

AMERICAN UNIVERSITY OF BEIRUT

Cross-document Analysis for Automatic  
Understanding of Electronic Medical  
Records

by

Batoul Masoumah Hussein Sharafeddin

A thesis  
submitted in partial fulfillment of the requirements  
for the degree of Master of Science  
to the Graduate Program in Computational Science  
of the Faculty of Arts and Sciences  
at the American University of Beirut

Beirut, Lebanon  
June 2020

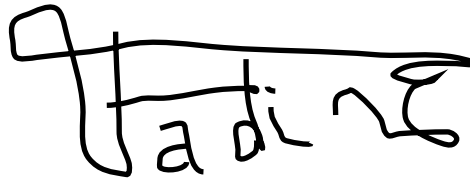
AMERICAN UNIVERSITY OF BEIRUT

Cross-document Analysis for Automatic  
Understanding of Electronic Medical  
Records

by

Batoul Masoumah Hussein Sharafeddin

Approved by:



---

Dr. Fadi Zaraket, Associate Professor

Advisor

Electrical and Computer Engineering

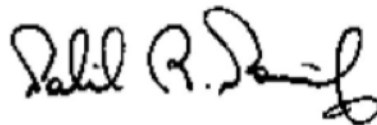


---

Dr. Abbas Alhakim, Associate Professor

Member of Committee

Mathematics



---

Dr. Nabil Nassif, Profe

Member of Committee

Mathematics

Date of thesis defense: June 17, 2020

# AMERICAN UNIVERSITY OF BEIRUT

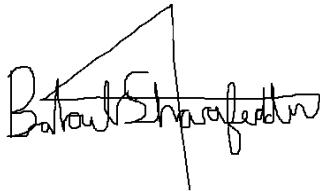
## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Sharafeddin Batoul Masoumah  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One  year from the date of submission of my thesis, dissertation or project.**  
**Two \_\_\_ years from the date of submission of my thesis, dissertation or project.**  
**Three \_\_\_ years from the date of submission of my thesis, dissertation or project.**



July 2 2020

Signature

Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

# Acknowledgements

Firstly, lastly, and forever more praise and thanks to Allah swt, and the fourteen stars he sent to guide and guard me.

To my parents and to my grandparents for being a huge inspiration in so many dimensions of life.

To my advisor Professor Fadi Zaraket, for his brilliance, patience, aid, and for everything he taught me.

To the Professors on my committee, Professor Nabil Nassif and Abbas Alhakim for their time and support.

To my siblings for making me want to be a better person.

Finally, to my fiancé and his family for being such an encouragement, and for always being on my side.

# An Abstract of the Thesis of

Batoul Masoumah Sharafeddin for Master of Science  
Major: Computational Science

Title: Cross-document Analysis for Automatic Understanding of Electronic Medical Records

Electronic medical records contain both structured entities such as diagnosis codes and results, and unstructured entities such as textual notes typed by health care providers to record patient information during encounters. They play an integral role in assisting health care providers to manage the diagnoses and treatment plans of individual patients. Automated understanding aims at extracting entities and relational entities from the notes, which can act as diagnosis indicators.

Existing research proposed to annotate and extract information with the lowest possible error, and surveys have been published to discuss existing work done in this field. Commercial HIS systems also exist. EPIC is an industrial system that supports automated understanding. Health Information Technology for Economic and Clinical Health is also commercial and has less support for automated understanding. The methods are often expensive and always lack support to records from developing countries.

In this thesis, we aim at improving automated understanding of electronic medical records for differential diagnosis analysis.

Differential diagnosis analysis considers several diagnostic algorithms from clinical diagnostic medical textbooks and relates them to the case at hand. For our computational model, we constructed several Bayesian networks that reflect the structure of the diagnostic algorithms. Then we devised a cross-document analysis method to learn the probabilistic parameters of these networks.

The cross-document analysis method performed cross-reference equivalence of the terms in the notes with (i) more rigorous English text with a similar distribution collected from United States Medical Licensing Examination questions, (ii) an expert based map of abbreviations, and (iii) a cor-

pora of medical publications extracted from Pubmed. For that we devised cross-reference equivalence metrics that augmented each term with equivalent terms. Then we estimated the truth of a Bayesian network node by estimating the existence of its terms from the diagnostic algorithms in the electronic medical record in question.

We applied our method to a corpora of 151,930 clinical notes of which 3,616 are annotated. The corpora is collected from American University of Beirut Medical Center and Rafic Hariri University Hospital. The annotations are organized in a tree of diagnoses and clinical differential analysis terms.

After computing the Bayesian network parameters, we queried each Bayesian network with its diagnostic decision node as an *explanation* and with the electronic medical note as *evidence*. We considered the networks with highest scores candidates for differential analysis.

Our method successfully identified the correct diagnosis among the top two diagnostic algorithms with an average recall of 93%, and a precision of 99%. When considering prediction of correct diagnosis, the average precision is 64%. The analysis often included prevailing diagnoses such as fatigue, headache, and joint sprain which health care providers refer to after eliminating more serious diagnoses.

The thesis work also presents a tool we developed for semi-automatic annotation of electronic medical records with diagnostic graphs, and use of other techniques such as Neural Networks and Hidden Markov Models that showed lower performance.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
1.1 Data . . . . .	3
1.2 Annotation Tools . . . . .	3
1.3 Distributional Similarity . . . . .	4
1.4 Model . . . . .	4
1.5 Augmentations . . . . .	4
1.6 Method . . . . .	5
1.7 Results . . . . .	6
<b>2 Background</b>	<b>7</b>
2.1 Electronic Medical Records . . . . .	8
2.2 Annotations . . . . .	8
2.3 USMLE Exams . . . . .	9
2.4 Diagnosis Graphs . . . . .	9
<b>3 Literature Review</b>	<b>12</b>
3.1 Data . . . . .	13
3.2 Techniques . . . . .	13
3.3 Systems . . . . .	15
<b>4 Bayesian Networks</b>	<b>25</b>
4.1 BNs . . . . .	26
4.2 Diagnostic Algorithms to Bayesian Networks . . . . .	27
<b>5 Distributional Similarity</b>	<b>32</b>
5.1 Similarity metrics: Distributional similarity . . . . .	33
5.2 Sources of Distributional Similarity Thesauruses . . . . .	33
5.3 DISCO . . . . .	34

5.4	Distance Value Matrix . . . . .	34
<b>6</b>	<b>Cross-Document Analysis Methodology</b>	<b>36</b>
6.1	Core word vector extraction . . . . .	38
6.2	Diagnostic graph enrichment . . . . .	38
6.3	Classification Boosted with Diagnostic Graphs . . . . .	39
6.4	Augmentation Mechanisms . . . . .	39
6.5	EMR Similarity Score . . . . .	41
6.6	Vowel Based Variations . . . . .	41
6.7	Dictionary Based Augmentation of Symptoms . . . . .	42
6.8	Word Similarity Score Calculation using Augmentations Algorithm . . . . .	42
6.9	Score Matrix . . . . .	43
6.10	Calculations . . . . .	44
6.11	Diagnostic Algorithms . . . . .	44
6.12	Score Aggregation . . . . .	47
6.13	Learning BN Parameters . . . . .	48
<b>7</b>	<b>Automatic Annotation Tool Code</b>	<b>49</b>
7.1	Constructors . . . . .	49
7.2	Dictionary Synonyms . . . . .	50
7.3	Co-occurrence Frequency Calculation . . . . .	50
7.3.1	EMR . . . . .	50
7.3.2	USMLE . . . . .	53
7.4	DescendingProbabilityComparator . . . . .	53
7.5	Finding Top <i>K</i> similarly distributed Words . . . . .	54
7.6	Automatic Annotations . . . . .	55
7.7	Objects . . . . .	56
7.8	Methods for Automatic Annotations . . . . .	57
7.8.1	Bagging . . . . .	57
7.9	Methods . . . . .	58
7.9.1	normedTopK contains w . . . . .	62
7.9.2	normedTopK does not contain w . . . . .	62
7.9.3	End of Method . . . . .	63
7.9.4	Pubmed version of Method using DISCO . . . . .	63
7.9.5	End of Method . . . . .	64
7.10	Saving the Automatically Annotated Notes . . . . .	64
<b>8</b>	<b>Website Implementation</b>	<b>66</b>
8.1	Website Layout (Using HTML) . . . . .	66
8.2	Displaying Automatically Annotated EMR Notes . . . . .	66



8.3	Automatically Annotating User Input . . . . .	68
8.4	Displaying Automatically Annotated EMR Notes . . . . .	68
8.5	Automatically Annotating User Input . . . . .	71
8.6	Pending Work . . . . .	72
<b>9</b>	<b>Other Work</b>	<b>73</b>
<b>10</b>	<b>Results</b>	<b>75</b>
10.1	Discussion . . . . .	86
<b>11</b>	<b>Conclusion</b>	<b>88</b>
<b>A</b>	<b>Abbreviations</b>	<b>90</b>

# List of Figures

1.1	Example EMR with clinical text . . . . .	1
2.1	The structured and the unstructured components of the EMR. .	9
4.1	A BN for symptoms of smokers . . . . .	27
4.2	Beginning of the diagnostic algorithm . . . . .	29
4.3	The first few BN nodes for the BN structure. . . . .	29
4.4	Full diagnostic algorithm. . . . .	30
4.5	Full BN structure with abbreviated nodes. . . . .	31
6.1	Flow chart for BN conditional probability learning procedure .	40
10.1	Precision and Recall [1] . . . . .	76
10.2	Anemia Notes Scores . . . . .	77
10.3	Anxiety Notes Scores . . . . .	78
10.4	Diabetes Notes Scores . . . . .	78
10.5	Fatigue Notes Scores . . . . .	79
10.6	Headache Notes Scores . . . . .	80
10.7	Hemoptysis Notes Scores . . . . .	81
10.8	Joint Sprain Notes Scores . . . . .	81
10.9	Kidney Disease Notes Scores . . . . .	82
10.10	Pruritus Notes Scores . . . . .	83
10.11	Tinnitus Notes Scores . . . . .	84

# List of Tables

2.1	Description of annotation scheme. N/A: Not applicable. Text within " and " indicates text within the same record/sentence. <b>Bolded</b> text denotes annotation labels. . . . .	10
2.1	Description of annotation scheme – continued from Table 2.1 . . . . .	11
6.1	Reference for notations . . . . .	44
6.2	This table shows the matrix we store of the scores of each word augmentation of a single word in a given EMR with each word augmentation of a single word in a given node of a given BN . . . . .	45
6.3	This table shows the matrix we store of the scores of each word in a given EMR with each word in a given node of a given BN . . . . .	45
10.1	Anemia Precision and Recall . . . . .	77
10.2	Anxiety Precision and Recall . . . . .	77
10.3	Diabetes Precision and Recall . . . . .	79
10.4	Fatigue Precision and Recall . . . . .	79
10.5	Headache Precision and Recall . . . . .	80
10.6	Hemoptysis Precision and Recall . . . . .	80
10.7	Joint Sprain Precision and Recall . . . . .	82
10.8	Kidney Disease Precision and Recall . . . . .	82
10.9	Pruritus Precision and Recall . . . . .	83
10.10	Tinnitus Precision and Recall . . . . .	83

# Chapter 1

## Introduction

Electronic medical records (EMR) are usually written during encounters with patients. They store structured entities alongside additional information the health care provider (HCP) finds important that may include symptoms, test results, family history, and possible treatment plans [2].

They assist HCPs in managing cases and diagnosing patients. Automating the process of finding the suitable diagnosis pertaining to these unstructured notes improves the work efficiency of the health care provider and helps extract potentially valuable information stored therein.

Tools that perform entity extraction from formal English exist [2]. However, EMR notes are far from formal English. Take for example the note in Figure 1.1.

It is clear in the figure 1.1 that the text information is not articulated properly. There is no proper grammar or punctuation. There are several abbreviations and spelling mistakes.

In addition, the notes often contain proprietary terminology and may in-

16/04/2009) -Low Vit D (25(OH) VITAMIN D: 13.70 ng/mL ON 07/12/2009), not under control -Bilateral severe Varicose veins in L.E. with Venous stasis (chronic venous insufficiency) -Bilateral Knee pain (severe osteoarthritis) , and she was told that she needs knees replacement. N.B. Last urinalysis was positive for infection bu culture: no growth in April 2009) P.S.H.: -Partial Thyroidectomy (Goiter) without secondary hypothyroidism (Last TSH done Lifestyly: -Non-smoker -Mild activities other than her home work Medications: -Co-Aprovel 300/12.5 mg daily -Amlor 5mg daily -Glibomet TID -Atorlip 20mg daily (to be replaced with Crestor 10mg) -Daflon 500 BID -Sterogyl 20000iu q week for 10 weeks -Orocal D3 BID -Aspirin Protect (was discontinued few months ago because of recurrent epistaxis) Plan: -New Lab tests -Mammography -Bilateral Knees X-Rays then C/S Orthopedist -C/S

Figure 1.1: Example EMR with clinical text

clude transliterated local Arabic words, arbitrary on the fly abbreviations, spelling mistakes, and other ambiguous information. This may be due to factors such as the responsibility of the HCP to maintain eye contact with the patient while typing.

Health care practitioners use health information management systems (HIS) to enter these notes. Examples of these systems are GNU Health[3] and Clinical Text Analysis and Knowledge Extraction System [4]. GNU Health is open-source and does not support automated understanding of unstructured entities. Clinical Text Analysis and Knowledge Extraction System is open-source and has limited support for automated understanding. It is based on data collected from developed countries.

Commercial HIS systems also exist. EPIC is an industrial system that supports automated understanding [5]. Health Information Technology for Economic and Clinical Health is also commercial and has less support for automated understanding [6]. Many techniques and methods have been proposed to annotate and extract information with the lowest possible error, and surveys have been published to discuss existing work done in this field [7], [8], [9]. The use of Natural Language Processing (NLP) in the field of medical information extraction has been successfully adopted by many researchers, however the methods are often expensive and always lack support to records from developing countries.

This thesis focuses on automating differential analysis of EMR notes using Bayesian networks (BNs).

Differential analysis involves discovering the two or three most related diagnostic algorithms to the complaint of the patient by analyze the relation the EMR has with the algorithms. This will enable us to come up with a relevant diagnostic.

To do that, it leverages EMR corpora, clinical diagnostic algorithms extracted from medical books and a subset of the notes annotated with a tree of diagnoses and clinical differential analysis terms.

We constructed the BNs corresponding to EMR corpora from the diagnostic algorithms manually. Each node in the BNs is associated with the text of each node in the diagnostic algorithm.

We also augment the text from EMR notes and BN nodes with statistically equivalent phrases extracted from the following

- An expert based map of abbreviations
- Our EMR notes
- United States Medical Licensing Examination questions

- A corpora of medical publications extracted from Pubmed

After computing the BN parameters we predicted diagnosis using BN explanation queries given some evidence.

We estimate the existence of each BN nodes in all EMR texts. This existence data enables us to learn the BN conditional probability functions which are the BN parameters. As a final step the BNs are queried for the most likely explanation of all the EMR notes, thus leading us to the most relevant diagnosis for every note queried.

Our results show that BNs successfully identify the relevant differential analysis by predicting the top two or three diagnostic algorithms as explanations of the EMR note.

In the following we will go over each of these briefly in the introduction and will explore in coming chapters.

## 1.1 Data

Our work builds on EMR corpora collected from American University of Beirut Medical Center and Rafic Hariri University Hospital to aid the automation of the diagnosis process. The corpora consists of 151,930 total medical notes, 3,616 of which are annotated [10]. We use all the annotated notes and a subset of the unannotated notes pertaining to ten diagnoses we would like to study. The diagnoses are anemia, anxiety, diabetes, fatigue, headache, hemoptysis, joint sprain, kidney disease, pruritus, and tinnitus.

We also have available one hundred and fifteen diagnosis graphs from medical books that represent clinical diagnostic algorithms extracted from clinical textbooks [11]. We leverage ten diagnostic graphs extracted from the clinical books pertaining to the mentioned ten diagnoses. The diagnostic graphs represent clinical diagnostic algorithms extracted from clinical textbooks.

We also leverage a set of USMLE exam questions that are very similar to our EMR data but adhere to the rules of proper English.

## 1.2 Annotation Tools

The manual annotations of a subset of the EMRs are provided by health care practitioners. These help identify terms relevant to one another through their classification. They are classified into twenty nine categories; SignsByDr, Medications, MedDose, EntityDescription, PreviousSurgeries, Labs, Imaging, ChiefComplaint, MissedNames, HistoryOther, Plan, LabResults, TimeWords,

Symptoms, PreviousDx, Differential, SocialHx, Radiology, CurrentDx, Frequency, Vitals, Negation, Surgeries, FamilyHx, Diet, Physical, Procedures, Site, Age.

These annotations help in creating automatic annotations as words that are similarly distributed to the annotated words may be labeled with the same annotation.

### **1.3 Distributional Similarity**

To find relations between words in medical notes we calculate the distribution of the EMR note words and USMLE note words to find sets of most similarly distributed words for each word.

This type of automated understanding aims at extracting diagnosis indicators from the notes using natural language processing and computational linguistics.

We will also be using a Java library DISCO [12]. This package allows us to retrieve the semantic similarity between arbitrary words and phrases based on the statistical analysis of large text collections. Distance similarities are based on the statistical analysis of very large text collections from PubMed.

### **1.4 Model**

We propose cross-referencing medical entities from different sources and establishing relations among them. We intend to identify matches between the words in the EMR notes and the words in the diagnostic graphs.

To do this, we manually translate the ten diagnostic algorithm graphs into BN structures following guidelines from [13]. To learn the parameters of the BNs, we estimated the truth value of each BN node relative to each EMR note. We developed a cross-document analysis approach for that purpose.

The method augments the EMR text and the text associated with BN nodes with equivalent text from different sources to provide vectors of similar words for each EMR note and BN node.

### **1.5 Augmentations**

We create vectors of similar words for every word in the EMR notes and the diagnostic graphs using five augmentation sources.

- The first is the manual annotations of EMRs we have. These help identify terms relevant to one another through their classification. Annotated words within the same note are related.
- The second is vowel-based variations of a word. So if a note word can pass as a vowel based variation of an annotated word we will include it in the vector of similar words.
- The third is based on sets of similarly distributed words from EMR.
- The fourth is based on sets of similarly distributed words from USMLE
- The fifth is based on sets of similarly distributed words from Pubmed corpora.

These Augmentation mechanisms are key in facilitating the communication between the diagnostic graphs and EMR analysis.

## 1.6 Method

Our method involves the followings steps:

- In the first phase, our method pre-processes the EMRs to remove irrelevant entities (noise) using JAVA code.
- We calculate the distributional similarities of words in the EMR and USMLE corpora using JAVA code.
- Our computational model enriches the annotations with the diagnostic graphs. We manually translate the diagnostic graphs to BN form. We input these structures into Python code.
- We use the augmentation methods to find matches of entities in the EMR notes to entities in the BN nodes using JAVA code. We fine tuned an aggregate metric to compute the score.
- The BNs are then trained using every entity's existence or non-existence in a note of the corresponding diagnosis for the conditional probability tables using Python package pomegranate.
- We then find the entity existence data of all the EMR notes with all the BNs using JAVA code.
- We then can find the BNs most relevant to an EMR based on its score with the BN models using Python package pomegranate.



- By querying the BNs for the likelihood of a note pertaining to different diagnoses, after finding its match data with each BN, we will be able to answer the question of which top three possible diagnoses best explains the note at hand.

## 1.7 Results

Our results show that BNs successfully identify the relevant differential analysis by predicting the top two or three diagnostic algorithms as explanations of the EMR note. The analysis often included prevailing diagnoses such as fatigue, headache, and joint sprain which are underlying symptoms to other more serious diagnoses.

The average recall is 0.93. The differential analysis based precision was 1 for all the models when we consider the precision according to whether the model appears in the top two models with the notes pertaining to them. If we weigh every single algorithm on its own, the average precision for the models is 0.18 when compared to all the other models, and 0.64 when compared to one other model. This may be due to a lack of data and a lot of missing information.

The rest of the thesis is structured as follows.

In Chapter 2 we present some background information for our work.

In Chapter 3 we present a literature review of work done in this field.

In Chapter 4 we present some information on BNs and how we turned our diagnostic algorithms into BN structures.

In Chapter 5 we discuss the word distributional similarity approach.

In Chapter 6 we discuss our cross-document analysis methodology.

In Chapter 7 we go over our code implementation.

In Chapter 8 we discuss the interactive website developed to display our work and the features it includes.

In Chapter 9 we discuss other work we completed along the process of searching for the best approach for our purpose.

In Chapter 10 we present the results of our chosen cross-document analysis methodology.

In Chapter ?? we discuss our results and analyze the work.

Finally, we conclude and discuss future work in Chapter 11.

# **Chapter 2**

## **Background**

Our work builds on existing EMR corpora collected from American University of Beirut Medical Center (AUBMC) and Rafic Hariri University Hospital (RHUH).

We will be cross-referencing the free-text notes in them with diagnosis graphs from medical books that represent clinical diagnostic algorithms extracted from clinical textbooks.

## 2.1 Electronic Medical Records

The corpora consists of 151,930 total EMRs. The notes in the EMRs are in the form of free text, and 3,616 of them contain annotated notes while 148,314 contain unannotated notes. The EMRs that contain annotated notes also contain structured information.

This structured information includes the patient record ID, the case code, a number encoding the patient's first and last name, the date of the visit, the age group and sex of the patient, and codes representing the identity of the doctor following up the case.

