. One of the goals of the project was to provide a trimmed output of icd's from which a user could select the best code.

To enable this, we implemented two different modules which would cut down the number of icds considered based to an upper limit specified by the user.

Word Hits Based Pruning
This method can be considered as an evidence based technique. We know that to generate the snomed codes, we utilized the snomed code descriptions to search in the document. In this method, we perform a reverse lookup using the icd descriptions.

Following are the steps used to prune the icds:

- Tokenize ICD Descriptions
- Eliminate stop words by using a predefined word list
- Match remaining words in the emr document
- Count the occurrence of these words in the document
- Rank ICD's based on the number of word hits
- Select top (N) ICD codes.
    - Number of ICD's selected can be modified in the UI

CAS Querying Based Pruning

From a more general perspective, the task of pruning can be considered as a search problem. The core task of pruning is to search for the outputs which satisfy certain constraints. Then we further process the returned outputs, e.g., remove them or reduce their weights. Once we consider that pruning as a search task, one general approach is to develop a query/search suite: A query language, and a search engine for it.

For our task, the general requirements of the suite are as follows:

1. **Reliability**
   The search suite should be able to match the patterns on UIMA annotations in Cas precisely. Furthermore, the search process should be reasonably efficient.

2. **Maintainability**
   The query language itself should come with certain level of generality, and not be too specific to any particular task. The query should also be simple enough for human experts to generate and maintain.

3. **Cross-annotation Search**
   One important ability expected for the suite is the ability to search among various annotations. In other word, we can set constraints on more than one type of annotations at the same time and search in the Cas. It's a common requirement needed in real-world applications. For instance, we might want to search on some certain sentences containing certain tokens.

Furthermore, we aim to develop a general UIMA query tool suite, so we add two extra UIMA-specific requirements:

1. **Independent from any UIMA type system**
   We aim to develop a general tool which is able to query various UIMA Cas among different system implementations. Therefore, the tool should be independent from any type systems.

2. **Able to query all non-empty direct attributes' value of annotations**
   The tool should be effective enough to support the query of all non-empty direct attributes' value. Note that UIMA type system allows nested annotation, i.e., the attributes of one annotation can be another annotation. We don't aim to recursively extract all attributes of all level of nested

annotations, but at least the tool should support all the first level, i.e., “direct”, attributes.

As a solution, we implemented a general tool, “UIMA Regex”, which lets users to query the UIMA Cas with Java regular expressions.

- **UIMA Regex**

UIMA Regex is a tool that allows you to query the annotations in UIMA Cas with Java regular expression. The runnable software, documentation, and the source codes are both accessible online.[12]

The basic idea is simple:

1. We first convert the UIMA annotations into a sequence of formatted string.
2. Then we use Java regular expression to query the string.
3. Once any sub-strings are matched, we convert it back to a list of UIMA annotations.

For example, if you have the UIMA annotation "Token" with the attribute "pos" (part of speech) in your UIMA system, you can use this Java regular expression pattern:

(&[^&]*%pos="JJ"%[^&]*&)(&[^&]*%pos="((NN)|(NNS))"%[^&]*&)+

To search for all adjective (pos="JJ") tokens followed by at least one noun (pos="[NN|NNS]+") token(s) in your UIMA JCas.

The following two figures demonstrate more details of the process. Figure 1 show that how we convert the annotations in a UIMA Cas into a sequence of String. We first select the needed annotation types in Cas, attach all selected annotations into a sequence, and then sort by their begin values and the hierarchy of annotations assigned by user. Finally, we convert the sorted annotation sequence into a formatted string sequence, which is the “string base” used in query phase.

[12] https://sites.google.com/site/uimaregex/, more details and documentations can be found in the website.

