ABSTRACT

ENRICHMENT OF ONTOLOGIES USING MACHINE LEARNING AND SUMMARIZATION

by
Hao Liu

Biomedical ontologies are structured knowledge systems in biomedicine. They play a major role in enabling precise communications in support of healthcare applications, e.g., Electronic Healthcare Records (EHR) systems. Biomedical ontologies are used in many different contexts to facilitate information and knowledge management. The most widely used clinical ontology is the SNOMED CT. Placing a new concept into its proper position in an ontology is a fundamental task in its lifecycle of curation and enrichment.

A large biomedical ontology, which typically consists of many tens of thousands of concepts and relationships, can be viewed as a complex network with concepts as nodes and relationships as links. This large-size node-link diagram can easily become overwhelming for humans to understand or work with. Adding concepts is a challenging and time-consuming task that requires domain knowledge and ontology skills. “IS-A links” (aka subclass links) are the most important relationships of an ontology, enabling the inheritance of other relationships. The position of a concept, represented by its IS-A links to other concepts, determines how accurately it is modeled. Therefore, considering as many parent candidate concepts as possible leads to better modeling of this concept.

Traditionally, curators rely on classifiers to place concepts into ontologies. However, this assumes the accurate relationship modeling of the new concept as well as the existing concepts. Since many concepts in existing ontologies, are underspecified in terms of their relationships, the placement by classifiers may be wrong. In cases where

the curator does not manually check the automatic placement by classifier programs, concepts may end up in wrong positions in the IS-A hierarchy. A user searching for a concept, without knowing its precise name, would not find it in its expected location.

Automated or semi-automated techniques that can place a concept or narrow down the places where to insert it, are highly desirable. Hence, this dissertation is addressing the problem of concept placement by automatically identifying IS-A links and potential parent concepts correctly and effectively for new concepts, with the assistance of two powerful techniques, Machine Learning (ML) and Abstraction Networks (AbNs).

Modern neural networks have revolutionized Machine Learning in vision and Natural Language Processing (NLP). They also show great promise for ontology-related tasks, including ontology enrichment, i.e., insertion of new concepts. This dissertation presents research using ML and AbNs to achieve knowledge enrichment of ontologies.

Abstraction networks (AbNs), are compact summary networks that preserve a significant amount of the semantics and structure of the underlying ontologies. An Abstraction Network is automatically derived from the ontology itself. It consists of "nodes," where each node represents a set of concepts that are similar in their structure and semantics. Various kinds of AbNs have been previously developed by the Structural Analysis of Biomedical Ontologies Center (SABOC) to support the summarization, visualization, and quality assurance (QA) of biomedical ontologies. Two basic kinds of AbNs are the *Area Taxonomy* and the *Partial-area Taxonomy*, which have been developed for various biomedical ontologies (e.g., SNOMED CT of SNOMED International and NCIt of the National Cancer Institute). This dissertation presents four enrichment studies of SNOMED CT, utilizing both ML and AbN-based techniques.

ENRICHMENT OF ONTOLOGIES USING MACHINE LEARNING AND SUMMARIZATION

by
Hao Liu

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Science

Department of Computer Science

August 2020

APPROVAL PAGE

ENRICHMENT OF ONTOLOGIES USING MACHINE LEARNING AND SUMMARIZATION

Hao Liu

Dr. Yehoshua Perl, Dissertation Co-Advisor Date
Professor of Computer Science, NJIT

Dr. James Geller, Dissertation Co-Advisor Date
Professor of Computer Science, NJIT

Dr. Michael Halper, Committee Member Date
Professor of Informatics and IT Division Director, NJIT

Dr. Zhi Wei, Committee Member Date
Professor of Computer Science, NJIT

Dr. Chunhua Weng, Committee Member Date
Professor of Biomedical Informatics, Columbia University

Dr. Huanying (Helen) Gu, Committee Member Date
Professor of Computer Science, NYIT

BIOGRAPHICAL SKETCH

Author:	Hao Liu
Degree:	Doctor of Philosophy
Date:	August 2020

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 2020

- Master of Science in Electrical Engineering,
  Columbia University, New York, NY, 2012

- Bachelor of Science in Electrical and Computer Engineering,
  New York Institute of Technology, New York, NY, 2011

- Bachelor of Science in Telecommunication Engineering,
  Nanjing University of Posts and Telecommunications, Nanjing, P.R. China, 2011

Major: Computer Science

Publications:

Published Journal Papers

Elhanan G, Ochs C, Mejino Jr JL, Liu H, Mungall CJ, Perl Y. From SNOMED CT to Uberon: transferability of evaluation methodology between similarly structured ontologies. Artificial Intelligence in Medicine. 2017 Jun 1;79:9-14.

Journal Papers in Progress

Liu H, Perl Y, Geller J. Concept placement using BERT trained by transforming and summarizing biomedical ontology structure. Journal of biomedical informatics. Submitted for review.

Liu H, Perl Y, Geller J, Elhanan G. Machine Learning with Embedding Similarity for Terminology Placement. Journal of biomedical informatics. Submitted for review.

Published Conference Papers

Liu H, Perl Y, Geller J. Transfer Learning from BERT to Support Insertion of New Concepts into SNOMED CT. In AMIA Annual Symposium Proceedings (p. 1129). American Medical Informatics Association. 2019.

Zheng L, Liu H, Perl Y, Geller J. Training a Convolutional Neural Network with Terminology Summarization Data Improves SNOMED CT Enrichment. In AMIA Annual Symposium Proceedings (p. 972). American Medical Informatics Association. 2019.

Liu H, Hildebrand PL, Perl Y, Geller J. Enrichment of SNOMED CT Ophthalmology Component to Support EHR Coding. In International Conference on Bioinformatics and Biomedicine (BIBM) (pp. 1990-1997). IEEE. 2018.

Liu H, Geller J, Halper M, Perl Y. Using Convolutional Neural Networks to Support Insertion of New Concepts into SNOMED CT. In AMIA Annual Symposium Proceedings (p. 750). American Medical Informatics Association. 2018

Zheng L, Liu H, Perl Y, Geller J, Ochs C, Case JT. Overlapping complex concepts have more commission errors, especially in intensive terminology auditing. In AMIA Annual Symposium Proceedings (p. 750). American Medical Informatics Association. 2018.

Liu H, Chen L, Zheng L, Perl Y, Geller J. A quality assurance methodology for ChEBI Ontology focusing on uncommonly modeled concepts. International Conference on Biological Ontology. 2018.

Liu H, Zheng L, Perl Y, Geller J, Elhanan G. Can a Convolutional Neural Network support auditing of NCI thesaurus neoplasm concepts? International Conference on Biological Ontology. 2018.

Zheng L, Ochs C, Geller J, Liu H, Perl Y, De Coronado S. Multi-layer Big Knowledge visualization scheme for comprehending neoplasm ontology content. IEEE International Conference on Big Knowledge (ICBK) (pp. 127-134). 2017.

Liu H, Dong Z, Gu H. Microblogging as a social sensing tool. In Proceedings of the 11th IEEE International Conference on Networking, Sensing and Control (pp. 513-517). IEEE. 2014

Posters

Liu H, Zheng L, Perl Y, Chen Y, Elhanan G. Correcting ontology errors simplifies visual complexity. Studies in health technology and informatics. 2017.

Presentations:

Transfer Learning from BERT to Support Insertion of New Concepts into SNOMED CT. 2019 AMIA Annual Symposium. Washington, D.C., USA, November 20, 2019.

Using convolutional neural networks to support insertion of new concepts into SNOMED CT. 2018 AMIA Annual Symposium. San Francisco, CA, USA, November 5, 2018.

Can a Convolutional Neural Network support auditing of NCI thesaurus neoplasm concepts? 2018 International Conference on Biological Ontology. Corvallis, OR, USA, August 7, 2018.

A quality assurance methodology for ChEBI Ontology focusing on uncommonly modeled concepts. 2018 International Conference on Biological Ontology. Corvallis, OR, USA, August 7, 2018.

To my beloved parents, Mr. Hutao Liu and Mrs. Peilan Cao

谨以此博士论文献给我挚爱的父母刘护涛先生和曹培兰女士

ACKNOWLEDGMENT

I would like to express my sincere gratitude and appreciation to my dissertation co-advisors Dr. Yehoshua Perl and Dr. James Geller for their continuous guidance, support, and encouragement of my Ph.D. study. Their expertise and professional dedication to scientific research has had a profound effect upon me. The journey of being their Ph.D. student is a truly precious experience and a lifelong benefit. I am grateful to Dr. Michael Halper for his help on some of the research projects described in the dissertation. I am grateful to Dr. Zhi Wei, Dr. Chunhua Weng, and Dr Hellen Gu for their insightful comments and feedbacks on my research and serving on my dissertation committee.

The research projects in this dissertation were partially supported by the National Cancer Institute of the National Institutes of Health under Award Number R01CA190779.

Special thanks are given to Dr. Christopher Ochs, Dr. Ling Zheng, Dr. Gai Elhanan, Dr. P Lloyd Hildebrand, Dr. Ling Chen, Dr. Hua Min, Dr. Yan Chen, and many other collaborators for all of their contributions to the work in the dissertation. I also would like to thank Prof. Janet Bodner for her continuous help with my English.

Lastly, I want to thank my parents, Hutao Liu and Peilan Cao; my friends, Qin Ling, Huayang Fang, Qizhao Xue, Pengwei Huang, Zheng Ling, and Vipina Kuttichi Keloth; Dr. Helen Gu and Dr. Ziqian Dong, who are respectable teachers and close friends.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF TABLES
(Continued)

LIST OF FIGURES

LIST OF FIGURES
(**Continued**)

LIST OF FIGURES
(**Continued**)

LIST OF FIGURES
(Continued)

CHAPTER 1

INTRODUCTION

1.1 Motivation

Biomedical ontologies are structured and organized knowledge systems within the domain of biomedicine. They play a major role in enabling precise communications and in support of healthcare applications, e.g., Electronic Healthcare Records (EHR) systems [1-3]. Biomedical ontologies are used in many different contexts to facilitate information utilization [4] and knowledge management [5]. For example, the Chemical Entities of Biological Interest (ChEBI) ontology [6] is a large important knowledge source that facilitates reference to chemical entities within the biological field. It annotates small distinguishable entities such as atoms, ions, and polymers and their relationships to each other.

Knowledge in ontologies is represented as *concepts* and *relationships*. A *concept* represents a unique entity while a *relationship* represents a connection between two entities. *Relationships* can be further categorized into *hierarchical relationships* (e.g., x **IS-A** y, or x is a ***SUBCLASS*** of y) and *lateral relationships* (i.e., non-hierarchical connections). The hierarchical IS-A relationship between two concepts represents the fact that one concept is a specialization of the other concept. For example, within ChEBI, the concept *Ionic Polymer* has an IS-A relationship to the concept *Polymer* because ionic polymer is a specialization of polymer. A *lateral relationship* represents a semantic connection between two concepts, for example, the concept *Vitamin E* has a *has role* relationship to the concept *Fat-soluble vitamin* because Vitamin E dissolves in fats and is stored in body tissues.

The NCBO BioPortal [7] is the largest existing repository of biomedical ontologies that hosts 837 ontologies (as of January 2020), covering more than 11,193,698 classes (concepts). This dissertation focuses on large biomedical ontologies, which typically have complex network structures consisting of many thousands of concepts and relationships. For example, the National Cancer Institute Thesaurus (NCIt) [8], serving cancer researchers inside and outside of National Institutes of Health (NIH), contains 156,172 concepts interrelated by more than 480,141 relationships in the February 2020 release.

Placing a new concept into its proper position in the hierarchy of an ontology is a fundamental task in the lifecycle of curation and enrichment of an ontology. It is a challenging and time-consuming task as it requires both domain knowledge and ontology skills. IS-A links and concepts form the backbone structure of an ontology, enabling the inheritance of lateral relationships. The position of a concept, represented by its IS-A relationships to other concepts, determines how accurately it is modeled in terms of granularity. Therefore, considering as many related parent candidate concepts as possible leads to a more comprehensive modeling of this concept. Traditionally, curators rely on classifiers such as Snorocket [9] or HermiT [10] to place concepts into ontologies based on a Description Logic. However, this approach relies on the relationship modeling of the new concept as well as the relationship modeling of existing concepts. Since many concepts in a Description Logic ontology, like SNOMED CT [11], are underspecified in terms of their relationships, the placement by classifiers may be wrong. In cases where the curator does not manually check the automatic placement by classifiers, concepts may end up in wrong positions in the hierarchy. Hence, a user searching for such a concept, without knowing its name in SNOMED, would not find it in its expected location. Thus, a machine learning model that automatically identifies a set of candidate parent concepts for a new concept can assist curators to improve the accuracy of placement of a new concept in the process of ontology curation.

Automated or semi-automated techniques that can help in placing a concept or narrow down the places where to insert it, are highly desirable. Hence, this dissertation is trying to address the challenges of concept placement by automatically identifying sets of potential parent concepts correctly and effectively for any new concepts, with the assistance of two powerful techniques, i.e., Machine Learning (ML) and Abstraction Networks (AbNs).

Machine Learning (ML) has been proven successful in many fields, e.g., Natural Language Processing (NLP) for knowledge mining. Some ML models were previously used in knowledge enrichment for ontologies [12-14]. Knowledge enrichment mines external sources for new knowledge that does not exist in the ontology. This dissertation presents experiments with using several ML and NLP models to address the tasks of knowledge enrichment of biomedical ontologies.

Abstraction networks (AbNs), are compact summary networks that preserve a significant amount of the semantics and structure of the underlying ontologies. An Abstraction Network is automatically derived from the ontology itself. It consists of “nodes,” where each node represents a set of concepts that are similar in their structure and semantics. Various kinds of Abstraction Networks (AbNs) have been previously developed by the Structural Analysis of Biomedical Ontologies Center (SABOC) [15] to support the summarization, visualization and quality assurance (QA) of biomedical ontologies [16]. Two basic kinds of AbNs are the area taxonomy and the partial-area taxonomy, which have been derived for various biomedical ontologies (e.g., SNOMED CT [17] and NCIt [18]).

To extend the applicability of existing AbNs and show the effectiveness of ML-based techniques for enrichment of biomedical ontologies, this dissertation presents five enrichment studies on SNOMED CT, utilizing both ML and AbN-based techniques.

1.2 Dissertation Overview

Chapter 2 provides background information on biomedical ontologies used in this dissertation, i.e., SNOMED CT. Chapter 2 also introduces the Abstraction Networks for biomedical ontologies developed by the SABOC team, and the previous enrichment studies of biomedical ontologies based on these Abstraction Networks. In addition, the ML techniques used for ontology enrichment in this dissertation are discussed.

Chapter 3 presents a study of employing Convolutional Neural Networks (CNNs) to support knowledge enrichment of SNOMED CT. In this study, a CNN model is trained to predict the placement of new concepts into SNOMED CT. This study employs a vectorization technique to map each concept into a vector representation.

Chapter 4 reports a study extending the CNN model used in Chapter 3 with one type of Abstraction Networks for the enrichment of SNOMED CT's *Clinical Finding* hierarchy. The results confirmed that the CNN model trained with data prepared by Abstraction Networks performed better than the model trained with randomly selected training data. This study shows that ML can benefit from using AbNs to obtain a better relationship classification model for biomedical ontologies.

Chapter 5 describes a study which employs a model from an NLP framework to address the enrichment task of biomedical ontologies. Using SNOMED CT's *Clinical*

Finding and *Specimen* hierarchies as testbeds, a language representation model, called *Bidirectional Encoder Representations from Transformers* (BERT), was used for the identification of IS-A relationships for newly added concepts. Utilizing the “next sentence prediction” capability of BERT, this study shows that the Fine-tuning strategy of Transfer Learning (TL) from the BERT model can support automatic terminology enrichment – namely insertion of new concepts at the correct locations.

Chapter 6 reports on a study that further improves the performance of the BERT model in Chapter 5. This study combines these two improvements: 1) utilizing ontology Abstraction Networks together with the BERT model; 2) an improved presentation of the training data. These two approaches further improve the model’s recall on relationship classifications.

Chapter 7 reports on a study that introduces a similarity-based algorithm to identify parent(s) for a new concept. The algorithm utilizes a similarity score of concept level embeddings to identify proper candidate parents and two rules to filter out improper parents. The results show that this approach can assist curators with the tasks of ontology versioning.

Chapter 8 describes ideas for future work and Chapter 9 concludes this dissertation. The studies in this dissertation have been accepted by related conferences on biomedical informatics. The study in Chapter 3 was published in the American Medical Informatics Association (AMIA) 2018 Annual Symposium proceedings. The studies in Chapter 4 and in Chapter 5 were published in the AMIA 2019 Annual Symposium proceedings.

CHAPTER 2

BACKGROUND

2.1 Biomedical Ontologies

Biomedical ontologies are helpful in providing a structured knowledge framework to support enhanced information encoding and interoperability in healthcare systems. They have been widely used to facilitate research in the domain of biomedical informatics, including for natural language processing (NLP) tasks, e.g., entity relationship retrieval [19, 20], knowledge mining/enrichment [21-23], and other applications [24-26]. This section will introduce several important and large biomedical ontologies relevant to this dissertation. In general, in this dissertation, ontologies and terminologies will be referred to by the common term “ontologies.” When referring to a specific ontology or terminology the appropriate term will be used as preferred by its curators (e.g., SNOMED CT terminology, NCIt ontology).

2.1.1 SNOMED CT

SNOMED CT (SNOMED Clinical Terms) [27] is a comprehensive and widely used clinical healthcare terminology, covering many subdomains of medicine and healthcare. It is utilized to represent and share clinical information accurately and consistently in Electronic Health Records (EHRs), thus facilitating the interfacing between different healthcare organizations and different systems in the same organization. SNOMED International, formerly the International Health Terminology Standards Development Organization (IHTSDO), oversees the maintenance, development, quality assurance, and distribution of SNOMED CT. Twice a year (in January and July), SNOMED International releases a new version of the SNOMED CT International Edition. The two essential components of SNOMED CT are concepts and the relationships connecting those concepts. Under the root concept of SNOMED CT there are 19 disjoint hierarchies of concepts, connected to the root by IS-A relationships. Examples of hierarchies include *Clinical finding*, *Procedure*, and *Specimen*.

There are two types of relationships to provide formal definitions of concepts. One type is the IS-A relationship, connecting a concept to a more general concept in the same hierarchy. The more general concept is called the parent of the more specific concept, while the more specific concept is the child concept. Every concept (with one exception, the root of the ontology) has at least one parent concept, but it may have several parents. The other type of relationship is called "attribute relationship." Attribute relationships, which are "lateral relationships," define characteristics of concepts. For example, the attribute relationship *Causative Agent* connects the concept *Bacterial cellulitis* to the bacteria causing it, such as *Superkingdom Bacteria*.

SNOMED CT's 2018 January release contains 341,105 active concepts connected by 511,766 IS-A links. Furthermore, there are 550,308 attribute relationships between pairs of concepts. The *Clinical finding* hierarchy and the *Procedure* hierarchy, two of the largest hierarches in SNOMED CT, consist of 111,081 concepts (33%) and 57,806 concepts (17%), respectively. The *Infectious disease* subhierarchy and the *Congenital disease* subhierarchy of the *Clinical finding* hierarchy are comprised of 6,681 and 8,126 concepts, respectively.

SNOMED CT is primarily released in its own RF2 format as a set of tab-delimited text files for its different components including individual files for concepts and relationships. For relationships, there are two different files. One is for stated relationships, which are explicitly entered by SNOMED CT curators/editors. The other one is for inferred relationships, which are obtained by running a classifier program on the stated relationships. This dissertation focuses on the inferred view including both kinds of relationships. Each RF2 release also provides 'delta' files to record the changes since the previous release. According to the SNOMED CT release notes in July 2015 and January 2016, revisions of the *Infectious disease* and the *Congenital disease* subhierarchies were initiated to detect and resolve their modeling inconsistencies brought up by external reviewers.

Figure 2.1 shows an excerpt of 15 concepts from the *Specimen* hierarchy with 1,620 concepts. The concept *Tissue specimen from digestive system* has two parents *Specimen from digestive system* and *Body substance sample*. The dashed green box summarizes three specimen concepts that share the same lateral relationship *Specimen source topography*. For example, the concept *Specimen from liver* has a lateral relationship *Specimen source topography* linking it to *Liver structure* (not shown in the diagram).

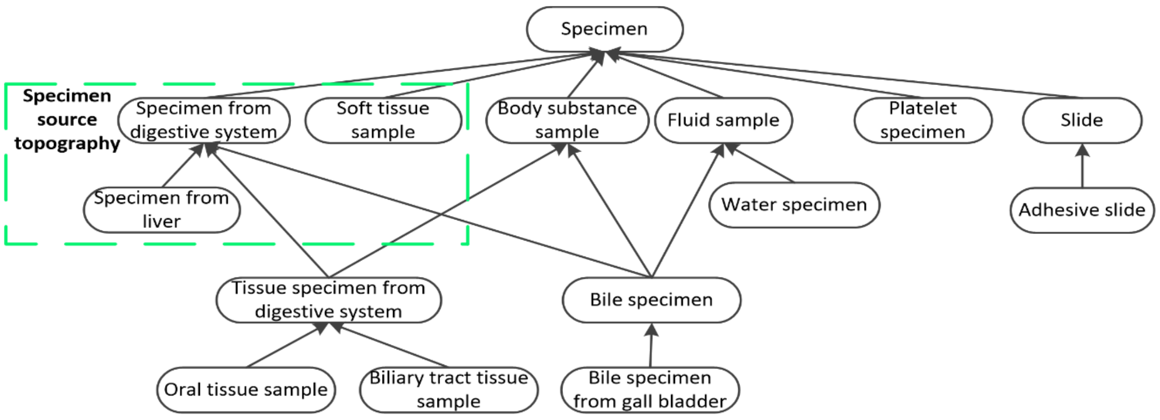


Figure 2.1 Excerpt of 15 concepts from the Specimen hierarchy of SNOMED CT. Concepts, represented by boxes with rounded corners, are connected by IS-A relationships shown as upward arrows.
Source: [28]

Extensive research has been conducted on SNOMED CT's content quality. Ceusters *et al*. [29] applied two algorithms to SNOMED CT and found several kinds of errors like improper assignment of IS-A relationships and attribute relationships. Ceusters [30] also utilized the Evolutionary Terminology Auditing technique to analyze 18 SNOMED CT releases and the results recommended explicitly documenting changes in SNOMED CT. Zhang and Bodenreider [31] demonstrated the Lattice-based Structural Auditing method for the quality assurance of SNOMED CT. Elhanan *et al*. [32] performed a user survey showing that direct users (working with SNOMED CT, as opposed to working with a tool that has SNOMED CT as a backend database) have a strong desire for an improved quality of SNOMED CT. Agrawal *et al*. [33] used a lexical approach to identify problematic concepts in SNOMED CT. Several Abstraction Network-based studies [34, 35] investigated the quality of different hierarchies in SNOMED CT. The journal special issue on auditing of terminologies [36] provides a summary of publications on terminology auditing and in particular on auditing of SNOMED CT [37].

2.2 Abstraction Networks for Biomedical Ontologies

Knowledge in large biomedical ontologies is beyond humans' comprehension ability; summarization or abstraction can help with this issue. In order to facilitate the comprehension of the complex content in biomedical ontologies, in a long range research program, the SABOC team [38] has developed an Abstraction Network-based framework to support the summarization and visualization of biomedical ontologies. An Abstraction Network (AbN) of an ontology is a compact summary network consisting of "nodes," each representing a set of concepts that are *similar* in their structure and semantics. Nodes are connected by hierarchical *child-of* links that are derived from the IS-A relationships in the ontology.