The EMRs that contain the unannotated notes only contain this one form of information. This sample of an EMR note is taken from A. Alawieh, Z. Sabra, A. Jaber, H. Hayek, F. Zaraket, and G. Hamadeh's work in the paper MENA-AMTC: Middle East and North Africa Annotated Medical Text Corpora. They used the same data-set we will be using.

The image 2.1 displays both the structured and the unstructured components of the EMR. The EMR begins with nine structured components in the form of numbers coding private patient information, they are separated by field delimiters in the form " | |".

When the unstructured free text component begins, the paragraph delimiters as symbolized by " " to avoid mistaking the EMR note for ending. Once a field delimiter is seen again, the end of the note is signaled and the rest of the structured components follow.

## 2.2 Annotations

In the annotated notes, the relevant words in the EMRs are annotated by health care providers according to an annotation schema containing 31 annotation labels.

Table 1 thoroughly explains the annotations made by health care practitioners available in the EMRs. Similarly, this table was adopted from the

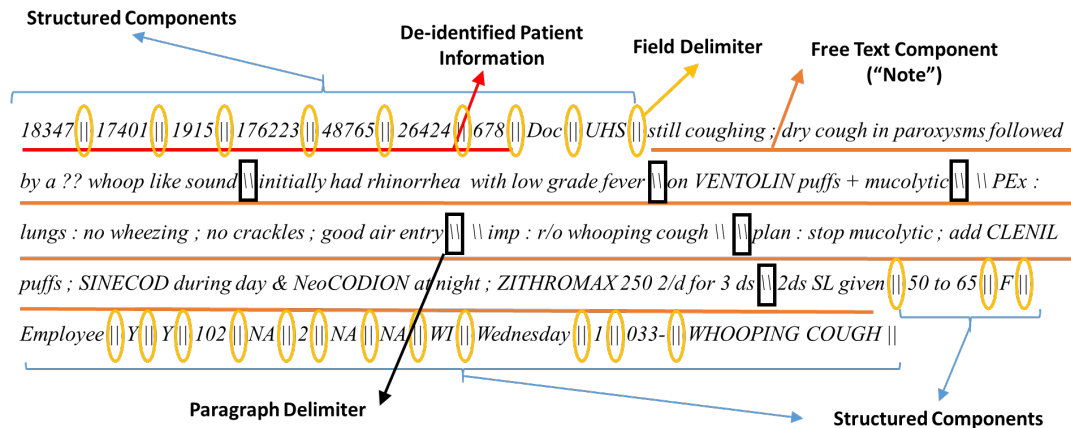


Figure 2.1: The structured and the unstructured components of the EMR.

paper MENA-AMTC: Middle East and North Africa Annotated Medical Text Corpora.

It displays the full set of annotations made by health care practitioners, alongside a description of each annotation and an example.

The labels in bold are the main annotation labels, and the labels not in bold are annotations that correspond to the last previous bold label.

## 2.3 USMLE Exams

We leverage a set of USMLE exam questions that are very similar to our EMR data but adhere to the rules of proper English.

## 2.4 Diagnosis Graphs

Our model leverages 115 diagnostic graphs that represent clinical diagnostic algorithms extracted from the 2012 clinical textbooks "An Evidence-Based Approach to Differential Diagnosis New York". Each graph has a corresponding image manifesting the information stored in sheets.

The sheets store this information in sets of nodes and edges. The nodes contain id numbers, entity label, entity type, and the color of the box containing the entity. The edges contain the id of the source entity, the id of the label entity, and the command to go from the source to the label depending on whether or not the target id condition is satisfied.

Table 2.1: Description of annotation scheme. N/A: Not applicable. Text within " and " indicates text within the same record/sentence. **Bolded** text denotes annotation labels.

<b>Label</b>	<b>Description</b>	<b>Examples</b>
<b>Age</b>	the age of the patient including number descriptions	22 years oldAge, youngAge, and elderlyAge
<b>ArabicWords</b>	labels the terms that are transliterated Arabic words	argilehArabicWords
<b>Differential</b>	labels the physician’s differential diagnoses once present individually	orthostatic dizziness
<b>CurrentDx</b>	labels the diagnosis that the physician writes during the visit	Mr. 6532 5698 has essential hypertensionCurrentDx
<b>EntityDescription</b>	labels an added feature describing a term	
Location	labels the body site at which the pain is or where a procedure is performed	bilateralLocation lower extremity pain Symptoms
Frequency	labels the terms related to the frequency of pain, medication intake or treatment,	dailyFrequency
MedDose	labels the terms that describe the dose of medication	MetforminMedications 500 mgMed-Dose.
<b>FamilyHx</b>	labels terms describing diseases, conditions, symptoms present in the family	hypertension in motherFamilyHx, heart disease in familyFamilyHx
<b>HistoryOther</b>	labels all components in the history	he always come with his momHistoryOther
Allergies	labels the patient’s allergies	allergic to penicillinAllergies
Diet	terms referring to diet habits	on low carb dietDiet
Physical	terms describing the physical activity of patient	walks one hour dailyPhysical, sedentary lifestylePhysical
<b>ImagingProc</b>	labels all radiology tests including X-ray, ultrasound, MRI, etc.	
Imaging	all imaging-type diagnostic tests for the patient	X-rayImaging
Procedures	all procedures performed on the patient	DialysisProcedures
<b>Labs</b>	labels the lab tests that the patient performed or that are prescribed.	CBC Labs, lipid panelLabs
<b>Missed Names</b>	labels all names for physicians, patients, or others that may be missed	N/A

Table 2.1: Description of annotation scheme – continued from Table 2.1

<b>Label</b>	<b>Description</b>	<b>Examples</b>
<b>Medications</b>	medications patient is on or had	lipitor, losartan,
<b>Negation</b>	words that indicate negation	no, nor, absent
<b>Plan</b>	physicians plan for the patient	ObservationPlan
<b>PreviousDx</b>	previous diagnoses that the patient had	diabetic
<b>SignsByDr</b>	physical exam results	scar of back surgery
<b>Vitals</b>	labels the vital signs of the patient	ESR: normal
<b>LabResults</b>	labels the results of laboratories experiments	Stable Hct
<b>SocialHx</b>	labels social history terms	nonsmoker
<b>Surgeries</b>	labels the previous surgeries of patient	patient had open heart surgery
<b>Symptoms</b>	the symptoms associated with the patient's current illness	headache, dry cough
<b>Negating a list</b>	If a negation term is labeled negation independent of the symptom	No cough, fever, or chills
<b>ChiefComplaint</b>	the prominent symptom	Noisy breathing
<b>TemporalWords</b>	terms and phrases that refer to time description	in 2005, yesterday

## **Chapter 3**

# **Literature Review**

Many efforts have been made to derive a meaningful use of EMRs.

NLP in this field uses the enormous and rich data found in EMRs to extract and annotate useful information. This chapter will discuss related work in terms of data I leverage, systems, and then techniques.

## 3.1 Data

EMRs contain both structured information, known as clinical text, and unstructured data, known as free text. Clinical text is the most abundant data type but is also the most difficult to analyze computationally since it is highly heterogeneous, whereas free text allows flexibility [14].

The use of NLP in the field of medical information extraction has been widely adopted by many researchers.

NLP can be utilized for information retrieval, information retraction, and recognizing biological entities in text as discussed in [15].

## 3.2 Techniques

Many techniques and methods have been proposed to annotate and extract information with the lowest possible error, and surveys have been published to discuss existing work done in this field [7], [8], [9].

Epic Systems Corporation, or Epic, is a privately held health-care software company. According to the company, hospitals that use its software hold EMRs of 64 percent of patients in the United States and 2.5 percent of patients worldwide [5]

GNU Health is a Free project for health practitioners, health institutions and governments. It provides the functionality of EMRs, Hospital Management (HMIS) and Health Information System (HIS) [3] .

Health Information Text Extraction (HITEx) is a clinical NLP system from Brigham and Women's Hospital and Harvard Medical School incorporated within the Informatics for Integrating Biology and the Bedside (i2b2) toolset.

It Manipulates text reports to extract specific terms and knowledge from them. HITEx is an open-source software application. The software is built on top of Gate framework and uses Gate as a platform. It consists of the collection of Gate plug-ins that were developed to solve problems in medical domain and works by assembling these plug-ins into pipeline applications, along with other standard NLP plug-ins. It is less successful with the sparsity of our data and data from developing countries in general. [16]



IBM developed a biomedical-domain NLP system by collaborating on the development of a prototype system for text analysis, search, and text-mining methods to support problem solving in life science. The system is called “BioTeKS” (“Biological Text Knowledge Services”), and it integrates research technologies from multiple IBM Research labs.

BioTeKS is also the first major application of the UIMA (Unstructured Information Management Architecture) initiative also emerging from IBM Research.

BioTeKS is intended to analyze biomedical text such as MEDLINE™ abstracts, medical records, and patents; text is analyzed by automatically identifying terms or names corresponding to key biomedical entities (e.g., “genes,” “proteins,” “compounds,” or “drugs”) and concepts or facts related to them. They describe the value of text analysis in biomedical research, the development of the BioTeKS system, and applications which demonstrate its functions. [17] It is not open source and not trained for success with data from developing countries.

IBM also created a system, MedTAS/P which automatically instantiates the knowledge representation model from free-text pathology reports. MedTAS/P is based on an open-source framework and its components use NLP principles, machine learning and rules to discover and populate elements of the model.

To validate the model and measure the accuracy of MedTAS/P, they developed a gold-standard corpus of manually annotated colon cancer pathology reports. MedTAS/P achieves F1-scores of 0.97-1.0 for instantiating classes in the knowledge representation model such as histologies or anatomical sites, and F1-scores of 0.82-0.93 for primary tumors or lymph nodes, which require the extractions of relations. An F1-score of 0.65 is reported for metastatic tumors, a lower score predominantly due to a very small number of instances in the training and test sets.

It is also not open source and not tailored for success with data from developing countries [18].

The HITECH Act (Health Information Technology for Economic and Clinical Health Act) is a US federal law encouraging the adoption of EMRs [19]. It uses financial incentives to encourage practicing HCPs to adopt certified EMRs; through the increased use of these EMRs they hope to achieve a national health information network that results in improved quality of care, patient safety, and lower costs. It has made available billions in federal funding to encourage hospitals and health care providers (HCPs) to adopt EMRs [20].

The rationale behind this push was improving patient care by reducing

errors, encouraging better coordination of care, and allowing patients to take greater control over their own health care decisions.

Apache Clinical Text Analysis and Knowledge Extraction System (cTAKES) was released open-source and builds on existing open-source technologies. The OpenNLP NLP toolkit aims to build and evaluate an open-source NLP system for information extraction from EMR clinical free-text [4].

The cTAKES algorithm consists of a sequence of steps which uses a dictionary for the Named entity recognition (NER) annotator and the status and negation annotators.

They aim to build and evaluate an open-source NLP system for information extraction from EMR clinical free-text. The cTAKES annotations are the foundation for methods and modules for higher-level semantic processing of clinical free-text.

However, not all complex medical terms could be found in a regular English dictionary. The system is not trained for success in developing countries. [21]

### 3.3 Systems

In 2012, John Carroll and his team tried using distributional similarity to acquire lexical information from notes typed by physicians. They created a distributional thesaurus by calculating the similarity of contexts of every pair of words and limited to the words occurring at least  $N$  times.

The thesaurus has for each word included an entry of size  $k$  consisting of the  $k$  most similar words to it. This method produce highly accurate results for high-frequency words; yet less accuracy for less frequent words [10].

Columbia University's proprietary Medical Language Extraction and Encoding System (MedLEE) was designed to process radiology reports was later extended to other domains. It was tested for transferability to another institution, and proved able to discover clinical concepts along with a set of modifiers. It is an open-source NLP system used to extract and encode medical information from patient medical reports. It was found to have high precision but a low tolerance for grammatical and spelling mistakes.

Association rules and distributional similarity approaches use correlation between entities in free text to identify keywords. One of the main advantages of this system is that the order of words is not of importance to the model making it more tolerable to grammatical mistakes.

In paper [22], distributional similarity and co-occurrence relations were used by utilizing the KeyGraph method. This technique identifies keywords by highlighting frequently co-occurring words and links them together. This

method has shown high accuracy in determining co-occurrences between items in a data set.

In paper [23], distributional similarity measurements were studied to find the best ones for semantic similarity prediction. It was observed that weighting features by point-wise mutual information appears to be the most beneficial. The intuition behind this is that the occurrence of a less common feature is more important in characterizing a word than a more common feature.

Paper [24], described an investigation into the automatic estimation of the incidence of symptoms using coded and free text information in medical records. The algorithm consists of three steps performed in sequence.

First, locating an occurrence of textual description of read code in the text. Second, checking whether there is evidence of negation. Third, determining whether the located textual description is within the scope of the negation.

Paper [25] proposes the use of recurrent neural networks to capture sequential patterns present in data and the use of word embedding to capture semantic similarity of words and their study showed the effectiveness of the model.

In 2014, [26] used NLP and BN classifiers to evaluate factors affecting performance of influenza detection.

Their dataset consisted of 124 influenza patients and 87 non-influenza (shigellosis) patients.

They measured the overall accuracy, recall, and precision of Topaz and MedLEE parsers for 31 influenza-related findings against a reference standard established by three physician reviewers to assess NLP finding-extraction performance.

To quantify the relative contribution of NLP and BN classifier to classification performance, they compared the discriminative ability of nine combinations of finding-extraction methods (expert, Topaz, and MedLEE) and classifiers (one human-parameterized BN and two machine-parameterized BNs).

Classifiers using human-annotated findings were superior to classifiers using Topaz/MedLEE-extracted findings.

They also found that the machine-parameterized classifiers were superior to the human-parameterized classifier.

The classifiers using the 17 'most influential' findings were more accurate than classifiers using all 31 subject-matter expert-identified findings.

They commented that to improve results researchers should improve NLP accuracies.

In 2008, [27] described a NLP technique called Extracting Association Rules from Text (EART). This technique aims to automatically extract rules of association from a corpora of textual documents. EART depends on key-

word features that function as the document labels, whilst ignoring the order in which the words may occur.

Instead of placing emphasis on the word order, importance is placed on the words' distributions in these documents. Along with this statistical approach, they used XML with a Information Retrieval scheme (TFIDF) for entity selection. An entity may be a keyword or a feature. This, along with Data Mining techniques, were the main pathways to find rules of association.

They employed three steps in their work. The first involved a text processing phase. They processed the text with stemming, indexing, and filtering the documents. The second step was a association rule mining (ARM) phase that applies the algorithm that uses a weighting scheme to generate association rules, and a visualization phase to display the results [27].

They experimented to evaluate their results using web-pages of news documents covering the outbreak of a disease, namely the bird flu. Their method was successful in describing the informative news, and the performance of their system, EART, did well.

In 2015, Luis et al. also recognised the challenge that comes with the pursuit of analyzing medical records. Their corpora had specific characteristics such as plain text, very specific technical vocabulary, and the unstructured layout characteristic our corpora had as well. [28].

Since medical record analysis is an interdisciplinary pursuit, it inevitably would need co-operation from experts from several fields. Yet this work is very important and such an analysis can potentially help achieve many goals, including helping in making clinical decisions, classifying and organizing different procedures in medicine, and providing evidence based support for hospital management decisions to name a few.

This paper delves into the concepts involved, the relevant existent related work and the primary issues still prevalent and need to be tackled in future research within the field of analyzing EMRs, relying mostly on data mining techniques with NLP and text analysis [28].

In 2018, Li et al. proposed a workflow to extract prospecting information by NLP based on convolutional neural networks (CNNs) [29]. Their aim was to classify the text data and extract the prospecting information automatically. They used CNN while classifying the geo-science text data. They chose four classes to do this: "geology, geophysics, geochemistry, and remote sensing" [28].

Each of these classes have three types of text scales pertaining to them; one for words, the second for sentences, and the last for paragraphs. The word frequency statistics was calculated for the word type, the co-occurrence matrix statistics was calculated for the sentence type, and the term frequency-inverse

document frequency (TF-IDF) statistics was calculated for the paragraph type. Through this they were able to derive from their content the most important and telling nodes and links. They also used word clouds, knowledge graphs, and TF-IDF statistical graphs to visualize their work.

In the test case, the prospecting information was extracted successfully.

In 2017, Luo et al. proposed models classifying relations from clinical notes based on recurrent neural networks [25]. They used Long Short-Term Memory - LSTM recurrent neural networks and tested the models on the i2b2/VA relation classification challenge dataset [25].

One of their models, the segment LSTM model had only word embedding feature and no manual feature engineering. This model was able to achieve a highly averaged a 0.661 f-measure in performance for classifying medical problem-treatment relations. It also achieved a 0.800 f-measure while testing medical problem relations, and 0.683 f-measure for medical problem-medical problem relations.

They compared this model with the sentence LSTM model, and explored the difference between concept text and context text, and between different contextual parts in the sentence.

They explored the influence word embedding had on the LSTM models' performance, proving that medical domain word embedding help improve the relation classification.

In 2016, Luo et al. attempted to tackle the issue of efficiency when storing certain types of annotations. Specifically, the efficiency of storing them during a NLP attempt on free-writing clinical notes in EMRs (EMRs). This approach also aims to efficiently retrieve such annotations "that satisfy position constraints" [30].

They tested the interval query problem by performing a time complexity analysis on the basic interval tree query algorithm. They proved its imperfection when being applied to a collection of 13 query types from Allen's interval algebra [30].

Their proposed algorithm achieved logarithmic time and solved the stabbing-interval query tasks on Allen's relations in logarithmic time [30].

They also discussed interval management in external memory models and higher dimensions [30].

In 2017, Metskera et al. developed algorithms for the proper interpretation of medical records using a NLP approach that relies on specific patterns identified while studying medical databases in collaboration with a medical assistant [31]. Their aim was to be able to accurately reflect the medical processes for through their data.

Their method was developed during the study of actual Russian language

medical data of Acute coronary syndrome (ACS) patients from the specialized medical center. They demonstrated the efficiency of their method with a correlation analysis of comorbidities on the treatment duration of ACS.

They also did this in the case of extracted data to develop process models with complexity metrics at the control-flow perspective of process mining techniques [31].

Henriksson et al. also noted that the prediction of diagnosis codes is typically based on free-text entries in clinical documents[32]. They approached this by attempting to build a word space model based on a corpus of coded patient records, associating co-occurrences of words and ICD-10 codes.

Their method used random indexing, a computationally efficient implementation of the word space model. By using random indexing, they hoped to be able to reach an effective way of assigning diagnosis codes. Their method was evaluated for feasibility by a medical doctor at a qualitative standard. The test was performed on clinical records from two Swedish clinics. Initially the proposed code was among the top 10 generated suggestions in 20% of the cases, but a partial match in 77% proved the method could lead to more telling results if employed properly[32].

In 2012 Kushim et al. decided to extract useful information from nursing records within EMRs by employing a text data mining technique [33].

They determined the information with regards to the condition and treatment of patients by this technique that works by identifying the relations between prominent entities seen in past chronic hepatitis in-patient records gathered from the University of Miyazaki Hospital's EMRs [33]. They discovered entities relating to proper treatment methods, and were able to summarize the information in the records. They also could successfully identify important entities that characterize each nursing and passage record were also revealed.

In 2014, A. Thomas et al. performed a study to assess the validity of an NLP program. This program aimed to accurately identify patients with prostate cancer and to retrieve pertinent pathologic information from the corresponding EMR [34].

This program was being used to identify patients with prostate biopsies that were positive for prostatic adenocarcinoma unanimously across all pathology reports within this time span of the implementation [34].

The program was then used for the processing of 100 consecutive patients with prostate adenocarcinoma to identify and return 10 important associated entities from their pathology reports [34]. To evaluate their work, they used the program to process 18,453 pathology reports. The program proved able to properly detect 117 out of 118 patients (99.1%) with prostatic adenocarci-

noma after TRUS-guided prostate biopsy [34]. The program had a positive predictive value of 99.1% with a 99.1% sensitivity and a 99.9% specificity to correctly identify patients with prostatic adenocarcinoma after biopsy [34]. The overall ability of the NLP application to accurately extract variables from the pathology reports was 97.6% [34].

In 2016, a study examined whether incorporating information from text into case-detection algorithms can improve research quality [35]. They did a systematic search that returned 9659 papers. 67 of these papers expanded on the retrieval of relevant information from the free text subset of EMRs with the in order to identify cases of a specific mentioned condition [35].

They also reviewed different methods for extracting information from text and the studied the accuracy of these case-detection processes. They found that the studies mainly used US hospital-based EMRs, and retrieved information from texts corresponding to 41 conditions by using highly relevant and related word searches, rule-based algorithms, and machine learning methods [35]. In their findings there was no difference between rule-based and machine learning methods of retrieval in the correct identification of the corresponding cases. When specific entities from the text were used in the search a significant improvement in algorithm sensitivity was observed [35].

In 2019, B. Helgheim et al. developed a framework process for the integration of data from different sources to increase its usability potential [36].

They used data from an private hospital database, non-private data, as well as structured data extracted from NLP (NPL) that was applied to EMRs [36]. They used a process named extract-transform and load (ETL) in order to merge different data sources into a single one [36]. This contributed to more efficient use of the available resources.

In 2019, a Zitkus et al. presented a method for coreference resolution in Lithuanian language and its application for processing e-health records from a hospital reception [37]. Their method processes coreferences with minimal linguistic resources, which is important in linguistic applications for rare and endangered languages.