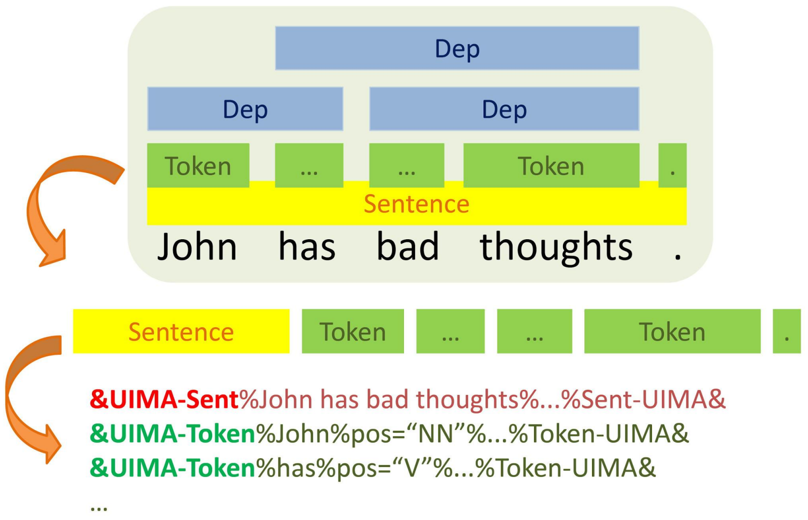

Dep
Dep
Dep
Token
...
...
Token
.
Sentence
John has bad thoughts .
Sentence
Token
...
...
Token
.
&UIMA-Sent%John has bad thoughts%...%Sent-UIMA&
&UIMA-Token%John%pos="NN"%...%Token-UIMA&
&UIMA-Token%has%pos="V"%...%Token-UIMA&
...

Fig 1. Convert UIMA Cas to a sequence of String

Figure 2 shows that how to query the Cas. After the "string base" is generated, we can query the string simply by Java regular expression. Once any sub-string is matched, we map it back to the original annotations.

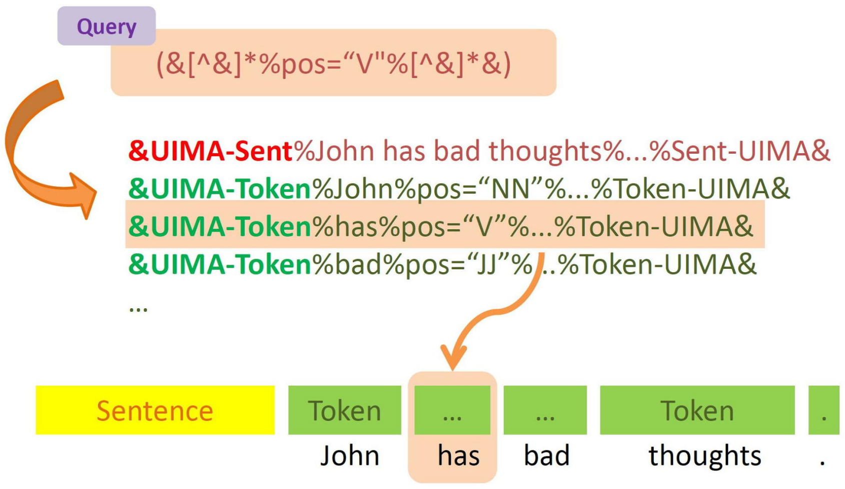

Query
(&[^&]*%pos="V"%[^&]*&)
&UIMA-Sent%John has bad thoughts%...%Sent-UIMA&
&UIMA-Token%John%pos="NN"%...%Token-UIMA&
&UIMA-Token%has%pos="V"%...%Token-UIMA&
&UIMA-Token%bad%pos="JJ"%...%Token-UIMA&
...
Sentence
Token
...
...
Token
.
John
has
bad
thoughts
.

Fig 2. Use Java regular expression to query the String, and map back to the annotation

- **UIMA Regex Pruning**

In our system, we adopt the UIMA Regex to pruning the resulting ICD-10 annotations with certain features. We observed the medical documents and found that doctors often write down the word "normal" to describe the normal result of an examination of patients. Therefore, we use UIMA Regex to prune out the ICD-10 annotations which cover the token "normal." The regular expression we use is as follow:

(&[^&]*((Normal)|(normal))[^&]*&)

The framework of this component is extensible and flexible, we could add more rules or search on other annotations once more features are observed.