The definition of “similar” depends on an ontology’s structural characteristics and is not the same for all ontologies, hence there are various types of Abstraction Networks. For example, the SABOC team has developed the *area taxonomies* and *partial-area taxonomies* [17, 18, 39] for the National Cancer Institute thesaurus (NCIt) [8], SNOMED CT [40], and the Gene Ontology [41]. Furthermore, the *disjoint partial-area taxonomies* [42] and the *tribal abstraction networks* [43] have been designed for SNOMED CT. Besides, they have introduced the *domain-defined partial-area taxonomy* [44, 45] for the Ontology of Clinical Research (OCRe) [46] and the Cancer Chemoprevention Ontology (CanCo) [47], the *restriction-defined partial-area taxonomy* [48] for the Sleep Domain Ontology (SDO) [49], and the *domain-defined and restriction-defined partial-area taxonomies* [50] for the Drug Discovery Investigations Ontology [51]. An extensive review of Abstraction Networks has been presented by Halper et al. [16]. The Ontology Abstraction Framework (OAF) created by Ochs et al. [52] is an open source software system and tool for deriving Abstraction Networks, which is available at http://saboc.njit.edu/. The following sections will describe the Abstraction Networks associated with this dissertation using as example neoplasm concepts from NCIt.

2.2.1 Area Taxonomy

Area Taxonomies were introduced by Min *et al*. [18] to achieve summarization of large ontologies. Ontology concepts with exactly the same set of lateral (i.e., non-IS-A) relationship types are grouped into an *area*. Areas, considered as nodes, are connected via child-of hierarchical links to form a network, called an Area Taxonomy, since it has only hierarchical relationships. Figure 2.2 illustrates the derivation of an Area Taxonomy. Figure 2.2(a) shows an excerpt of 14 concepts from SNOMED CT's *Clinical Finding* hierarchy, drawn as labeled ovals. A dashed rectangle contains a set of concepts each of which has exactly the same lateral relationship type(s). The list of relationship types for the concepts in each dashed rectangle appears in bold. The arrows denote IS-A links. Lateral relationships are inherited down along the IS-A links between concepts.

Figure 2.2(b) shows the Area Taxonomy for the excerpt of the subhierarchy in Figure 2.2(a). All colored dashed rectangles are represented as "nodes," (shown as colored rectangles), which are connected by hierarchical child-of links (drawn as bold arrows) that are derived from the IS-A relationships in the ontology. The list of the relationship types of an area is used as its name (in bold). For example, because *Bradyarrhythmia* and *Diastolic heart failure* (and two other concepts) in Figure 2.2(a) all have the same lateral relationship types, *Finding site* and *Has definitional manifestation*, they are grouped together as area node (represented as a green rectangle) in Figure 2.2(b).

Areas shown at the same level are displayed in the same color, indicating that all of their concepts have the same number of lateral relationship types. For example, the areas {*Finding site, Occurrence*} and {*Finding site, Has definitional manifestation*} appear in the second level in green. The concept *Heart disease* and its descendants in the grey rectangle in Figure 2.2(a) are represented by the area {*Finding site*} in Figure 2.2(b). Similarly, the concept *Neonatal bradycardia* and the concept *Fetal bradycardia* are represented by the red area {*Finding site, Has definitional manifestation, Occurrence*} in level 3. Areas inherit relationships along the hierarchical child-of links. For example, the red area inherits its relationships from both green areas.

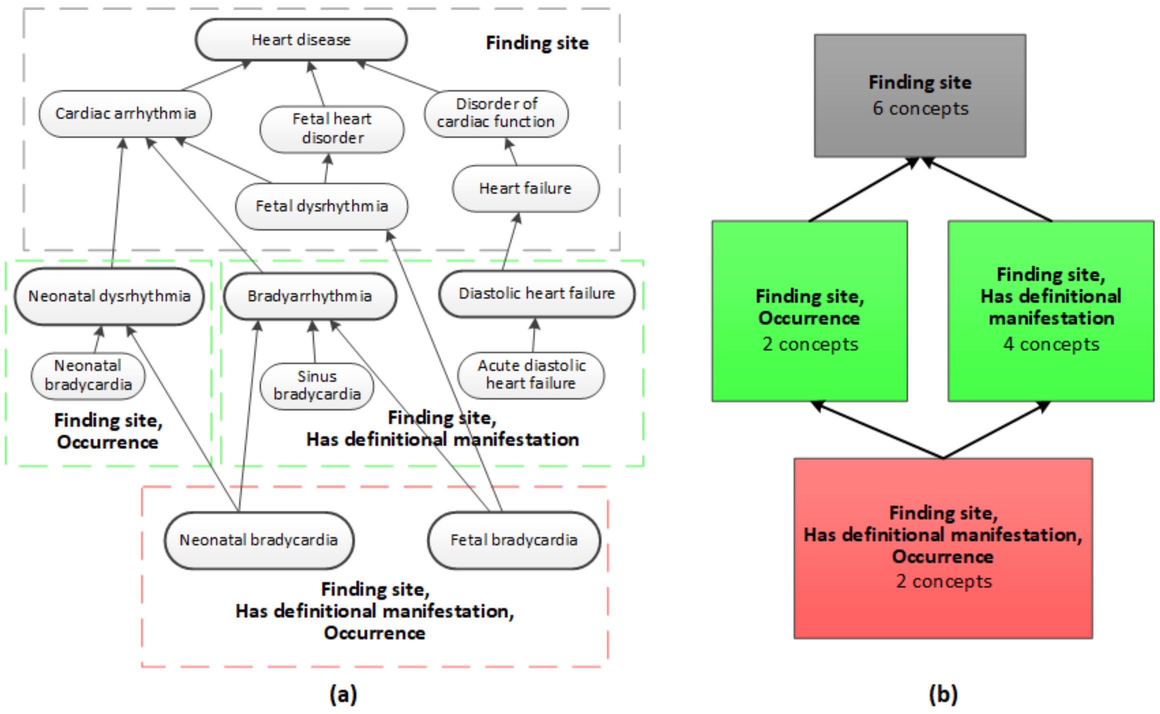

Heart disease
Finding site
Cardiac arrhythmia
Fetal heart disorder
Disorder of cardiac function
Heart failure
Fetal dysrhythmia
Neonatal dysrhythmia
Bradyarrhythmia
Diastolic heart failure
Neonatal bradycardia
Sinus bradycardia
Acute diastolic heart failure
Finding site, Occurrence
Finding site, Has definitional manifestation
Neonatal bradycardia
Fetal bradycardia
Finding site, Has definitional manifestation, Occurrence
(a)
Finding site 6 concepts
Finding site, Occurrence 2 concepts
Finding site, Has definitional manifestation 4 concepts
Finding site, Has definitional manifestation, Occurrence 2 concepts
(b)

Figure 2.2 Derivation of Area Taxonomy. (a) Excerpt of a subhierarchy of 14 concepts from SNOMED CT's Clinical Finding hierarchy. (b) Area Taxonomy for the excerpt subhierarchy in (a).

2.3 Enrichment of Biomedical Ontologies

Ontology enrichment [53] plays a critical role in the life cycle of an ontology. It is a process to periodically extend a terminology with the evolvement of domain knowledge by adding new concepts, relationships, axioms and rules. Early terminology enrichment relied on advances in the NLP field, with their enrichment processes being based on linguistic analysis of domain specific corpora, exploiting syntactic relations to identify new concepts, and extracting hierarchical and non-hierarchical relations and rules [53, 54]. With the success of Neural Networks and Deep Learning in the NLP field, recent work on terminology enrichment has mainly focused on deriving high-quality word embeddings for terminology Machine Learning (ML) techniques for concepts from large corpora. Pembeci *et al.* [14] proposed to use concept similarity scores computed via Word2vec models to discover related concepts as a domain-independent enrichment framework. Jayawardana *et al.* used word vector embeddings to derive candidate vectors and then trained an SVM model to calculate representative vectors for concepts [55].

This dissertation focuses on the task of placing a list of new concepts into their proper positions in an ontology. Moreover, the enrichment of adding a new concept implies adding of new hierarchical IS-A relationships required to connect this new concept to its parent(s) in order to place it in the proper position. The evaluation of ontology enrichment results is time-consuming, because the ontology curators need to manually review the added concepts and relationships. Zhang et al. [56] and Ceusters [30] used the differences between two consecutive releases of SNOMED CT [25] to test the performance of a terminology quality assurance technique. Similarly, evaluation of the enrichment results can be automated by comparing two consecutive releases of ontologies, referred to as old release and new release, respectively. The placements of new concepts in the new release of an ontology (done by the curators of the ontology) are used as gold standards for algorithmic placement of those same concepts. To test how many new concepts are properly positioned by a proposed technique, the equivalent problem is investigated: how many of the new concepts' IS-A relationships to their parents can be found by the proposed technique, with knowledge of only the old release.

2.4 Machine Learning Techniques

Natural Language Processing and Deep Learning are two fields that have attracted increasing research interest from both academia and industry. This dissertation proposes the combination of applying both NLP and deep learning techniques to the specific areas of ontology enrichment and quality assurance.

Distributed representations of words/phrases rely on the co-occurrence information between words obtained from large corpora of text. This assumes that words with similar or related meanings tend to occur in similar contexts. In the biomedical domain, this approach has been adopted to assist with various NLP tasks such as defining similarity and relatedness measures between clinical terms [57], word sense disambiguation [58], and entity linking by mapping text to concepts [59]. Most of these tasks have been studied with clinical records or biomedical knowledge bases, e.g., the Unified Medical Language System.

As most of the ML algorithms require the input in numeric format, vectorization of biomedical ontology data as fixed-length feature vectors is necessary before feeding the data into downstream ML algorithms or models. Word2vec, originally proposed by Mikolov *et al*. [60], is an efficient technique for learning high-quality word-wise distributed vector representations, while capturing syntactic and semantic word relationships from large corpora of general English text.

Figure 2.3 illustrates an example of using context words (“the,” “cat,” and “sat”) to predict the fourth word (“on”). In applying this method to biomedical text in the OHSUMED corpus (a collection of 348,566 biomedical research articles) [61], Word2vec has also been shown as a successful vectorization method for tasks such as grouping similar medical concepts [62]. In this research, Paragraph Vector (Doc2vec) [63], an extension of Word2vec to the paragraph or document level, is employed and its applicability tested with biomedical data.

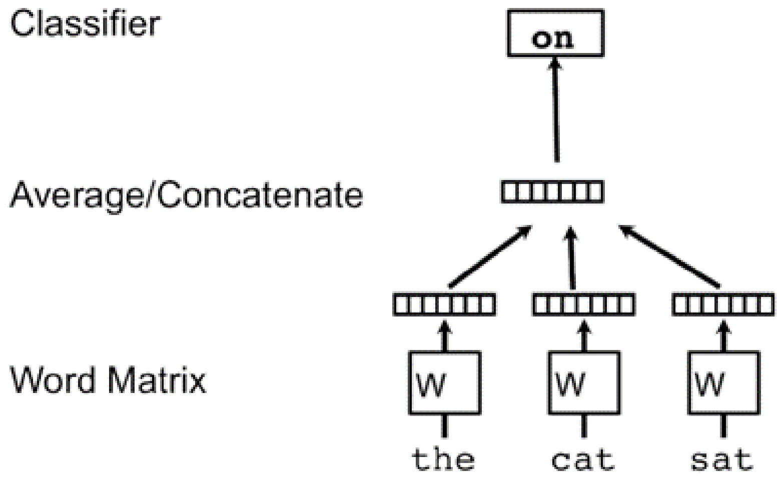


Figure 2.3 A framework for learning word vectors. Context of three words (“the,” “cat,” and “sat”) is used to predict the fourth word (“on”). The input words are mapped to columns of the matrix W to predict the output word.
Source: [63]

Deep Learning [64] has been successfully applied in many fields, such as computer vision [65] and speech recognition [66]. Convolutional Neural Networks (CNNs) as one basic and popular type of deep learning model, have been extensively studied and used for various applications including image recognition, semantic parsing, search query retrieval, sentence modeling, classification, prediction and other traditional NLP tasks. CNNs utilize layers with convolving filters that are applied to local features [67], which serve as input to trainable classifiers for prediction tasks. This dissertation focusses on using CNNs as the downstream model to solve ontology related classification and prediction problems that are essential to ontology enrichment and quality assurance.

2.4.1 Paragraph Vector (Doc2vec)

Numeric representation of variable-length texts, ranging from sentences to documents is a challenging task. Doc2vec, (Paragraph Vectors) [63], an extension of Word2vec (word embedding) [60], maps variable-length texts to fixed-length vectors. It is an unsupervised framework that learns continuous distributed vector representations from unlabeled text data of a paragraph/document, while preserving the inter-relationships within the text in a numeric format. In such vector representations, similar pieces of text are close to each other in Euclidean or cosine distance in lower dimensional vector spaces.

The Doc2vec representation inherits the semantics of the words in the context and takes the word order into consideration when constructing the representation. The latter is important to some hierarchy-related ontology problems, in which the concept order expresses a concept's topological/hierarchical position in the ontology. This is useful information for feature learning. An intensive literature search did not turn up any publications that uses Doc2vec to derive vector representations for biomedical ontology concepts.

The Distributed Memory Model of Paragraph Vectors (PV-DM) and the Distributed Bag of Words version of Paragraph Vectors (PV-DBOW) are two models introduced [63] to derive vectors for paragraphs (or documents, if working at the document level). Before training, words in the texts are mapped to unique vectors using Word2vec and every paragraph is also mapped to a unique vector. At every step of the training, a fixed-length context is sampled from a random paragraph and used to compute the error gradient in order to update the parameters in the model. In the PV-DM model, the processing involves sliding a window over a paragraph so that the order of words is taken into consideration. Then, the paragraph vector is concatenated with the word vectors to predict the next word in a context. For example, in Figure 2.4, the PV-DM model concatenates this paragraph vector with a context representation of three words "the," "cat," and "sat" to predict the fourth word "on." The paragraph vector is used to represent the missing information from the current context; thus it can be viewed as a memory of the topic of the paragraph.

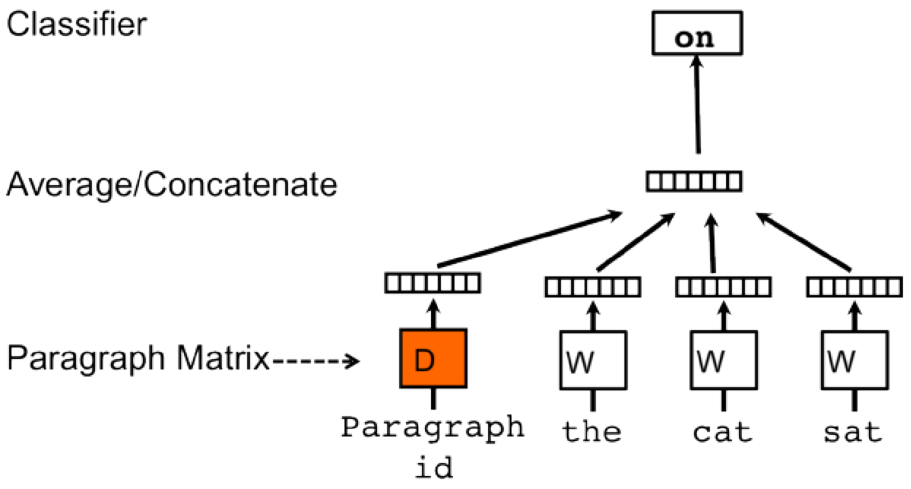


Figure 2.4 A framework for learning paragraph vector. This framework is similar to the framework presented in Figure 2.3; the only change is the additional paragraph token that is mapped to a vector via matrix D.
Source: [63]

In the PV-DBOW model, instead of using a sliding window, words in the context window are randomly sampled from paragraph texts. Then the model is trained to predict words randomly picked from the paragraph, which is analog to the Bag of Words version of Word2vec. According to Le *et al.* [63], paragraph vectors are shared for all contexts generated from the same paragraph, but not across paragraphs. In contrast, word vectors are shared across all paragraphs.

2.4.2 Convolutional Neural Networks

A Convolutional Neural Network (CNN) resembles a connectivity pattern between neurons in the human brain. A CNN is typically composed of an input layer, multiple hidden layers and an output layer. The hidden layers typically include convolutional layers, pooling layers, and fully-connected layers. A nonlinear function is calculated inside each neuron of every layer. The loss function is used in evaluating the performance of a classification model. Some optimization methods are employed to minimize the loss function. Dropout is a popular technique to prevent the network from overfitting. Figure 2.5 illustrates a typical CNN architecture for a toy image classification task.

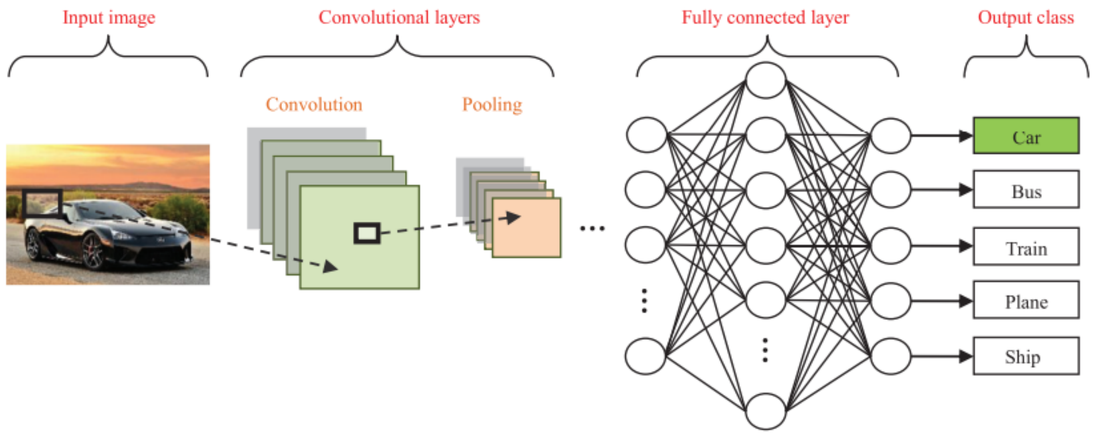


Figure 2.5 CNN image classification pipeline.
Source: [68]

Convolutional layer. The convolution operation is used between two consecutive layers in a CNN. The input to the operation is a set of feature maps, which can be the numeric representation of the raw input data or convolved feature maps from the previous convolutional layer. The element that carries out the convolution operation on the feature maps is called filter (or kernel). Filters share the same size but are initialized with different weight so that different features can be extracted at each location [64, 67]. A filter's size represents its receptive field, which is connected to a neighborhood of neurons in the previous layer via a set of trainable weights [64]. A filter traverses a fixed length defined by the *stride* parameter, till the entire input is scanned. As the filter slides over the input, it convolves the input values by multiplying the filter's weights with its receptive field. The convolved results are sent through a nonlinear activation function to compute the new feature map. Stacking the feature maps for all filters along the depth dimension forms the full output of the convolutional layer.

Figure 2.6 demonstrates a concrete example of the convolution operation. The input feature map is of size width=5, height=5, and depth=3, as shown in blue. The output is of size width=3, height=3, and depth=2 (same as the number of filters), as shown in green. The parameters for this convolutional layer are Filters=2, Stride=2, and Padding=1. W0 and W1 are the two filters, each with size width=3, height=3, and depth=3, as shown in red.

Stride=2 means that a filter will slide two features (i.e., move by two positions) each time. The input feature map is padded with zeroes at both ends. By setting Padding=1, the width of the padding is one feature at each end, in both dimensions. In Figure 2.6 (a), filter W0 (red) performed the convolution by *elementwise* multiplying its weights with the receptive field of the input, highlighted in blue. Thus, 9 grey/blue values are multiplied with the corresponding 9 red values and then summed up. This happens independently for every depth value. For example, for the first step, 0 * -1 + 0 * 0 + 0 * 0 + 0 * 1 + 1 * -1 + 1 * 0 + 0 * 0 + 0 * -1 + 1 * 1 = -1 + 1 = 0.

As the depth=3, the results of summing three elementwise multiplications were 0 (as shown in the previous paragraph), 1, and -3, respectively. By summing these three numbers and offsetting by adding the Bias b0=1, the corresponding first unit (first row, first column) in the output feature map is -1 (highlighted in green). Filter W0 then slides over by two features, as shown in Figure 2.6 (b), to repeat the convolution. By summing the elementwise multiplications, the results were -2, 3, and -3. After offsetting by b0=1, the obtained the second unit (first row, second column) is -1 (highlighted in green). This process was repeated until Filter W0 sild over the entire input, so that the first layer of the new feature map was fully generated (the top green matrix) in the output. Similarly, sliding

Filter W1 over the input generated the second layer of the new feature map (the bottom green matrix) in the output.

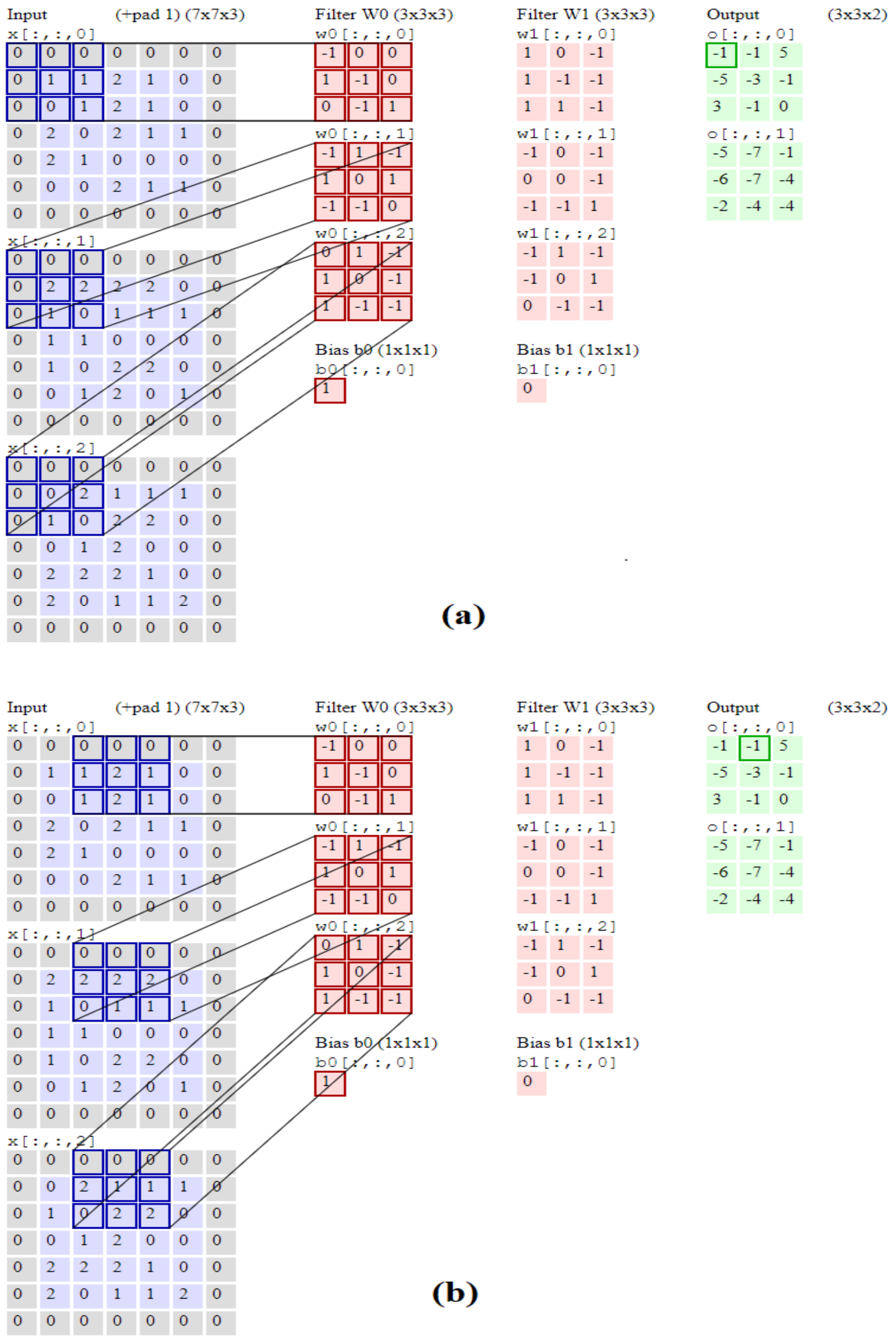


Figure 2.6 Convolution operation on a 5x5x3 input with two 3x3x3 filters with Padding=1 and Stride=2. (a) Filter W0 performs element wise multiplications with the input receptive fields. The results are summed and offset with Bias b0 to get -1 as the first unit of output. (b) Filter W0 slides 2 features and perform the convolution to get -1 as the second unit of the output.
Source: [69]

Pooling layer. The pooling layer is used to pick out the salient values from a local patch of units. It can reduce the spatial resolution of the feature maps and thus achieve spatial invariance to input distortions and translations [67, 70-72]. Common pooling techniques including max pooling, average pooling and L2-norm pooling. The max pooling operation is most used, which selects the max value in a local patch of units and then the local patch shifts a step with the stride size. Figure 2.7 illustrates the difference between max pooling and average pooling.