The experimental results show that coreference resolution is applicable to the development of NLP-powered online healthcare services in Lithuania [37].

In 2019, Assale et al. demonstrated the potential of NLP data extractions techniques, and discuss challenges still being faced that the technical communities of IT and medical practitioners should cooperate on understanding and solving. In this way they can work towards making full use of the unexplored resources unstructured content has to offer [38].

They gave a comprehensive literature review of the most recent and relevant contributions to leverage the application of NLP techniques to the free-

text content electronic patient records. Of the application fields they focused on four: data quality, information extraction, sentiment analysis and predictive models, and automated patient cohort selection [38]. Then, they presented a few empirical studies that were undertaken at a major teaching hospital specializing in musculoskeletal diseases [38]. They thus proved the possibility of reaching very high performance levels with a non-English dataset, that would meet academic and practical needs, taking a step further towards using the note data at our disposal [38].

In 2020, Lee et al. proposed a method to convert HLA genotype information stored in an unstructured format into a reusable structured format by extracting serotype/allele information [39].

They queried HLA electronic reports from the data-set of Seoul National University Hospital (SUPPREME). They did this over a time span of 18 years, from 2000 to 2018 as a rule-development data set (64,024 reports). In addition, from the 2018 reports they used 6,181 as a test set [39]. They used a rule-based NLP approach using a Python regex method to retrieve three main variables. Firstly the number of patients involved in the report, secondly medical informative traits such as information regarding the HLA testing, and finally specific HLA genotypes [39].

The performance of the rules and codes were tested through a comparison made between the retrieval results from the test set and a control test set manually performed [39].

They found that from an original 11,287 reports as a training set and 1,107 reports working as the test set, the method they employed successfully developed 124 extracting rules and 8 cleaning rules for HLA genotypes [39]. The application of these rules was able to retrieve HLA genotypes with 0.892-0.999 precision and 0.795-0.998 recall for the five HLA genes [39].

The precision and recall of the extracting rules for the number of patients in a report were 0.997 and 0.994 and those for the clinical variable extraction were 0.997 and 0.992, respectively. All extracted HLA alleles and serotypes were transformed according to formal HLA nomenclature by the cleaning rules [39].

In 2019, Koleck et al. searched 1964 records from PubMed and EMBASE was narrowed to 27 eligible articles to find data related to the NLP methods [40]. For each study, they looked at free-text corpus, patients, symptoms, NLP methodology, evaluation metrics, and quality indicators [40].

They found that symptom-related information was presented as a primary outcome in 14 studies [40]. EHM free-text represented many different types of patient data, including but not limited to general, cardiology, and mental health. These mentioned three had a pattern of being observed more than



other patient data. Studies included a large selection of symptoms, many of which happen to be shortness of breath, pain, nausea, dizziness, disturbed sleep, constipation, and depressed mood [40].

Their NLP method made use of previously developed NLP tools, classification methods, and manually curated rule-based processing [40].

Only one-third ( $n = 9$ ) of studies reported patient demographic characteristics [40].

In 2019, Green EP and Whitcomb A and Kahumbura C and et al.. applied a NLP approach to characterize the ways that Kenyan men and women communicated with the first iterations of askNivi, a free reproductive health information service [41].

They divided into entities and further processes over 179,000 anonymous messages that users exchanged with live agents [41]. As a follow-up, they performed two manual coding exercises. The first involved classifying the intent of 3,834 user messages in a training data-set [41]. The second involved coding all conversations between a random subset of 100 users whose chats were longer than was more common. They found that between September 2017 and January 2019, 28,021 users (mean age 22.5 years, 63% female) sent 87,180 messages to askNivi, and 18 agents sent 92,429 replies [41]. They observed different patterns of discussing family planning methods, contraception, side effects, pregnancy, menstruation, and sex. They found that good difference predictors happened to be sex and age [41]. User intents largely reflected the marketing focus on reproductive health, seeking factual information [41]. They also found that requests for advice and symptom reports were common [41].

In 2018, Xing et al.. introduced parallel processing on a supercomputer [42].

They developed paraBTM, a runnable framework that enables parallel NLP on the Tianhe-2 supercomputer [42].

It employs a low-cost yet effective load balancing strategy to maximize the efficiency of parallel processing. They evaluated its performance on several datasets, utilizing three types of named entity recognition tasks as demonstration [42].

Their results indicate in most instances that the processing efficiency can be greatly improved with parallel processing, and the proposed load balancing strategy is simple and effective [42].

In 2018, Wencheng et al. expanded on the process of EMR processing and emphatically analyzed the key techniques. They also make an extensive study on the programs developed based on NLP together with the current remaining challenges and research issues for future work [43].

They explain how since the EMR system has been recognized as a valuable resource for large-scale analysis, it must be utilized despite the hindrances that come with the EMR characteristics of diversity, incompleteness, redundancy, and privacy. Although these make it difficult to carry out data mining and analysis directly, an appropriate solution can be implemented by processing the source data in order to improve data quality and improve the data mining results [43].

They recognized that different processing methods perform best with different types of data. Most structured data commonly needs classic processing methods, including data cleansing, data integration, data transformation, and data reduction [43].

For semistructured or unstructured data, such as medical text, containing more health information, it requires more complex and challenging processing methods [43].

The task of information extraction for medical texts mainly includes NER (named-entity recognition) and RE (relation extraction)[43].

In 2013, Cohen et al. used text-mining methods for phenotype extraction since such methods can help disease modeling by mapping named-entities to terminologies and clustering semantically related terms [44]. They used EMR corpora and made use of the fact that health care practitioners typically copy and paste information from previous notes when documenting a current patient encounter [44].

They aimed to explore the problem of redundancy quantification whilst handling in large-scale text corpora. They also explore the problem of their observed EHR redundancy affecting NLP despite the large quality of their corpora that is believed to be boosting to results. They suspected that this may develop a bias that confuses accomplished models. They also question whether the redundancy is potentially beneficial as it emphasizes the most telling part of the data-set. They also aimed to tackle controlling the roll and impact of redundancy has in textual analysis. [44]

After analyzing large-scale EHR corpus and quantifying redundancy both in terms of word and semantic concept repetition, they observe redundancy levels of about 30% and non-standard distribution of both words and concepts [44].

They evaluated redundancy on two basic text-mining applications: collocation identification and topic modeling [44]. These methods performed on synthetic data were compared with controlled levels of redundancy and significant performance variation was observed [44]. To avoid redundancy-induced bias they compared two controlling strategies, namely a baseline strategy, keeping only the last note for each patient in the corpus; and a dele-

tion strategy that employed a fingerprinting based algorithm [44].

For NLP, preprocessing the EHR corpus with fingerprinting yields significantly better results [44].

They thus found that fingerprinting enables text-mining techniques to leverage available data in the EHR corpus, while avoiding the bias introduced by redundancy [44].

In 2016, Lv et al. proposed a method using entity dictionaries and dependency parser as the feature to do the classification of short texts in EMR [45].

It used NLP to preprocess the texts first including sentence segmentation, word segmentation, part of speech and entity extraction [45].

Then several entity dictionaries were built according to the result of thier performed NLP. After that the TF-IDF and LSA were deployed to select the vocabulary feature. After that, using the entity information relevant to the EMRs, dependency parser was done to the texts and triple dependency relation features would be used as the expanding feature for text classification [45].

Their results proved that even if a classification process depends solely on NLP techniques which make use of vocabulary features, the performance of classifier can be substantially improved by the performed methods.

This was proven using recall, precision, and F-value standards [45].

## **Chapter 4**

# **Bayesian Networks**

In this chapter will explain what BNs are, and introduce diagnostic algorithms. We will then give an example of how a diagnostic graph turned into a BN.

## 4.1 BNs

BNs provide a systematic method for structuring probabilistic information about a situation into a coherent whole.

It is a compact representation of a probability distribution that is usually too large to be handled using traditional specifications from probability and statistics such as tables and equations.[13]. They help automate the derivation of useful implications within the data to form conclusions and decisions about the corresponding issue at hand. BNs are a type of graphical model that encode the conditional probability between different learning variables in a directed acyclic graph.

There are benefits to using BNs compared to other unsupervised machine learning techniques.

It is easy to exploit expert knowledge in BN models. BN models have been found to be very robust in the presence of

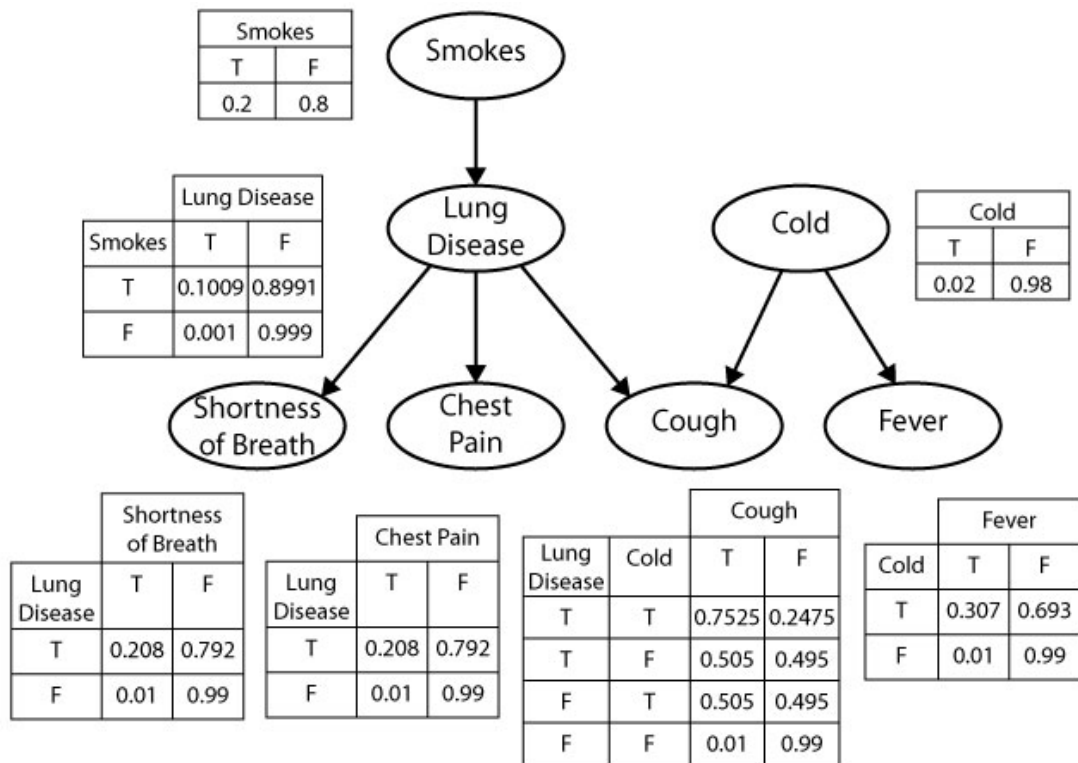
- Noisy data
- Missing data
- Sparse data.

Unlike many machine learning models (including Artificial Neural Network), which usually appear as a “black box,” all the parameters in BNs have an understandable semantic interpretation.

Every BN has two components: a directed acyclic graph (called a structure), and a set of conditional probability tables (CPTs). The nodes of a structure correspond to the variables of interest, and its edges have a formal interpretation in terms of probabilistic independence.

The network edges usually signify direct causal influences. A BN must include a CPT for each variable. The CPT of a variable  $v$  holds the probability of each possible value of the given values of other variables [46] In a BN, the CPT only needs to quantify the relationship between every variable and its parents, as the probability is only reliant on the parent nodes.

The figure ?? illustrates a BN for symptoms of smokers. An example of what we can query the BN is the probability of shortness of breath in a human given they do not have lung disease. We can also query for the probability of a cough symptom given the subject has a cold.



[47]

Figure 4.1: A BN for symptoms of smokers

## 4.2 Diagnostic Algorithms to Bayesian Networks

In clinical textbooks, students learn diagnostic algorithms that represent methods health care practitioners use for making a diagnosis. These algorithms are represented by graphs that involve a combination of symptoms, signs, or test results [11]. We took clinical diagnostic algorithms from the book *The Patient History: An Evidence-Based Approach to Differential Diagnosis* [11]. We coded them into graphs and used those graphs to build BN structures.

The structure of the BNs was found by a manual interpretation of the diagnosis algorithms. Since the diagnosis graphs can start with either a diagnosis or a symptom, the graphs we used changes the placement of the first node to the last node of each direction the BN path could follow.

An example of how the beginning of the diagnostic algorithm is turned into the first few BN nodes for the BN structure is displayed in the figures. Figure 4.2 shows the beginning of the diagnostic algorithm, figure 4.3 that shows the first few nodes of the BN, figure 4.4 shows the full diagnostic algorithm, and figure 4.5 shows the full abbreviated BN.

We learn the BN parameters by passing training data from EMRs we know to be relevant to the diagnosis at hand. We use an explanation query on our BN models the probability of each BN model explaining the note at hand.

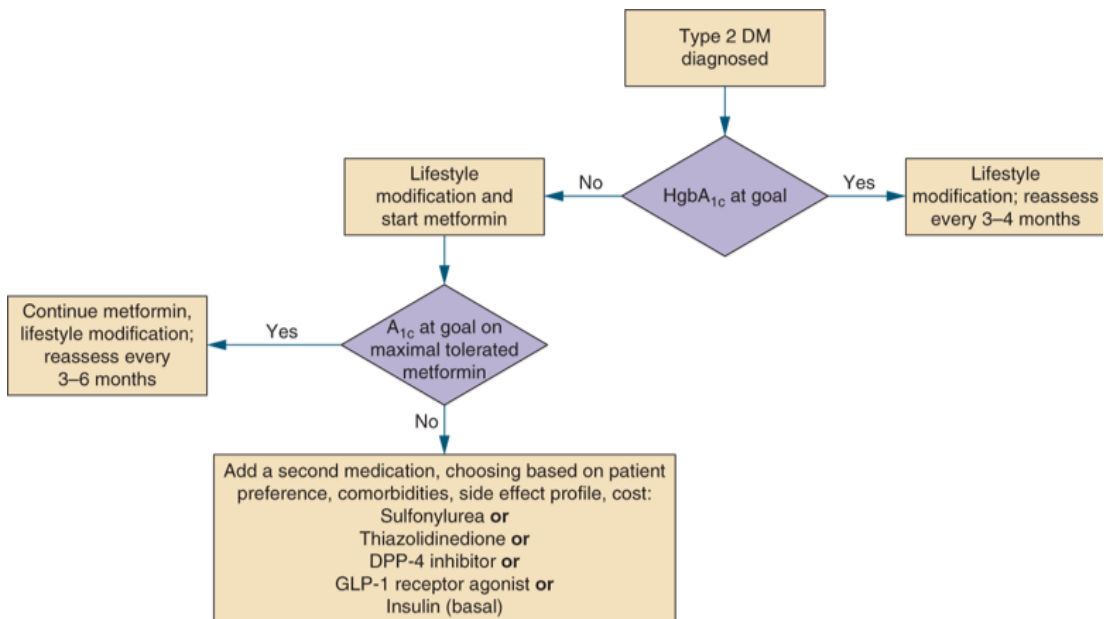


Figure 4.2: Beginning of the diagnostic algorithm

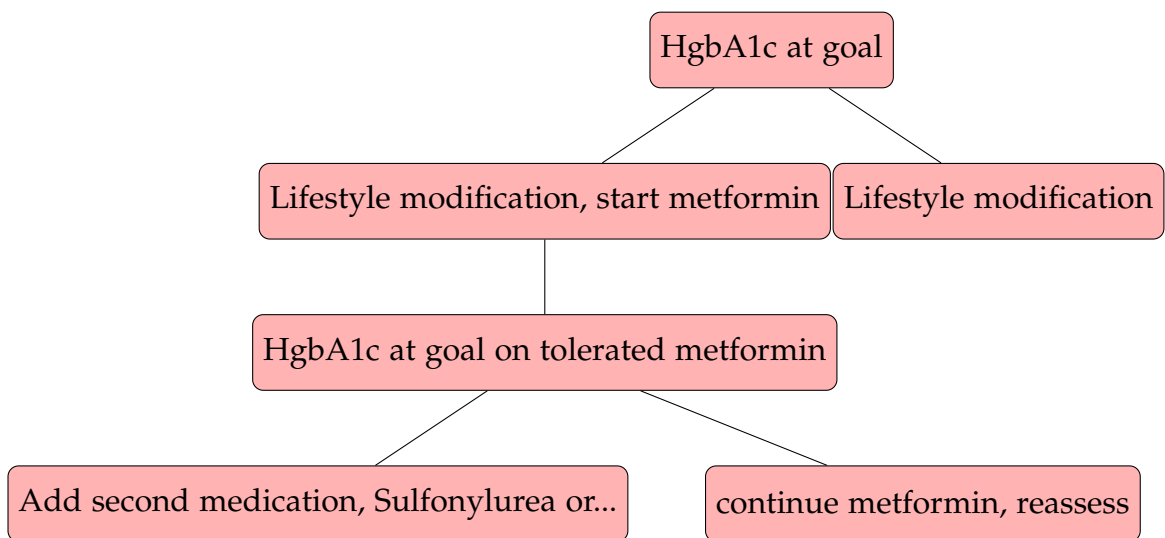
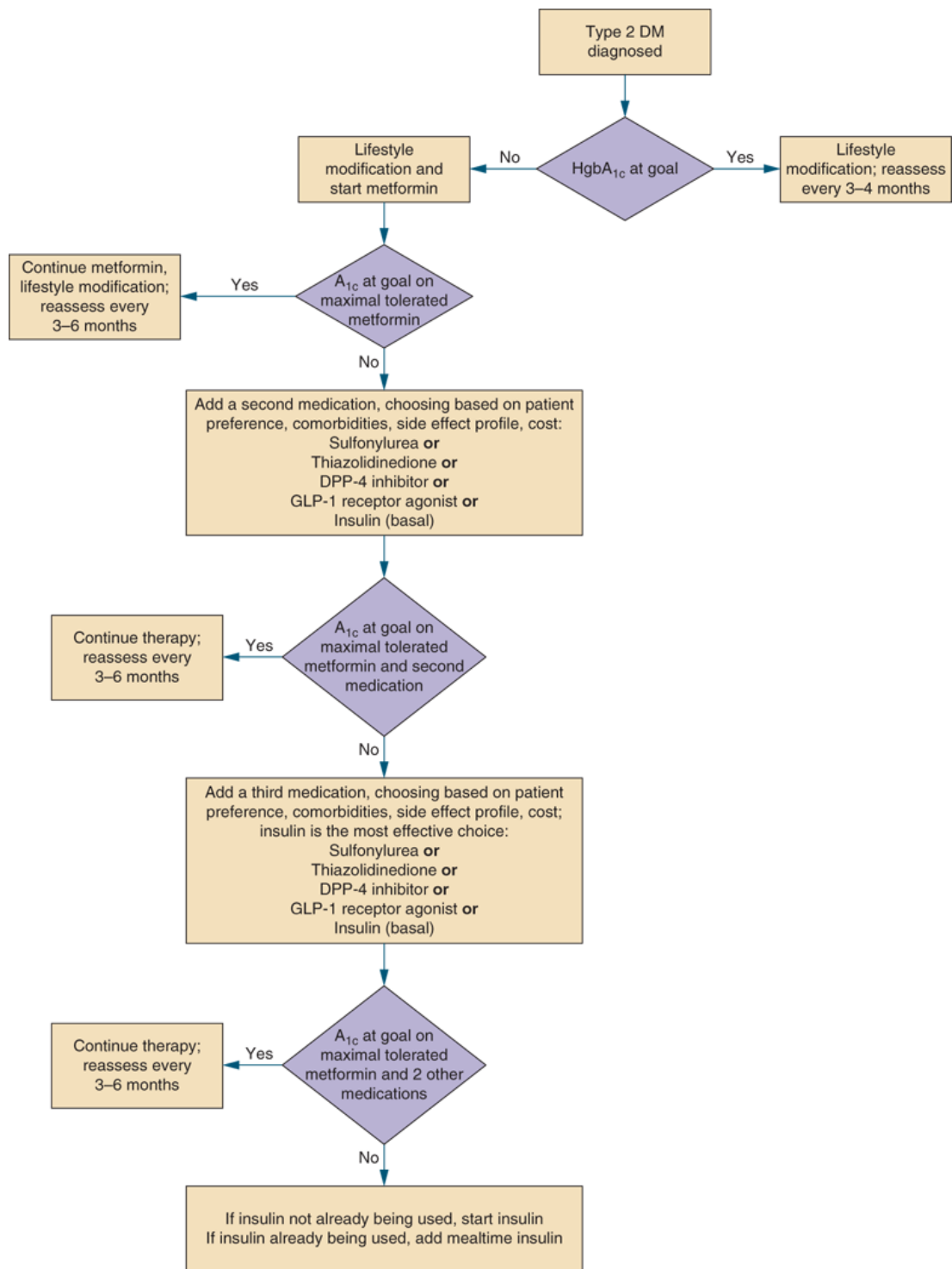


Figure 4.3: The first few BN nodes for the BN structure.





DM, diabetes mellitus; DPP4, dipeptidyl peptidase 4; GLP-1, glucagonlike peptide 1.

Figure 4.4: Full diagnostic algorithm.