Aggregation

- **Requirements to generate pipeline**

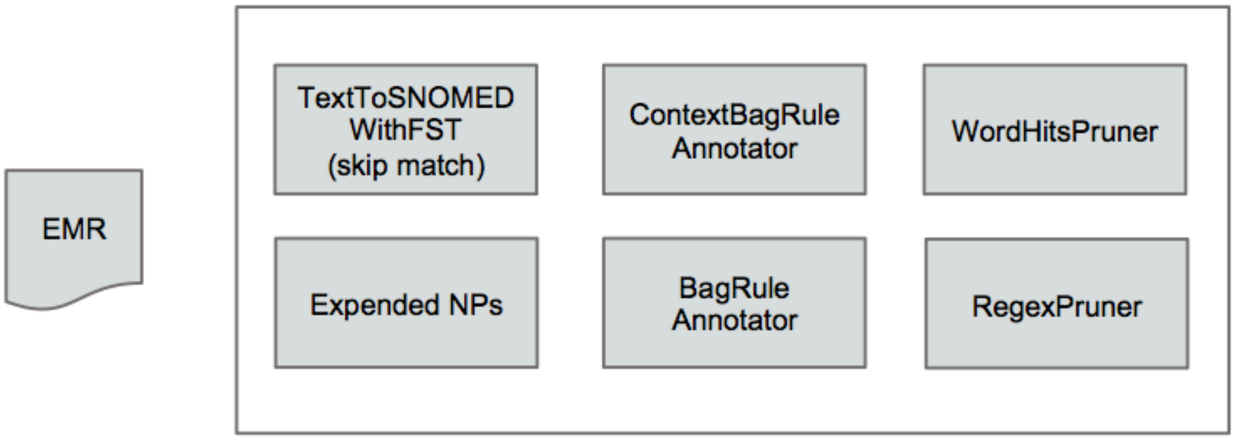
In EMR system, there are several requirements to generate pipeline as follows:

1. Dependency between Analysis Engines

First, some Analysis Engine require other specific Analysis Engine to be in the pipeline. For example, Context Rule Annotator needs Context Annotator such as Sentence Annotator in advance. Thus, we should consider this dependency carefully.

2. Different combinations of Analysis Engines

Second, we have different pipelines and each pipeline needs different Analysis Engine. As shown below the figure, we have mainly three modules and two options for each. Therefore, we should be able to make different combinations of these modules.



3. Various Configuration Parameters

Moreover, each Analysis Engine has various configuration parameters. For example, in Text to SNOMED annotator, we should be able to configure the number of words to skip matching for FST algorithms. Also, in Text Hits Pruner, we need to configure the maximum number of ICD codes. So, we need to try out various values for the configuration parameters.

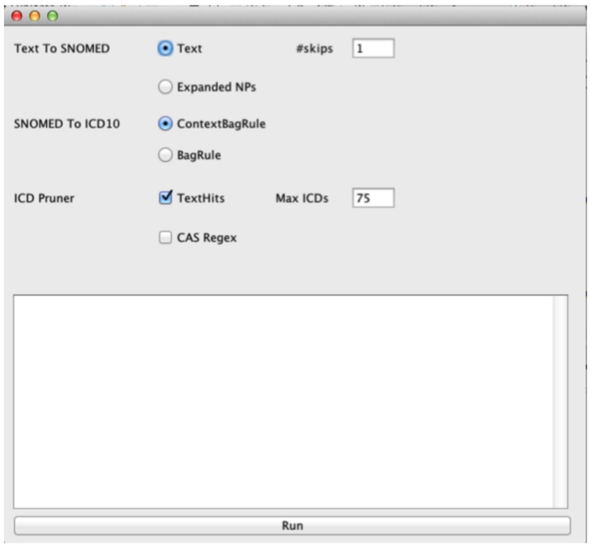
From these reasons, we need a smart aggregator module.