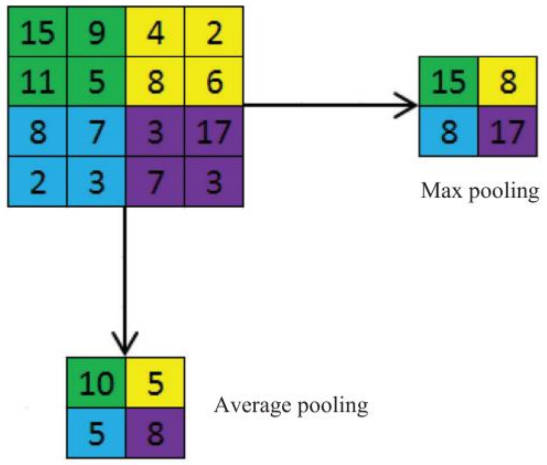


Figure 2.7 Average versus max pooling.
Source: [68]

Fully-connected layer. The fully-connected layer connects all units in the previous layer to all units in the next layer. In a fully-connected layer, the number of units in the next layer is required to be set as a hyperparameter (a parameter whose value is set before the learning process begins). Every entry in the output volume can be interpreted as an output of a neuron that looks at a small region in the input and shares parameters with neurons in the same filter bank. The fully-connected layers interpret these feature representations and perform the function of high-level reasoning [73-75]. For classification problems, it is standard to use the softmax operator on top of a CNN [65, 74-78]. The softmax function is used to squash the output from fully-connected layer into normalized positive values (each value is between zero and one) that sum to one, which can be interpreted as the (normalized) probability assigned to all classes.

Nonlinear function. In a neural network, the common ways to model a neuron's output vector **f** as a function of its input vector **z** are with tanh, sigmoid, or Rectified Linear Unit (ReLU) [79]. Figure 2.8 shows examples of sigmoid and ReLU functions. ReLU outperforms the other two with its simple form (f(x) = max(0, **z**)) for its fast convergence of stochastic gradient descent.

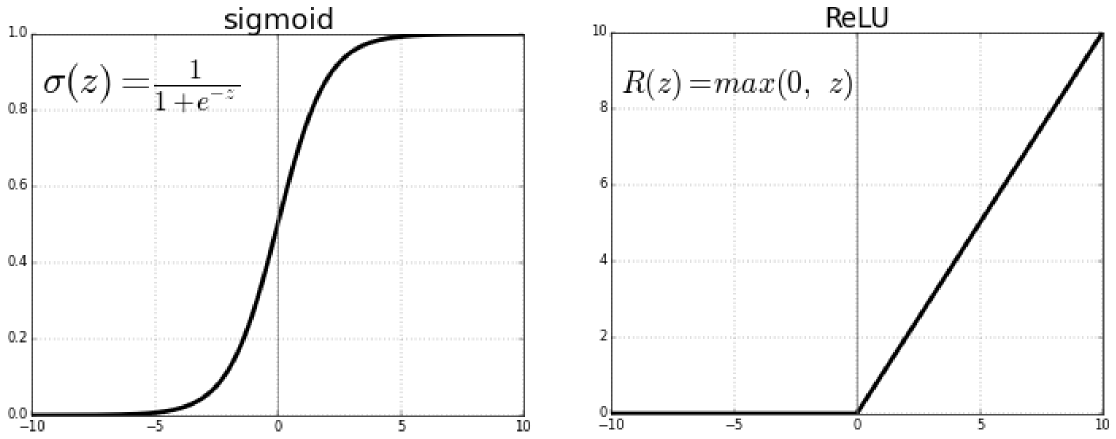


Figure 2.8 Logistic Sigmoid and ReLU functions.

Loss function. A loss function is used to evaluate how labels derived by the current classification model deviate from the corresponding true labels. The loss function is usually composed of two parts, the variance between the estimated labels and true labels, and the regularization. Regularization is usually used to prevent the model from fitting the noise of the training data by adding a penalty on the different parameters of the model. It helps to reduce the freedom of the model to improve the generalization abilities of the model. The loss L is shown in the Formula 2-1.

$$L = V + \lambda \cdot R \qquad (2\text{-}1)$$

where V denotes the variance and R is the regularization value. λ is the weight decay determining how much the regularization affects the final loss. Two widely used regularization techniques are Lasso Regression (L1) and Ridge Regression (L2) [80].

Dropout. Dropout is frequently used to reduce overfitting. At each training stage, individual neurons are either "dropped out" of the network with probability 1-p or kept with probability p. The incoming and outgoing edges to a dropped-out neuron are “turned off.” Only the reduced network is trained with the data in that stage. In the next training iteration, the dropped-out neurons are then “turned on” again with their original weights.

CNN utilizes convolving filters to automatically learn and extract local features from various layers, regardless of the input size. This makes CNN a very powerful tool for classification or prediction tasks, e.g., text classification [81] and relation extraction [82], even if the data or features have not been manually labeled for learning purposes.

2.4.3 BERT

For the language representation model, there are two main research streams: *Context-free* and *Contextual* representations. Traditional word embeddings such as *word2vec* [83], *GloVe* [84], or *fastText* [85], are *Context-free* embeddings, which generate a single “word embedding” representation for each token in the vocabulary. Therefore, they are not likely to capture any word meaning changes caused by surrounding context changes. *Contextual* models, instead, generate a representation of each word that is based on the other words in the context. *Contextual* representations can further be categorized into *unidirectional* or *bidirectional*.

BERT is the first *unsupervised, deeply bidirectional* system that outperforms previous methods [86]. BERT’s model architecture is a multi-layer *bidirectional* *transformer encoder,* based on the original implementation proposed by Vaswani et al. [87]. Figure 2.9 shows the overall structure of BERT with a sequence of words (e.g., sentences with multiple words) as input. Each word in the input is transformed to a word/token vector, adding a position vector (e.g., a word's index in a sentence) and a segment vector (e.g., index of the sentence that a word is in). The input vectors obtained this way are embedded with representations of words and the relationships between words. The input vectors then go through a network of transformers. As opposed to directional models, in which transformers are connected sequentially (either left-to-right or right-to-left), BERT's transformers are fully-connected (non-directional) so that it can read the entire sequence of words at once. This characteristic allows BERT to learn the context of a word including all of its surroundings (all other words in the sentence, even across sentences, based on allowed input length).

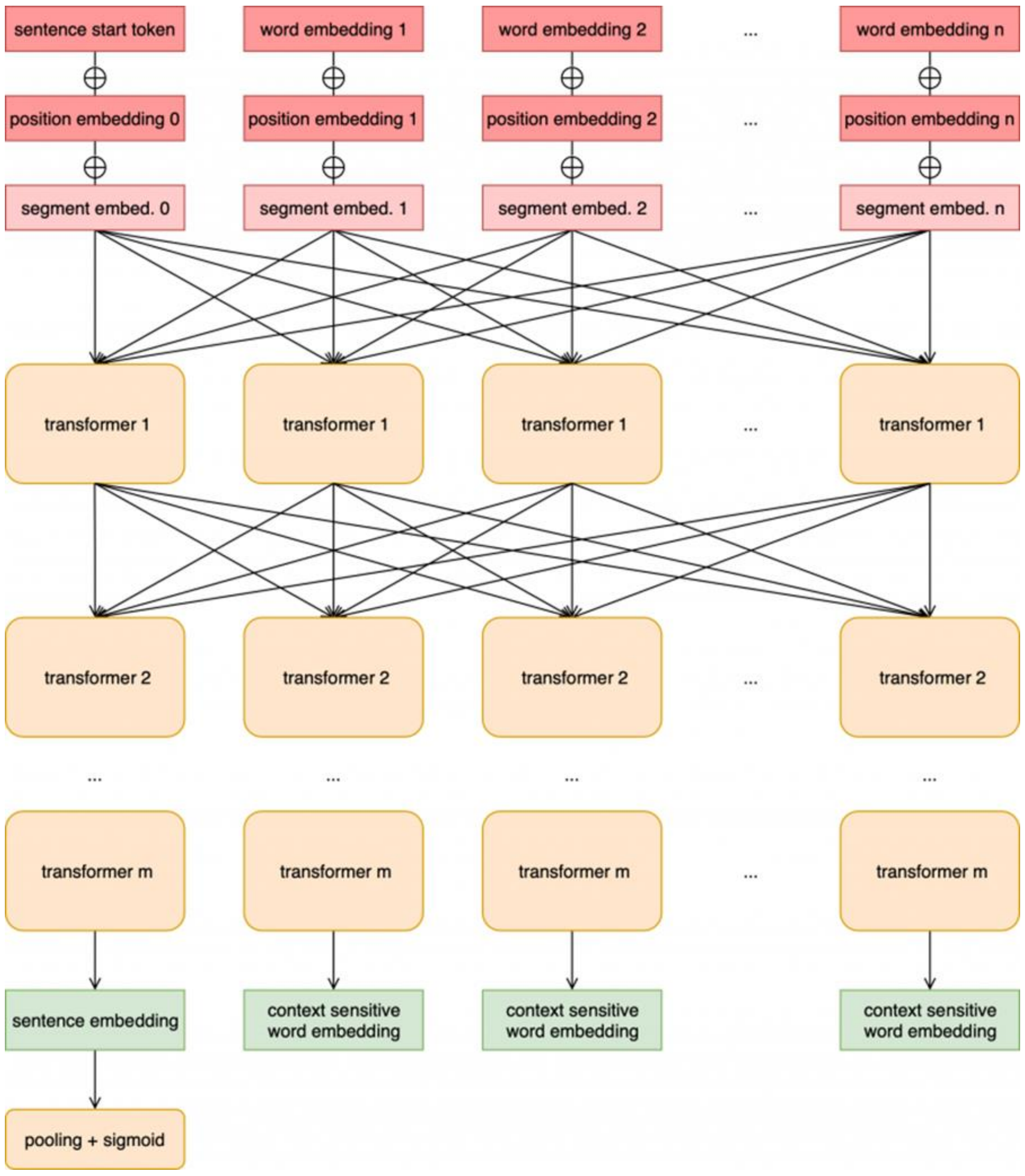


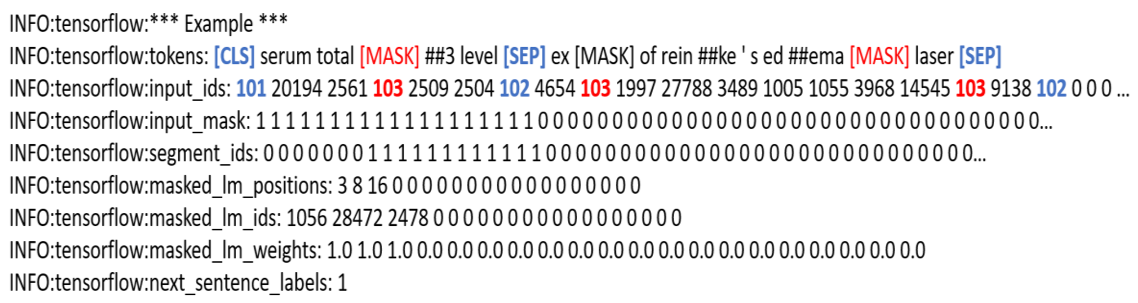
Figure 2.9 Overall structure of BERT, enabling context sensitive word representations. *Source: [88]*

An example of BERT's input representation of two sentences of "Serum total T3 level" and "Excision of Reinke's edema using laser" is shown in Figure 2.10. These two sentences are tokenized (following initial basic tokenization — converting all tokens to lower case, punctuation split) to "[CLS] serum total [MASK] ##3 level [SEP] ex [MASK] of rein ##ke ' s ed ##ema [MASK] laser [SEP]".

This sequence of tokens is further mapped to three components that are token embeddings, segment embeddings and position embeddings. Token embeddings are general word embeddings, which use vectors to represent tokens (or words). For example, the embedding index for token "serum" is 20194.

Segment embeddings are used when there are multiple sentences. For instance, if the input includes two sentences A and B, then tokens from sentence A will be assigned E_A as its corresponding sentence embeddings while tokens from sentence B will be assigned E_B. If the input only includes one sentence, only one sentence embeddings will be used. For instance, the segment id for "serum" is 0 while the segment id for "laser" is 1.

Position embeddings represent the token sequence of the input. The position for tokens will be accumulated regardless the number of sentences. For example, the position for masked token "T" is 3. "CLS" (embedding index 101) is the reserved token to represent the start of sequence while "SEP" (embedding index 102) separate segment (or sentence). The embedding index for [MASK] is 103. Note that token embeddings are randomly initialized into a vector with a length of 768 by default if a BERT model is trained from scratch. More detail of these three component are explained in [87].



```
INFO:tensorflow:*** Example ***
INFO:tensorflow:tokens: [CLS] serum total [MASK] ##3 level [SEP] ex [MASK] of rein ##ke ' s ed ##ema [MASK] laser [SEP]
INFO:tensorflow:input_ids: 101 20194 2561 103 2509 2504 102 4654 103 1997 27788 3489 1005 1055 3968 14545 103 9138 102 0 0 0 ...
INFO:tensorflow:input_mask: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...
INFO:tensorflow:segment_ids: 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0...
INFO:tensorflow:masked_lm_positions: 3 8 16 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:masked_lm_ids: 1056 28472 2478 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
INFO:tensorflow:masked_lm_weights: 1.0 1.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
INFO:tensorflow:next_sentence_labels: 1
```

Figure 2.10 An example of BERT input representation for two concepts of "Serum total T3 level" and "Excision of Reinke's edema using laser."

A transformer is an element for transforming one sequence into another one using an *Encoder* module and a *Decoder* module [89] (Figure 2.11). The *Encoder* module is on the left and connected to the *Decoder* module, which is on the right. Both Encoder module and Decoder module consist of multiple identical encoders and decoders stacked on top of each other, which is denoted by "Nx" in Figure 2.11. The number of encoder and decoder units is a hyperparameter that can be changed. An encoder consists of two components: *Multi-Head Attention* and *Feed Forward Neural Network*.

The input sequences are first embedded into an n-dimensional vector space, then the embedded input flows through the encoder's self-attention layer. In this process, the encoder encodes each specific unit (e.g., a word or token) while considering other units in the input sequence.

Multi-Head Attention is concatenated from multiple self-attention heads (explained later). If a transformer uses eight self-attention heads, it contains eight sets for each encoder/decoder. The *Feed Forward Neural Network* consists of two linear transformations (multiply with a weight and add a bias) with a ReLU activation in between. While the linear transformations are the same across different positions, the weights and biases are different from layer to layer.

The decoder has the same two components as the encoder, with an extra *Masked Multi-Head Attention* component. Masking is used to hide some words from the input sequence so that the model is trained to predict the masked words, using non-masked words. This helps the decoder focus on the parts of the input sequence that are most relevant to the masked words.

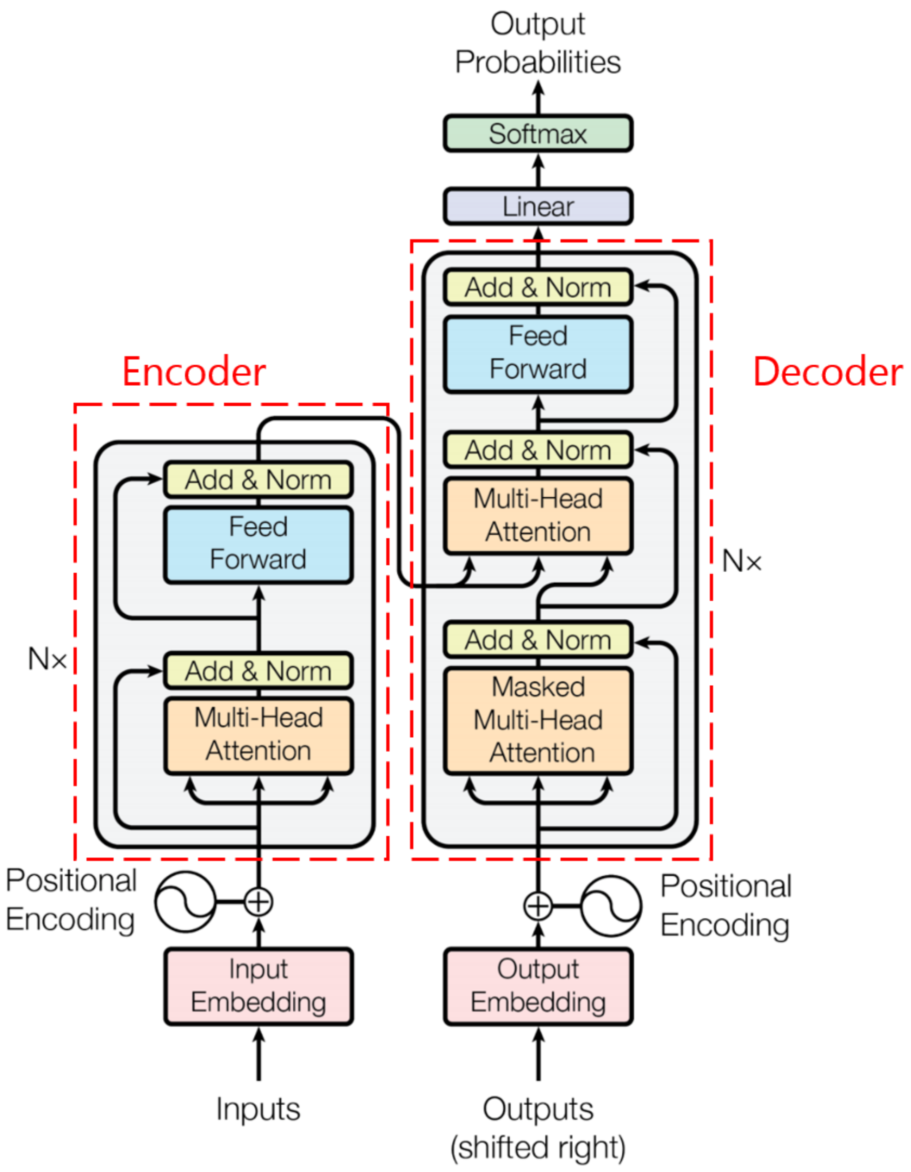


Figure 2.11 The Transformer – model architecture.
Source: [87]

Each of the transformers is implemented with an attention mechanism, that is, to obtain a word's transformed representation in the output, how much attention it should pay to each word (including itself) in the input. A single self-attention module is demonstrated in Figure 2.12 with an input of "The dog ran away." For each of the word vectors, a (*key*, *query*, *value*) triple representation is computed using simple matrix multiplication with Query/Key/Value weight matrices. Each of these matrices is randomly initialized. After training, they are used to calculate the attention for each input embedding (or vectors from lower encoders/decoders). For example, the amount of attention that the word-to-be-transformed "ran" should pay to each word (including itself), is computed by the scaled dot product of the *query* vector (q_3) of this word with the *key* vectors (k_i, where $i = 1 \ldots 4$) of the other words.

The query vector q_3 for "ran" is multiplied with k_1, k_2, k_3, and k_4 to get 64, 96, 112, and 80, respectively. Then these numbers are scaled by dividing them by 8, which is the square root of the matrix dimension of the key vectors (64, as used in [87]), to get 8, 12, 14, and 10. A softmax function is applied to these scaled numbers to obtain weights (0.002, 0.117, 0.865, and 0.016, which add to 1.0) for each corresponding word. The resultant weights determine how much each word will be expressed at this position. It is clear that the word "ran" at this position will have the highest softmax weight 0.865, and it is also useful to attend to any other word that might be relevant to the current word, such as the subject word "dog" with the weight 0.117. The output vector of this word "ran" is based on the weighted sum of *value* vectors (v_i, where $i = 1 \ldots 4$) of all words, where the weights are the softmax-normalized dot products (S_i, where $i = 1 \ldots 4$) of the S and v vectors. (See the dashed area in Figure 2.12).

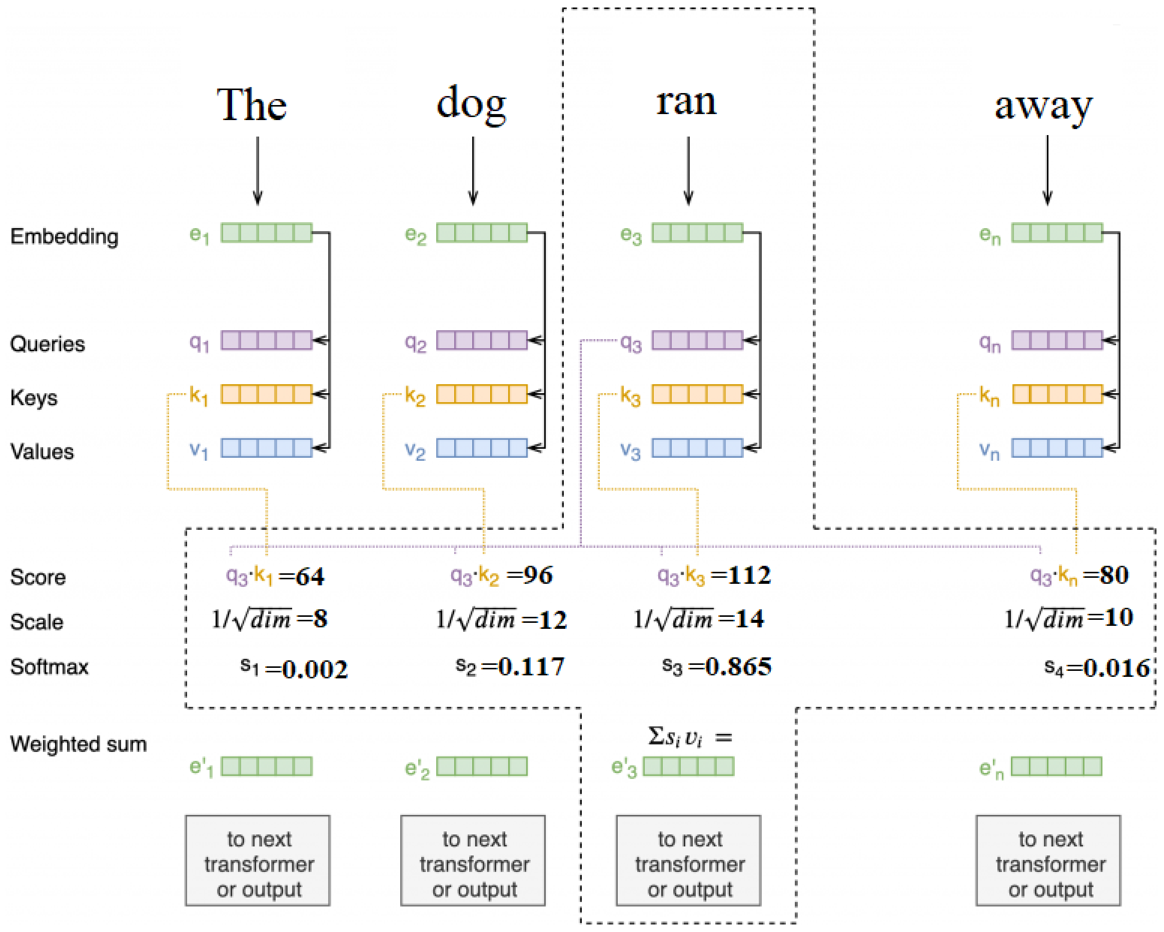


Figure 2.12 A single attention module.
Source: [88]

A single attention module (also called one "head") is largely limited to a single type of "attention" between words, such as one type of semantic or grammatical relationship. This is because in a single attention module, only one softmax layer is evaluated, which means attention is paid predominantly to one individual word at a time. To overcome this limitation, the transformer used by BERT consists of multiple attention heads. Figure 2.13 shows how multiple single Scaled Dot-Product Attention modules (Figure 2.13 (a)) can be concatenated in parallel to form a Multi-Head Attention module (Figure 2.13 (b)).