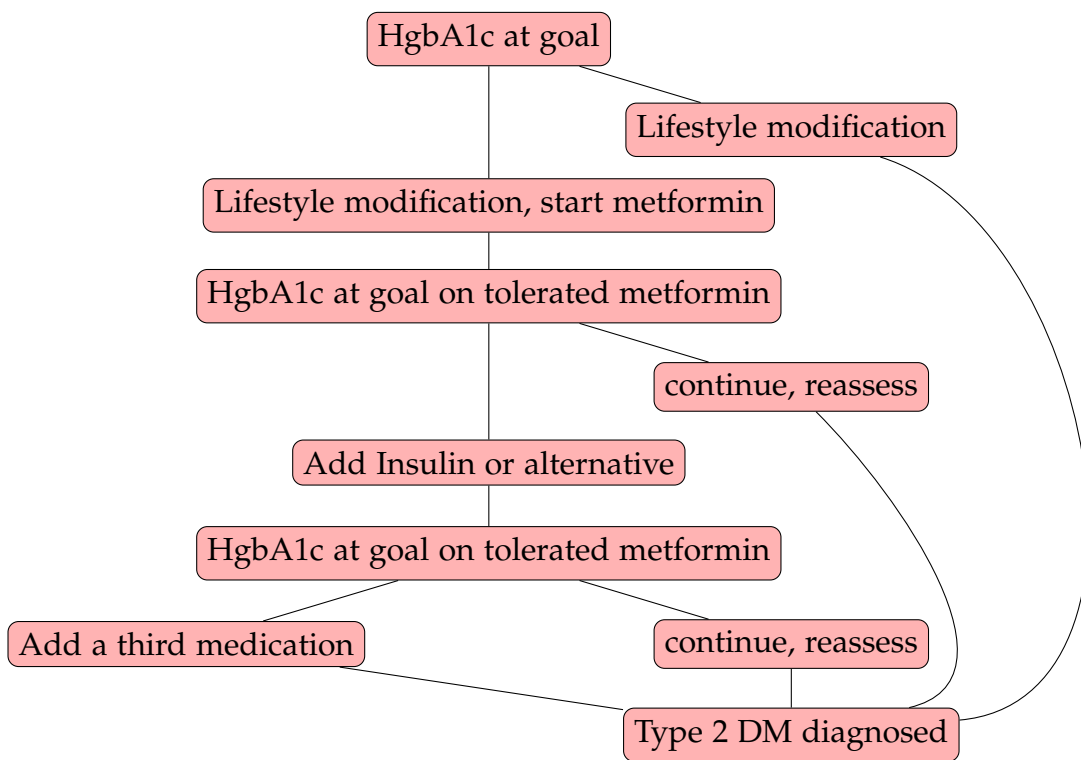


Figure 4.5: Full BN structure with abbreviated nodes.

## **Chapter 5**

# **Distributional Similarity**

## 5.1 Similarity metrics: Distributional similarity

The distributional similarity of a pair of words is computed based on the shared contexts of the two words. Distributional similarity scores between words using a specific distance measure. These scores enable us to create a distributional thesaurus, in which each word is associated with a list of other words with the highest distributional similarity scores.

Every word  $w$  will be associated with a set of features having specific frequencies. Each feature is a pair

$\langle r; x \rangle$  consisting of a relation name  $r$  and a word  $x$  that is related to word  $w$  via  $r$ .

To create the full distributional thesaurus, similarity of contexts for every pair of words are calculated and limited to those words that have a total feature frequency of at least  $N$ . A thesaurus entry of size  $k$  for some word  $w$  consists of the  $k$  most similar words to  $w$ . [2]

## 5.2 Sources of Distributional Similarity Thesauruses

For our distributional similarity thesauruses, we will be using three sources.

- The first is based on the distributional analysis of our EMR notes.
- The second is based on the distributional analysis of the USMLE test texts.
- The third the distributional analysis of phrases in Pubmed corpora based on the statistical analysis of large text collections and the creators offer 181 million token word space specifically for terminology used in the medical field [12]

To calculate the distributional similarity of word vectors, Lin's measure is used. Lin's measure builds on the concepts following. It is an information theory metric that provides a semantic distance. This is what we are looking for.[2]

Let  $w1$  and  $w2$  be the two words to be compared, and  $r$  be the position with respect to each other in the context at hand. Since we are working with a window size  $\pm 3$ ,  $r$  could be -3, -2, -1, 1, 2, or 3. Let  $f$  be the frequency of occurrence of the described state in the this equation describing Lin's measure. The asterisk \* stands for multiplication.

$$\log \frac{(f(w1, r, w2) - 0.95)(f(*, r, *))}{f(w1, r, *)f(*, r, w2)}$$

(5.1)

[2]

We also supplement this measure with the Euclidean distance measure where faced with a lack of information.

The formula for the Euclidean distance is as follows:

$$\sqrt{\sum_{i=1}^n (w1_i - w2_i)^2} \text{ [48]}$$

### 5.3 DISCO

DISCO package is a Java library that allows us to retrieve the semantic similarity between arbitrary words and phrases based on the statistical analysis of large text collections.

The metric is useful for retrieving the semantically most similar words for an input word, the value of the semantic similarity between two input words, collocations for an input word, and the semantic similarity between short texts.

The creators offer 181 million token word space specifically for terminology used in the medical field [12]. The corpus was taken from PubMed and tokenized, and highly frequent function words were eliminated.

They used a simple context window of size 3 words for counting co-occurrences.

The features that describe a word's distribution are ordered pairs of word and window position, and relies neither on part of speech tagging nor on lemmatization.

### 5.4 Distance Value Matrix

Let  $W$  be a vector of words of interest. An individual word distance (iwd) similarity vector has  $n$  elements where vector  $iwds(i)$  for word  $w$  is its distributional similarity with the  $i$ th word. For any given word  $w$ , this index provides us with its distributional similarity score with all the other core words, creating a very large distance value matrix.

This distance value matrix is used by the following two functions.

- a) Function  $similarlyDist(w, k)$  provides the top  $k$  closest words to  $w$ .
- b) Function  $distScore(w1, w2)$  provides a distance score between  $w1$  and  $w2$ .

We did the same for USMLE and constructed a USMLE based index. We have a set of 5,904 USMLE exam questions to enrich the relevant entity extraction and classification process. These exam questions are very similar to the EMR notes we have but instead follow the rules of proper English.

We also supplement this measure with the Euclidean distance measure where faced with lack of information.

The similarity of two words to one another is one minus their distance.

# Chapter 6

## Cross-Document Analysis Methodology

Cross-document analysis methodology aims at extracting entities and relational entities from unstructured information in EMRs, and relating them to diagnosis charts. It leverages diagnostic graphs and annotations made by health care providers (HCPs) to extract information from the EMRs. The EMRs will be processed by distributional similarity measurements[2] to identify words of importance in the EMRs.

To enrich the EMR notes with the diagnosis graphs, we will start with a simple string matching technique, and improve the results using a variety of word distance methods, such as the Levenshtein distance [49], Euclidean distance[48], Jaccard Similarity [50], the Jaro metric [51], and the distance metric DISCO [12].

The DISCO distance metric calculates semantic similarities between words and phrases based on the statistical analysis of large text collections and the creators offer 181 million token word space specifically for terminology used in the medical field [12].

The following metric is used to determine semantic similarity. We consider a EMR text  $t$ , made of words  $t_1, t_2, \dots, t_m$ , and diagnostic algorithm node  $u$  made of words  $u_1, u_2, \dots, u_n$ .

For each word  $t_i$  in  $t$ , we select  $p$  words most similar to it, using our cross DISCO/USMLE/EMR/Dictionary method. Thus,  $t_i$  has a corresponding set  $A_i$  of most similar words where  $A_i = \langle a_i^1, a_i^2, \dots, a_i^p \rangle$ . The union of the  $A_i$ s lies in set  $A$ , where  $A = \langle A_1, A_2, \dots, A_m \rangle$ .

Similarly, for each word  $u_i$  in  $u$ , we select  $p$  words most similar to it, using our cross DISCO/USMLE/EMR/Dictionary method. Thus,  $u_i$  has a corresponding set  $W_i$  of most similar words where  $W_i = \langle w_i^1, w_i^2, \dots, w_i^p \rangle$ .

The union of the  $W_i$ s lies in set  $W$ , where  $W = \langle W_1, W_2, \dots, W_n \rangle$ .

We denote the intersection of sets  $A$  and  $W$  with  $D$ . The intersection and union of sets  $A$  and  $W$  are described in the equations following.

$$A \cap W = D = \langle d_1, d_2, \dots, d_k \rangle \quad (6.1)$$

$$A \cup W = \langle d_1, d_2, \dots, d_k, A - D, W - D \rangle \quad (6.2)$$

Where  $D$  has  $k$  elements,  $A - D$  has  $k_a$  elements, and  $W - D$  has  $k_w$  elements.

To calculate the similarity between EMR text  $t$  and algorithm node  $d$ , we first consider the similarity measures of each word in  $D$ .

We consider the similarity measure of each word in  $D$  as occurring in the EMR text and as occurring in the diagnostic algorithm node. These are denoted in the following equations.

$$s_a = \langle s_{d_1}^a, s_{d_2}^a, \dots, s_{d_k}^a \rangle \quad (6.3)$$

$$s_w = \langle s_{d_1}^w, s_{d_2}^w, \dots, s_{d_k}^w \rangle \quad (6.4)$$

We then calculate the difference of the context metrics.

$$Difference = \langle s_{d_1}^a - s_{d_1}^w, s_{d_2}^a - s_{d_2}^w, \dots, s_{d_k}^a - s_{d_k}^w \rangle \quad (6.5)$$

Next, we consider the context metric of each word in  $A - D$  ( $ad$ ) as occurring in the EMR text and in  $W - D$  ( $wd$ ) as occurring in the diagnostic algorithm node.



$$p_a = \langle p_{ad_1}, p_{ad_2}, \dots, p_{ad_{k_a}} \rangle \quad (6.6)$$

$$p_w = \langle p_{wd_1}, p_{wd_2}, \dots, p_{wd_{k_w}} \rangle \quad (6.7)$$

To arrive at a word's distributionally similar words the next step is to compare every word vector with all other word vectors.

If the distances are relatively small and subsequently the similarity measures are large as determined by a certain threshold, the EMR note and the algorithm are taken to be similar, and pertaining to the same diagnosis.

We will build word lists from existing data, and add to it the results of the natural language processing techniques implemented. We intend to optimize our results with a reasonable trade-off between precision and recall. Leveraging diagnostic graphs by EMR notes enriched by annotations is a simple task. Understanding the notes in EMRs is more complicated. By projecting the diagnostic graphs on the annotated EMR notes, we hope our task to extract relevant and relational entities from the notes will be made easier.

## 6.1 Core word vector extraction

The annotated words are our core focus in our relational analysis. The informational vectors corresponding to these words will be extracted and used to compare with new data in order to find patterns and begin the automatic annotation of entities that are classified as symptoms and diagnosis.

## 6.2 Diagnostic graph enrichment

The diagnostic graphs will be processed through our algorithm after training in order to extract annotations found within it. Our goal is to enrich our annotations with relevant entities found in the graphs.

We then aim to translate the diagnostic graphs to a BN structure to be trained for its parameters (conditional probability tables).

## 6.3 Classification Boosted with Diagnostic Graphs

We discuss the implemented algorithm for the BN used to identify the most likely diagnosis of a given EMR 6.1.

We discuss the Augmentation Mechanisms in providing vectors of similar words for each word in the EMR and the BN nodes, as well as the respective scores of similarity for each word-to-word match.

We expand on the implemented algorithm for the how the augmentation results are saved, processed, and how we build information based on the initial results.

We then discuss how we calculated EMR-to-node and then EMR-to-BN scores based on an optimum threshold discovered by our initial results.

Next, we discuss how the information was organized and saved to be used in the training process for the BN.

We discuss the results the BN has been trained to derive; its conditional probabilities, to then be able to deduce the top most-likely BNs for a given EMR according to the similarity scores of then EMR at hand and the BNs it could correspond to.

The sources of our augmentations in order of importance are:

- 1) Our Manual Annotations of EMRs
- 2) Vowel-Based variations
- 3) EMR Distributional Similarity corpora
- 4) USMLE Distributional Similarity corpora
- 5) Pubmed corpora

## 6.4 Augmentation Mechanisms

These Augmentation mechanisms are key in facilitating the communication between the diagnostic BNs and EMR analysis.

The manual annotations of EMRs are searched for a match within the BN nodes that would clearly indicate when a match is identified. In such a case, as with the case of an exact string match, the word score match would be the highest it could; 1.

We also used the EMR distribution we calculated to find the most similarly distributed words to the particular word being searched for, and we add that to its vector of similar words.

The distribution we calculated and inferred word groups from also stored the similarity score between any two matches, that will come of use we deciding the similarity score threshold when it comes to deciding whether or not a

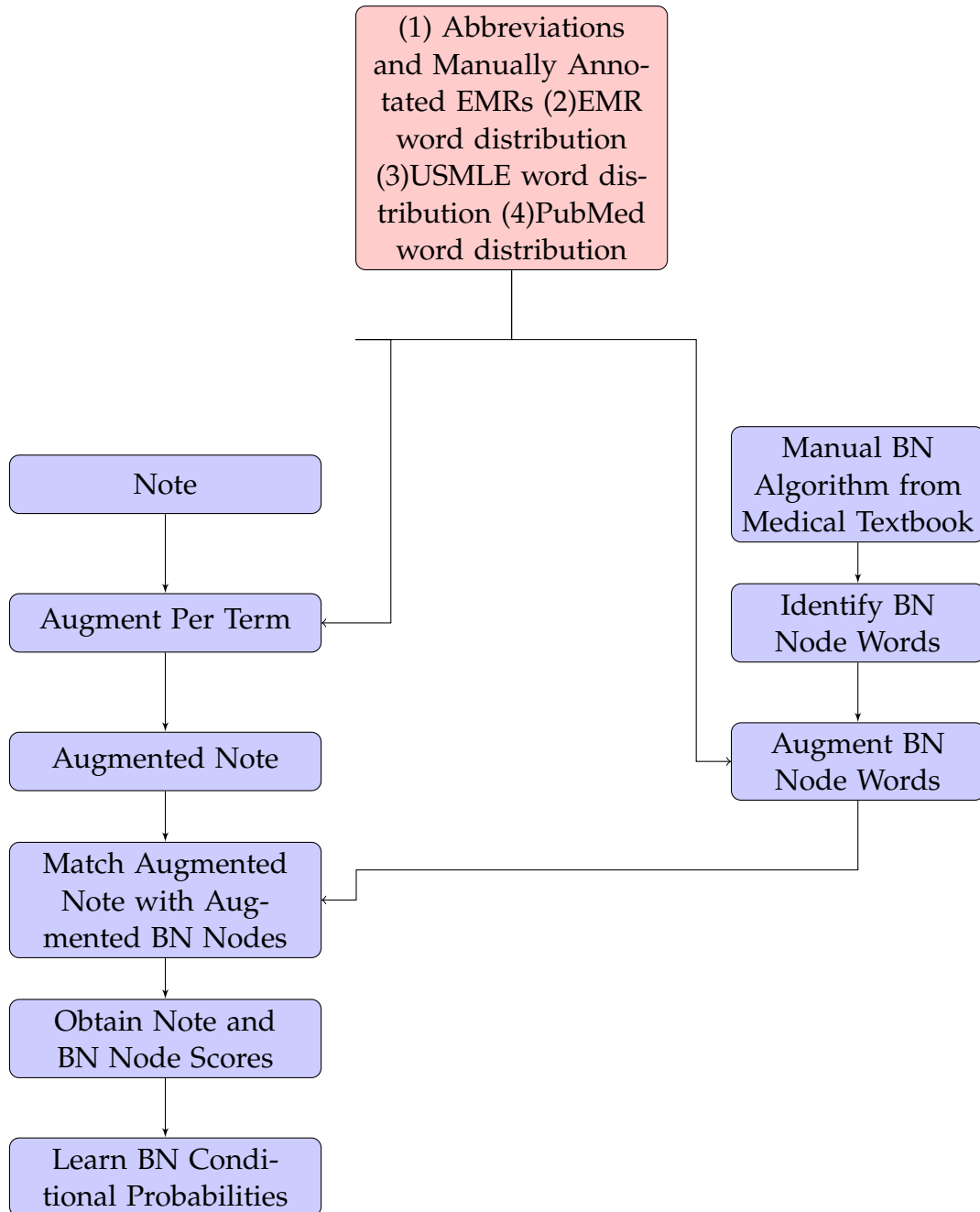


Figure 6.1: Flow chart for BN conditional probability learning procedure

word in an EMR, and potentially consequently the entire EMR, matched with a BN node.

We also check for word abbreviations; or vowel-based variations of our annotated EMRs. If a word is a vowel-based variation of a word we have information on, we use that word's information.

Similarly, we used the USMLE distribution we calculated to find the most similarly distributed words to the particular word being searched for, based on that corpora. We add that information to its vector of similar words.

The distribution we calculated based on the USMLE corpora that we then inferred word groups from also stored the similarity score between any two matches in that context. That score plays an indirect role in deciding the similarity score threshold when it comes to deciding whether or not a word in an EMR, and potentially consequently the entire EMR, matched with a BN node.

We also use the distribution based on PubMed corpora stored in the DISCO tool. This works in a similar way to our EMR and USMLE distribution-based methods, however this tool was developed prior to our work and we are using it along with the rest of our augmentation mechanisms.

The online synonym dictionaries are useful for string matching, but not always accurate, so they are given the smallest priority in the augmentation process. If this is the only source of a positive augmentation match between an EMR word and a BN word this is given a score of 0.8, which is lower than high match scores.

These scores will be combined into a score tailored for the purpose of our cross-document analysis. The supplementary dictionaries were built to assist the correction of typing errors and group words with similar meanings.

## **6.5 EMR Similarity Score**

A distributional similarity score based on our corpora was created. This was done by finding the distributional similarity of the annotated words. The euclidean distance between them was evaluated to produce a similarity score.

Our findings were stored and we use the function `distScoreEMR()` to obtain a distance score for any word pair.

## **6.6 Vowel Based Variations**

The final similarity score is based on a series of variations of the annotated words, produced automatically by removing the vowels in a given word.

A JAVA structure (HashMap) was built mapping every symptom to all the words that are a variation of it and represent the same word (vowel version and abbreviations). The scores of the closeness of the word variations to the original word were calculated using DISCO.

We built a function `variation()` which takes a variation and returns the original word along with the DISCO similarity score.

## 6.7 Dictionary Based Augmentation of Symptoms

The International Statistical Classification of Diseases and Related Health Problems was used to extract a dictionary of symptoms [52] [53].

Our implementation confirmed that all words annotated as symptoms in our corpora were included in this dictionary.

Using this dictionary, we produced a function `isDictionarySymptom()` that returns a Boolean specifying whether a word is classified as a symptom. We also use Roget's 21st Century Thesaurus [54] to relate symptoms to synonyms. We produced a function `Synonym()` that returns a list of synonyms accommodating all symptoms. This work can be applied to other interesting classes such as diagnoses, tests, etc.

The third dictionary is a list of common medical abbreviations that are found online [55]. We developed function `abbreviation()` that takes a word `w` and returns suggested medical terms such that `w` may be an abbreviation of each medical term `m` in the set of medical terms `mt`.

These dictionaries were combined and used for string matching and symptom identification.

## 6.8 Word Similarity Score Calculation using Augmentations Algorithm

The algorithm to calculate and store all the word similarity scores using the augmentation starts with a loop over the EMRs we want to augment words for and find respective scores for.

Inside the loops, we retrieve the EMR text, and then loop through the nodes of the BN we would like to compare the text to and find the scores with.

For each node of the BN, we loop through its individual node words.

If the word of a given node is not a stop-word that does not give any information, we continue with the loop.

We derive the augmentation vector (augNW) for the BN node word at hand using the augmentation sources we have.

We then loop through every word in the EMR note, and find the augmentation vector (augEMRW) for the EMR note word at hand using the augmentation sources we have.

If the augmentation vectors augEMRw and augNW have one or more identical terms, we store them in an ArrayList and treat them as matches.

We loop through these matches and find the scores of the match word with the original EMR notes word and the BN node word.

Whichever score is higher is taken to be the matched score of that matched word with the EMR word and BN Node word.

The way the augmentation vectors and there scores are stored is using two new constructor classes, bnConnections and augConnections.

augConnections has three objects; two are exactly the same; they are the two match strings that are identical matches from the EMR word augmentation vector and the BN node word augmentation vector.

The third one is for the score of this augmentation match. It is the higher of the two scores of the EMR word score with the augmentation match and the BN node word score with the augmentation match.

bnConnections has four objects. It has two strings, one for the EMR word being compared against a BN node word, and the other for the BN node word.

The third object is the score of that match, which was calculated after finding the match scores of the two augmentation vectors and their match scores.

The final object is an array list of augConnections which stores all the values of the matched augmentations of the two augmentation vectors and their match scores.

We added to the original Note constructor has an object for an array list of bnConnections for each of the non-stop words in that EMR note.

## 6.9 Score Matrix

We built a matrix of all the scores of every vector of augmentations for each BN node word and EMR word. Table 6.1 is a reference for the notations we will be using to explain our algorithm. We will define every term as we go through the thesis. The matching augmentation scores were calculated as follows; if augmentation  $nWA_i$  pertains to  $nW_j$  with a score 0.7, and  $nWA_i$  is an exact match to  $eWA_k$  that pertains to  $eW_n$  with a score 0.8, we match  $nW_j$

Term	Represents
$BN_i$	BN index i
$n_i$	BN Node index i
$E_i$	EMR index i
$nW_i$	BN Node Word index i
$n_i$	Node index i
$eW_i$	EMR Word index i
$nWA_i$	BN Node Word Augmentation index i
$eWA_i$	EMR Word Augmentation index i
$s(E, n)$	Score of EMR and Node
$S(wX, wY)$	Score of word x and word y

Table 6.1: Reference for notations

to  $eW_n$  with the higher of the two scores, namely 0.8 in this case. The scores stored as shown in the tables.