- **What is the challenge?**

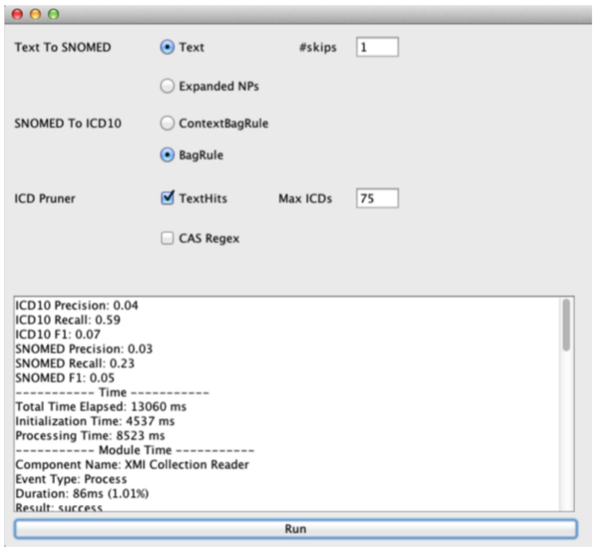
Each UIMA component has a metadata via XML descriptor file which describes the framework how components should be instantiated and deployed. In order to configure UIMA components, different set of descriptor files should be created for each parameter or pipeline and it can be quite laborious. For example, we have 16 different pipelines so we need 16 aggregator descriptors. Moreover, to test each pipeline, we should run the system with the corresponding descriptor. Also, for changing the maximum number of ICD codes or the number of words to skip matching, we have to change the values in XML descriptor one by one before running the system. These make the testing the system be hard.

- **How to do it?**

As a solution, we created XML descriptor files dynamically at runtime by using uimaFIT library(https://code.google.com/p/uimafit/). uimaFIT library provides APIs to describe UIMA components in code. With using uimaFIT library, we generate XML descriptor files based on the selected option in the GUI of the system.
The figures below show snapshots of the system. The left figure shows a snapshot before running, and the right figure shows a snapshot after running. The result displays the ICD 10 and SNOMED precision, recall, F-score, and processing time for each module.