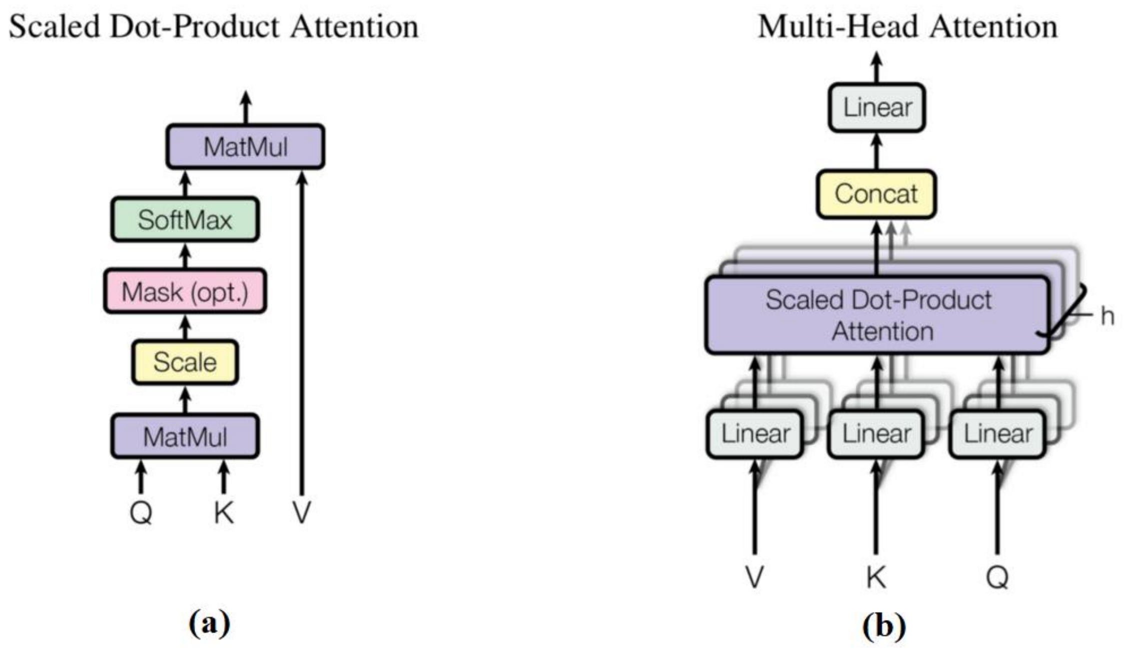


Figure 2.13 (a) Scaled Dot-Product Attention. (b) Multi-Head Attention consists of several attention layers running in parallel.
Source: [87]

Figure 2.14 shows a graphical representation of an encoder's internal architecture of a transformer [88]. For a multi-head attention module, the output is obtained by concatenating the output of each head. Concatenating vectors (the same length as the input) from multiple heads would quickly grow the length of an output vector to an infeasible size. Hence, only a fraction of the width of the original vectors from each of the attention heads is extracted, and then these "fractional" vectors are concatenated to achieve the original length (the same as the length of the input vector). *Addition* and *Normalization* operations (see Figure 2.14) are performed on the concatenated output before it is fed through a single-layer *Feed Forward Neural Network*. The *Addition* operation calculates the element-wise sum of two vectors. The *Normalization* used in the Transformer is *Layer Normalization*, which calibrates ("normalizes") a vector based on the mean and the variance of the summed vectors within the same layer [90]. The *Addition* and *Normalization* operations are used to improve the training convergence.

BERT is a general-purpose "language understanding" model trained on a large text corpus (like Wikipedia), which can be used for various downstream NLP tasks without heavy task-specific engineering. The BERT model is pre-trained with two tasks Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). The training objective of the MLM task is to predict the masked words of a text sequence. The training objective of NSP is to classify whether two sentences are consecutive for any given sentence-pair. This generic training produces a model that represents the meaning of a large variety of text. Thus, the model pre-trained with these two tasks can be easily adapted to other types of NLP tasks. BERT has advanced the state-of-the-art for several major NLP benchmarks, including named entity recognition on CoNLL-2003 [91], question answering on SQuAD [92], and sentiment analysis on SST-2 [93].

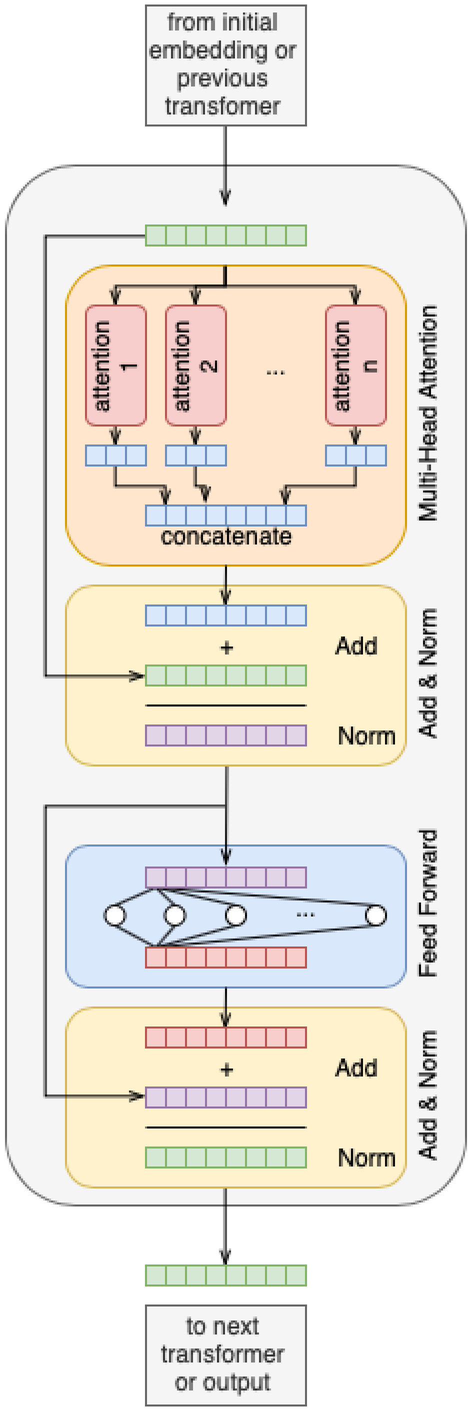


Figure 2.14 A full transformer module.
Source: [88]

2.4.4 Embeddings

The research on *Embeddings* (text vectorization) has advanced from context-free to contextual embeddings. Context-free embeddings, such as *word2vec* [83], *GloVe* [84], or *fastText* [85], generate a single "word embedding" for each word in the vocabulary. Several methods were proposed to rectify this limitation [94, 95]. Subsequent research extended the word level embeddings to paragraphs or documents, e.g., Doc2vec [63], Sentence Encoder [96], and Sent2Vec [97].

Contextual representations generate vectors for words or tokens by “considering” other text in their surroundings. These include various Deep Learning (DL) techniques, such as Semi-supervised Sequence Learning [97], Generative Pre-Training [98], ELMo [99], and ULMFit [100], which mainly rely on LSTM [101] or BiLSTM [102] models, thus they are *unidirectional* or *shallowly bidirectional*, meaning that each word is only contextualized using the words to its left (or right). With the advent of Attention and Transformers [87], research has shifted to training general language representation models rather than competing on improving task-specific embeddings. Bert [86], GPT-2 [103], and XLNet [104] are transformer-based language models, which are deeply bidirectional. These general-purpose language models are usually trained with a large text corpus without binding to specific downstream tasks.

In bioinformatics, different textual resources have been utilized to train word/document embeddings, such as Word2vec trained on PubMed and PMC data [105] (http://bio.nlplab.org/), and Word2vec trained on 27 million articles from the MEDLINE/PubMed Baseline 2018 (http://nlp.cs.aueb.gr/). Cui2vec [106] has been trained with an insurance claims database of 60 million members, a collection of 20 million clinical notes, and 1.7 million full text biomedical journal articles. Wang *et al.* evaluated the word embeddings trained with four different kinds of textual resources including news [107]. Onto2vec [108] automatically learns feature vectors for proteins from their annotations in

Gene Ontology. These embeddings have been commonly leveraged as feature input to downstream ML models.

2.5 Semantic Similarity

In this dissertation, Pirró's [109] definition of *Semantic Similarity* is adopted, which considers only the subsumption relationship between two concepts (i.e., 'IS-A'). Another concept is *Semantic Relatedness,* which considers a broader range of relationships (e.g., 'part-of' [110, 111]). There exist various measurement methods for both semantic similarity and relatedness, because of the lack of consistent definitions of those two across different publications. The Lesk measure, proposed by Banerjee et al.[112], increases linearly as the overlap between two terms increases. Patwardhan generates a first-order co-occurrence matrix for each word based on extended definitions of biological terms and then builds a second-order co-occurrence matrix as a gloss vector [113]. The cosine similarities calculated between gloss vector pairs are used as the semantic relatedness score between terms. Pesaranghader used Pointwise Mutual Information (PMI) as an adjunct feature to cut-off co-occurrence data, improving the gloss vector-based relatedness measures [114]. Taieb *et al.* exploited WordNet features, such as the noun and the verb taxonomy and the shared words (overlaps) in glosses as a multi-strategy measure of semantic similarity [115].

CHAPTER 3

MACHINE LEARNING FOR ENRICHMENT OF SNOMED

The goal of using ML for enrichment of ontologies is to utilize it to identify IS-A relationships. Consider a missing IS-A relationship from an existing concept A to a concept B. If the concept B is not in the ontology and is added together with adding an IS-A link from concept A to it, then this is knowledge enrichment. For knowledge enrichment, a concept is recognized as missing in the ontology and then inserted into the proper place. The aforementioned problems are similar to classification/prediction of relationships between two concepts.

This chapter presents a study of integration of embeddings and a neural network model to support ontology enrichment. The previously discussed vectorization techniques, i.e., Doc2vec, are employed to transform each concept into its vector representation. Then a CNN model is trained to perform classification or prediction tasks on the relationships between concepts.

3.1 Concept Vector Representation Using Doc2vec

As noted, ontology concepts in string representations cannot be directly processed by a numeric ML technique such as CNN or SVM (Support Vector Machine), for classification or regression tasks. Thus, concepts are converted into numeric vector representations while maintaining as much hierarchical and semantic information as possible. Note that this information is critical, because similar concepts should be close to each other given their vector representations. It is necessary to maintain this kind of association between concepts and their meaning in the vector representation.

Concepts are primarily represented as text strings in the form of English word(s), sometimes combined with ID numbers. This is exactly what Word2vec and Doc2vec can deal with. Thus, it is necessary to construct text in a way that Word2vec can learn about single words or word sequences, and Doc2vec can learn about documents on a high level. To employ this technique, the following steps were performed:

i. Create one "document" per focus concept (Figure 3.1). The ID for a document is the corresponding SNOMED CT concept ID. The content of this document consists of the concepts that are hierarchically related to this focus concept: the focus concept's parents (targets of IS-A links from this concept), its strict siblings (that share exactly the same parents as this concept), and its children (sources of IS-A links to this concept). This construction is based on the idea that a concept is the topic of a document and that the closely related concepts are descriptions of the meaning of this concept in the ontology hierarchy.

ii. Arrange the order of these related concepts such that the underlying hierarchical relationships are maintained. In order to do that, a concept's parent(s) should be at the beginning of the text, followed by the concept itself. Then, a concept's siblings are put before its children. So, the order of the whole document text is: Parent(s) – Focus concept – Sibling(s) – Child(ren) (See Figure 3.1). No separators between concept groups are needed, because of the design of the ML method applied below. For example, "(Parents) *finding of abnormal level of heavy metals in blood*, *finding of trace element level* – (Focus concept) *blood copper abnormal* – (Siblings) *serum iron level abnormal*, *zinc in blood specimen outside reference range*, *cobalt in blood specimen outside reference range* – (Children) *raised blood copper level*, *serum copper level abnormal"* defines the content of the document for the concept *blood copper abnormal.*

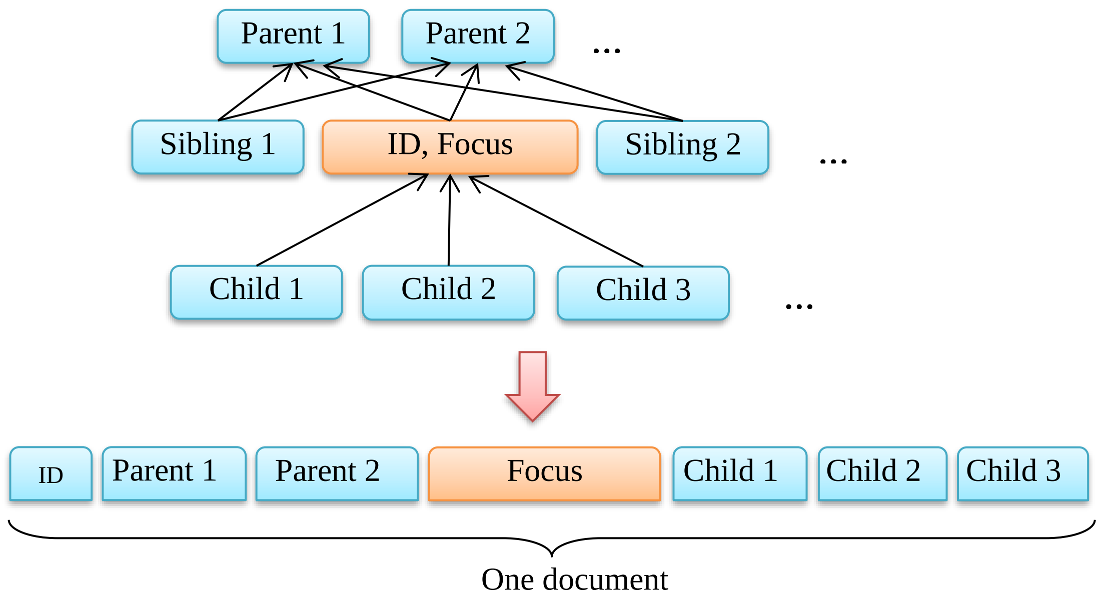


Figure 3.1 Serializing the hierarchical structure of one concept into one document.

iii. After constructing a document for each concept, a list of documents is obtained and fed to a Gensim [116] Doc2vec system, which automatically extracts semantic topics from the documents, with high efficiency. After the training of this Gensim's Doc2vec model (Figure 3.2) is finished, a "weighted matrix under the hood" is constructed. In order to get the corresponding vector for each concept, a concept's "one-hot vector" (a positional coding of a concept in a long vector of many 0s and a single 1) is multiplied with this weighted matrix. Intuitively, one can view this Doc2vec model as a lookup table where all trained concepts in their vector forms with unique IDs are stored. The vector length is chosen by the user in the training phase. To better fit with the CNN model used in a later step, 512 features per vector were chosen. (The input dimensionality is preferably a power of 2.) In addition, the Doc2vec model can infer a vector for a concept that was not in the training data, by looking at this concept's name. Metaphorically, Doc2vec "guesses" a vector by looking for the concept name (words) in its "vocabulary." The inaccuracy introduced by deriving a vector can propagate to the CNN classifier.

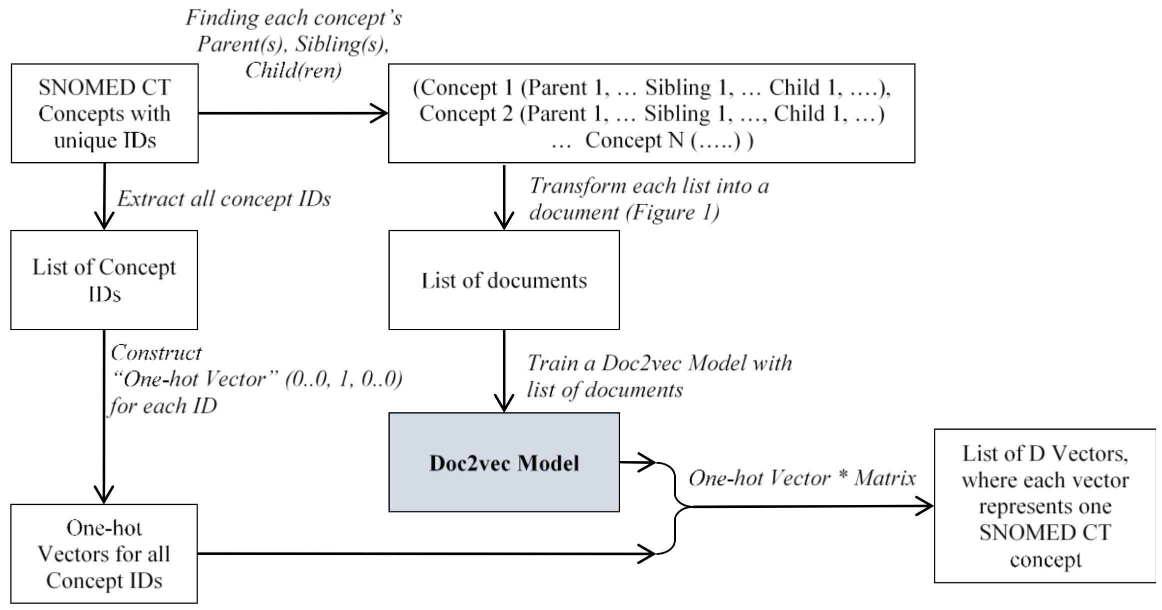

Finding each concept's Parent(s), Sibling(s), Child(ren)
SNOMED CT Concepts with unique IDs
(Concept 1 (Parent 1, … Sibling 1, … Child 1, ….), Concept 2 (Parent 1, … Sibling 1, …, Child 1, …) … Concept N (…..))
Extract all concept IDs
Transform each list into a document (Figure 1)
List of Concept IDs
List of documents
Construct "One-hot Vector" (0..0, 1, 0..0) for each ID
Train a Doc2vec Model with list of documents
Doc2vec Model
One-hot Vector * Matrix
List of D Vectors, where each vector represents one SNOMED CT concept
One-hot Vectors for all Concept IDs

Figure 3.2 Data flow during vectorization phase. D = 336,893 is the total number of concepts. The vector length is 512.

3.2 CNN Model as an IS-A Relationship Classifier

In this dissertation, a CNN model was built to deal with the concept classification problem. The design of the CNN model was inspired by a model used to detect transportation modes from smartphone data by Liang *et al*. [117]. The input to both models is a set of one-dimensional feature vectors. This model differs from Liang *et al*.'s work in both input vector length and the number of convolutional layers, as well as in the intricacy of the application domain.

The CNN was designed for one-dimensional data. Figure 3.3 demonstrates the CNN architecture used in this study. The architecture consists of a succession of convolutional layers, max pooling, and fully-connected layers. Specifically, the input is a 1024 × 1 vector (created by concatenating two 512 × 1 vectors). The first convolutional layer puts 32 filters on a 15-feature window with a stride of one feature. The following max pooling layer is set with a 4-feature window and a stride of two features. The convolutional layer and max pooling layer are repeated N times. In the proposed network, N is equal to 7. The six hidden layers are shown in Figure 3.4. All the max pooling layers are set to the same parameters. The second and third convolutional layers have 64 filters on a 10-feature window with a stride of one feature. The other four convolutional layers filter the data with 64 kernels on a 5-feature window with a stride of one feature. Conducting the convolution and max pooling processes N times, the data become 8×64. The data are fed into a fully-connected layer with a hyperparameter of 200 to become 200×1. Finally, the data are transferred into a 2×1 output vector.

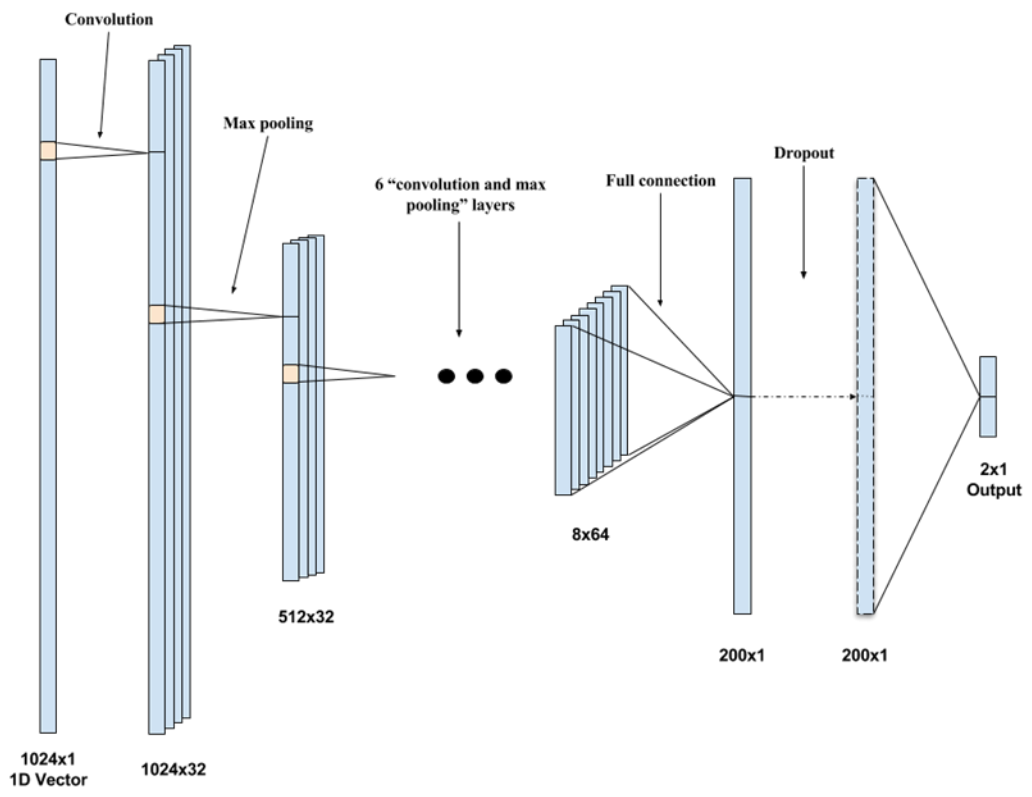


Figure 3.3 CNN architecture in this system. It is composed of a succession of convolutional, max pooling and fully-connected layers.