## 6.10 Calculations

We have the entire matrix of word augmentation scores with EMR words and BN node words.

Once the code is done running, we need to determine the following.

- Whether a word augmentation scores high enough to match the EMR word and BN node word (EMR-word to node-word match).
- Whether a BN node has enough words that score high enough to match with an EMR (EMR to node match).
- Whether the EMR note matches the BN (EMR to BN match).

We will now be discussing how we are going to decide how to determine these, and how we need to display the word augmentation matrix of scores so that we can look at them and decide the thresholds.

## 6.11 Diagnostic Algorithms

The structure of the BNs was found by a manual interpretation of the diagnosis algorithms. Since the diagnosis graphs can start with either a diagnosis or a symptom, the graphs we used changes the placement of the first node to the last node of each direction the BN path could follow.

	$nW_1$	$nW_2$ ...
	$nWA_1$   $nWA_2$	$nWA_1$   $nWA_2$
	...	...
$eWA_1$	$s(eWA_1, nWA_1)$ $s(eWA_1, nWA_2)$  ...	$s(eWA_1, nWA_1)$  ...
$eWA_2$	$s(eWA_2, nWA_1)$ $s(eWA_2, nWA_2)$  ...	$s(eWA_2, nWA_1)$  ...
$eWA_3$	$s(eWA_3, nWA_1)$ $s(eWA_3, nWA_2)$  ...	$s(eWA_3, nWA_1)$  ...
·		
·		
·		

Table 6.2: This table shows the matrix we store of the scores of each word augmentation of a single word in a given EMR with each word augmentation of a single word in a given node of a given BN

	$n_1$	$n_2$
...		
	$nW_1$   $nW_2$  ...	$nW_1$   $nW_2$  ...
...		
$eW_1$	$s(eW_1, nW_1)$   $s(eW_1, nW_2)$	$s(eW_1, nW_1)$   $s(eW_1, nW_2)$
...	...	...
$eW_2$	$s(eW_2, nW_1)$   $s(eW_2, nW_2)$	$s(eW_2, nW_1)$   $s(eW_2, nW_2)$
...	...	...
$eW_3$	$s(eW_3, nW_1)$   $s(eW_3, nW_2)$	$s(eW_3, nW_1)$   $s(eW_3, nW_2)$
...	...	...
·		
·		
·		

Table 6.3: This table shows the matrix we store of the scores of each word in a given EMR with each word in a given node of a given BN



In this way, each BN graph is defined by a set of vertices and directed edges.

The function `augment()` takes a word and returns a vector of words using one of the augmentation mechanisms, or a combination of more than one.

Training the BNs:

Each state of our BN represents a node in the BN graph.

Since we already have the structure of our BN, all we need is to learn the conditional probabilities of this BN.

The data that was given to the BN for training was based on the matrix that represents the score of an EMR with each node discussed earlier.

The BN works by calculating a probability of the existence of a state *Y* given the existence or non-existence of state *X*. Therefore, probabilistic data cannot be fed as is. If we were to, would not be giving the BN model the existence information, but the probability of a state existence.

Since the score numbers cannot be states since they are not patterns of true/false, we translate the score numbers into states so that the BN packages can make sense of them.

A state existing will be represented by "true" and it not existing will be represented by "false". In each array of data there is as many states as there are nodes, keeping the number of columns in each row consistent with the number of nodes of the BN at hand.

We removed the words pertaining to the BN that scored very high in any medical record.

This is either due to (1) it being an highly common word used in medicine or (2) it having too many relations to words.

We are left only with words that are more specific to the BN at hand. We made BNs for ten different diagnosis: Anemia, Anxiety, Diabetes, Fatigue, Headache, Hemoptysis, Joint Sprain, Kidney Disease, Pruritus, and Tinnitus. The words retained from each diagnostic algorithm in the corresponding BN are displayed below.

Words retained for the Anemia BN:

anemia, WBC, iron, tbc, ferritin, normocytic, myeloma, Sickle, hemolytic, mcv, marrow, b12, thrombocytopenia, macrocytic

Words retained for the Anxiety BN:

anxiety, phobia, agrophobia, trauma, ocdapprehension, ptsd, uneasiness, fear, impairment, attack, panic, trigger

Words retained for the Diabetes BN:

diabetes, mellitus, Lifestyle, metformin, comorbidities, Sulfonylurea, DPP—4, inhibitor, GLP-I, receptor, basal, Insulin, agonist, mealtime, HgbA1c

Words retained for the Fatigue BN:

fatigue, depression, anxiety

Words retained for the Headache BN:

headache, seizure, throbbing, migraine, meningitis, triggers, cough, sexual, tumor, cns, neuroimaging

Words retained for the Hemoptysis BN:

hemoptysis, airway, chest, bleeding, parenchymal, respiratory, pseudohe-  
moptysis, hematemesis, gastrointestinal

Words retained for the Joint Sprain BN:

joint, periarticular, monoarticular, inflammatory, inflammation, gout, arthropathy, arthritis, rheumatologic

Words retained for the Kidney Disease BN:

kidney, nephrotoxic, hydronephrosis, toxin, nephritic, renal, glomerulonephri-  
tis, hypoperfusion, rehydration, creatinine

Words retained for the Pruritis BN:

pruritis, menstrual, menopause, polyuria, thyroid, hiv, parasitic, weight,  
fever, jaundice, malignancy

Words retained for the Tinnitus BN:

tinnitus, osteosclerosis, neuroma, meniere, spasm, sound, pulsatile, cardiac,  
hyperthyroidism, vascular, intracranial, cerumen, malformation

The second thing we did was stabilize the thresholds of similarity for all the word augmentation sources at 0.4 and test different the thresholds (0.6, 0.65, 0.7, 0.75, 0.8, 0.9) to identify a match to see if the results are closer to what we expect.

This showed results as expected. For higher thresholds the matches became little to none in the non-related notes while score matches are being found for the notes that are related. For lower thresholds the matches were existent in both related and non-related notes, while the percentage of matches are usually higher in notes that are related.

## 6.12 Score Aggregation

We decided to aggregate the EMR word - BN node word score to include the highest three scores of their augmentation vector string match. It is calculated by adding the maximum string match score to 10% of the second to maximum string match score and 5% of the third to maximum string match score. If this is higher than 1 the score becomes 1.

The we translated the augmentation scores to a form the BN program can understand (true/false).

## 6.13 Learning BN Parameters

We used the python package pomegranate to learn the parameters of the BN network for which the structure has been previously identified.

The network structure is passed as a tuple of tuples, along with the samples from the data to the `from_structure` api.

Pomegranate takes the desired structure, and calculates the parameters of the BN network model in accordance to the data.

Once the parameters are calculated, the BN network is complete and ready to be queried.

We use `predict_proba` to compute the missing probabilities when we present evidence.

It is also very useful for computing conditional probabilities of the values of certain nodes, given others.

The command `predict` gives us missing values in terms of the structure.

For each set of notes corresponding to a diagnosis, we calculated it's score with all ten available BNs at seven different thresholds (0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9).

We did this by finding the match data, translating it to the correct form corresponding to each BN to query the likelihood it corresponds to this BN.

In this way, we can find the top BNs capable of explaining a note at hand according to our work.

The results are displayed in the results section (Section 10).

# Chapter 7

## Automatic Annotation Tool Code

This section discusses the automatic annotation tool code.

### 7.1 Constructors

We created a constructor to define an Annotation; Annotation.java.

Each annotation has a label classifying it, the text that was classified, and the index of the first letter's positioning in the note text.

We created a constructor to define a Note; Note.java.

Each note has saves the patient's record ID, case code, doctor identification code, two numbers representing the patient's first and last names, the date of the visits, the note text to be analyzed, the patient's age group, gender, the number of diagnoses associated to the patients, a code representing the diagnosis and a list of those diagnoses.

We added a feature that allows us to save the number of annotations for this note text and a list of the annotations manually made by real doctors for the annotated notes. This is used for annotated notes and left empty for the non-annotated notes. We automatically annotate these later on in our work.

A Node is a constructor used to store a node in a BN. It contains the node's ID, its Label, its Type, its Color, and its Chart availability.

The node's ID is a number attributed to each node in the BN.

Its Label is the actual text of Node.

Its Type classifies the Label, it is similar to the label of the annotated notes.

Its Color is the color of the node in the visual version of the BN.

Its chart availability indicates whether this node is present in the visual version of the BN.

An Edge is a constructor used to store the structure of the BN. Each edge contains a SourceID, a TargetID, and a Command.

The SourceID represents the node ID of the origin of the BN path.

The TargetID represents the node ID of the destination of the BN path.

The Command is either "yes" or "no". The code checks for status the of the condition described in the label that pertains to the node with ID SourceID. Depending on the existence of this condition; "yes" or "no", the ID of the next Node to check for ( TargetID ) is specified.

A graph is a constructor used to store BNs. It contains an array of Nodes ( Node[] ), an array of edges ( Edge[] ), and a HashMap of integers ( that pertain to Node IDs ) that point to a list of edges to summarize the structure of the BNs and simplify the process of handling any future queries.

## 7.2 Dictionary Synonyms

The method *synonymsOnline* creates the dictionary word vector for any relevant word. A word of importance (  $w$  ) is passed to it and it returns an ArrayList of the words that are identified as synonyms to this word by the online dictionary. The dictionary used is found online at <https://www.thesaurus.com>.

The code accesses the internet and queries for  $w$ . It then parses the results and extracts each word identified as synonyms to  $w$ . It does this by first using a while loop to ensure the entire text of the page of results read. Once it saves the results, it searches for the definition of  $w$  and the first 10 synonyms in the page text. It adds these to the ArrayList of synonyms ( *wvector* ) as soon as they are found.

## 7.3 Co-occurrence Frequency Calculation

### 7.3.1 EMR

The method *buildallNotes* takes an String with the address of an xml file and builds a public array list ( allNotes ) of the notes in the file. The function does this by reading the passed xml file that contains notes and processing them. It reads the lines of the file in order to identify each note and its attributes.

It then uses the note constructor to define and save each note. Once the note is constructed and all the information it hold is saved, the note is added to the array list allNotes. Once all the lines of the xml file are read we are left with the final version of allNotes that holds a comprehensive list of notes that were processed.

The notes can now be easily accessed at any point in the code.

Two versions exist for this code. One is for the annotated notes and the other is for the non-annotated notes. This is because the content of the files

that contain the annotated notes differ slightly from the the files that contain the non-annotated notes and this minimizes the chance of error.

The method *clean* takes a word and removes any punctuation marks with spaces and removes highly frequent and repetitive words that are not likely to benefit the analysis process. This method is called by *buildallNotes* in the stage of reading and saving the note text.

To build the co-occurrence matrix, we create a public nested *HashMap* *coreDist* that is prone to alteration in all methods, and undergoes alteration in a specific sequence as specified by the main method.

*coreDist* is a *HashMap* with keys that are words ( $w_1(i)$ ) of type *String* that point to *HashMap*s containing keys that are co-occurring words ( $w_2(j)$ ) of type *String* that point to the frequency  $f_{12}(i - j)$  of the co-occurrence of  $w_1(i)$  and  $w_2(j)$  of type *Double*.

The method *getOrMakeRow* takes a word and returns the *HashMap* in *coreDist* corresponding to the passed word. It does this by first checking if *coreDist* has a key that matches that word.

If it does, it returns the *HashMap*s containing keys that are co-occurring words with the passed word that point to the respective co-occurrence frequencies.

If it does not, it creates a new *HashMap* with the corresponding structure of *String* keys pointing to objects of type *Double* and assigns it to the word passed.

The method *addPairToCorehandleW2* takes a *HashMap* of words to frequencies ( an object of one of the keys in *coreDist* ) and a word ( $w_2(j)$ ) that should be included as a key in this *HashMap*.

It alters the *HashMap* by locating the object  $f_{12}(i - j)$  of the word ( $w_2(j)$ ) if it exists and adding one to it, or creating a  $f_{12}(i - j) = 1$  for ( $w_2(j)$ ) and adding it to the *HashMap*.

The method *addPairToCore* takes two words that have co-occurred, *word1* and *word2* , and updates *coreDist* accordingly.

It does this by first retrieving the *HashMap* (*w1Row*) that stores the words and respective frequencies that *word1* co-occurs with. To do this, it calls *getOrMakeRow* and passes *word1* to it.

It then updates this *HashMap* by passing *w1Row* and *word2* to *addPairToCorehandleW2* .

To calculate the frequency of occurrence of *word1* in general, at this point the code also passes *w1Row* and "\*" to *addPairToCorehandleW2* .

In this way, to determine the frequency of occurrence of *word1* we can get the object in *w1Row* stored with key "\*" after we are done processing all the notes. The code then updates *coreDist* by pointing *word1* to *w1Row*.

The method `buildAnntoRand` takes the array list of notes and builds the *coreDist* .

It does this through looping through all the notes in `allNotes` and accessing the note text for each note. The size of this loop is the number of notes in `allNotes`.

Once the text is accessed, the words are separated into an array of words ( `words[]` ).

Another loop of the same size as `words[]` goes through each word in the text. In this loop, the word at index *i* in `words[]` ( `words[i]` ) is treated as *word1* , and the method `addPairToCore` needs to be called with all the words that count as *word2* to *word1* .

In other words, we will be passing *word1* to `addPairToCore` along with each of the words co-occurring with it. The words co-occurring with it are defined as words either receding or preceding *word1* , and for our code we use an index window to capture a certain number of words both immediately after and immediately before *word1* .

Our window was taken to be 3, which means the three preceding words and the three receding words are treated as co-occurring words. These six words thus need to each be passed through a loop to `addPairToCore` as *word2* .

This is done through two nested loops. The first loop size depends of the index *i* of *word1* ( `words[i]` ) and on the size of `words[]`.

So for example if  $i=0$ ,  $i-3 = -3$  which is not a valid index. The first word cannot co-occur with what is before it as there is nothing before it. To tend to this issue, we define *kminus* and *kplus*. *kminus* is either zero or  $i - \text{window}$ .

It is zero if  $i - \text{window}$  is less than zero and it is  $i - \text{window}$  otherwise. *kplus* is either the length of `words[]` -1 or  $i + \text{window}$ . If  $i + \text{window}$  is less than length of `words[]` *kplus* is  $i + \text{window}$  and *kplus* is length of `words[]` -1 ( the index of the last word in `words[]` ) otherwise.

The first loop thus goes from  $\text{cursor} = \text{kminustocursor} = \text{kplus}$ , and we increment by adding 1 and skipping *i* itself in this range, so the loop size is  $\text{kplus} - \text{kminus} - 1$ . Inside this loop, *i* is skipped using an if statement; if `cursor` is equal to *i*, continue to the next increment of `cursor` and ignore the rest of the commands in the loop.

We then use the second loop to pass *word1* and *word2* to `addPairToCore`.

The second loop size depends on the co-occurrence distance between *word1* and *word2* . This is because our window will include words immediately preceding *word1* and words immediately receding *word1* as co-occurring with a window of 1, 2, 3, and all higher numbers.

Similarly, our window will include words preceding *word1* by two words

and words receding *word1* by two words as co-occurring with a window of 2, 3, and all higher numbers.

Therefore, with a window of 3, words immediately receding and preceding *word1* will be counted as co-occurring three times ( with a window of 1, 2, and 3 ), words immediately receding and preceding *word1* by two words will be counted as co-occurring two times ( with a window of 2 and 3 ), and words immediately receding and preceding *word1* by three words will be counted as co-occurring one time ( with a window of 3 ).

The size of this loop is the difference between absolute difference between the cursor and *i* ( called width ) and the window size + 1. In other words, the size of the loop is  $\text{width} + 1 - |\text{cursor} - i|$ . So if  $\text{window} = 3$  and  $|\text{cursor} - i| = 3$ , the loop will run  $3 + 1 - 3$  times ( once ). This will happen when *word2* is preceding or receding *word1* by three words.

If *word2* immediately precedes or recedes *word1* ,  $|\text{cursor} - i|$  will be 1, so the loop will run  $3 + 1 - 1$  times ( thrice ). Inside the loop, *word1* and *word2* are finally passed to *addPairToCore*.

### 7.3.2 USMLE

A similar method to finding the co-occurrence frequencies of the EMR note texts was implemented here.

The main difference was the processing of the USMLE test texts.

The method *buildallTests* read the text file containing the USMLE test texts and processing them, adding each test text to the Array of texts *allTests*. The method *clean* defined previously is called by *buildallTests* as well to remove irrelevant information from the test texts before adding them to *allTests*.

The methods *getOrMakeRow*, *addPairToCorehandleW2*, and *addPairToCore* remain the exact same.

The method *buildAnntoRand* only differs in that the outermost loop loops over *allTests* and is of the same size as *allTests*. For each test text in *allTests* it splits the words to *words[]* and continues in an identical manner to that found in the EMR co-occurrence frequency calculation process.

## 7.4 DescendingProbabilityComparator

The class *DescendingProbabilityComparator* is a manual comparator that compares pairs that contain words and numbers of type double and returns the sorted version in a descending order.



## 7.5 Finding Top $K$ similarly distributed Words

The *kTopSimilarWordsforW* method aims to identify the  $K$  most similarly distributed words to the word queried. This method works for both EMR and USMLE data as we pass it the name of the file it needs to load depending on which of the two sets of data.

This file contains all the co-occurrence frequencies in the corpus and it works the same from there. At the end our code saves the results to a file that we specify a name for depending on which of the two sets of data.

*kTopSimilarWordsforW* takes as input the word in question ( called  $w$  ), the distribution map of the corpus - either USMLE, DISCO, EMR - ( called *map* ), the distance metric the distributions will be measured against each other by ( called *metric* ), and the number of most similarly distributed words we would like to retrieve ( called  $K$  ).

It returns an ordered Array List of size  $K$  ( called *topK* ) containing pairs of words and distances ( the format is `ArrayList<Pair<String,Double>>` ).

The words in *topK* are the most similarly distributed ones to  $w$ , and the distances are the difference between the distribution of each of the words in *topK* and  $w$ .

It does this by defining the `DescendingProbabilityComparator dpc` and defining *topK*. It then retrieves the distribution of  $w$  from *map* ( called *forW* ).

It loops over the Strings in *forW*, calling each  $w2$  and checks if *map* contains a distribution for  $w2$ . If it does ( it should ), we retrieve the distribution of  $w2$  ( called *forW2* ) and calculate the distance between  $w$  and  $w2$ .

The distance metric used depends on the metric passed to

*kTopSimilarWordsforW*. Once the specific distance method is called and saved as distance, we create a new `Pair` ( called  $p$  ) of a string and a double, saving  $w2$  and distance in this pair. We then add  $p$  to *topK*.

After doing this for all the words  $w2$  in *forW*, we sort *topK* using `dpc`, and then keep the first  $K$  entries in *topK*, erasing the rest. *topK* is then returned.

The *kTopSimilarWords* method calculates the  $K$  most similarly distributed words to all the words in the distribution map of the corpus (Used for EMR notes and USMLE texts).

*kTopSimilarWords* takes the distribution map ( called *map* ), the distance metric the distributions will be measured against each other by ( called *metric* ), and the number of most similarly distributed words we would like to retrieve ( called  $K$  ).

It returns a `HashMap` called `kSimilarWordsMap`. `kSimilarWordsMap` consists of words that point to an `ArrayList` of size  $K$  containing pairs of Strings

( the top relevant words ) and doubles ( their distances from the key words ).

It does this by loading the saved map, and looping over all the words in the map ( we call each word  $w$  ). Inside the loop, we pass  $w$  to  $kTopSimilarWordsforW$  along with  $map$ ,  $metric$ , and  $k$ .

It thus retrieves  $wk$ , the ArrayList of  $K$  pairs relevant to  $w$ .

Then we add  $w$  and  $wK$  to  $kSimilarWordsMap$ .

Once the loop is completed, the method returns  $kSimilarWordsMap$ .

The method  $resultsToHashmap$  is passed  $kSimilarWordsMap$  and returns a HashMap that points the words in the map to a HashMap of the top  $K$  co-occurring words and the respective frequency of co-occurrence. This is easier to save to a file for later use.

## 7.6 Automatic Annotations

The method  $annotate$  aims to automatically annotate non-annotated notes using different approaches in attempt to achieve the most accurate automatic annotation method.

In the first attempt, we start by retrieving the BNs for each of the EMRs diagnosed as diabetes and anemia. We process the file containing all the manually annotated notes, and proceed to extract the manually annotated notes diagnosed as diabetes and the manually annotated notes diagnosed as anemia.

We store all the words that are manually annotations for the notes diagnosed as diabetes and in a separate object we store all the words that are manually annotations for the notes diagnosed as anemia.

We use a method that checks whether a word could be an abbreviation of a word that should be annotated to supplement the automatic annotation results.

Then we make use of the manually annotated notes, word abbreviations, and BNs to automatically annotate non-annotated notes diagnosed as diabetes and anemia.

So for the first trial if the BN words are included in the non-annotated notes, they get annotated. If the annotated words in a specific diagnosis are included in the non-annotated notes for the same diagnosis, they get annotated.