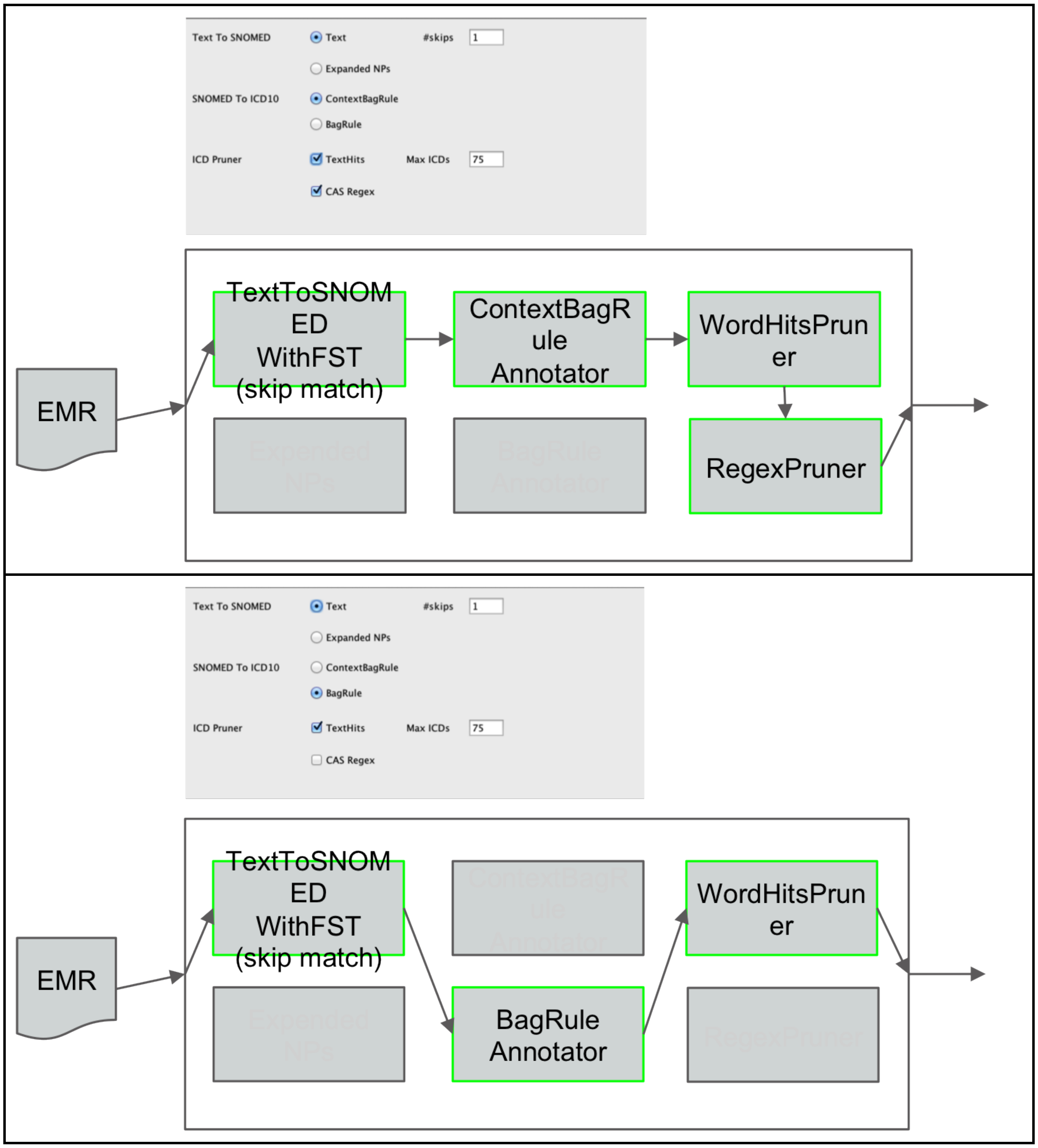
Text To SNOMED: [✓ Text] Expanded NPs — #skips: 1
SNOMED To ICD10: [✓ ContextBagRule] BagRule
ICD Pruner: [✓ TextHits] CAS Regex — Max ICDs: 75



Text To SNOMED: [✓ Text] Expanded NPs — #skips: 1
SNOMED To ICD10: ContextBagRule [✓ BagRule]
ICD Pruner: [✓ TextHits] CAS Regex — Max ICDs: 75

```
ICD10 Precision: 0.04
ICD10 Recall: 0.59
ICD10 F1: 0.07
SNOMED Precision: 0.03
SNOMED Recall: 0.23
SNOMED F1: 0.05
----------- Time -----------
Total Time Elapsed: 13060 ms
Initialization Time: 4537 ms
Processing Time: 8523 ms
----------- Module Time -----------
Component Name: XMI Collection Reader
Event Type: Process
Duration: 86ms (1.01%)
Result: success
```

User can select the Text to SNOMED modules and choose rule policy and also exclude the pruner options or include one of the pruners. When user click the Run button, the descriptors for each selected component are generated, and then the descriptor of Aggregate Analysis Engine is generated. The Aggregate Analysis Engine is for combining the sequence of Analysis Engines and it defines the order of execution of the components. The following two figures demonstrate the examples of the process.


Text To SNOMED [✓ Text] Expanded NPs #skips 1

SNOMED To ICD10 [✓ ContextBagRule] BagRule

ICD Pruner [✓ TextHits] [✓ CAS Regex] Max ICDs 75

Text To SNOMED [✓ Text] Expanded NPs #skips 1

SNOMED To ICD10 ContextBagRule [✓ BagRule]