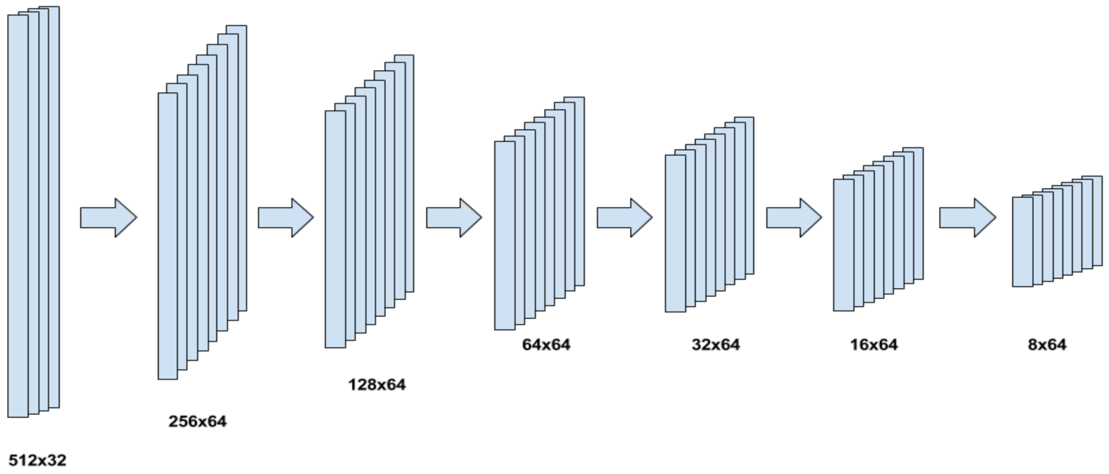


Figure 3.4 The six hidden layers.

The elements of the final 2 × 1 output vector represent the probability for Class 0 and Class 1, respectively. Class 0 means there should *not* be an IS-A link connecting two given concepts, and class 1 means there should be an IS-A link. This probability of Class 1 can be used as the score of how confident the model is about the existence of an IS-A link between two concepts.

The CNN model was implemented with the Tensorflow [118] framework for training and testing. The implemented neural network contained over 100,000 parameters. The nonlinear activation function attached to every convolutional layer and the fully-connected layer is the Leaky ReLU function [78]. The Adaptive Moment Estimation (Adam) is employed to optimize the loss function for its fast convergence and adaptive learning rate [119].

3.2.1 Training and Testing

Figure 3.5 summarizes the data flow during the CNN training phase. The training samples for the CNN model are a set of IS-A linked concept pairs and a second set of non-IS-A linked pairs.

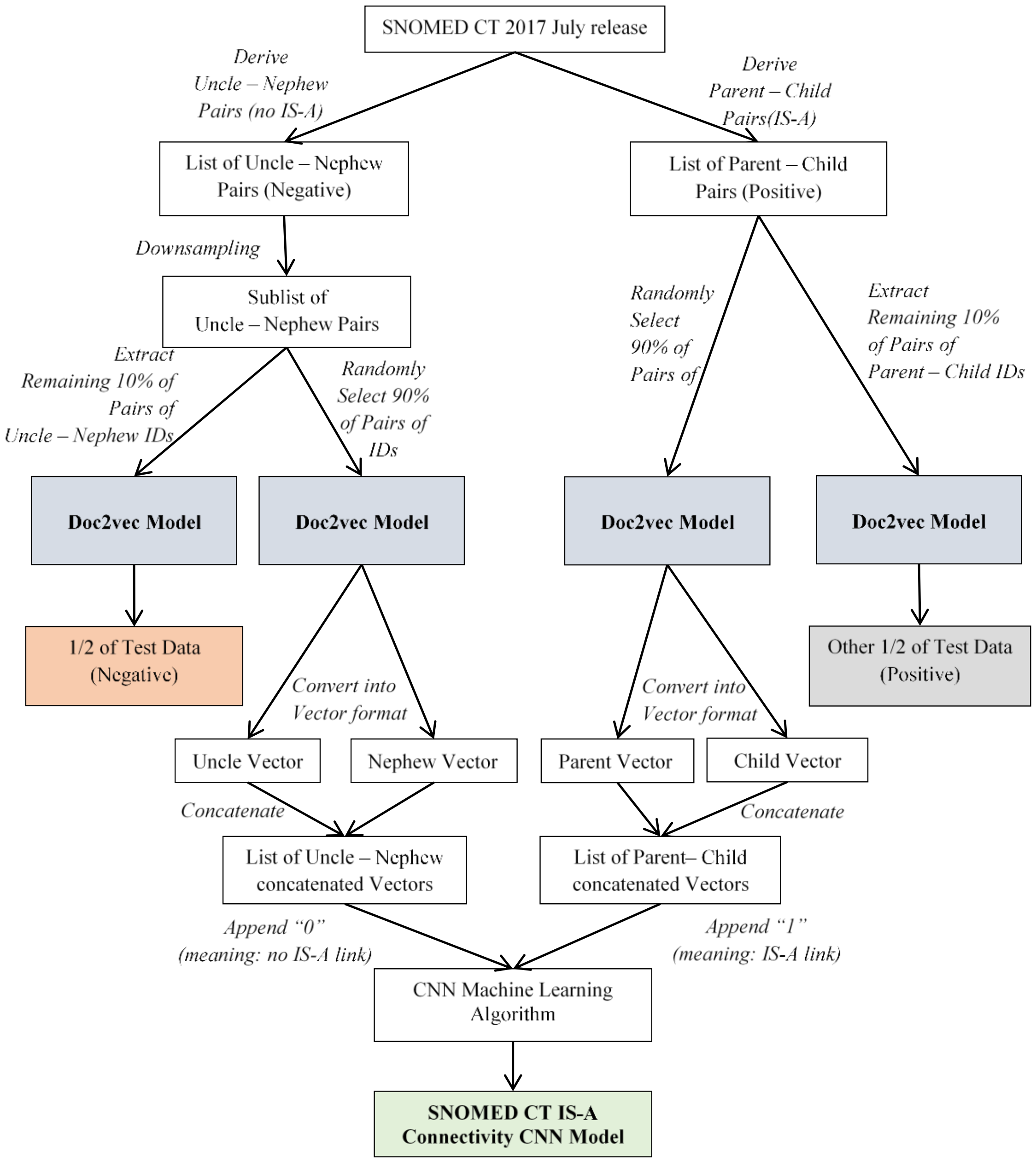

SNOMED CT 2017 July release
Derive Uncle – Nephew Pairs (no IS-A)
Derive Parent – Child Pairs(IS-A)
List of Uncle – Nephew Pairs (Negative)
List of Parent – Child Pairs (Positive)
Downsampling
Sublist of Uncle – Nephew Pairs
Extract Remaining 10% of Pairs of Uncle – Nephew IDs
Randomly Select 90% of Pairs of IDs
Randomly Select 90% of Pairs of
Extract Remaining 10% of Pairs of Parent – Child IDs
Doc2vec Model
Doc2vec Model
Doc2vec Model
Doc2vec Model
1/2 of Test Data (Negative)
Other 1/2 of Test Data (Positive)
Convert into Vector format
Convert into Vector format
Uncle Vector
Nephew Vector
Parent Vector
Child Vector
Concatenate
Concatenate
List of Uncle – Nephew concatenated Vectors
List of Parent– Child concatenated Vectors
Append "0" (meaning: no IS-A link)
Append "1" (meaning: IS-A link)
CNN Machine Learning Algorithm
SNOMED CT IS-A Connectivity CNN Model

Figure 3.5 Data flow during training the CNN model. Test data is generated "on the fly."

In SNOMED CT, all pairs of concepts that are directly connected by an IS-A link are known. Thus, the IS-A linked pairs are given, so they can be viewed as positive sample. On the other hand, it is both impossible and counterproductive from a machine learning point of view to create all pairs of concepts that are *not connected* as training data. (These would be labeled 0 = “No.”) It is impossible (with the limited computational resources available), because, for example, in the SNOMED CT 2017 January release, there are 336,893 active concepts. The total pairs of concepts are then 56,748,278,278. From these the 502,459 existing IS-A links need to be subtracted, giving a total of 56,747,775,819 pairs of concepts that are not connected. Training a model with such an unbalanced dataset would create a biased and meaningless results (the model would attempt to make all predictions negative and could still achieve high prediction accuracy). Thus, a strategy was developed to reduce the negative sample size, making it both computationally manageable and balanced with the positive sample.

Concept pairs such that one concept in the pair is the strict sibling of the other concept’s parent (“uncle”) were picked. In total, 502,459 IS-A linked pairs (positive sample instances) and 6,167,243 non-IS-A linked pairs (negative sample instances) were selected. Even this positive vs. negative sample ratio is about 1:12, but this could be handled with common techniques such as oversampling or downsampling.

First, the negative pairs were randomly downsampled to make the negative sample’s size the same as the positive sample’s size. This was done at the beginning of each training round. This resulted in a training data set with 1,004,918 pairs (502,459 pairs from both positive and negative groups). Secondly, the training dataset was shuffled and then 90% of it was used as the training set and 10% was kept as the test set. Thirdly, both concepts in a pair of concepts (i.e., in a positive or a negative sample instance) were transformed into vectors by “querying” the Doc2vec model that was already trained before. Two vectors, either a child-parent vector pair as a positive sample instance, or a nephew-uncle vector pair as a negative sample instance, were concatenated into one vector (1 × 1024) before sending it into the CNN model for feature learning.

The CNN model compares its predictions with the truth labels (0 or 1) to perform self-correction. The errors are back-propagated to the model, so it updates the weights in each layer in order to minimize the cross-entropy loss.

Since it is not possible to read all 1,004,918 training sample instances into the computer’s memory at the same time, this training dataset was divided into “batches” with 8,000 instances per batch. In other words, in one iteration the CNN model was trained with 8,000 instances of one batch, and then the system moved to the next batch in the next iteration. The classification accuracy of every 1000 batches was tracked to check on the training gains in the process. Note that each batch could be used for training multiple times, because after all the batches were iterated, the system restarted with the first batch and continued to the next. This process was repeated until a pre-defined accuracy threshold was reached. The training was repeated for 20 rounds to deal with the imbalance issue introduced by downsampling the non-IS-A linked pairs.

After training the CNN model, the model was accessed with the remaining 10% data in the test set (Figure 3.6). The concatenated vectors from both the positive and the negative group were sent to the trained CNN model. The model output is a stream of 0s and 1s. A “0” for a pair of concepts indicates that the model concludes there should not be an IS-A link, and *vice versa* for a “1.” For a negative test instance, the label 0 is correct and for a positive test instance the label 1 is correct. These correctness values were sent to a scoring function, where Precision and Recall were calculated to get the F1 score as the performance/quality measure of the trained CNN model. Precision, Recall and F1 score were calculated with the scikit-learn [120] built-in toolkit.

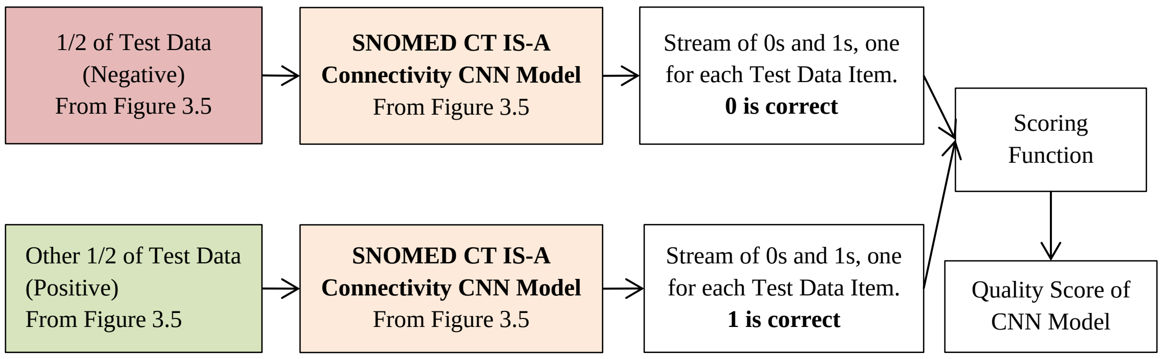


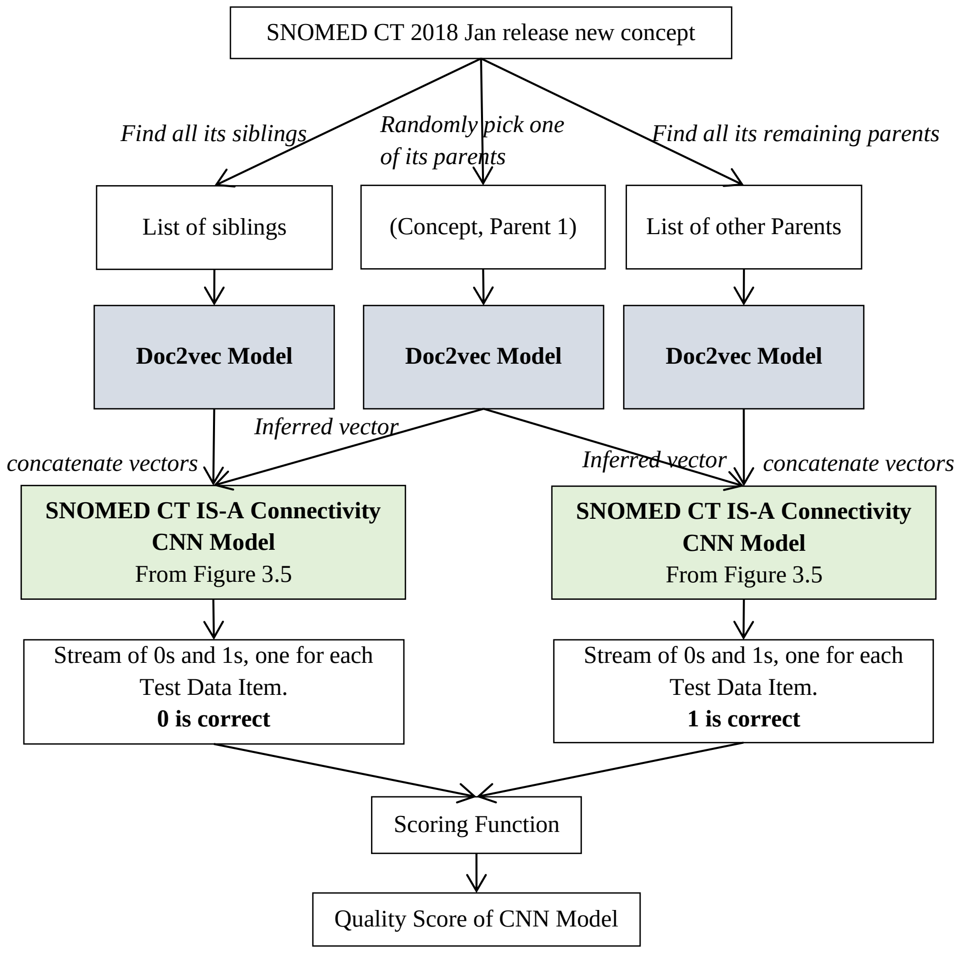
Figure 3.6 Data flow during testing CNN mode phase.

3.2.2 Prediction of IS-As for New Concepts

To evaluate the model's performance on real, previously unseen data, a test task using new concepts from the SNOMED CT 2018 January release (Figure 3.7) was created to confirm the efficacy of the methodology. For each new concept that was added to this release, it and its strict siblings were extracted pair-wise as negative sample instances, and it and its parents pair-wise as positive sample instances. Then they were converted into vectors, before they were fed into the CNN model. If a concept existed in the SNOMED CT 2017 July release, its corresponding vector was queried directly from the pre-trained Doc2vec model, otherwise the model had to infer a vector for it. Since a new (2018) concept was not used for training with the Doc2vec model (trained with data from SNOMED CT's 2017 July release), the model can only infer a vector by using this concept's name as data. However, if all the word(s) in a new concept's name are not in the trained vocabulary, the inferred vector for this concept would be totally meaningless.

To avoid this situation, testing the CNN model was limited to new concepts with multiple parents. To get the inferred vector for a new concept, one of its parents was randomly selected and this parent’s name and its name were concatenated as the “query” to the Doc2vec model. This cannot guarantee the avoidance of a completely random vector, but the accuracy improvement in the experiments showed that it is a workable solution. As this selected parent must be excluded from the testing sample, the positive test sample set is then composed of all other remaining parent(s).

The pair-wise vectors, either extracted or inferred from the Doc2vec model, were concatenated into single vectors before sending them to the trained CNN model. The CNN model processed each input vector, using the weights that it had already learned before, computing a class label (0 or 1) and returned it as prediction result. Both positive and negative samples were fed into the model in random order, and the output was a stream of 0s and 1s. For negative sample instances, the label 0 is correct, indicating the there is no IS-A link in the new SNOMED CT release for these two concepts. Label 1 is correct for positive sample instances, indicating the existence of an IS-A link. The model’s predicted result labels were compared with the true labels to calculate prediction accuracy in terms of F1 score.



SNOMED CT 2018 Jan release new concept
Find all its siblings
Randomly pick one of its parents
Find all its remaining parents
List of siblings
(Concept, Parent 1)
List of other Parents
Doc2vec Model
Doc2vec Model
Doc2vec Model
Inferred vector
concatenate vectors
Inferred vector
concatenate vectors
SNOMED CT IS-A Connectivity CNN Model From Figure 3.5
SNOMED CT IS-A Connectivity CNN Model From Figure 3.5
Stream of 0s and 1s, one for each Test Data Item. 0 is correct
Stream of 0s and 1s, one for each Test Data Item. 1 is correct
Scoring Function
Quality Score of CNN Model

Figure 3.7 Data flow during testing CNN model phase with new data (use case).

3.3 Performance Comparison between CNN and SVM Models

The prediction results of the CNN model with the remaining 10% of the sample (See "Test Data" in Figure 3.5) extracted from the SNOMED CT 2017 July release are as follows. The average Precision, Recall, and F1 score among ten tests are presented in Table 3.1. In each test, the CNN is tested against 1024 sample instances (mixed positives and negatives). For example, in Test 4, the average Precision is 0.88, Recall is 0.77 and the F1 score is 0.82. The results show that the proposed CNN can achieve an accuracy of approximately 80% with reasonable fluctuations.

Table 3.1 Precision Score and F1 Score for Ten Tests

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Average**
Precision	0.90	0.82	0.91	0.88	0.89	0.88	0.90	0.89	0.87	0.91	0.885
Recall	0.84	0.65	0.75	0.77	0.71	0.61	0.75	0.62	0.76	0.76	0.722
F1 Score	0.87	0.73	0.82	0.82	0.79	0.72	0.82	0.73	0.81	0.83	0.793

It is also interesting to compare these results with traditional ML methods. Specifically, an SVM stochastic gradient descent (SGD) classifier was tested against the same test data. The classification results under different classification models are shown in Table 3.2. The table shows that in general, the proposed CNN model outperforms SVM. In fact, CNN (0.794) outperforms the traditional method SVM (0.705) by 8.8% when comparing the F1 scores.

Table 3.2 F1 Score Comparison between CNN and SVM Models Among Ten Tests

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Average**
CNN	0.87	0.73	0.82	0.82	0.79	0.72	0.82	0.73	0.81	0.83	0.793
SVM	0.61	0.68	0.72	0.67	0.75	0.74	0.75	0.73	0.72	0.68	0.705

Regarding the prediction of IS-A links for new concepts, an example is shown in Table 3.3 of the CNN model's prediction results for the new concept *Bone island of left pelvis*, which was added to SNOMED CT in the 2018 January release. For each test, *Bone island of left pelvis* was paired with one of its parents (*Disorder of body wall, Disorder of pelvic girdle, Disorder of pelvis, Finding of bone of pelvis, Hypertrophy of bone*) to get its inferred vector, then the CNN model predicted IS-A links between it and the other parents and also to the sibling concept *Bone island of right pelvis*.

For instance, when *Disorder of pelvis* was chosen as the first parent and paired with *Bone island of left pelvis,* then the task became to predict IS-A links between the pairs (*Bone island of left pelvis, Disorder of body wall*), (*Bone island of left pelvis, Disorder of* *pelvic girdle*), (*Bone island of left pelvis, Finding of bone of pelvis*), and (*Bone island of left pelvis, Hypertrophy of bone*). No IS-A link should be found for the sibling pair (*Bone island of left pelvis, Bone island of right pelvis*). The CNN model correctly predicted the IS-A links between the first four child-parent pairs but was wrong about the Sibling pair.

Table 3.3 CNN Prediction Result for *Bone island of left pelvis*

Bone island of left pelvis	**Parent**	**Parent**	**Parent**	**Parent**	**Parent**	**Sibling**
Predicting / Paired with	*Disorder of body wall*	*Disorder of pelvic girdle*	*Disorder of pelvis*	*Finding of bone of pelvis*	*Hypertrophy of bone*	*Bone island of right pelvis*
Disorder of body wall	■	1	1	1	1	1
Disorder of pelvic girdle	1	■	1	0	1	0
Disorder of pelvis	1	1	■	1	1	1
Finding of bone of pelvis	1	1	1	■	1	0
Hypertrophy of bone	1	1	0	1	■	0

NOTE: Grey background indicates a correct prediction.

Table 3.4 shows the Precision, Recall and F1 score of the CNN model testing against all new concepts with multiple parents in the SNOMED CT 2018 January release. On average, this model achieved around 70% accuracy out of a total of 21,945 predictions using the content of SNOMED CT (January 2018) as the gold standard.

Table 3.4 Precision, Recall, and F1 Score for New Tests

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Average**
Precision	0.73	0.71	0.69	0.80	0.71	0.75	0.82	0.81	0.73	0.72	0.747
Recall	0.69	0.67	0.63	0.65	0.60	0.67	0.69	0.73	0.62	0.64	0.661
F1 Score	0.71	0.69	0.66	0.72	0.65	0.71	0.75	0.77	0.67	0.68	0.701

The results were compared with the previously employed SVM model, testing it on the new concept data. The classification results are shown in Table 3.5. In fact, CNN (0.701) still outperformed the SVM (0.643) model by 5.8% in regard to the F1 score.

Table 3.5 Average F1 Score Comparison between CNN and SVM Models among New Concept Tests

Index	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Average**
CNN	0.71	0.69	0.66	0.72	0.65	0.71	0.75	0.77	0.67	0.68	0.701
SVM	0.64	0.63	0.62	0.69	0.61	0.61	0.68	0.62	0.70	0.63	0.643

The presented CNN model achieved good results when testing against pre-trained concept vectors. This supports two claims: first, the method used to construct documents for concepts preserves enough semantic and hierarchical information in the context of the whole ontology, even after the complex encoding with Doc2vec; secondly, the trained CNN model was able to predict IS-A links with scores in line with many other machine learning techniques. This suggests that the pretrained vectors maintained semantic and hierarchical features that can be utilized for various classification tasks with modern machine learning techniques.

Meanwhile, it was found that the quality and accuracy of a vector representation of a concept, in a consistent way among all concepts, is critical to the downstream classification and prediction models. For a new concept that has a name that contains no words in the trained vocabulary, the inferred vector from the Doc2vec model carries very limited or no useful information. Also, the inferred vector is not always consistent, because the inference is an incremental optimization process with random initialization.