If the an abbreviated version of any of the above words are included in the non-annotated notes, they get annotated.

In future methods we will be using USMLE disco, pubmed disco, dictionaries for this purpose as well. The details of the described attempt above are found below.

## 7.7 Objects

*map* is a HashMap that points the name of the file that contains a BN to the graph it becomes after being processed.

*dwBNbag* is a HashMap of ArrayLists of words as keys pointing to a word. Each node in the diabetes BN is split into the individual words that constitute it, and these fill the ArrayList key.

The word the ArrayList points to is the type of the node, which serves as an annotation. In this way the diabetes BN words are bagged.

*awBNbag* is a HashMap of ArrayLists of words as keys pointing to a word. Each node in the the anemia BN is split into the individual words that constitute it, and these fill the ArrayList key.

The word the ArrayList points to is the type of the node, which serves as an annotation. In this way the anemia BN words are bagged.

*anotWords* is an ArrayList of all the annotated words. It has only the specific annotation text and not their respective labels.

*anotDiabetesWords* is an ArrayList of all the annotated words in diabetes notes. It has both the specific annotation texts and their respective labels.

*anotAnemiaWords* is an ArrayList of all the annotated words in anemia notes. It has both the specific annotation texts and their respective labels.

VOWELS is a string of the vowels; "aeiouy". It is split and used for the method that checks if a word can count as an abbreviated version of a longer word of significance.

*annotatedNotes* is an ArrayList of all the annotated Notes.

*anDiabNotes* is an ArrayList of the Notes annotated for the diabetes diagnosis.

*anAnemNotes* is an ArrayList of the Notes annotated for the anemia diagnosis.

*nonAnDiabNotes* is an ArrayList of the Notes non-annotated for the diabetes diagnosis.

*nonAnAnemNotes* is an ArrayList of the Notes non-annotated for the anemia diagnosis.

*autoNonAnDiabNotes* is an ArrayList of the Notes non-annotated for the diabetes diagnosis automatically annotated.

*autoNonAnAnemNotes* is an ArrayList of the Notes non-annotated for the anemia diagnosis automatically annotated.

## 7.8 Methods for Automatic Annotations

### 7.8.1 Bagging

For our annotation process, and for the purpose of relating EMRs with their corresponding BNs, we need to bag words of a common function in the natural language processing method. Different word corpora will bag words in accordance to the information available in that source.

We have BN word bags, abbreviations, annotation word bags, USMLE word bags, Pubmed word bags, and Dictionary word bags from the separate corpora.

The BN word bags are words in each of the nodes of the diabetes and anemia BNs respectively and their annotation types. It therefore bags the words in the same node and associates them with their annotation.

The abbreviation word bags are based on a vowel-based variation of the full version of any given word. These bags aim to identify no-vowel tokens that are variations of longer words with vowels, and bag them together.

The annotation word bags come from the manual annotations holders of medical degrees made to help with the EMR NLP work.

The USMLE word bags are extracted from the USMLE word vector Kmost-SimilarUSMLE. The words with the closest distributional similarity scores are bagged in the same context.

The Pubmed corpora provides word bags we extract from DISCO. DISCO also provides the similarity and distance scores that we need.

The Dictionary word bags consist of the dictionary word vector for any relevant word. These are built through the method that extracts synonyms for words of relevance from an online thesaurus.

These will be compared with the EMR vectors to match words of relevance.

In this section, we will automatically annotate non-annotated notes using BN word bags, abbreviations, and annotation word bags.

In future sections, we still use non-annotated notes using BN word bags, abbreviations, and annotation word bags along with USMLE word bags.

Another section will involve non-annotated notes using BN word bags, abbreviations, annotation word bags, and Pubmed word bags.

The following section will have non-annotated notes using BN word bags, abbreviations, annotation word bags, and Dictionary word bags.

The final sections will incorporate all our findings into one main method.

## 7.9 Methods

The method `processWorkbook` processes the files that contain the BNs. It does this by loading the file, and using our method `findCol` to read column by column the contents of the file. Each column represents either a node or an edge attribute.

We have the format of the files so they are organised as expected and simple to process. This method builds the object map ( the `HashMap` that points the name of the file that contains a BN to the graph it becomes after being processed ).

In this way, the BNs for the diabetes and anemia diagnosis are saved.

The method `fillBNwordBag` fills in the objects `dwBNbag` and `awBNbag` with the words in each of the nodes of the diabetes and anemia BN graphs respectively and their annotation types.

It therefore bags the words in the same node and associates them with their annotation.

It does this by locating the graphs of the requested diagnosis and looping through the nodes, splitting each node to the words constituting it.

Once it removes redundant and irrelevant words, it saves the individual words of significance and bags them.

The method `abbreviationCheck` is passed two words, the original word (  $w1$  ) and the word that needs to be checked (  $w2$  ).  $w2$  is being checked for whether or not it can be considered an abbreviation of  $w1$ .

This is done by firstly splitting the string `VOWELS` into characters. The next step is creating an `ArrayList` ( named `vars` ) of all the versions of  $w1$  altered by removing vowels. It creates an `ArrayList` removing vowels from the beginning of the word to the end, adding each alteration as it goes. It skips the first letter in case it is a vowel as usually the first letter is not removed in abbreviations.

It then does the same backwards, removing vowels from the end of the word to the beginning of it, adding each alteration as it goes.

The last step is to check if `vars` contains the  $w2$ . The method returns a `Boolean`; true is  $w2$  can be considered an abbreviation of  $w1$ , and false if not.

The method `BuildAnNotes` processes the file that contains the annotated notes and fills the object `annotatedNotes`. It has been described before as a version of `buildallNotes` that accounts for the annotations in the annotated notes.

The difference between the files that contain the annotated notes and the file that contain the non-annotated notes are a string of lines for each note describing the annotated text, the position of the annotated text in the note text, and the label of this particular annotation word / annotation phrase.

The annotated notes therefore fill the ArrayList of Annotations in the note constructor while the non-annotated notes does not fill it in.

The method `specificNotes` is passed the name of a particular diagnosis and returns an ArrayList of the Notes annotated for that specific diagnosis.

It is called to fill `anDiabNotes` and `anAnemNotes` with the annotated notes for the diabetes diagnosis and the anemia diagnosis respectively. It disregards any notes that are diagnosed irrelevantly to the particular diagnosis queried for.

The method `buildUnAnNotes` processes the file that contains the non-annotated notes and fills both the objects `nonAnDiabNotes` and `nonAnAnemNotes`. It is identical to previously described `buildallNotes`.

The name was altered in this class for simpler differentiation purposes.

The method `diagSpecificAnnotations` is passed the ArrayList of notes annotated for a specific diagnosis and returns the HashMap of the specific annotations found exclusively in those notes that point to the label that those annotations pertain to.

This method was passed `anDiabNotes` and `anAnemNotes` built `anotDiabetesWords` and `anotAnemiaWords` for the specific annotations labels for the diabetes diagnosis and the anemia diagnosis respectively. It therefore disregards any annotations made in notes that are diagnosed irrelevantly to the particular diagnosis queried for.

The method `returnLowestDistanceScoreAnn` takes an ArrayList of Annotations and returns the Annotation with the lowest distance score.

This method is designed to select the most likely annotation for a non-annotated word from a set of possible annotations that carry a distance score from the non-annotated word on hand.

We will be using this returned annotation, the closest annotated word to the non-annotated one, to determine the label under which the automatic annotation method will classify it.

`public static Annotation`. This method returns the annotation by initiating a double variable, named `low`, at a high distance. It proceeds to iterate over the list of Annotations, switching its value to an Annotation distance score if it is less than it.

In this iterative way it is able to find the smallest distance score. Once the smallest distance score it identified, the code loops to find the corresponding annotation. The final step returns this annotation.

The method `annotate` automatically annotates the non-annotated notes.

We use several methods as a first step to automatically annotate these notes. We thus use several versions of the method "annotate" as a first step to automatically annotate these notes.

The first method involves comparing the non-annotated note words to the word distributions according to their occurrence in the EMR notes. If a word distribution for the note word at hand is found within those distributions, either through a direct string match or if the note word can pass for an abbreviation of a word that has a distribution according to its occurrence in the EMR notes, we proceed.

The next step is to retrieve this word's top  $K$  ( $K$  is taken to be 10) words similarly distributed to it as occurring in the EMR notes. This forms a bag of words that we then compare to:

- a) The annotations for the specific diagnosis and the top  $K$  ( $K$  is taken to be 10) words similarly distributed to every annotation word.
- b) The BN graph words and respective word bags of the co-occurring tokens in each node.

The second method involves comparing the non-annotated note words to the word distributions according to their occurrence in the USMLE texts. If a word distribution for the note word at hand is found within those distributions, either through a direct string match or if the note word can pass for an abbreviation of a word that has a distribution according to its occurrence in the USMLE texts, we proceed.

The next step is to retrieve this word's top  $K$  ( $K$  is taken to be 10) words similarly distributed to it as occurring in the USMLE texts. This forms a bag of words that we then compare to:

- a) The annotations for the specific diagnosis and the top  $K$  ( $K$  is taken to be 10) words similarly distributed to every annotation word
- b) The BN graph words and respective word bags of the co-occurring tokens in each node

The third method involves comparing the non-annotated note words to the word distributions according to their occurrence in the Pubmed texts using DISCO.

If a word distribution for the note word at hand is found within those distributions through a direct string match we proceed. The next step is to retrieve this word's top  $K$  ( $K$  is taken to be 10) words similarly distributed to it as occurring in the Pubmed texts. This forms a bag of words that we then compare to:

- a) The annotations for the specific diagnosis and the top  $K$  ( $K$  is taken to be 10) words similarly distributed to every annotation word.
- b) The BN graph words and respective word bags of the co-occurring tokens in each node.

Since the implementation of the first two methods is identical save the different files containing the word distributions according to their occurrence in

the EMR notes as opposed to word distributions according to their occurrence in the USMLE texts, we will explain this version implementation first.

The version for the Pubmed texts only involves a slight alteration pertaining to the fact that the Pubmed text distribution was not built by us but by DISCO and therefore has its own implications that will be discussed after this implementation is thoroughly studied.

The first two versions of the method automatically annotate the non-annotated notes by passing the ArrayList of non-annotated notes, along with the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

For the diabetes case, we pass *nonAnDiabNotes* as the ArrayList of non-annotated notes, *anotDiabetesWords* as the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and *dwBNbag* the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

For the anemia case, we pass *nonAnAnemNotes* as the ArrayList of non-annotated notes, *anotAnemiaWords* as the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and *awBNbag* the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

The method returns an ArrayList of notes that are automatically annotated. It therefore builds *autoNonAnDiabNotes* ( EMR version ) and then builds

*autoNonAnDiabNoteswithUSMLE* ( USMLE version ) for the diabetes case and *autoNonAnAnemNotes* ( EMR version )

*autoNonAnAnemNoteswithUSMLE* ( USMLE version ) for the anemia case.

The method begins by defining the ArrayList of automatically annotated notes.

It proceeds to loop through the non-annotated notes, retrieving the note text and splitting the words. Each word is called *w*.

This outer loop has two sets of instructions.

a) A set that is followed if the vector of EMR words ( or USMLE words ) with their top *K* matches contains *w*. The HashMap *normedTopK* contains the EMR words ( or USMLE words ) and points to a HashMap of the top *K* word matches pointing to their match scores.

It is loaded from the code *kTopSimilarWords* that built it under the name *kSimilarWordsMap*.

b) A set if *normedTopK* does not contain *w*.



### 7.9.1 normedTopK contains w

If normedTopK contains  $w$ , we retrieve the corresponding HashMap ( called tTopK ) of the top  $K$  similar words pointing to the distance scores.

The next inner loops goes through the keys of the HashMap specific to the diagnosis at hand of the annotations pointing to the labels called specificAnnWords. The keys are called  $w_2$ , and we check if normedTopK has a key with the same identity as  $w_2$ .

If so, we retrieve its HashMap, called sAnnTopK, and calculate the Euclidean distance between tTopK and sAnnTopK.

If the distance is less than a specific threshold THRESH we add  $w$  to the ArrayList of annotations in the note, including the label ( from the object of *specificAnnWords* at  $w$  ) and position of  $w$  in the text.

Then this annotation is added to the ArrayList of annotations for the note at hand ( called *autoAnnotation* ).

Once the loop over *specificAnnWords* ends, we loop over the BN node words.

We use this with two loops, one looping over the keys of the HashMap with ArrayLists of BN node words pointing to their labels ( called  $wBNbag$  ), and an inner loop looping over each ArrayList ( called bag ) of words in an individual BN node.

Each of these individual words is  $w_2$ .

If normedTopK has a distribution for  $w_2$ , we retrieve the corresponding HashMap and name it bagTopK.

We then calculate the Euclidean distance between tTopK and bagTopK.

If the distance is less than a specific threshold THRESH we add  $w$  to the ArrayList of annotations in the note, including the label ( from the object of  $wBNbag$  at bag ) and position of  $w$  in the text.

Then this annotation is added to *autoAnnotation* .

### 7.9.2 normedTopK does not contain w

If normedTopK does not contain  $w$ , we loop through the words that normedTopK does contain ( called st ). Our method abbreviationCheck is called to know whether  $w$  can pass for an abbreviation of st.

If abbreviationCheck confirms that  $w$  can pass for an abbreviation of st, we retrieve the distribution HashMap corresponding to st ( and thus  $w$  ), tTopK, and then proceed to go through the same set of loops the previous set of instructions described ( *specificAnnWords* loop followed by  $wBNbag$  loop ).

### 7.9.3 End of Method

In the outermost loop, once these commands are executed, we add to the note  $n$  the ArrayList of automatic annotations. We then add  $n$  to the ArrayList of automatically annotated notes, *autoAnned*.

Once this outermost loop is concluded, and we have looped over all the non-annotated notes and automatically annotated them, we return *autoAnned*.

This method builds the ArrayList of automatically annotated note objects *autoNonAnDiabNotes* and *autoNonAnAnemNotes* for the EMR note words case, and *autoNonAnDiabNoteswithUSMLE* and *autoNonAnAnemNoteswithUSMLE* for the USMLE text words case.

### 7.9.4 Pubmed version of Method using DISCO

The third Pubmed version of the method automatically annotate the non-annotated notes also by passing the ArrayList of non-annotated notes, along with the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

For the diabetes case, we pass *nonAnDiabNotes* as the ArrayList of non-annotated notes, *anotDiabetesWords* as the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and *dwBNbag* the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

For the anemia case, we pass *nonAnAnemNotes* as the ArrayList of non-annotated notes, *anotAnemiaWords* as the HashMap specific to the diagnosis at hand of the annotations pointing to the labels, and *awBNbag* the HashMap of an ArrayList of words included in each node of the respective BN pointing to their labels.

The method returns an ArrayList of notes that are automatically annotated. It therefore builds *autoNonAnDiabNoteswithDISCO* for the diabetes case and *autoNonAnAnemNoteswithDISCO* for the anemia case.

The method begins by defining the ArrayList of automatically annotated notes. It proceeds to loop through the non-annotated notes, retrieving the note text and splitting the words. Each word is called  $w$ .

If the Pubmed corpora contains a distribution for  $w$ , we retrieve the corresponding top  $K$  similar words and their distance scores. We call this *simResult*. We next loop through the *simResult* bag of words, namely  $w$  and its top  $K$  similar words.

We first check if *specificAnnWords* ( the HashMap specific to the diagnosis at hand of the annotations pointing to the labels ) contains a key ( called  $w_2$

) for a word in the *simResult* bag. If so, we retrieve its label, and add *w* to the ArrayList of annotations in the note, including the label ( from the object of *specificAnnWords* at *w* ) and position of *w* in the text. Then this annotation is added to the ArrayList of annotations for the note at hand ( called *autoAnnotation* ).

We then check if the BN node words HashMap ( called *wBNbag* ) contains a key ( called *w2* ) for a word in the *simResult* bag. If so, we retrieve its label, and add *w* to the ArrayList of annotations in the note, including the label ( from the object of *specificAnnWords* at *w* ) and position of *w* in the text. Then this annotation is added to *autoAnnotation* .

### 7.9.5 End of Method

In the outermost loop, once these commands are executed, we add to the note *n* the ArrayList of automatic annotations. We then add *n* to the ArrayList of automatically annotated notes, *autoAnned*.

Once this outermost loop is concluded, and we have looped over all the non-annotated notes and automatically annotated them, we return *autoAnned*.

This method builds the ArrayList of automatically annotated note objects *autoNonAnDiabNoteswithDISCO* and *autoNonAnAnemNoteswithDISCO*.

## 7.10 Saving the Automatically Annotated Notes

To save the results of annotate, namely *autoNonAnDiabNotes* and *autoNonAnAnemNotes*, we need to create an object compatible with the way Java saves objects.

Since Java does not save an ArrayList of our Note constructor, we fill two HashMap objects of Strings as keys pointing to an ArrayList of Strings as objects.

The keys will be filled with the automatically annotated text, and the ArrayList of Strings will be filled with the annotations in that text alongside their labels.

We therefore define and fill *autoAnnDiab* and *autoAnnAnem* through two loops, one looping through *autoNonAnDiabNotes* and the other looping through *autoNonAnAnemNotes*.

We then call the method that saves objects in Java to a file, by passing *autoNonAnDiabNotes* and *autoNonAnAnemNotes* to it, along with the names of the files we would like them saved to.

For the EMR case, we saved *autoNonAnDiabNotes* to *autoAnnotatedNonAnDiabNotes*,

and *autoNonAnAnemNotes* to *autoAnnotatedNonAnAnemNotes*.

For the USMLE case, we saved *autoNonAnDiabNotes* to *autoAnnotatedNonAnDiabNoteswithUSMLE*, and *autoNonAnAnemNotes* to *autoAnnotatedNonAnAnemNoteswithUSMLE*.

For the DISCO case, we saved *autoNonAnDiabNotes* to *autoAnnotatedNonAnDiabNoteswithDISCO*, and *autoNonAnAnemNotes* to *autoAnnotatedNonAnAnemNotes* with DISCO.

# Chapter 8

## Website Implementation

The website implementation that contains my work is written in HTML and JavaScript. The website communicates with python and then java to provide feedback, correct annotations, and automatically annotate new texts.

### 8.1 Website Layout (Using HTML)

The site title is EMR Annotation Tool.

The page title is "Electronic Medical Record Automatic Annotation Tool".

Once the user enters the page they can choose to display the automatically annotated EMR notes and they can input their own medical notes for automatic annotation.

If they choose to display the automatically annotated EMR notes they can also add a missing annotation or correct an incorrect annotation displayed.

### 8.2 Displaying Automatically Annotated EMR Notes

To display the automatically annotated EMR notes, they can choose using training with either USMLE texts and EMR manual annotations, PubMed DISCO index and EMR manual annotations, or EMR manual annotations on their own.

The website then asks the user to specify a diagnosis for the automatically annotated notes to display; either Diabetes Mellitus, or Anemia.

The website highlights the annotations in every note displayed according to a color-label legend displayed above the text box where the current note is being displayed.

The web-page has four buttons to begin displaying the notes.

The first button is named "Display Note" and will display the note with index 1 each time it is pressed.

The button "Previous" will display the note with the index directly preceding the index of the current note being displayed.

The button "Next" will display the note with the index directly succeeding the index of the current note being displayed.

The button "Go To" comes with a text box that it will read and then display the note with the corresponding index.

Once any of these buttons are clicked, the legend table as well as the note at hand are displayed.

In addition, two tables containing characteristics of the automatic annotation at hand are displayed.

The first table is named the labels table. It contains:

- the labels that correspond to words automatically annotated in the note text
- the words in the note text annotated with that label
- the automatically annotated words intersecting with the manually annotated words in our corpora
- the automatically annotated words not intersecting with the manually annotated words in our corpora
- the missed words that were not automatically annotated but were included in the manual annotations in our corpora
- the precision measurement for this particular note text (number of intersecting annotations divided by the sum of the number of intersecting annotations and the number of non-intersecting annotations)
- the recall measurement for this particular note text (number of intersecting annotations divided by the sum of the number of intersecting annotations and the number of annotations not included - missed)

The second table is named the total table. It contains:

- all the labels
- the total precision measurement for each label in the entire training method specified by the user (the number of all the intersecting annotations in this method divided by the sum of the number of all the intersecting annotations and the number of all the non-intersecting annotations in this method)

- the total recall measurement for each label in the entire training method specified by the user (the number of all the intersecting annotations in this method divided by the sum of the number of all the intersecting annotations and the number of all the annotations not included - missed in this method)

The user can also add a missing annotation or correct an incorrect annotation displayed

To add a missing annotation or correct an incorrect annotation displayed, the user double-clicks on a word in the EMR text displayed, then proceeds to right-click. On the user's right-click, a window pops up asking the user what the annotation label is of the word they double-clicked on.

The user must input an annotation label from the list of labels available on the page in the text-box inside the pop-up. This pop-up is case-sensitive.

If the user enters an incorrect label, they will be asked to input an annotation label from the list of labels available on the page and they will be informed that the input is case-sensitive.

If the user does not enter anything another pop-up will inform the user that they have cancelled the annotation.

If the user input a valid annotation label it alters the annotation on the page and sends the new annotation to JavaScript to update the annotation information.

### **8.3 Automatically Annotating User Input**

At the bottom of the page the website asks the user to enter a text to be annotated automatically and specify the training method.

The user can pick for their input to be annotated using any combination of the following: USMLE exam questions Notes, PubMed DISCO index, and EMR manual annotations.