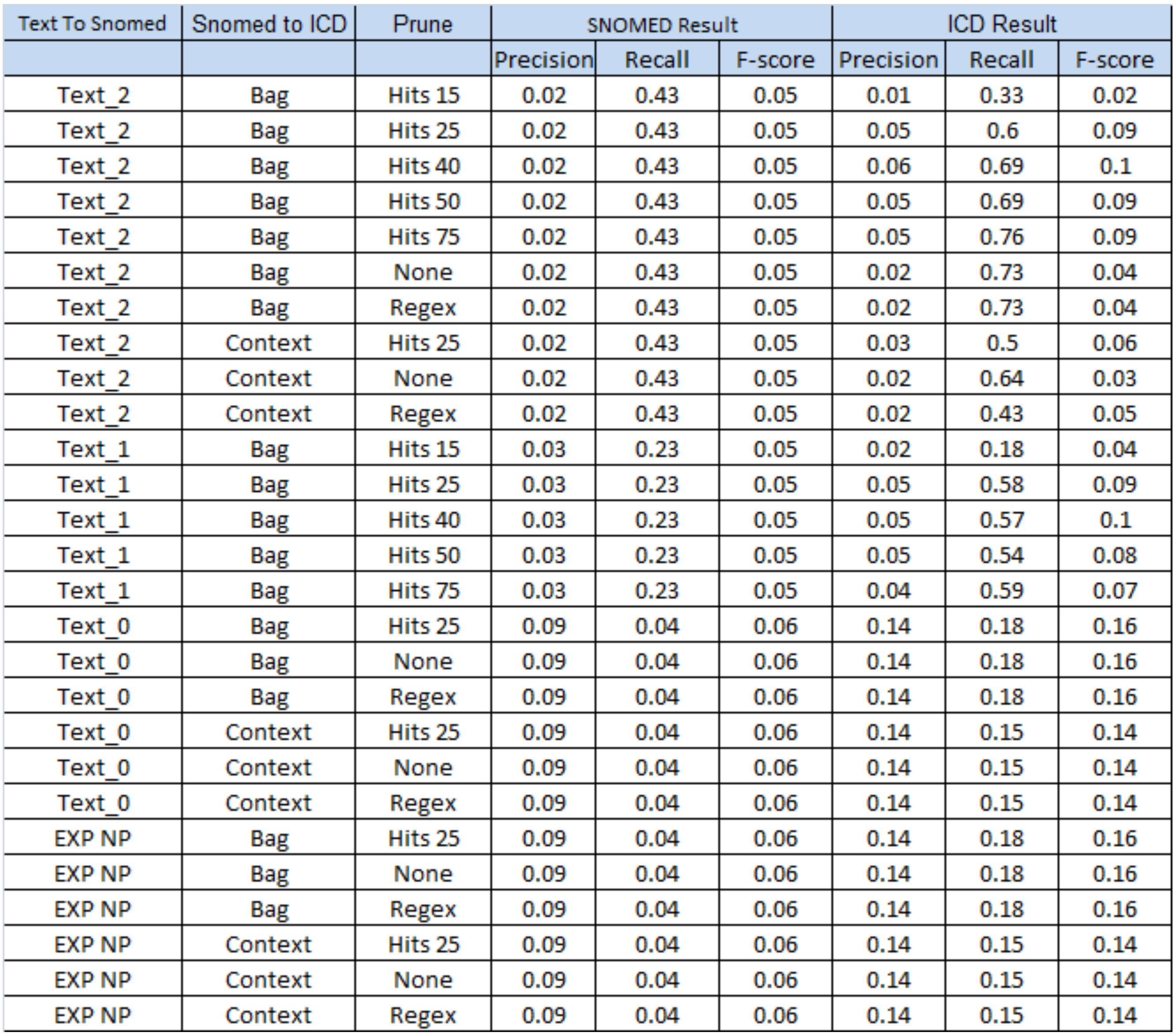
ICD Pruner [✓ TextHits] CAS Regex Max ICDs 75

Results

Evaluation on EMR Document

We evaluated the performance of the different techniques employed in our system on 14 EMR Document by configuring the parameters of the GUI of the system.