Automatically predicting the placement of new concepts in an ontology would be both of practical importance by helping ontology curators and would stretch the state-of-the-art of ML methods in the direction of semantic tasks. A limitation of the presented work is that *one parent* had to be given for a new concept, and the system predicted the other parents. Ideally, all parents should be predicted. While there are some difficulties in principle, the main obstacle is one of computing power. Operating with 57 billion pairs (the negative training set) is not within the range of current computing resources. This work has concentrated on predicting IS-A links. Future work will extend the paradigm to attribute relationships (semantic relationships) and to other ontologies besides SNOMED CT.

CHAPTER 4

ENRICHMENT OF SNOMED USING MACHINE LEARNING WITH ABSTRACTION NETWORKS

In Chapter 3, the negative sample for training was based on nephew-uncle pairs. The assumption was that the uncles, while not being parents, are similar to parents in their content, providing a realistic negative training set for ontology enrichment. However, the concept may have many uncles, some of which are more similar to the parent(s) than others. If the first group of the more similar concepts could be distinguished from the second group of the less similar concepts, then the first group could serve to provide a more accurate negative sample for the training.

The area taxonomy, a kind of Abstraction Network (Section 2.2), can potentially help to identify the set of uncles relatively more similar to the parent(s). The uncle concepts that are within the same area as the parent(s) are all more similar to the parent(s) than uncle concepts in other areas, since they have the exact same sets of relationships as the parent(s).

This chapter presents the effectiveness of using Abstraction Networks to enhance the model introduced in Chapter 3 to predict IS-A relationships. It is a summarization approach for the automatic identification of IS-A relationships for new concepts. This approach is based on the area taxonomy. SNOMED CT's *Clinical Finding* hierarchy was the test-bed for evaluating the effectiveness of this approach.

4.1 Concept Vector Representation from Two Models

The SNOMED CT July 2017 release was used as training set and the subsequent January 2018 release as testbed. The data preprocessing follows the same workflow as in the previous study (Chapter 3 and [121]) to generate documents and train embedding vectors, but is limited to the *Clinical Finding* hierarchy of SNOMED CT. The document generation process is illustrated below, as it is crucial for understanding the next step.

i. Create one "document" per focus concept (see Figure 4.1). A concept's corresponding SNOMED CT ID was used as the unique ID for a document. The hierarchical information related to this focus concept is used as the content of this document. Specifically, this concept's parents (targets of IS-A links from this concept), and its children (sources of IS-A links to this concept) were chosen. Unlike in the previous study, a concept's sibling(s) are neglected in this study without showing accuracy loss. By constructing a document this way, a concept becomes the focus topic of this document and its closely related concepts "describe" the meaning of it in the ontology hierarchy.

ii. To maintain the existing hierarchical relationships of the focus concept to some degree, these concepts were arranged as follows: A concept's parent(s) are placed at the beginning of the text, followed by the concept itself, and then by the child(ren) of this concept. Thus, the order of the whole document text is: Parent(s) – Focus concept — Child(ren) (Figure 4.1). Tabs are used as separators between concept groups. For example, *Infectious thyroiditis* has two parents, *Thyroid infection* and *Thyroiditis*, and two children, *Acute suppurative thyroiditis* and *Viral thyroiditis*. Thus, the content of the document for the concept *Infectious thyroiditis* is defined as "*Thyroid infection, Thyroiditis* \t *Infectious thyroiditis* \t *Acute suppurative thyroiditis, Viral thyroiditis*" where \t is the ASCII tab character, and the document ID is 3511005, which is the SNOMED ID of the focus concept *Infectious thyroiditis* .

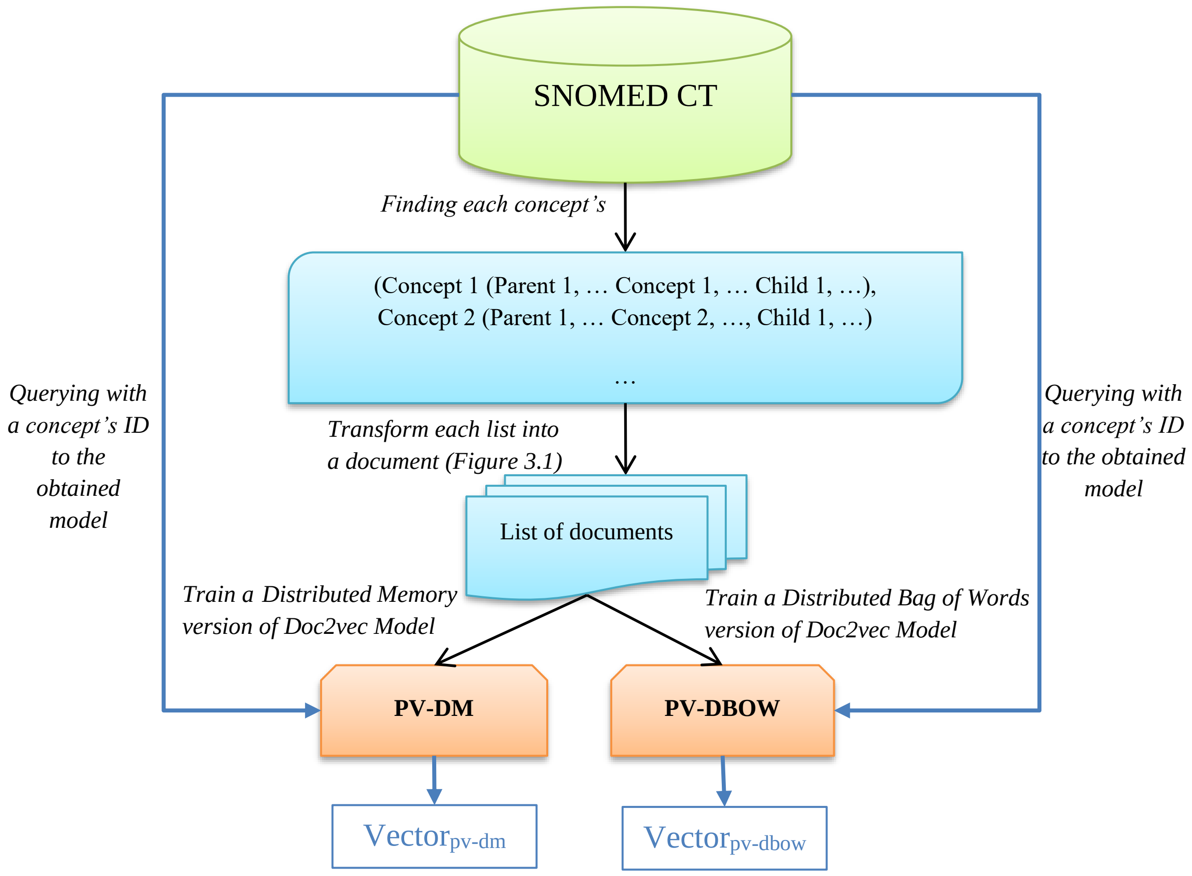

SNOMED CT
Finding each concept's
(Concept 1 (Parent 1, … Concept 1, … Child 1, …),
Concept 2 (Parent 1, … Concept 2, …, Child 1, …)
…
Transform each list into a document (Figure 3.1)
List of documents
Train a Distributed Memory version of Doc2vec Model
Train a Distributed Bag of Words version of Doc2vec Model
Querying with a concept's ID to the obtained model
Querying with a concept's ID to the obtained model
PV-DM
PV-DBOW
Vector pv-dm
Vector pv-dbow

Figure 4.1 Data flow during vectorization phase. N = 109,366 is the total number of concepts in the *Clinical Finding* hierarchy of the SNOMED CT July 2017 release.

After constructing a document for each concept, the list of all documents was passed to the Gensim implementation of Doc2Vec to generate two embeddings (displayed as black arrows in Figure 4.1). Gensim automatically generates the document-level vectors, while maintaining the semantic topics of the documents. Experiments with both the Distributed Memory version (PV-DM) and the Distributed Bag of Words (PV-DBOW) version of Paragraph Vector [63] were performed. The PV-DM model and the PV-DBOW model function as two "vector dictionaries." For each concept, its corresponding two vectors (denoted as $\text{Vector}_{\text{pv-dm}}$ and $\text{Vector}_{\text{pv-dbow}}$ in Figure 4.1) were retrieved by sending its corresponding document ID to the two models, respectively. This is indicated by two blue arrows emanating from the green SNOMED CT repository. The vector dimension was set to 128. The use of those two embeddings is new and was not done in Chapter 3 [121].

4.2 Training Data from Hierarchy and Area Taxonomy

To train a CNN model that can verify an IS-A link between a pair of concepts, two models were trained with both IS-A connected concept pairs and concept pairs with no IS-A connection. The IS-A connected concept pairs are explicitly defined in the terminology hierarchy. The judicious selection of non-IS-A concept pairs is critical for the accuracy of the model. In Chapter 3 [121], the non-IS-A pairs were limited to only nephew-uncle pairs. Figure 4.2 (a) illustrates this. Consider an IS-A relationship from a concept A to a concept B. A sibling C of B is an uncle of A. Thus, the pair of A and C is called a nephew-uncle pair. The concept D has an IS-A relationship to concept Q, and Q is **not** an uncle of A. The concept C is hierarchically related to B as its sibling. Thus, C has a higher probability of being mistaken as A's parent than D, which is not hierarchically related to B. In other words, the non-IS-A pair (A, C) is more similar to the IS-A pair (A, B) than the non-IS-A pair (A, D) is to (A, B). It is more useful to learn to distinguish between an IS-A pair and (one of) its nephew-uncle non-IS-A pairs than between an IS-A pair and an arbitrary non-IS-A pair, which might be “far away” from the IS-A pair.

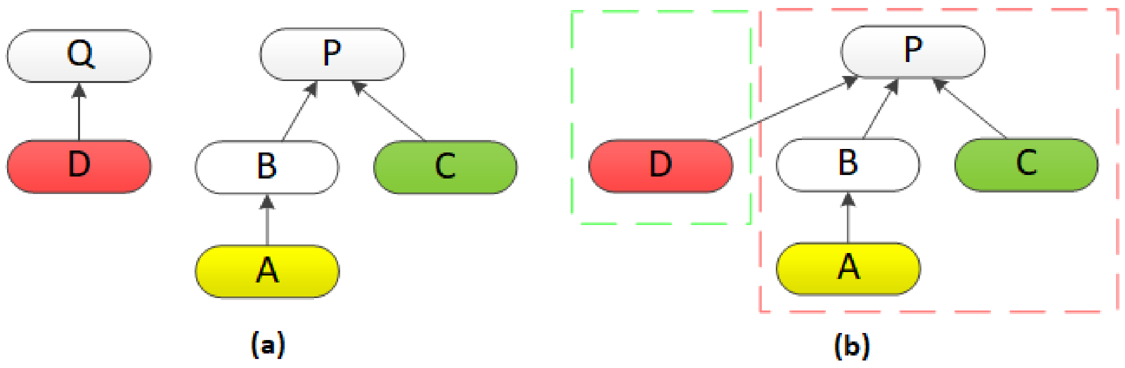


Figure 4.2 Illustrating two potential training data patterns.

Utilizing the Area Taxonomy, the nephew-uncle pairs can be further divided into two types: uncle and nephew concepts are from the same area versus uncle and nephew concepts are from different areas. In Figure 4.2 (b), both C and D are A's uncles, because they are siblings of B. However, C is in the same (red) area as A, while D resides in another (green) area. The nephew-uncle pair (A, C) is more similar to the IS-A pair (A, B) than the nephew-uncle pair (A, D), since concepts from the same area share the same set of relationships types, and are more similar to each other.

The hypothesis is that the CNN model will become more accurate by training it with nephew-uncle pairs from the same area, because it can learn to distinguish between IS-A pairs and closely related non-IS-A pairs. As a result, the derived CNN model should be better at verifying whether a concept pair should be connected by an IS-A link or not, achieving better accuracy.

The above observation can be demonstrated with a concrete example (Figure 4.3) from the *Clinical Finding* hierarchy. Let the nephew concept be *Conjunctival diphtheria* (in yellow). Its uncle concepts are *Viral conjunctivitis*, *Parasitic conjunctivitis,* and *Tumorlet*. *Conjunctivitis* (also called pinkeye) is an infection of the conjunctiva, the transparent tissue that covers the white part of the eye. These three concepts are the siblings of the concept *Bacterial conjunctivitis*, which is the parent of *Conjunctival diphtheria*. As Figure 4.3 shows, the uncle concept *Viral conjunctivitis* (in green) is in the same area {*Associated morphology*, *Causative agent*, *Finding site*, *Pathological process*} as *Conjunctival diphtheria*, while the other two uncle concepts *Parasitic conjunctivitis* and *Tumorlet* (in red) are in different areas {*Causative agent*, *Finding site*, *Pathological process*} and {*Associated morphology*, *Finding site*, *Pathological process*}, respectively, than *Conjunctival diphtheria*. *Conjunctival diphtheria* is structurally more similar to *Viral conjunctivitis* as both of them have the same four lateral relationships, while it is structurally different from *Parasitic conjunctivitis* and *Tumorlet*, which have only three lateral relationships. Indeed, *Viral conjunctivitis* is closer to *Bacterial conjunctivitis*, the parent of *Conjunctival diphtheria*, than to *Parasitic conjunctivitis* and *Tumorlet*.

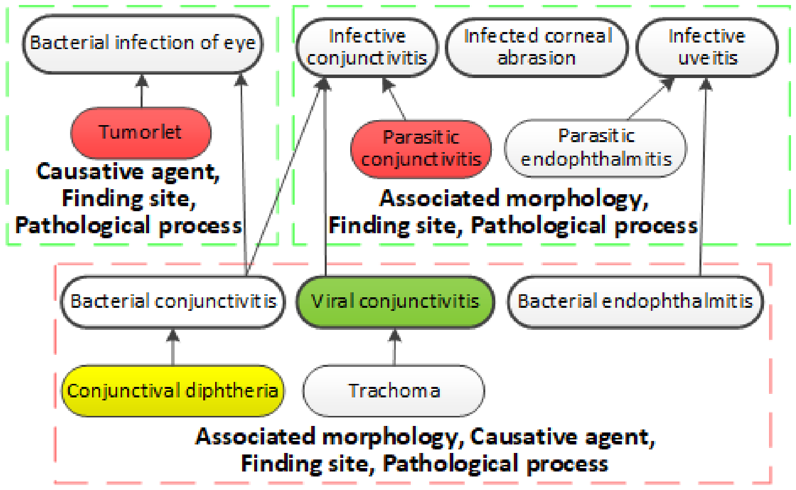

Bacterial infection of eye
Tumorlet
Causative agent, Finding site, Pathological process
Infective conjunctivitis
Infected corneal abrasion
Infective uveitis
Parasitic conjunctivitis
Parasitic endophthalmitis
Associated morphology, Finding site, Pathological process
Bacterial conjunctivitis
Viral conjunctivitis
Bacterial endophthalmitis
Conjunctival diphtheria
Trachoma
Associated morphology, Causative agent, Finding site, Pathological process

Figure 4.3 Nephew-uncle pairs within/without the same area.

4.3 Training and Testing two CNN Models as IS-A Relationship Classifiers

Two CNN models were trained with two sets of training data from the *Clinical Finding* hierarchy: one set with nephew-uncle non-IS-A pairs, with both the uncle and newphew concepts taken from the same area; the other set with uncle and nephew randomly chosen from the hierarchy. The two models shared the same CNN architecture as in Section 3.2. The data was tailored to accommodate two vectors with a dimension of 128. The modification of stacking two vectors in depth for each concept as input does not require changing the previous CNN model, because the convolutional operations work across the depth dimension, such as, for example, when processing RGB channel data for color image input [64].

To evaluate the model's performance on real, previously unseen data, new concepts from the SNOMED CT 2018 January release were used. A pre-trained Doc2Vec model returned a pre-determined vector for an existing concept or inferred a vector for a new concept from the new concept's name. However, if all the word(s) in a new concept's name were not in the trained vocabulary (i.e., they were never seen before when training the model), the inferred vector for this concept would be a random vector. To avoid this situation, the testing was limited to new concepts with multiple parents existing in the July 2017 release. This limitation restricts the model's applicability to about one-third of the concepts in SNOMED CT. For new concepts with only one existing parent, it is better to use them for training the model for the following release, rather than testing against them without having any knowledge about them in the model. The detailed procedures for preparing test data for both IS-A concept pairs and non-IS-A concept pairs are as follows (Figure 4.4).

New concepts that were added in the SNOMED CT January 2018 release and their parents were used. To get the inferred vector for a new concept N, one of its parents was randomly selected – say Parent 1 in Figure 4.4(a). Then Parent 1's name was concatenated with N's name as the "query" to the Doc2Vec model. The vector inferred by the Doc2Vec model was then paired with one of the remaining parents' vectors, say the vector that represents Parent 2, to form a vector pair (N, Parent 2). This vector pair, as one test case, was sent to the pre-trained CNN model for IS-A link verification. Similarly, the inferred vector for N was paired with vectors for Parent 3 and Parent 4, respectively to create two more test cases, namely, whether the CNN model can verify the IS-A links for the pairs (N, Parent 3) and (N, Parent 4).

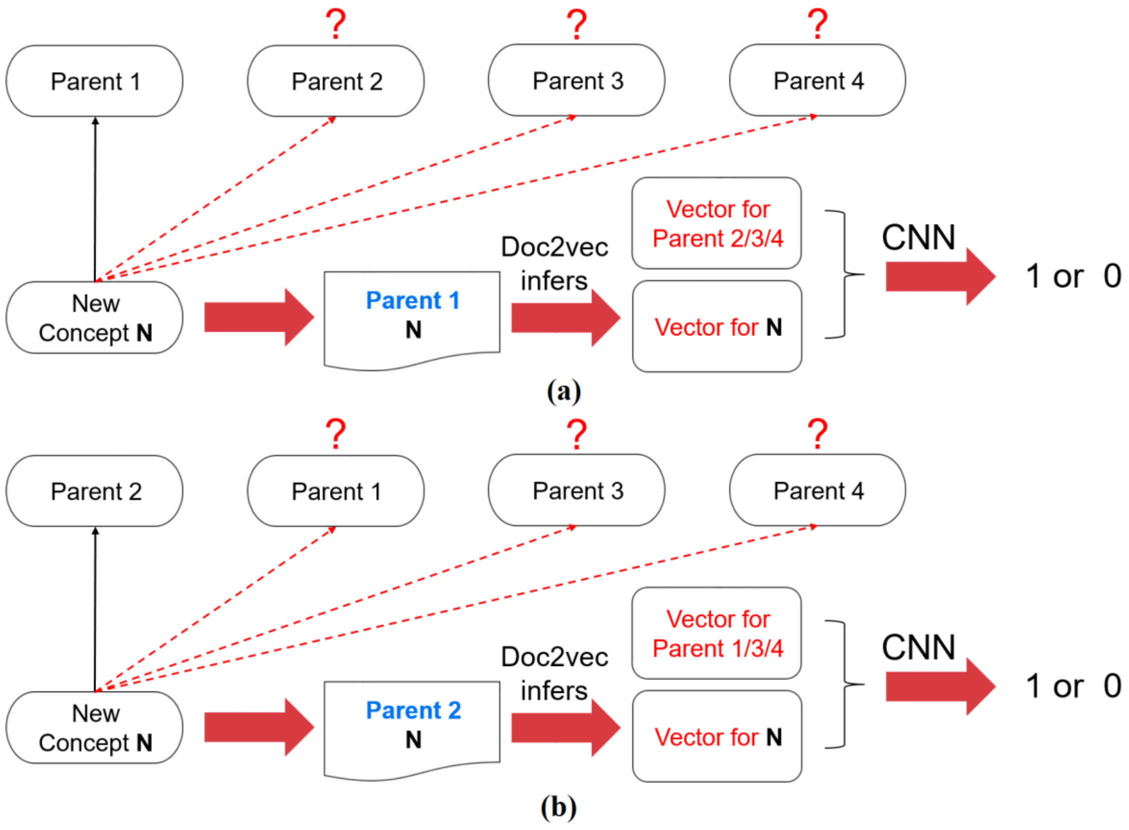


Figure 4.4 (a) Three IS-A test cases if pairing up the new concept N with its first parent. (b) Repetition of (a) by pairing up N with Parent 2.

As the Doc2Vec model is pre-trained, i.e., it is not updated during the inference process, the inferred vectors are independent of each other. Therefore, other sets of test cases were created by generating all the possible pairs between N and its parents. For example, as shown in Figure 4.4(b), the names of N and Parent 2 were concatenated to get the inferred vector for N. Then we paired the inferred vector with vectors for Parent 1, Parent 3, and Parent 4 as three additional test cases. This can be continued by assuming for every parent (Parent 3, Parent 4) that it is known, while all the remaining parents are tested.

Following this setup, out of 2,005 new concepts added to the *Clinical Finding* hierarchy of the January 2018 release, 1,027 concepts were selected to generate a total of 7,494 IS-A concept pairs for testing the CNN model's accuracy when verifying IS-A links for the new SNOMED CT release. Out of the 1027 concepts, 797 concepts were leaf nodes.

For non-IS-A test pairs, the existing parents of **all** the new concepts that were used for IS-A testing were collected to form a "Parent set." Then for each new concept, it was paired with all the concepts from the "Parent set" except its own parents, generating non-IS-A test pairs (Figure 4.5). By doing this, the non-IS-A test data is not limited to only nephew-uncle pairs, but allows general combinations of new concepts with other existing parent concepts. The CNN model's performance on such concept pairs reveals its generalizability to the broader population of non-IS-A test data. To achieve balanced testing data, 7,494 non-IS-A concept pairs were randomly selected to match the number of IS-A concept pairs.

The process of converting a test pair of concepts into a vector pair is demonstrated in Figure 4.5. After the test pairs for both positive and negative samples were selected, the PV-DM and PV-DBOW models were queried to get the corresponding concept vectors. For each concept pair, four vectors (two vectors for each concept) were obtained. To get the corresponding vector pair, the two vectors generated by the same model were concatenated into one vector. Then the two long vectors were sent into the pre-trained CNN model in a random order to compute a class label (0 or 1) as the prediction result. For an

IS-A test concept pair, label 1 is correct if there is an IS-A link between these two concepts in the new SNOMED CT release. For a non-IS-A concept pair, label 0 is correct if there is no IS-A link between these two concepts in the new SNOMED CT release. The training and testing were repeated ten times with randomly chosen 90:10 splits between training and validation data to obtain a consistent evaluation of the efficacy of the methodology.

Besides examining the quality of the training data, the quality of the concept vectors from the Doc2Vec embedding was also scrutinized. The "iterations/epochs" parameters of the PV-DM and PV-DBOW models were adjusted to control how many times to iterate over the training corpus. The typical iteration counts suggested in the original "Paragraph Vectors" paper [63] are 10-20 for tens-of-thousands to millions of documents. Thus, 10, 20, and 50 iterations were experimented with, for comparison.