The user then inputs their text and presses the submit button.

### **8.4 Displaying Automatically Annotated EMR Notes**

The automatically annotated EMR notes are generated by the Java codes and stored in xml files in a readable way. There is a Java code that takes all the automatically annotated EMR notes trained with different methods and prints them in a readable way to xml files. This is because JavaScript reads xml files using a built in method (`getElementById().childNodes`), so this choice of

storage was the most convenient and made the JavaScript end of the process simple.

JavaScript receives the request from the user to display a specific set of those notes, depending on what they were trained with and what diagnosis they want displayed. Then JavaScript reads the respective files and organises them into a table for displaying purposes.

We implemented a highlight method that we call during the process of reading the files and organizing each unit of the table content to be displayed. This method allows JavaScript to highlight the annotations in every note displayed according to a color-label legend displayed above the text box where the current note is being displayed.

The notes in the xml files are read by JavaScript per index. We do this by passing to the JavaScript function that reads the xml files an index  $i$  of childNodes to specify. The button "Display Note" will pass index 1 to JavaScript. The button "Previous" will pass 1 minus the current index to JavaScript, while the button "Next" will pass 1 plus the current index. The button "Go To" comes with a text box that it will read and then pass the input index to the JavaScript method.

The default index is 1, and so if the user presses "Display Note" first, they will be viewing the note with index 1 first, if they press previous first, they will be viewing the note with index 0 first, and so on.

Once the initial button is pressed (any of "Display Note", "Previous", "Next", and "Go To"), JavaScript then changes the display of the website by communicating to HTML `getElementById().innerHTML` to set the element indicates between brackets to the table created of the automatically annotated EMR note at hand. Once any of these buttons are clicked, the legend table as well as the note at hand are displayed.

JavaScript also organises the labels table for each note in the same method that organises the table displaying the note. It reads the

- the labels that correspond to words automatically annotated in the note text
- the words in the note text annotated with that label
- the automatically annotated words intersecting with the manually annotated words in our corpora
- the automatically annotated words not intersecting with the manually annotated words in our corpora
- the missed words that were not automatically annotated but were included in the manual annotations in our corpora



- the precision measurement for this particular note text (number of intersecting annotations divided by the sum of the number of intersecting annotations and the number of non-intersecting annotations)
- the recall measurement for this particular note text (number of intersecting annotations divided by the sum of the number of intersecting annotations and the number of annotations not included - missed)

from the same xml file it uses for the note and the highlights within each note.

Since the total table does not vary with the individual notes in every file, a separate xml file contains the information necessary for this table, depending on the training method and diagnosis selected. Consequently, JavaScript organises this table in a separate method than it does the notes and labels tables. The information read from JavaScript, as previously stated, includes

- all the labels found in this training method and diagnosis analyzed
- the total precision measurement for each label in the entire training method specified by the user (the number of all the intersecting annotations in this method divided by the sum of the number of all the intersecting annotations and the number of all the non-intersecting annotations in this method)
- the total recall measurement for each label in the entire training method specified by the user (the number of all the intersecting annotations in this method divided by the sum of the number of all the intersecting annotations and the number of all the annotations not included - missed in this method)

The user can also add a missing annotation or correct an incorrect annotation displayed.

To add a missing annotation or correct an incorrect annotation displayed, JavaScript has a double-click listener for when the user double-clicks on a word in the EMR text displayed. It also has a right-click listener so when the user then proceeds to right-click it can use the alert() function to communicate to the user with a window pop-up. JavaScript adds a script to the alert box asking the user what the annotation label is of the word they double-clicked on.

The user must input an annotation label from the list of labels available on the page in the text-box inside the pop-up. This pop-up is case-sensitive.

JavaScript stores a string of all the words that are correct label for a given word to be annotated as, separated with white spaces. A JavaScript function we implemented called contains() is passed two strings to check if the first

string is included in the second. Using this method we can tell if the user enters an incorrect label by passing it to the contains method along with the string that contains the correct labels. If the user had indeed entered an incorrect method, JavaScript will use the alert() function prompting the user to input an annotation label from the list of labels available on the page. It also informs them that the input is case-sensitive.

If the user does not enter anything another alert() function will be used by JavaScript to send the user a pop-up informing them that they have cancelled the annotation.

If the user input a valid annotation label it alters the annotation on the page and sends the new annotation to JavaScript to update the annotation information.

Once JavaScript receives the valid annotation label, it sends it along with the word that it detected the double-click take place on to the code that corrects inaccurate annotations and adds new ones.

This method uses The Common Gateway Interface, or CGI to communicate to python. CGI is a set of standards that define how information is exchanged between the web server and a custom script. In this way, JavaScript communicates to python to write to an xml file a line containing the word to be annotated along with its valid annotation label, separated by a white space.

The Java code in turn reads this file, and updates the structure that has all the words annotated linked to their correct annotation labels.

It then updates all the xml files containing the automatic annotations organized by training method and diagnosis to include the correct annotation for each occurrence of the word in any note.

In this way, when JavaScript reads the xml file at hand it reads from the updated version that includes the annotation(s) provided by the user.

## **8.5 Automatically Annotating User Input**

At the bottom of the page the website asks the user to enter a text to be annotated automatically and specify the training method.

The user can pick for their input to be annotated using any combination of the following: USMLE exam questions Notes, PubMed DISCO index, and EMR manual annotations.

The user then inputs their text and presses the submit button.

Once the submit button is pressed, JavaScript stores the training methods selected in a string form along with the text to be annotated.

Then two JavaScript functions are called.

The first function uses CGI to communicate with python. It is passed the training methods selected in a string form along with the text to be annotated, and sends then to a python class. This python class prints two lines to a file, the first line containing the training method(s) specified by the users, and the second containing the text to be annotated.

The second function uses CGI to communicate with another python class. This python class runs a Java code. This Java code reads the file containing the training method(s) specified by the users and the text to be annotated. It then accordingly implements the methods corresponding to the training techniques selected by the user to automatically annotate the user's medical text.

Once this is completed, the Java code proceeds to print the text along with the annotations in an xml form to an xml file. The python code that ran java then prints every line of the last xml file to be easily sent to JavaScript. JavaScript then reads the annotations, and displays the automatically annotated user text on the web-page. The annotations are shown in ghlights along with the annotation to highlight legend.

## **8.6 Pending Work**

We are working on adding a feature to the website where the user's medical text can be processed for the top three or four likely diagnoses pertaining to the input. This is based on the entity analysis of the existence of the words in the BNs corresponding to diagnostic algorithms as discussed in the previous sections.

# Chapter 9

## Other Work

Some previous work was done in regards to automation of understanding the unstructured notes stored within the EMR notes. A simple distributional similarity algorithm was deployed to identify the symptoms in any given EMR.

Using Java, distribution matrices were built using the annotated set of EMR notes.

A distribution matrix of all the words in a training set of the annotated data was calculated according to the order in which they show up in the text, with respect to the word immediately preceding and succeeding it.

Another distribution matrix followed the same formula but took into consideration two preceding and succeeding words per single entity instead of one in both directions. These were obtained by running through the annotated texts once, and using a nested hashmap to record the probability of one word occurring next to another.

To evaluate the success of the distributions in predicting whether or not any given word is a symptom, the two centers of gravity of the distribution of the symptoms was calculated, and the distance between the distribution of a word and the symptom center of gravity was calculated.

The same process was repeated by comparing the distribution of a word with the mean distribution of the symptoms. Different distance metrics were used, including Euclidean, Jacobian, Manhattan, Binary, Canberra, Minkowski, and Maximum.

For both the first and the second matrices, comparing a word distribution with the center of gravity had better precision and accuracy levels than comparing with the mean distribution. The distance metric with the best precision and accuracy scores was the Euclidean distance.

For the first matrix, accounting for one word directions, the precision was 0.159 and the recall was 0.87 at a threshold of 0.28. For the second matrix,

accounting for two words in both directions, the precision was  $332/(332+26) = 0.919668$  and the recall was  $332/333 = 0.997$  at a threshold of 0.177. Therefore it is most useful to take a wider locus into consideration during distributional similarity testing.

Hidden Markov Models were explored as well, by substituting words in notes with random numbers. Running the algorithm as is produced no feasible results. We then tried altering the original text notes and adding the annotation label after the annotated word to identify the distributions of high interest. This not only should increase the chances of the model to detect words, of high interest, but should also improve accuracy, seeing as a word annotated in one text is not necessarily annotated underneath the same label in another. After the alteration, and no result improvement, we have decided to further explore the method to discover the optimal way to utilize the algorithm, by representing words by numbers according to a different metric than the random one being used thus far.

The diagnostic graphs have been built through an algorithm that reads and interprets the files, and a visualization joining the EMR notes has been produced. The relation linking the EMR notes and the most relevant diagnostic graph has not yet been made.

# Chapter 10

## Results

Our results show that Bayesian networks successfully identify the relevant differential analysis by predicting the top two or three diagnostic algorithms as explanations of the EMR note.

The analysis often included prevailing diagnoses such as fatigue, headache, and joint sprain which are underlying symptoms to other more serious diagnoses.

The average recall is 0.93. The differential analysis based precision was 1 for all the models when we consider the precision according to whether the model appears in the top two models with the notes pertaining to them. If we weigh every single algorithm on its own, the average precision for the models is 0.18 when compared to all the other models, and 0.64 when compared to one other model. This may be due to a lack of data and a lot of missing information.

After all the graphs are displayed, a discussion of the results will follow.

Each set of EMR notes has three corresponding graphs relating how the ten different BN models explain the notes at the seven thresholds 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9.

One graph for the score of the BN model with the EMR notes, one for the precision of the BN model with the EMR notes, and one for the recall the BN model with the EMR notes.

To evaluate the effectiveness of our method, we use precision and recall as illustrated in Figure 10.1. The purpose of precision and recall is to see how well our models work. The precision metric answers the question; "How many selected items are relevant?". Therefore, it is the number of true positives divided by the sum of true and false positives. Recall answers the question; "How many relevant items are selected?". Therefore, it is the number of true positives divided by the sum of true positives and false negatives.

These models trained on yes/no data, not attempting to higher the score

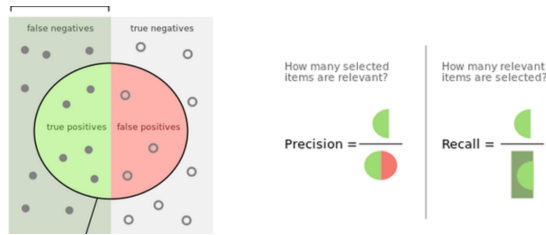


Figure 10.1: Precision and Recall [1]

of a match that surpasses a high threshold.

In the following graphs, the following abbreviations are used for the following diagnoses:

Anemia = A1, Anxiety = A2, Diabetes = D1, Fatigue = F1, Headache = H1, Hemoptysis = H2, Joint Sprain = J1, Kidney Disease = K1, Pruritus = P1, Tinnitus = T1.

Dif Precision represents the differential precision.

Precision 1 represents the precision of the model when the correct data is measured against the data from all the other sets of diagnosis notes.

Precision 2 represents the average precision of the model when the correct data is measured against the data from one other set of diagnosis notes.

To choose the other set we paired the notes into sets of two that would be measured against each other. The sets were paired depending on how many notes we had in their corpora. The closest two in note number were paired, in ascending order. So the two diagnoses we had the most notes for, diabetes (3,280 notes) and anemia (1,211 notes), were paired together. Similarly, anxiety (709 notes) was paired with headache (555 notes), joint sprain (429 notes) was paired with fatigue (253 notes), kidney disease (55 notes) was paired with pruritus (50 notes), and tinnitus (39 notes) was paired with hemoptysis (4 notes).

The following graphs show the results of our work. Throughout the graphs, a prevailing presence of the fatigue, headache, and joint sprain diagnoses is notable, and this is to be expected. As we mentioned, these are underlying symptoms to other more serious diagnoses and we therefore will include them in our score graphs and our precision 1 and when needed precision 2 calculations, but not for the differential precision purposes.

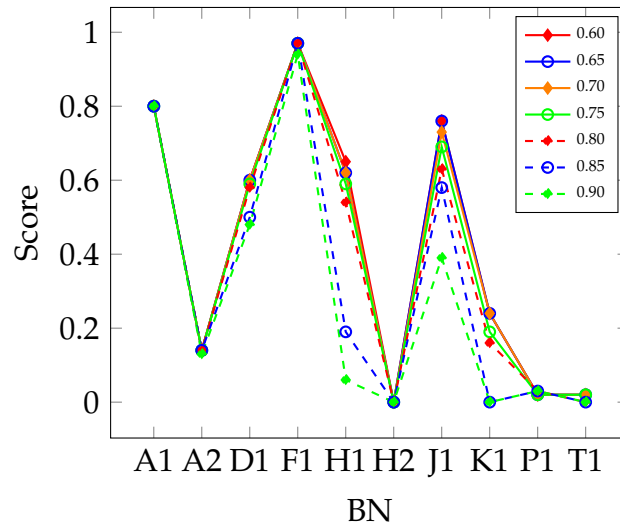


Figure 10.2: Anemia Notes Scores

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.2	0.67	1
0.65	1		0.2	0.67	0.99
0.7	1		0.2	0.67	0.99
0.75	1		0.2	0.67	0.99
0.8	1		0.2	0.67	0.99
0.85	1		0.2	0.67	0.99
0.9	1		0.21	0.68	0.95

Table 10.1: Anemia Precision and Recall

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.1	0.5	0.72
0.65	1		0.1	0.5	0.72
0.7	1		0.1	0.5	0.72
0.75	1		0.1	0.5	0.72
0.8	1		0.1	0.5	0.72
0.85	1		0.13	0.57	0.7
0.9	1		0.14	0.58	0.7

Table 10.2: Anxiety Precision and Recall



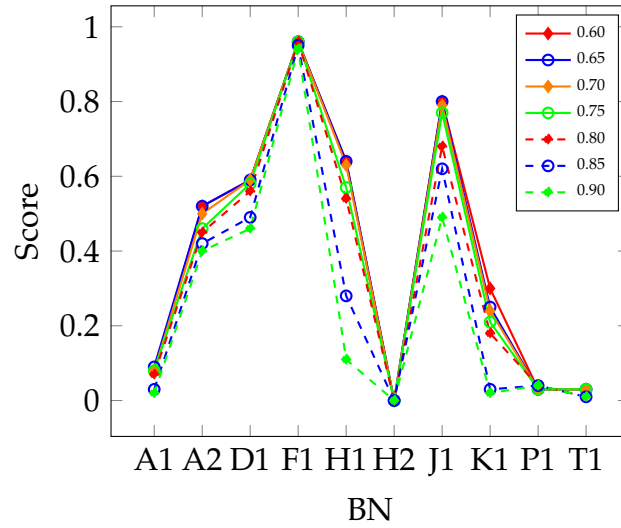


Figure 10.3: Anxiety Notes Scores

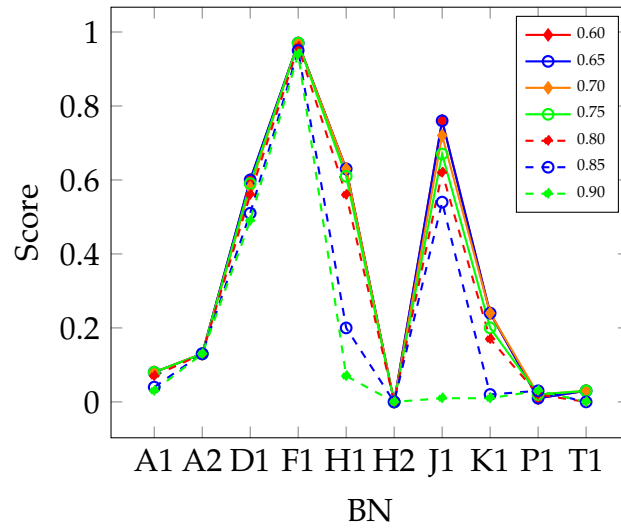


Figure 10.4: Diabetes Notes Scores

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.19	0.66	0.99
0.65	1		0.19	0.66	0.99
0.7	1		0.19	0.66	0.98
0.75	1		0.19	0.63	0.97
0.8	1		0.2	0.67	0.96
0.85	1		0.22	0.69	0.91
0.9	1		0.24	0.71	0.88

Table 10.3: Diabetes Precision and Recall

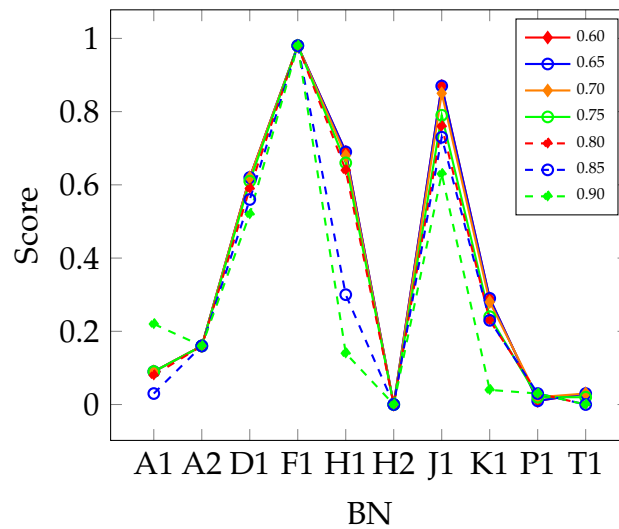


Figure 10.5: Fatigue Notes Scores

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.17	0.63	1
0.65	1		0.18	0.64	1
0.7	1		0.18	0.64	1
0.75	1		0.18	0.64	1
0.8	1		0.19	0.66	1
0.85	1		0.21	0.68	1
0.9	1		0.23	0.7	0.99

Table 10.4: Fatigue Precision and Recall

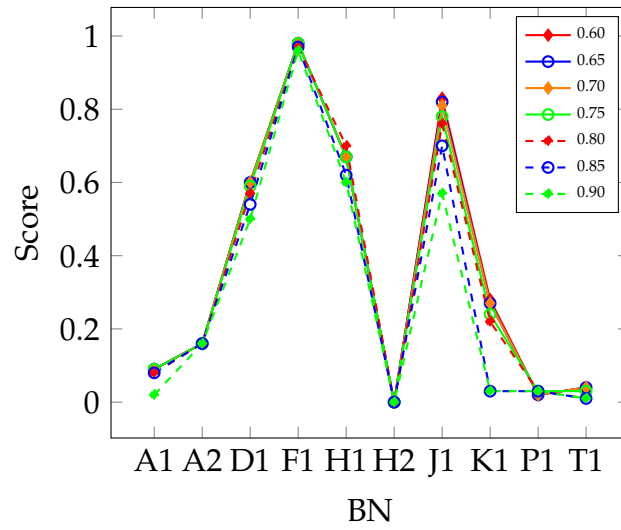


Figure 10.6: Headache Notes Scores

Similarity threshold	Dif sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.17	0.63	0.98
0.65	1		0.17	0.63	0.97
0.7	1		0.18	0.64	0.97
0.75	1		0.18	0.64	0.97
0.8	1		0.19	0.66	0.96
0.85	1		0.2	0.67	0.96
0.9	1		0.21	0.68	0.92

Table 10.5: Headache Precision and Recall

Similarity threshold	Dif sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.14	0.58	1
0.65	1		0.14	0.58	1
0.7	1		0.14	0.58	1
0.75	1		0.14	0.58	1
0.8	1		0.14	0.58	1
0.85	1		0.14	0.58	0.9
0.9	1		0.14	0.58	0.9

Table 10.6: Hemoptysis Precision and Recall

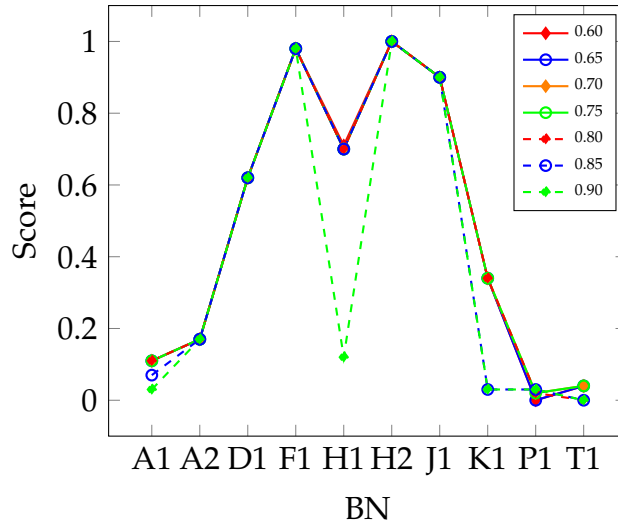


Figure 10.7: Hemoptysis Notes Scores

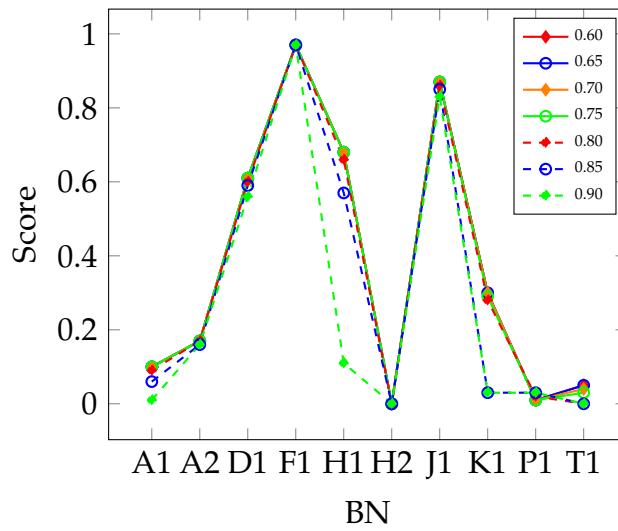


Figure 10.8: Joint Sprain Notes Scores

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.17	0.63	0.97
0.65	1		0.17	0.63	0.97
0.7	1		0.17	0.63	0.97
0.75	1		0.17	0.63	0.97
0.8	1		0.17	0.63	0.97
0.85	1		0.19	0.66	0.97
0.9	1		0.2	0.67	0.95