The following table shows an in-depth analysis of the performance using different combinations of the technique



Text To Snomed	Snomed to ICD	Prune	SNOMED Result			ICD Result		
			Precision	Recall	F-score	Precision	Recall	F-score
Text_2	Bag	Hits 15	0.02	0.43	0.05	0.01	0.33	0.02
Text_2	Bag	Hits 25	0.02	0.43	0.05	0.05	0.6	0.09
Text_2	Bag	Hits 40	0.02	0.43	0.05	0.06	0.69	0.1
Text_2	Bag	Hits 50	0.02	0.43	0.05	0.05	0.69	0.09
Text_2	Bag	Hits 75	0.02	0.43	0.05	0.05	0.76	0.09
Text_2	Bag	None	0.02	0.43	0.05	0.02	0.73	0.04
Text_2	Bag	Regex	0.02	0.43	0.05	0.02	0.73	0.04
Text_2	Context	Hits 25	0.02	0.43	0.05	0.03	0.5	0.06
Text_2	Context	None	0.02	0.43	0.05	0.02	0.64	0.03
Text_2	Context	Regex	0.02	0.43	0.05	0.02	0.43	0.05
Text_1	Bag	Hits 15	0.03	0.23	0.05	0.02	0.18	0.04
Text_1	Bag	Hits 25	0.03	0.23	0.05	0.05	0.58	0.09
Text_1	Bag	Hits 40	0.03	0.23	0.05	0.05	0.57	0.1
Text_1	Bag	Hits 50	0.03	0.23	0.05	0.05	0.54	0.08
Text_1	Bag	Hits 75	0.03	0.23	0.05	0.04	0.59	0.07
Text_0	Bag	Hits 25	0.09	0.04	0.06	0.14	0.18	0.16
Text_0	Bag	None	0.09	0.04	0.06	0.14	0.18	0.16
Text_0	Bag	Regex	0.09	0.04	0.06	0.14	0.18	0.16
Text_0	Context	Hits 25	0.09	0.04	0.06	0.14	0.15	0.14
Text_0	Context	None	0.09	0.04	0.06	0.14	0.15	0.14
Text_0	Context	Regex	0.09	0.04	0.06	0.14	0.15	0.14
EXP NP	Bag	Hits 25	0.09	0.04	0.06	0.14	0.18	0.16
EXP NP	Bag	None	0.09	0.04	0.06	0.14	0.18	0.16
EXP NP	Bag	Regex	0.09	0.04	0.06	0.14	0.18	0.16
EXP NP	Context	Hits 25	0.09	0.04	0.06	0.14	0.15	0.14
EXP NP	Context	None	0.09	0.04	0.06	0.14	0.15	0.14
EXP NP	Context	Regex	0.09	0.04	0.06	0.14	0.15	0.14

Results on Complete Pipeline

We merged our pipeline with the MS-BIC students. We evaluated the performance of our complete pipeline on the ICD Description. We evaluated based on fuzzy match of ICD Codes. The following table presents our evaluation on the 14 EMR Document and ICD Description File.

Test Data	EMR-ICD10	EMR- Snomed	Snomed - ICD
EMR Document	Precision 0.05 Recall 0.52 F-Score 0.03	Precision 0.02 Recall 0.38 F-Score 0.04	Precison 1.00 Recall 0.84 F-Score 0.89
ICD Description	Precision 0.44 Recall 0.79 F-Measure 0.57	N/A	N/A

Conclusion

Process and Evolution

We started our system design development with dummy rules for Text to Snomed and Snomed to ICD Mapping. From our early meetings, we decided to focus on the Text to Snomed and Snomed To ICD components. First, by using Snomed Manual as our resource, we performed Exact Matching of Text to convert to Snomed. We used one-to-one rules to convert from Snomed to ICD Codes. Although, the precision was reasonable, we were losing on recall.

One of the reasons was due to the very few matches of the Snomed from the Text. We approached the problem by proposing and implementing two solutions. The first is using Expanded Noun Phrases to extract more Snomed Codes from the text. The second is to give FST based Skip Matches to extract more codes which are not extracted by performing exact matches. We compared the performance of the two approaches and incorporate both the options in out GUI.