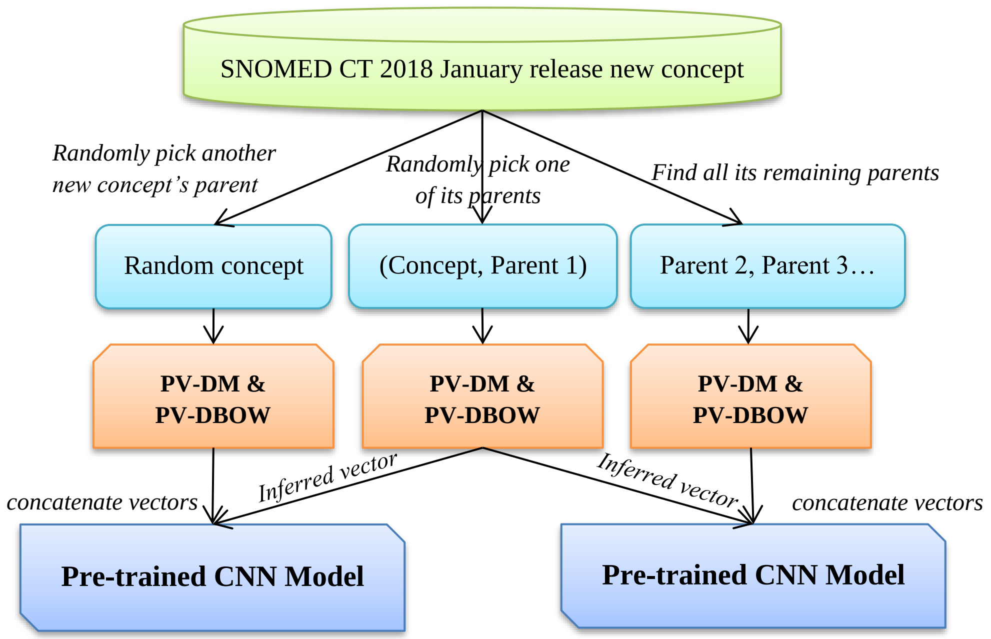


Figure 4.5 Data flow during testing CNN model with new data (use case).

4.4 Performance Comparison between Two Classifiers

The prediction results of the CNN model with the 7,494 IS-A connected concept pairs will be reported now. Table 4.1 summarizes the performance comparison between using the original (= "unsummarized") training data from the whole *Clinical Finding* hierarchy (shown with the column title "hierarchy") versus training data from the same area only (shown as "area"). The Precision, Recall, and F1 scores for ten tests are presented in Table 4.1. The model trained using the Area Taxonomy data is superior to the model trained using hierarchy data in both best F1 score (0.78 vs. 0.73) and average F1 score (0.75 vs. 0.69). The standard deviations of the ten F1 scores for both trained models are the same (0.018).

Table 4.1 IS-A Testing Results for Ten Tests, Comparing SNOMED CT Hierarchy Training Data with "in area" Training Data

Index	**Precision**		**Recall**		**F1**	
	hierarchy	**area**	**hierarchy**	**area**	**hierarchy**	**area**
1	0.85	0.83	0.59	0.65	0.69	0.73
2	0.86	0.83	0.63	0.70	0.73	0.76
3	0.86	0.82	0.58	0.67	0.69	0.74
4	0.87	0.78	0.59	0.74	0.70	0.76
5	0.86	0.86	0.58	0.64	0.69	0.73
6	0.86	0.80	0.58	0.69	0.69	0.74
7	0.87	0.82	0.57	0.71	0.69	0.76
8	0.87	0.80	0.53	0.76	0.66	**0.78**
9	0.85	0.83	0.62	0.64	0.71	0.72
10	0.86	0.81	0.58	0.69	0.69	0.74
Average	0.86	0.82	0.59	0.69	0.69	0.75
Standard deviation	0.007	0.022	0.027	0.041	0.018	0.018

Next, the model's performance on non-IS-A concept pairs was evaluated. The Precision, Recall, and F1 scores for ten tests are presented in Table 4.2. Table 4.2 summarizes the performance comparison for non-IS-A concept pairs between using data from the whole hierarchy (shown as "hierarchy") versus using training data from the same area (shown as "area"). The model trained with Area Taxonomy data achieved the same

F1 value as the model trained with the whole hierarchy (0.78). The standard deviation of the ten F1 scores for the "hierarchy" model is 0.005, and it is 0.008 for the "area" model.

Table 4.2 Non-IS-A Testing Results among Ten Tests, Comparing SNOMED CT Hierarchy Training Data with "in area" Training Data

Index	**Precision**		**Recall**		**F1**	
	hierarchy	**area**	**hierarchy**	**area**	**hierarchy**	**area**
1	0.69	0.71	0.89	0.86	0.78	0.78
2	0.71	0.74	0.90	0.86	0.79	**0.80**
3	0.68	0.72	0.90	0.85	0.78	0.78
4	0.69	0.76	0.91	0.79	0.78	0.77
5	0.68	0.71	0.90	0.89	0.78	0.79
6	0.68	0.73	0.90	0.83	0.78	0.78
7	0.68	0.74	0.91	0.85	0.78	0.79
8	0.66	0.77	0.92	0.81	0.77	0.79
9	0.70	0.71	0.89	0.87	0.78	0.78
10	0.68	0.73	0.90	0.84	0.78	0.78
Average	0.69	0.73	0.90	0.85	0.78	0.78
Standard deviation	0.014	0.021	0.009	0.029	0.005	0.008

The training time for the 10, 20, and 50 iterations was 485, 947, and 2035 seconds, respectively, with the same hardware configuration. The CNN models trained on top of these three vector spaces achieved an average F1 score of approximately 0.745 for IS-A testing, for each of the 10, 20, and 50 iterations. For non-IS-A testing, the average F1 scores for the 10, 20, and 50 iterations were 0.784, 0.775, and 0.753, respectively. Notably, the F1 scores were going down when performing more iterations.

CHAPTER 5

ENRICHMENT OF SNOMED USING BERT

Chapter 3 presented the effectiveness of using neural network-based methods, such as Convolutional/Recurrent Neural Networks to enrich biomedical ontologies by predicting IS-A relationships. In this chapter, we conducted experiments with a more advanced model for the same task. This model is called Bidirectional Encoder Representations from Transformers (BERT), a relatively new language representation model, which obtains state-of-the-art results on a wide array of general English NLP tasks. This chapter explores BERT's applicability to medical terminology-related tasks. Utilizing the "next sentence prediction" capability of BERT, it is shown that the fine-tuning strategy of Transfer Learning (TL) from the $\text{BERT}_{\text{BASE}}$ model can address the challenging problem of insertion of new concepts into a terminology. Adding a pre-training strategy enhances the results.

BERT was not trained with medical literature. In order to harness the high performance of BERT for medical terminology enrichment, two approaches are introduced in this chapter. One approach is to add medical knowledge to the general knowledge of BERT. For this purpose, the SNOMED CT knowledge was used, providing a "document" for each concept of SNOMED CT. The second approach is to train BERT to be able to distinguish between concept pairs that should be connected by IS-A relationships and pairs that shouldn't. This kind of learning utilizes the "next sentence prediction" feature of BERT. The technical issues and the details involved in implementing these ideas will be described in the following subsections.

5.1 Fine-tuning BERT Model for IS-A Relationship Classification

Step 1. Fine-tuning the BERT$_{BASE}$ model (Figure 5.1): IS-A linked concept pairs and concept pairs not connected by IS-A links were extracted as supervised fine-tuning data from the SNOMED CT July 2017 release. These pairs are referred to as IS-A and non-IS-A pairs, respectively. Then a relationship classifier was trained on top of BERT$_{BASE}$ with the IS-A and non-IS-A pairs to obtain the BERT$_{BASE+CLF}$ model (CLF = Classifier).

Step 2. Prediction on new release data (illustrated in the rightmost process of Figure 5.1): The trained BERT$_{BASE+CLF}$ model was tested to verify IS-A links and the absence of IS-A links for newly added concepts in the SNOMED CT January 2018 release.

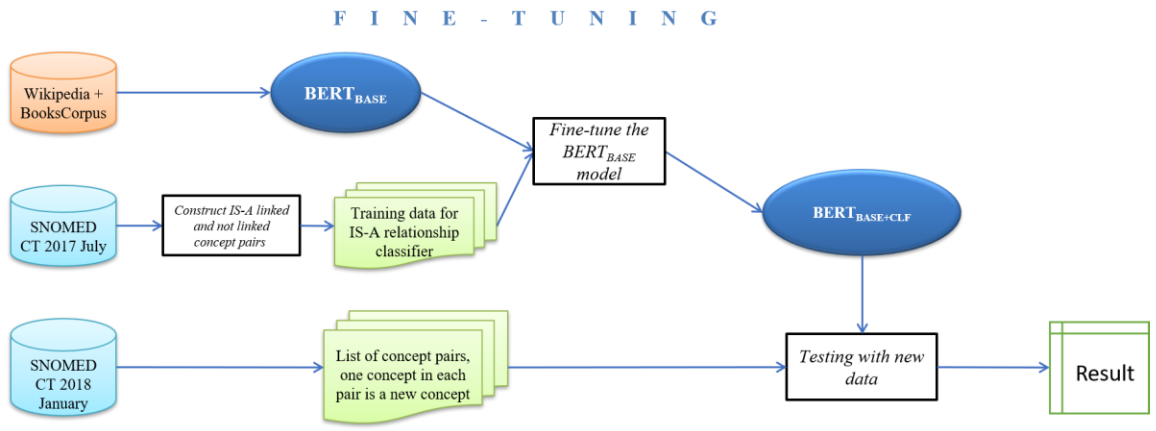

FINE-TUNING
Wikipedia + BooksCorpus
BERT_BASE
Fine-tune the BERT_BASE model
SNOMED CT 2017 July
Construct IS-A linked and not linked concept pairs
Training data for IS-A relationship classifier
BERT_BASE+CLF
SNOMED CT 2018 January
List of concept pairs, one concept in each pair is a new concept
Testing with new data
Result

Figure 5.1 The pipeline for Strategy 1 Fine-tuning. CLF is short for Classifier.

5.1.1 Data Preparation

The training sample passed to the fine-tuning process was a set of IS-A and non-IS-A concept pairs. In SNOMED CT, the IS-A concept pairs were given. Thus, they were used as positive sample instances. On the other hand, the negative sample could consist of all the non-IS-A pairs of concepts. This would create an imbalance between the positive and negative samples, because there are always many more pairs not connected by IS-A links. Thus, non-IS-A pairs for the negative sample were picked as follows. For each IS-A pair

(A, B) siblings C_1, C_2, …C_k of B were selected to form the non-IS-A pairs (A, C_i) with i=1, 2, ... k. The advantage of such pairs is that they are closely related to the corresponding IS-A pair. This will sharpen the distinction between IS-A and non-IS-A pairs in training. For example, (*Crushing injury of back*, *Crushing Injury*) defines an IS-A link, while (*Crushing injury of back*, *Shear injury*) is a similar pair that is not connected by an IS-A link. The reason is that *Shear injury* is a sibling of *Crushing Injury*, with the same parent *Injury by mechanism*.

In the data preparation for fine-tuning, the positive and negative samples were first extracted, and the negative sample was randomly downsampled to the size of the positive sample, at the beginning of each training round. The dataset was shuffled and 90% of it was used for training and 10% was kept as the test set. The samples went through three preprocessing steps: Text normalization (e.g., Excision of Reinke's edema, → excision of reinke's edema), Punctuation splitting (e.g., excision of reinke's edema, → excision of reinke ' s edema), and WordPiece tokenization (excision of reinke ' s edema, → ex ##cision of rein ##ke ' s ed ##ema). Then the samples were processed by $\text{BERT}_{\text{BASE}}$, which performed its own preprocessing, including input embeddings, segment masking, labeling, etc. [86]. The input embeddings are the sum of the token embeddings, the segmentation embeddings and the position embeddings.

For example, *Urine xanthine level* is the child of *Evaluation of urine specimen* in the SNOMED CT *Procedure* hierarchy. The input sequence will be "1 Evaluation *of urine specimen (\t) Urine xanthine level (\t)*". This input will be converted as one training instance to "[CLS] evaluation of urine specimen [SEP] urine x ##ant ##hine level [SEP]" as shown in Figure 5.2(a). The first token of the sequence is the classification embedding

([CLS]), representing a classification label. A special token ([SEP]) is used to separate sentences. Out-of-vocabulary words are split into word pieces and denoted with ##. For example, "xanthine" is denoted as three items "x", "##ant", and "##hine." Similarly, *Rubella screening* is <u>not</u> a child of *Down's screening – blood test*. The input sequence "0 *Rubella screening (\t) Down's screening - blood test (\t)*" will be converted to "[CLS] rub ##ella screening [SEP] down ' s screening - blood test [SEP]" in Figure 5.2(b).

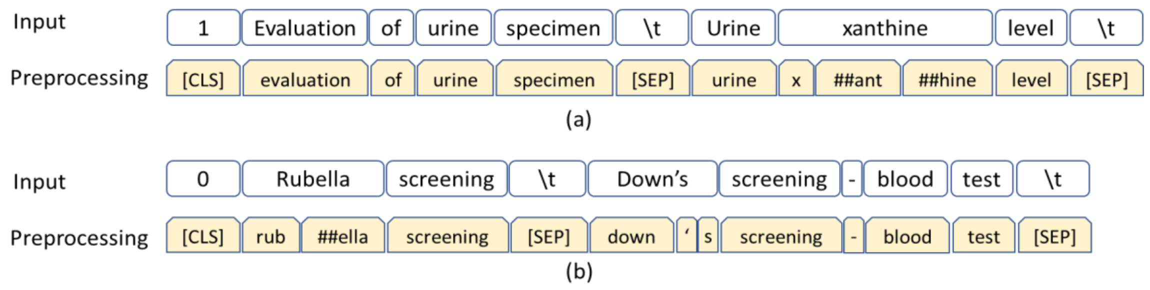
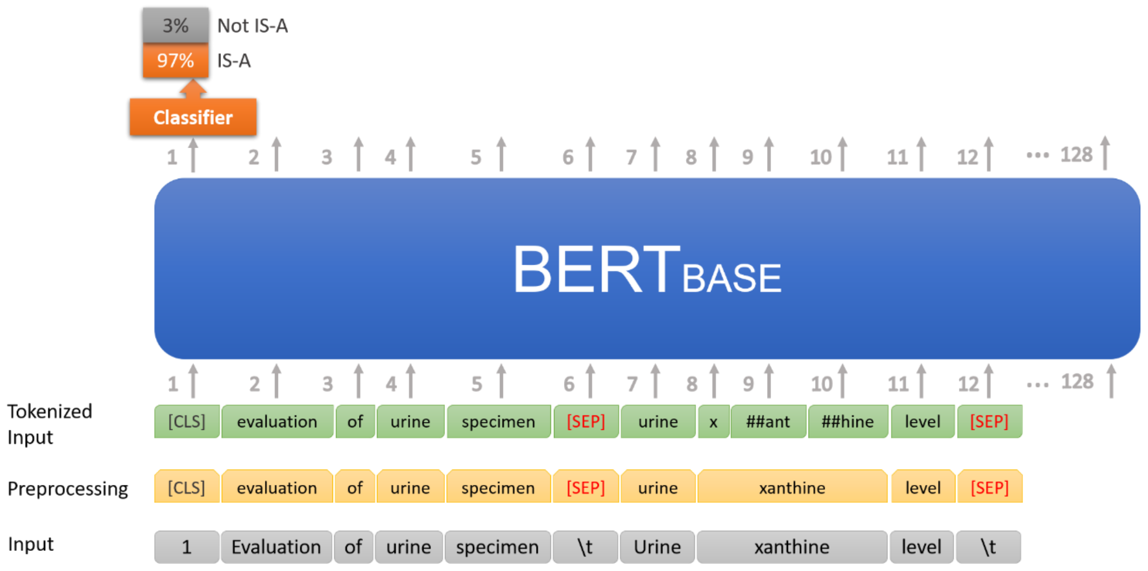


Figure 5.2 Fine-tuning data: Preprocessing (a) IS-A and (b) non-IS-A concept pairs.

5.1.2 Fine-tuning the BERT$_{\text{BASE}}$ Model

The BERT$_{\text{BASE}}$ model was fine-tuned to predict the IS-A and non-IS-A linking for the concept pairs in the test data. This is similar to a binary sentence-pair classification task. Utilizing its sentence prediction capability, BERT$_{\text{BASE}}$ was trained as BERT$_{\text{BASE+CLF}}$, to predict IS-A links between concept pairs of a terminology. A classifier was employed by using *softmax* with categorical cross-entropy on top of BERT$_{\text{BASE}}$. The parameters of BERT$_{\text{BASE}}$ were fine-tuned to maximize the log-probability of the correct label (IS-A or non-IS-A). Given two concepts A and B, described as two "sentences," the classifier learned to determine whether B is the next "sentence" following A. This was modeled as equivalent to classifying whether the concept B should be a child of A, i.e., B *IS-A* A (Figure 5.3).



3% Not IS-A
97% IS-A
Classifier
1 2 3 4 5 6 7 8 9 10 11 12 ··· 128
BERTBASE
1 2 3 4 5 6 7 8 9 10 11 12 ··· 128
Tokenized Input
[CLS] evaluation of urine specimen [SEP] urine x ##ant ##hine level [SEP]
Preprocessing
[CLS] evaluation of urine specimen [SEP] urine xanthine level [SEP]
Input
1 Evaluation of urine specimen \t Urine xanthine level \t

Figure 5.3 Fine-tuning the BERT$_{\text{BASE}}$ model with concept pairs to obtain BERT$_{\text{BASE+CLF}}$ model.

The input "1 Evaluation *of urine specimen (\t) Urine xanthine level (\t)*" was converted as one training instance to "[CLS] evaluation of urine specimen [SEP] urine x ##ant ##hine level [SEP]" with Class label = 1. Class 1 means that there should be an IS-A link between the two concepts, and Class 0 means that there shouldn't be such a link. The BERT$_{\text{BASE}}$ model computed the probabilities for Class 0 and Class 1 and recorded the result as a 2×1 vector. The classifier reported the class label with the higher probability. The error between the true label and the label predicted by the model was back-propagated through the model to improve the CNN network's parameters. The obtained model is denoted as BERT$_{\text{BASE+CLF}}$ (CLF = classifier), the model after fine-tuning. For this default model, the same hyperparameters were used as in the pre-trained BERT$_{\text{BASE}}$, with the exception of the sequence length (=128), batch size (=64), learning rate (=2e^{-5}), and number of training epochs (=3).

5.2 Pre-training BERT Model with SNOMED Hierarchical Data

Step 1. Pre-training the BERT$_{BASE}$ model (Figure 5.4): Concept-related information from the July 2017 release was preprocessed to generate documents that were used as unsupervised pre-training data. Then BERT$_{BASE}$ was trained with unsupervised concept level data so that the trained BERT$_{BASE+SNO}$ model integrated terminology information from SNOMED CT (=SNO). Then the fine-tuning process (of Strategy 1) was applied to train a classifier on top of BERT$_{BASE+SNO}$ to derive the BERT$_{BASE+SNO+CLF}$ model.

Step 2. Prediction on new release data (illustrated in the rightmost process of Figure 5.4): The trained BERT$_{BASE+SNO+CLF}$ model was tested to verify IS-A links and non-IS-A pairs for newly added concepts in the SNOMED CT 2018 January release.

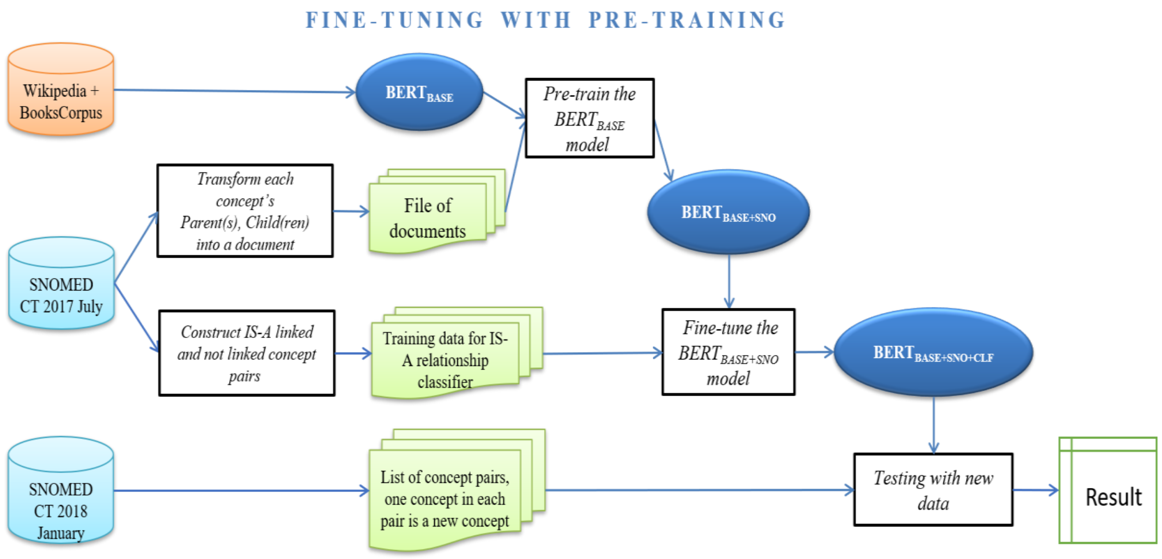

FINE-TUNING WITH PRE-TRAINING
Wikipedia + BooksCorpus
BERT BASE
Pre-train the BERT BASE model
Transform each concept's Parent(s), Child(ren) into a document
File of documents
BERT BASE+SNO
SNOMED CT 2017 July
Construct IS-A linked and not linked concept pairs
Training data for IS-A relationship classifier
Fine-tune the BERT BASE+SNO model
BERT BASE+SNO+CLF
SNOMED CT 2018 January
List of concept pairs, one concept in each pair is a new concept
Testing with new data
Result

Figure 5.4 The pipeline for Strategy 2 combining Fine-tuning with Pre-training. SNO is short for SNOMED CT.

5.2.1 Data Preparation

In the setup of unsupervised pre-training, BERT$_{BASE}$ is not trained for a specific task, but the purpose is integrating medical knowledge into its representation. This is done by training with the non-task related sample taken from SNOMED CT. BERT was originally trained with millions of documents that are composed of sentences. Similarly, a list of terminology-oriented documents were generated by creating one "document" per focus concept F (Figure 5.5), with related concept(s) as "sentences" of such documents. The ID for a document is the corresponding SNOMED CT concept ID. The content of this document consists of the concepts that are hierarchically related to F (Figure 5.5(a)). Specifically, F's parents (targets of IS-A links from F), F itself, and its children (sources of IS-A links to F) were chosen. Thus, the whole document text is: Parent(s) – Focus concept – Child(ren) (Figure 5.5(b)). The concept groups are separated into lines, e.g., "(Parents) *finding of abnormal level of heavy metals in blood*, *finding of trace element level* –NEW LINE– (Focus concept) *blood copper abnormal* –NEW LINE– (Children) *raised blood copper level*, *serum copper level abnormal"* is the document for the focus concept *blood copper abnormal.* This construction is based on the idea that a concept is the topic of a document and that the closely related concepts are descriptions of the meaning of this concept in the terminology hierarchy. To feed sentences into the BERT$_{BASE}$ model, all the documents are concatenated in one text file, separated by empty lines.

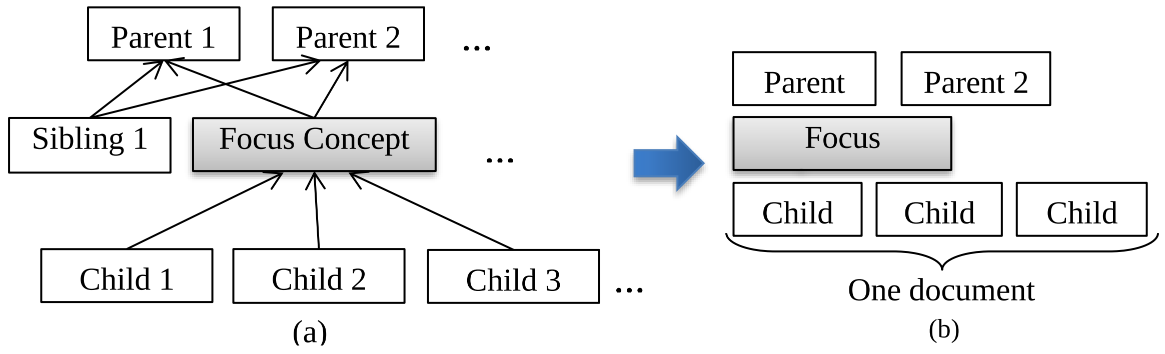


Figure 5.5 Pre-training data: Serializing **(a)** the hierarchical structure of one concept into **(b)** one document.