Table 10.7: Joint Sprain Precision and Recall

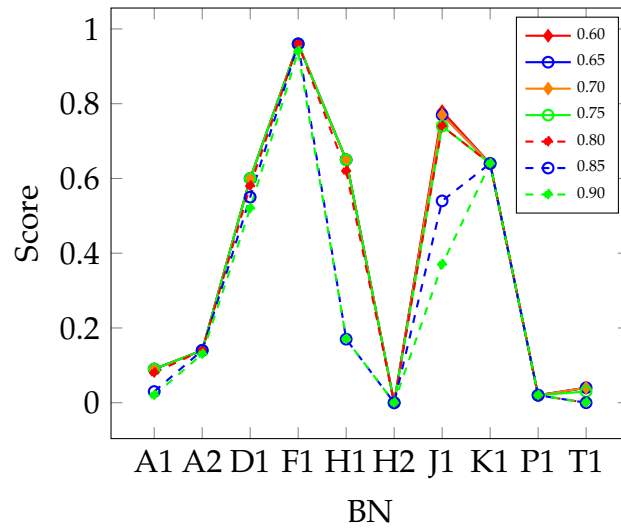


Figure 10.9: Kidney Disease Notes Scores

Similarity threshold	Dif- sion	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.16	0.62	1
0.65	1		0.16	0.62	1
0.7	1		0.16	0.62	0.99
0.75	1		0.16	0.62	0.8
0.8	1		0.16	0.62	0.7
0.85	1		0.16	0.62	0.7
0.9	1		0.16	0.62	0.63

Table 10.8: Kidney Disease Precision and Recall

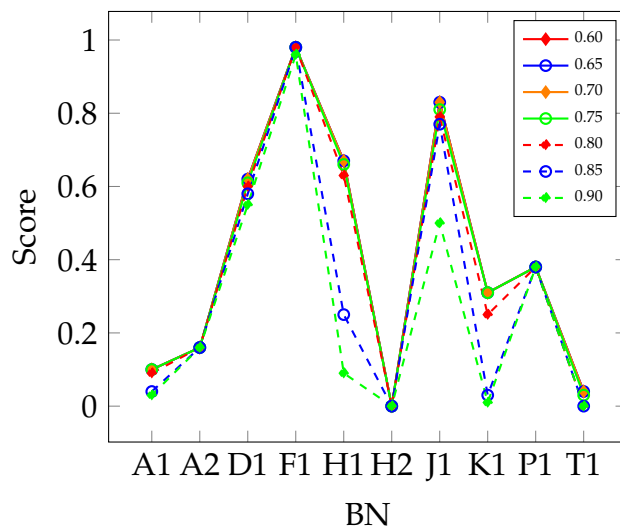


Figure 10.10: Pruritus Notes Scores

Similarity threshold	Dif	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.12	0.55	0.74
0.65	1		0.12	0.55	0.74
0.7	1		0.12	0.55	0.74
0.75	1		0.12	0.55	0.74
0.8	1		0.12	0.55	0.74
0.85	1		0.14	0.58	0.74
0.9	1		0.2	0.67	0.74

Table 10.9: Pruritus Precision and Recall

Similarity threshold	Dif	Preci- sion	Precision 1	Precision 2	Recall
0.6	1		0.14	0.58	0.93
0.65	1		0.14	0.58	0.93
0.7	1		0.15	0.6	0.93
0.75	1		0.15	0.6	0.93
0.8	1		0.17	0.63	0.93
0.85	1		0.17	0.63	0.93
0.9	1		0.18	0.64	0.93

Table 10.10: Tinnitus Precision and Recall

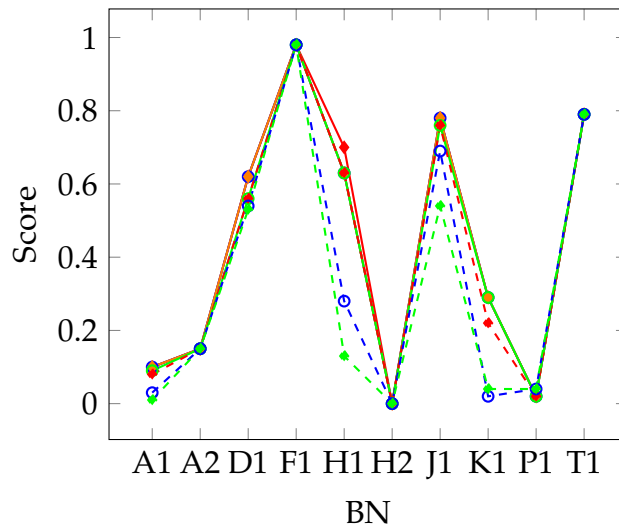


Figure 10.11: Tinnitus Notes Scores

As we can see, in the Anemia score figure 10.2 Anemia scores the highest out of the non-common diagnoses. When we compare the scores we will not be taking into account the prevailing diagnoses fatigue, headache, and joint sprain. Therefore Anemia has the top position in it's score table.

In the Anemia Precision and Recall table 10.1 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model when we don't take into account fatigue, headache, and joint sprain. Precision 1, 2, and recall remain consistent at around 0.2, 0.67, and 0.99 respectively.

In the Anxiety score figure 10.3 anxiety scores the second-highest out of the non-common diagnoses. It is second to the diabetes diagnosis.

In the Anxiety Precision and Recall table 10.2 the differential precision is consistently 1 throughout the similarity thresholds, since it is still in the top two highest explaining models when we don't take into account fatigue, headache, and joint sprain. Precision 1, 2, and recall remain consistent at around 0.1, 0.5, and 0.7 respectively.

As we can see, in the Diabetes score figure 10.4 diabetes scores the highest out of the non-common diagnoses.

In the Diabetes Precision and Recall table 10.3 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.2, 0.66, and 0.98 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.24 and 0.71, while the recall drops notably to 0.88 at the highest threshold.

In the Fatigue score figure 10.5 fatigue scores the highest out of all the

diagnoses.

In the Fatigue Precision and Recall table 10.4 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.18, 0.64, and 1 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.23 and 0.7, while the recall drops slightly to 0.99 at the highest threshold.

As we can see, in the Headache score figure 10.6 headache scores the highest out of the other non-common diagnoses.

In the Headache Precision and Recall table 10.5 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.18, 0.66, and 0.97 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.21 and 0.68, while the recall drops notably to 0.92 at the highest threshold.

In the Hemoptysis score figure 10.7 hemoptysis scores the highest out of all diagnoses, common diagnoses included.

In the Hemoptysis Precision and Recall table 10.6 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.14, 0.58, and 1 respectively. However, at the higher thresholds, the the recall drops notably to 0.9 at the two highest thresholds.

As we can see, in the Joint Sprain score figure 10.8 joint sprain scores the highest out of the other non-common diagnoses.

In the Joint Sprain Precision and Recall table 10.7 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.17, 0.63, and 0.97 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.2 and 0.67, while the recall drops slightly to 0.95 at the highest threshold.

In the Kidney Disease score figure 10.9 kidney disease scores the highest out of the non-common diagnoses.

In the Kidney Disease Precision and Recall table 10.8 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.16, 0.62, and 1 respectively. However, as the thresholds get higher, the recall drops first to 0.99 at the threshold 0.7, then to 0.8 at the 0.75 threshold, then down to 0.7 at thresholds 0.8 and 0.85, to finally drop to 0.63 at the highest threshold.

As we can see, in the Pruritus score figure 10.10 pruritus scores second-



highest out of the non-common diagnoses. It is second to the diabetes diagnosis

In the Pruritus Precision and Recall table 10.9 the differential precision is consistently 1 throughout the similarity thresholds, since it is the second-highest explaining model. Precision 1, 2, and recall remain consistent at around 0.12, 0.63, and 0.74 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.2 and 0.67.

Finally, in the Tinnitus score figure 10.11 tinnitus scores the highest out of the non-common diagnoses.

In the Tinnitus Precision and Recall table 10.10 the differential precision is consistently 1 throughout the similarity thresholds, since it is the highest explaining model. Precision 1, 2, and recall remain consistent at around 0.14, 0.6, and 0.93 respectively. However, at the higher thresholds, the precision metrics 1 and 2 do rise to 0.18 and 0.64.

## 10.1 Discussion

Every single BN scored the highest and did the best with the notes corresponding to it. That is, for all the notes, the BN model scores the highest with the notes pertaining to it, and is most likely to identify notes pertaining to it. This is true for all BN models on all three fronts; score, precision, and recall (accuracy).

However, not all notes, were necessarily best explained by the BNs pertaining to the same diagnosis.

So for example, a Pruritis note most likely would score the highest with the fatigue BN, but the Pruritis BN will identify a Pruritis note with a much higher likelihood that it will identify a note corresponding to any other diagnosis. This is evident in the precision and recall data.

Therefore is important to note that precision and recall are the most important factors in determining what BN is most likely to explain a note. While the score of a BN may be higher, it may not consistently be able to identify a note with that diagnosis to begin with.

Another important factor to note about these results is that the BN models that are explaining notes (score, precision, and recall wise) with a higher likelihood than they should are made up of extremely common entities that have many links and relations with words that can come up in a typical medical note.

For example, the fatigue BN words "fatigue", "depression", and "anxiety" have high scoring matches with almost all notes.

This is not necessarily inaccurate, as fatigue can be a side-effect of the original diagnosis we would like to obtain.

The headache BN contains many words (such as headache, throbbing, triggers, cough, tumor, seizure, sinusitis, ...) that are commonly found in EMR notes. This is due to the fact that the diagnostic algorithm for the headache diagnosis check for many possible diagnoses of which headache may be a common symptom.

Again, this is not necessarily inaccurate, as headache is likely to be a side-effect of the original diagnosis we would like to obtain.

The word "joint" corresponding to the joint sprain BN matches with an extreme amount of words and therefore the BN is more likely to score high with an irrelevant EMR note.

This is problematic, and is to be looked into further.

The entities relevant to the diabetes BN are also very common. For example, words such as lifestyle, sedentary, and obese and are related to a large selection of medical problems and thus medically significant words.

This is problematic as well, and is to be looked into further.

One suggestion to resolve these issues is by finding more diagnosis specific diagnostic algorithms and placing more importance on finding words that relate exclusively to the disease at hand.

Our results support that our BNs are successful to a degree in identifying the top matches for a note.

# Chapter 11

## Conclusion

EMR notes assist HCPs in managing cases and diagnosing patients.

The automation of understanding the unstructured notes stored therein improves health-care quality. Therefore, applying natural language processing methods on EMR notes improves a HCP's ability to diagnose patients and extract useful information. We utilized automated understanding to predict diagnoses from the notes by querying a set of BNs for the likelihood of their relevance to it.

We utilize a cross-document analysis method of EMR notes that pertain to the diagnoses diabetes mellitus and anemia. This method identifies closely related entities in the data. It establishes a score to rate the percentage of their similarity. We did this with distributional similarity measurements based on HCP annotations, such as abbreviations and words of particular importance to the diagnosis at hand, and vowel based variations of words, and three distributional similarities, the first based on EMR note texts, the first based on USMLE exam texts, and the third based on Pubmed corpora.

We enriched our annotations by diagnosis graphs and to help deduce a suitable diagnosis for the note in a faster and more efficient way.

The diagnostic algorithms from medical textbooks allowed us to build BNs structures for the diagnosis of ten diagnoses: anemia, anxiety, diabetes, headache, hemoptysis, fatigue, joint sprain, kidney disease, pruritis, and tinnitus.

Each structured model was trained with yes/no data based on the existence of a node match in the EMR notes pertaining to its diagnosis.

The existence of a node is determined by whether or not a word vector to word vector match score surpasses a threshold. The thresholds used are 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9. This therefore generates a BN model for each threshold.

Once the BN parameters are found, all the notes of all the diagnoses are

processed to find their match data with all the BNs at all the thresholds.

We did this to identify the top three BNs that could explain each set of notes pertaining to different diagnoses.

We used off the shelf tools Java and Python to stimulate our final results.

We validated our results and have found the optimal BN models for the ten diagnoses given our data.

The models yielded good results, however, some diagnoses (fatigue, headache, joint sprain, and sometimes diabetes) were more common in irrelevant notes that is favorable.

This could perhaps be due to the fact that the entities relevant to the corresponding BNs are very common.

This may be resolved by finding more diagnosis specific entities to add to the BN structure. This could place more importance on finding words that relate exclusively to the disease we are trying to identify.

Our BN models help to speed up the diagnosis process of these diseases, improve organization of sparse medical data, and improve meaningful use of available information.

Analyzing the EMRs and relating them with the diagnosis graphs could be very helpful in detecting diagnoses early on.

# Appendix A

## Abbreviations

BN	Bayesian Networks
EMR	Electronic Medical Record
HCP	Health Care Practitioner
A1	Anemia
A2	Anxiety
D	Diabetes
F	Fatigue
H1	Headache
H2	Hemoptysis
J	Joint Sprain
K	Kidney Disease
P	Pruritus
T	Tinnitus

# Bibliography

- [1] C. Riggio, "'what's the deal with accuracy, precision, recall and f1?',"
- [2] J. A. Carroll, R. Koeling, and S. Puri, "Lexical acquisition for clinical text mining using distributional similarity," in *Computational Linguistics and Intelligent Text Processing*, p. 232 to 246, 2012.
- [3] L. Falcon, "Gnu health: Health and hospital information system,"
- [4] G. K. Savova, J. J. Masanz, P. V. Ogren, J. Zheng, S. Sohn, K. K. Schuler, and C. G. Chute, "Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications," vol. 17 5, pp. 507–13, 2010.
- [5] Glaze and Jeff, "Epic systems draws on literature greats for its next expansion," 2015.
- [6] "The benefits of ehra department of health and human services office of the national coordinator for health information technology,"
- [7] Rijo, Rui, Martinho, Ricardo, and C. Pereira, Silva, "Text mining applied to electronic medical records: A literature review," vol. 6, pp. 1–18, July 2015.
- [8] S. M. Meystre, K. S. Guergana, C. K.-S. Karin, and F. H. John, "Extracting information from textual documents in the electronic health record," 2008.
- [9] W. Sun, Z. Cai, Y. Li, F. Liu, and S. Fang, "Data processing and text mining technologies on electronic medical records: A review," in *Journal of healthcare engineering*, 2018.
- [10] "'mena-mtc: Middle east and north africa medical text corpora',"
- [11] H. MC, T. LM, Jr., and S. GW, "The patient history: An evidence-based approach to differential diagnosis," NY: McGraw-Hill.

- [12] P. Kolb, "Disco: A multilingual database of distributionally similar words," in *In Proceedings of KONVENS*.
- [13] A. Darwiche, "What are bayesian networks and why are their applications growing across all fields?," 2010.
- [14] P. B. Jensen, L. J. Jensen, and S. Brunak, "Mining electronic health records: towards better research applications and clinical care," 2012.
- [15] D. B. K. S. Ananiadou and J. i. Tsujii, "Text mining and its potential applications in systems biology," 2006.
- [16] W. S. e. a. Zeng Q, Goryachev S, "Extracting principal diagnosis, comorbidity, and smoking status for asthma research: evaluation of a natural language processing system.," 2006.
- [17] S. A. e. a. Mack R, Mukherjea S, "Text analytics for life science using the unstructured information management architecture.," 2004.
- [18] S. I. T. M. M. J. S. K. C. J. G. W. d. G. P. Coden A, Savova G, "Automatically extracting cancer disease characteristics from pathology reports into a disease knowledge representation model," 2009.
- [19] "Health information technology for economic and clinical health act title xiii american recovery and reinvestment act of 2009,"
- [20] B. Shilling, "The federal government has put billions into promoting electronic health record use: How is it going?," 2011.
- [21] O. P. e. a. Savova GK, Masanz JJ, "Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications.," 2009.
- [22] Kushima, Muneo, Araki, Kenji, Suzuki, Muneou, and Araki, "Text data mining of the electronic medical record of the chronic hepatitis patient," vol. 2195, pp. 569–573, 03 2012.
- [23] J. Weeds and D. Weir, "Co-occurrence retrieval: A flexible framework for lexical distributional similarity," vol. 31, pp. 439–475, 2005.
- [24] Koeling, Rob, Tate, A. Rosemary, Carroll, and J. A., "Automatically estimating the incidence of symptoms recorded in gp free text notes," in *Proceedings of the First International Workshop on Managing Interoperability and Complexity in Health Systems, MIXHS '11*, 2011.

- [25] Y. Luo, "Recurrent neural networks for classifying relations in clinical notes," vol. 72, pp. 85–95, 2017.
- [26] Y. Ye, F. Tsui, M. Wagner, J. U. Espino, and Q. Li, "'influenza detection from emergency department reports using natural language processing and bayesian network classifiers'," vol. 21, pp. 815–823, 01 2014.
- [27] H. Mahgoub, D. Rösner, N. Ismail, and F. Torkey, "A text mining technique using association rules extraction,"
- [28] P. Luis, R. Rui, S. Catarina, and M. Ricardo, "Text mining applied to electronic medical records: A literature review," 2015.
- [29] S. Li, C. Jianping, and X. Jie, "Prospecting information extraction by text mining based on convolutional neural networks—a case study of the lala copper deposit, china," 2018.
- [30] Y. Luo and P. Szolovits, "Efficient queries of stand-off annotations for natural language processing on electronic medical records," vol. 8, 2016.
- [31] O. Metskera, E. Bolgova, A. Yakovleva, A. Funknera, and S. Kovalchuk, "Pattern-based mining in electronic health records for complex clinical process analysis," vol. 119, 2017.
- [32] A. Henriksson, M. Hassel, and M. Kvist, "Diagnosis code assignment support using random indexing of patient records—a qualitative feasibility study," p. 348 to 352.
- [33] M. Kushim, K. Araki, M. Suzuki, S. Araki, and T. Nikama, "Text data mining of the electronic medical record of the chronic hepatitis patient," vol. 1, 2012.
- [34] A. A. Thomas, C. Zheng, H. J. A. Chang, B. Kim, J. Gelfond, J. Slezak, K. Porter, S. J. Jacobsen, and G. W. Chien, "Extracting data from electronic medical records: validation of a natural language processing program to assess prostate biopsy results," vol. 32, 2014.
- [35] F. E, C. JA, S. HE, S. D, and C. JA, "Extracting information from the text of electronic medical records to improve case detection: a systematic review.," in *J Am Med Inform Assoc*, 2016.
- [36] H. BI, M. R, F. JC, and M. AL., "Merging data diversity of clinical medical records to improve effectiveness," in *Int J Environ Res Public Health*, 2019.



- [37] Z. V, B. R, B. R, M. R, D. R, and W. M., "Minimalistic approach to coreference resolution in lithuanian medical records," in *Comput Math Methods Med*, 2019.
- [38] A. M, D. LG, C. A, S. A, and C. F, "The revival of the notes field: Leveraging the unstructured content in electronic health records," in *Comput Math Methods Med*, 2019.
- [39] L. KH, K. HJ, K. YJ, K. JH, and S. EY, "Extracting structured genotype information from free-text hla reports using a rule-based approach," in *J Korean Med Sci*, 2020.
- [40] K. TA, D. C, B. PE, and B. S, "Natural language processing of symptoms documented in free-text narratives of electronic health records: a systematic review," 2019.
- [41] G. EP, W. A, K. C, and et al., "'what is the best method of family planning for me?': a text mining analysis of messages between users and agents of a digital health service in kenya.," 2019.
- [42] X. Y, W. C, Y. X, W. W, Z. E, and Y. J, "Parabtm: A parallel processing framework for biomedical text mining on supercomputers," 2018.
- [43] W. Sun, Z. Cai, Y. Li, F. Liu, S. Fang, and G. Wang, "Data processing and text mining technologies on electronic medical records: A review," 2018.
- [44] C. R, E. M, and E. N, "Redundancy in electronic health record corpora: analysis, impact on text mining performance and mitigation strategies," 2013.
- [45] L. Y, D. Y, L. M, C. Y, and L. Q, "Zhongguo yi liao qi xie za zhi," 2016.
- [46] K. Murphy, "'machine learning: a probabilistic perspective',"
- [47] Stats and Bots, "'probabilistic-graphical-models-tutorial',"
- [48] W. contributors, "'euclidean distance'," Wikipedia, The Free Encyclopedia.
- [49] Yujian, Li, Bo, and Liu, "A normalized levenshtein distance metric," vol. 29, p. 1091 to 1095, June 2007.
- [50] Vorontsov, I. Kulakovskiy, and I. Makeev, "Algorithms molecular biology,"

- [51] J. Feigenbaum, "Jarowinkler: Stata module to calculate the jaro-winkler distance between strings,"
- [52] ICD=10, "" international statistical classification of diseases and related health problems","
- [53] ICD-9, "" international statistical classification of diseases and related health problems","
- [54] SynonymThesaurus, ""free online thesaurus","
- [55] M. Net, "" common medical abbreviations list (acronyms and definitions) center","