The other issue we faced was the less coverage for Snomed to ICD Codes. We resolved the problem by implementing Conditional Rules which lets the system output a more fine-grained set of ICD Codes. However, by using Skip Matches in Text to Snomed conversion and conditional rules for Snomed to ICD Mapping, we accounted the problem of over generation of ICD codes. We resolved the issue by adding another component to our pipeline - ICD Pruner. We implemented and compared two solutions to this problem. The first is based on Word Based Pruning which matches the ICD Codes by matching the text of the EMR Document with the ICD Description. The second is based on Querying the CAS by using Regular Expression.

Overall, through our complete evolution cycle we learnt the precision-recall tradeoff and which factors favours which metrics and why does the results vary by using different components.We focused mostly on the system design answering what would be a design that is generic enough to improve upon in future with few modifications, but not about the accuracy. All of us have learnt so much about the critical decision making that shapes up the final design of a typical software engineering project.

Reflection

When we started this project,

1. We took time in understanding the problem
2. what is the input give to us
3. what resources could be employed
4. The need for Snomed Codes was not clear to us in the beginning
5. The web has different versions of ICD, which at that time was confusing for us.

Given, the problem again, we would be well acquainted with the details of the existing system, and what and how the improvements can be made to the system. Some of the main changes we can now think of are as follows:

- **Adding a Rule Generator**: The rules should be generated by a module on the fly in an initialization step. Currently, the rules are read from an XML document, whose logic is hard coded. Typically, different options in a rule based system like simple Vs complex rules should ha[[en through programmatic logic instead of reading rules from different files. This is one change we want to do to our system.
- **Evaluating the necessity of SNOMEDs**: The decision to go to SNOMEDs in an intermediate step is motivated by the philosophy that the valuable SNOMED-to-ICD resources can be utilized in the pipeline. However, it's also interesting to see how well a system that's trained on ICD descriptions performs without going to SNOMEDs first. Since we cannot make an informed choice unless we know for sure whether SNOMEDs help or not, we would give preference to devising an evaluation scheme for the usefulness of SNOMEDs if we were to start the project today.
- **Relevant area identification in EMR**: At several points in the current pipeline, we match text in the EMR to assign codes. In an attempt to generate relevant codes instead of codes resulting from any match, we could have gone for identifying the area of the document that pertains to the patient condition and not patient's details, family history etc.. In the current system, there is some manual work being done to disregard some of the meta-data. Since finding relevant areas was previously done in NLP and since it benefits the pipeline by acting against over-generation, we would have gone for that.
- **Configuration space exploration**: In a completely rule based system, one should expect several decisions to be taken while system building that are merely choices and they end up as options for the configuration parameter. We didn't put much thought into how we would be tuning the system until the end. In the end, we came up with the idea of using UIMAfit, to be able to select various options one can select in the pipeline. If we were to start today,we would have gone for CSE right from the beginning so that we might be able to take more informed decisions at several stages. However, we should note that most of the decision making is based on the 14 samples we have and hence cannot be trusted so much.

Future

There are many future directions to our work. Since we mostly worked on the system design, there is a lot more that can be done in the individual modules in the system. Some generic ideas are listed below:

1. Delving deeper into each component: As we said, each component in the current system is by no way state-of-the-art, but only a prototype that tells us that the system works. By improving the resources used by each component, the accuracies can be improved on both the samples and larger corpora.
2. Apart from using better resources, one can also think of comparing the current modules with the existing NLP state-of-the-art and see if ideas from such systems can be incorporated into the system. For example, as mentioned before, there is work in NLP on identifying related areas in an unstructured document, which looks like a prominent thing that wasn't looked into, in this project. A module that does specifically this would be very useful.
3. In the current system, we first map the text to the Snomed, and then by using Snomed we extract the ICD Codes. As mentioned in the reflection section, it will be interesting to directly convert from text to ICD without undergoing the need to first generate Snomed Codes. That will help us understand the necessity of the SNOMED resources.
4. The values that can be given as "context" in the ContextBagRuleAnnotator are as of now types in the cas. However, context in NLP is much richer and contains a snapshot of the values the types in the cas have and so on. We tried to incorporate this in the pipeline by representing the context with UIMA regex, but we couldn't do it because of UIMA regex still being in development for such complex patterns.