5.2.2 Pre-training the BERT$_{BASE}$ Model

To utilize BERT's powerful language representation, BERT$_{BASE}$ was enriched with terminology knowledge by using new training data. BERT was originally trained for two unsupervised prediction tasks: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) on an arbitrary text corpus. The same two training tasks and objective were adopted with concept-based documents from SNOMED CT.

In the MLM phase, the training objective is to predict only the masked words. Across all the concept-based documents, 15% of the words were randomly masked out, and then the complete BERT$_{BASE}$ model was trained to output the masked words. In the NSP phase, the objective is to learn relationships between concepts: Given two concepts A and B, is B a child of A, or not (Figure 5.6). Two "sentences" *Colitis* and *Phlegmonous colitis* from the document were extracted for the focus concept *Colitis*. After preprocessing these two concepts (treated as two "sentences") as shown in the middle level, two token – "##mon" and "##tis" were masked out. The BERT$_{BASE}$ model was trained to raise the probabilities of two correct tokens "##mon" and "##tis" over other tokens in the vocabulary.

In the Figure 5.6, we symbolize this vocabulary using a list of tokens starting from "Apple" to "Zoo." The actual vocabulary used by BERT is a WordPiece vocabulary with a list of 30522 tokens. In addition, as *Phlegmonous colitis* IS-A *Colitis,* the BERT$_{BASE}$ model was also trained to output the correct classification label "IsNext." The obtained model is denoted as BERT$_{BASE+SNO}$ (SNO=SNOMED CT). Then the fine-tuning process (Section 5.1.2) was applied to the BERT$_{BASE+SNO}$ model to get the BERT$_{BASE+SNO+CLF}$ model.

The training parameters used for pre-training are as follows: batch size = 64, sequence length = 128, training steps = 15,000 for *Procedure* and 200,000 for *Clinical*

Finding, learning rate =2e^{-5}, dropout rate = 0.1, and activation function = gelu (Gaussian error linear unit).

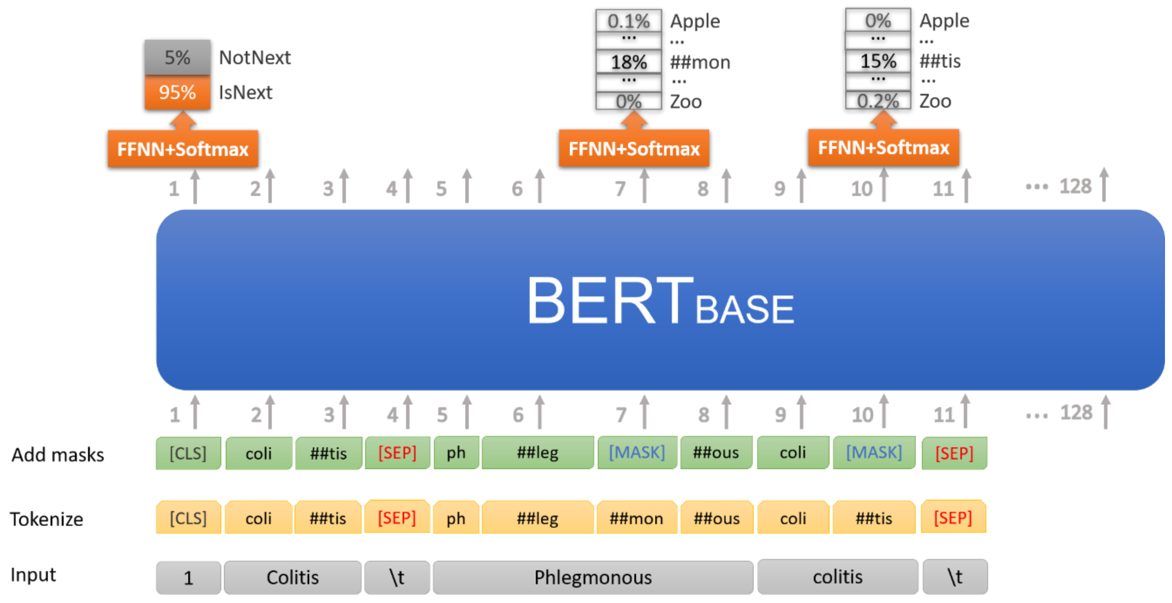

5%
NotNext
95%
IsNext
FFNN+Softmax
0.1% Apple
18% ##mon
0% Zoo
FFNN+Softmax
0% Apple
15% ##tis
0.2% Zoo
FFNN+Softmax
BERTBASE
Add masks
[CLS] coli ##tis [SEP] ph ##leg [MASK] ##ous coli [MASK] [SEP]
Tokenize
[CLS] coli ##tis [SEP] ph ##leg ##mon ##ous coli ##tis [SEP]
Input
1 Colitis \t Phlegmonous colitis \t

Figure 5.6 Pre-training the BERT$_{\text{BASE}}$ model with concept-based documents to obtain BERT$_{\text{BASE+SNO}}$ model. FFNN is short for Feedforward neural network.

5.3 Performance Comparison between Two Classifiers

5.3.1 Testing the Prediction on New Release Data

To evaluate the BERT$_{\text{BASE+CLF}}$ and the BERT$_{\text{BASE+SNO+CLF}}$ models on previously unseen data, separate test tasks were created, using new concepts from the *Procedure* and the *Clinical finding* hierarchy of the January 2018 release (Figure 5.7). This description will focus on the *Procedure* hierarchy. For each new concept that was added to the *Procedure* hierarchy in this release, its parents and itself were extracted as positive sample pairs. For example, *Local excision of lesion of kidney* has two parents—*Local excision* and *Excision of lesion of kidney*. The corresponding positive testing sample instances are "*Local excision* (\t) *Local excision of lesion of kidney*" and "*Excision of lesion of kidney* (\t) *Local excision* *of lesion of kidney*" with the true class label = 1. For the negative sample, each new concept was paired with a randomly chosen parent taken from the other new concepts' parents. For example, *Ultrasonography of left lower limb*, which is the parent of *Ultrasonography of left knee region*, was selected randomly and paired it with *Local excision of lesion of kidney* to form the instance "*Ultrasonography of left lower limb* (\t) *Local excision of lesion of kidney*" with the label = 0.

The positive and negative samples were randomly arranged into a sequence and sent to the trained $\text{BERT}_{\text{BASE+CLF}}$ and $\text{BERT}_{\text{BASE+SNO+CLF}}$ models for testing. The models processed each input pair, using the weights that it had learned before, returning a class label (0 or 1) as prediction result. For negative sample instances, label 0 is correct, indicating that there is no IS-A link between these two concepts in the new SNOMED CT release. Label 1 is correct for positive sample instances, indicating the existence of an IS-A link. The predicted result labels were compared with the true labels to calculate the prediction accuracy in terms of Precision, Recall, and F1 score.

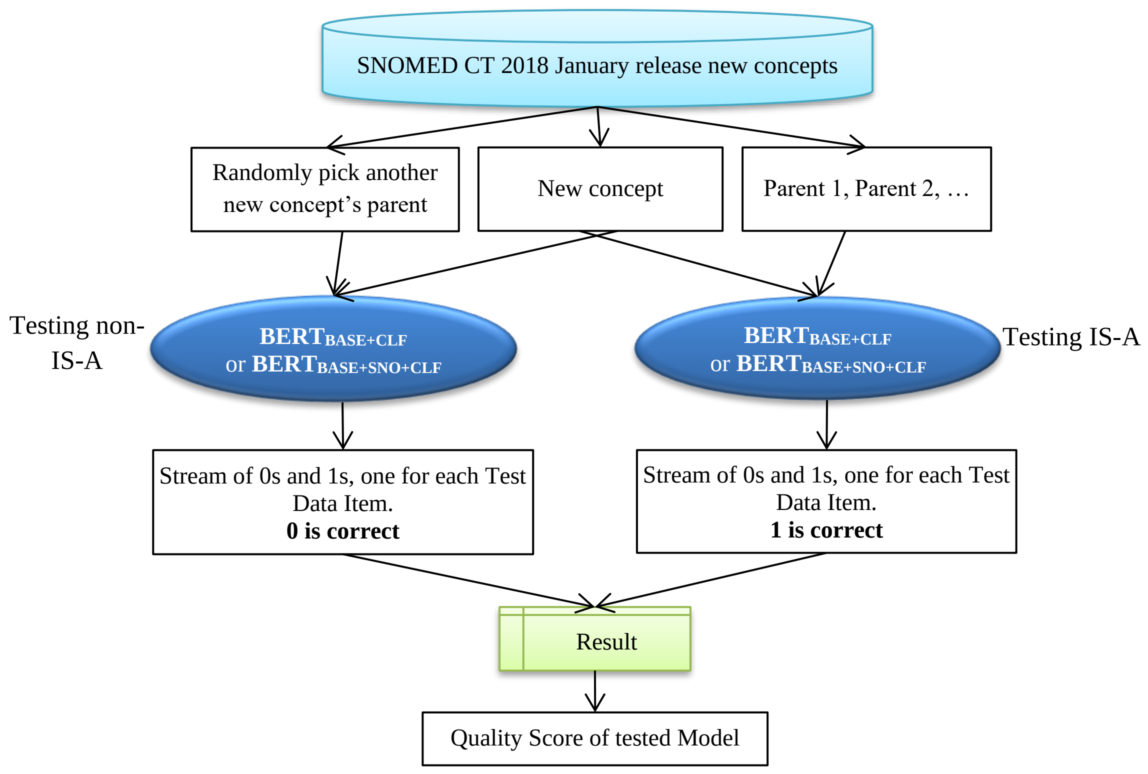

SNOMED CT 2018 January release new concepts
Randomly pick another new concept's parent
New concept
Parent 1, Parent 2, …
Testing non-IS-A
BERTBASE+CLF or BERTBASE+SNO+CLF
BERTBASE+CLF or BERTBASE+SNO+CLF
Testing IS-A
Stream of 0s and 1s, one for each Test Data Item. 0 is correct
Stream of 0s and 1s, one for each Test Data Item. 1 is correct
Result
Quality Score of tested Model

Figure 5.7 Data flow for testing the trained BERT$_{\text{BASE+CLF}}$ or BERT$_{\text{BASE+SNO+CLF}}$ models with unseen data.

5.3.2 Results

The prediction results of the fine-tuned model and the pre-trained & fine-tuned model with samples extracted from the *Procedure* hierarchy of the SNOMED CT 2018 January release, with 15,000 training steps, will be reported first. The Precision, Recall, and F1 scores for ten tests are presented in Table 5.1. For the *Procedure* hierarchy, the model was tested against 3,908 pairs (1,954 positives and 1,954 negatives). For example, in Test 7 for IS-A classification, the Precision is 0.69, Recall is 0.98, and F1 score is 0.81 for fine-tuning. When adding pre-training, Precision is 0.73, Recall is 0.98, and the F1 score is 0.84. The F1 score improved by about 3.7% with pre-training. Similarly, for Non-IS-A tests, the F1 score increased from 0.71 to 0.77, an 8.5% improvement. On average, by adding pre-training, we achieved 6.3% (from 0.80 to 0.85) and 14.5% (from 0.69 to 0.79) improvements of F1 for IS-A and Non-IS-A classifications, respectively.

Table 5.1 Precision, Recall, and F1 Score for Ten Tests of *Procedure* Hierarchy (Training Steps = 15,000)

Procedure	**IS-A Classification**						**Non-IS-A Classification**					
	Fine-tuning			**Pre-training & Fine-tuning**			**Fine-tuning**			**Pre-training & Fine-tuning**		
No.	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1	0.67	0.98	0.79	0.75	0.98	0.85	0.96	0.51	0.67	0.97	0.68	0.80
2	0.66	0.98	0.79	0.77	0.98	0.86	0.97	0.48	0.64	0.98	0.70	0.82
3	0.70	0.98	0.81	0.75	0.98	0.85	0.96	0.58	0.72	0.97	0.68	0.80
4	0.66	0.98	0.79	0.74	0.98	0.84	0.97	0.50	0.66	0.97	0.66	0.79
5	0.69	0.98	0.81	0.74	0.98	0.84	0.96	0.56	0.70	0.97	0.66	0.79
6	0.71	0.97	0.82	0.74	0.98	0.85	0.96	0.59	0.73	0.97	0.66	0.79
7	0.69	0.98	0.81	0.73	0.98	0.84	0.97	0.56	0.71	0.97	0.64	0.77
8	0.67	0.98	0.80	0.71	0.98	0.82	0.96	0.52	0.68	0.97	0.59	0.73
9	0.69	0.98	0.81	0.73	0.98	0.84	0.96	0.56	0.71	0.97	0.63	0.77
10	0.66	0.98	0.79	0.76	0.98	0.86	0.96	0.49	0.65	0.97	0.69	0.81
Average	0.68	0.98	**0.80**	0.74	0.98	**0.85**	0.96	0.54	**0.69**	0.97	0.66	**0.79**
Standard Deviation	0.02	0.00	0.01	0.02	0.00	0.01	0.00	0.04	0.03	0.00	0.03	0.03

The summary of the Precision, Recall, and F1 scores of ten tests for the *Procedure* hierarchy with 10,000 training steps is reported in Table 5.2. In each test, the model was tested against 3,908 pairs (1,954 positives and 1,954 negatives). On average, by adding pre-training to fine-tuning, the improvements are 3.75% (from 0.80 to 0.83) and 10.1% (from 0.69 to 0.76) for IS-A and Non-IS-A classifications, respectively.

Table 5.2 Precision, Recall, and F1 Score for Ten Tests of *Procedure* Hierarchy (Training Steps = 10,000)

Proce-dure	**IS-A Classification**						**Non-IS-A Classification**					
	Fine-tuning			**Pre-training & Fine-tuning**			**Fine-tuning**			**Pre-training & Fine-tuning**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Average	0.68	0.98	0.80	0.72	0.98	0.83	0.96	0.54	0.69	0.97	0.62	0.76
Max	0.71	0.98	0.82	0.75	0.99	0.85	0.97	0.59	0.73	0.98	0.67	0.79
Min	0.66	0.97	0.79	0.71	0.98	0.82	0.96	0.48	0.64	0.96	0.59	0.73
Standard Deviation	0.02	0.00	0.01	0.01	0.00	0.01	0.00	0.04	0.03	0.01	0.02	0.02

For the *Clinical finding* hierarchy, the summary of ten test results with training steps = 200,000 is reported in Table 5.3. In each test, the model was tested against 8,574 pairs (4,287 positives and 4,287 negatives). On average, by adding pre-training, the improvements that were achieved are 7.5% (from 0.80 to 0.86) and 15.3% (from 0.72 to 0.83) for IS-A and Non-IS-A classifications, respectively.

Table 5.3 Precision, Recall, and F1 Score for Ten Tests of *Clinical Finding* Hierarchy (Training Steps = 200,000)

Clinical Finding	**IS-A Classification**						**Non-IS-A Classification**					
	Fine-tuning			**Pre-training& Fine-tuning**			**Fine-tuning**			**Pre-training& Fine-tuning**		
	Precision	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**	**Precision**	**Recall**	**F1**
Average	0.70	0.94	**0.80**	0.79	0.94	**0.86**	0.90	0.60	**0.72**	0.93	0.76	**0.83**
Max	0.72	0.94	0.82	0.82	0.95	0.88	0.91	0.64	0.75	0.94	0.80	0.86
Min	0.69	0.93	0.79	0.77	0.94	0.85	0.89	0.58	0.70	0.92	0.73	0.81
Standard Deviation	0.01	0.00	0.01	0.01	0.00	0.01	0.01	0.02	0.02	0.01	0.02	0.01

Regarding the prediction of IS-A links for new concepts, examples of the two models' prediction results (Table 5.4) are shown for ten pairs for which the second concept was newly added to SNOMED CT's *Clinical finding* hierarchy in the 2018 January release. For each test, one *test concept* was paired with one *new concept* as one test instance, then the model predicted whether there should be an IS-A link between them. For instance, for Example 2, *Arthropathy of knee joint* was chosen as the first concept and paired with *Aseptic necrosis of right lateral femoral condyle*. Then the task became to predict whether there is an IS-A link between the two concepts. Both the fine-tuning and pre-training & fine-tuning models returned the correct label (=1). For *Example 4*, the fine-tuning model is wrong, and the combined model is correct that *Chronic pain following radiotherapy* IS-A

Persistent pain following procedure. Both models are wrong about *Injury of bilateral optic tracts*, because it is not a *Soft tissue injury* (Example 9).

Table 5.4 Prediction Results of Two Models on Five IS-A & Five non-IS-A Examples from *Clinical Finding* Hierarchy.

Index	**Test Concept**	**New Concept**	**True Label**	**Fine-tuning**	**Fine-tuning and Pre-training**
1	*Injury of trachea*	*Crushing injury of trachea*	1	1	1
2	*Arthropathy of knee joint*	*Aseptic necrosis of right lateral femoral condyle*	1	1	1
3	*Lesion of neck*	*Stenosis of right vertebral artery*	1	1	1
4	*Persistent pain following procedure*	*Chronic pain following radiotherapy*	1	0	1
5	*Joint injury*	*Traumatic rupture of ligament of wrist*	1	1	1
6	*Bursitis of shoulder*	*Injury of toenail*	0	0	0
7	*Atherosclerosis of artery*	*Crushing injury of trachea*	0	0	0
8	*Finding of employment status*	*Social isolation in parenthood*	0	0	0
9	*Soft tissue injury*	*Injury of bilateral optic tracts*	0	1	1
10	*Injury of wrist*	*Injury of peripheral nerve of abdomen*	0	1	0

CHAPTER 6

ENRICHMENT OF SNOMED USING BERT AND ABSTRACTION NETWORKS

The study in Chapter 4 showed that it is possible to further improve the performance of the CNN model by using summarization of ontologies, based on Abstraction Networks. In addition, the study in Chapter 5 and [122] utilized the "next sentence prediction" capability of BERT for IS-A relationship classifications and demonstrated a performance improvement with combining pre-training and fine-tuning of BERT. Hence, two independent ways were demonstrated to improve on the performance of previous work described in Chapter 3, namely a method to automatically predict the presence of IS-A relationships between a new concept and existing concepts based on the language representation model BERT. The first improvement was achieved by using the BERT model rather than the CNN model and the second by utilizing ontology summarization to provide a more accurate training data set of a CNN model. However, the utilization of the "next sentence prediction" capability of BERT was not optimal.

This chapter combines the two improvements by utilizing ontology summarization together with the BERT model and with an improved presentation of the training data to better utilize the "next sentence prediction" capability of BERT. It is a challenge to further improve the performance of the BERT model, which is already high, with a recall of 0.94.

The SNOMED CT 2017 July release was used as training set and the following 2018 January release as testbed. Due to the different sizes and inconsistent modeling schemas across hierarchies, SNOMED's largest hierarchy, *Clinical Finding,* was selected as the data source. The models were implemented with Tensorflow [118]. The models were trained and tested on a machine with two Nvidia Tesla P100 “Pascal” video cards with 16 GB RAM per GPU and two Intel Xeon E5-2630-v4 CPUs with 2.2 GHz processor speed and 128 GB memory per CPU.

Google released two BERT models: $\text{BERT}_{\text{BASE}}$ (12 Transformer layers) and $\text{BERT}_{\text{LARGE}}$ (24 Transformer layers). Both models were trained on their Cloud TPUs (Tensor Processing Units), which have 64GB of RAM. They were trained on English Wikipedia (2,500M words of text) and BookCorpus [123] (800M words of text) with one million update steps. As recommended by the BERT creators, $\text{BERT}_{\text{LARGE}}$ was not used on our GPUs with 16GB of RAM, because the RAM size limited the number of training instances in each batch to avoid out-of-memory issues. Therefore, only $\text{BERT}_{\text{BASE}}$ was used in this experiment. The number of parameters for the pre-trained $\text{BERT}_{\text{BASE}}$ model is 110M, with the default training settings L=12, H=768, A=12, where L is the number of layers (i.e., Transformer blocks), H is the hidden size, and A is the number of self-attention heads. The feed-forward/filter size is set to 4 times H, i.e., 3072 for H = 768.

Figure 6.1 illustrates the process of training and testing an IS-A relationship classifier using taxonomy data from SNOMED CT. In the following sections the three major steps of the process will be described in detail: (a) pre-train the BERT model with concept-level documents (blue arrows), (b) fine-tune the BERT model with data derived from our area taxonomy (black arrows), and (c) test the two trained models with data from the new SNOMED CT release (red arrows).

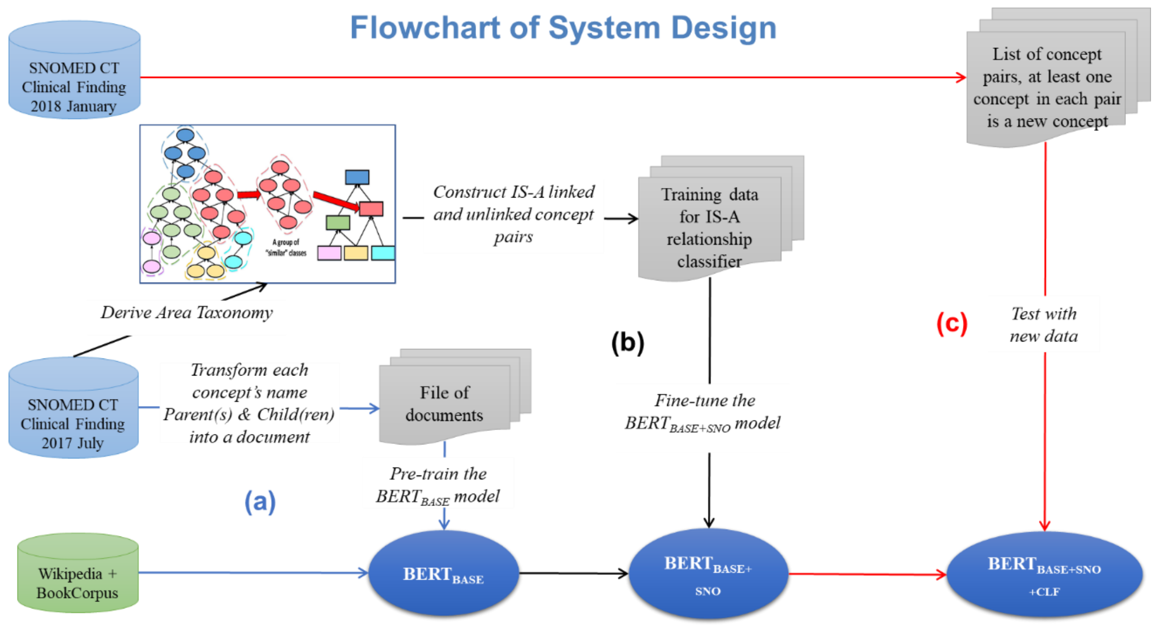

Flowchart of System Design
SNOMED CT Clinical Finding 2018 January
List of concept pairs, at least one concept in each pair is a new concept
Construct IS-A linked and unlinked concept pairs
Training data for IS-A relationship classifier
A group of "similar" classes
Derive Area Taxonomy
(b)
(c)
Test with new data
SNOMED CT Clinical Finding 2017 July
Transform each concept's name Parent(s) & Child(ren) into a document
File of documents
Fine-tune the BERT_BASE+SNO model
Pre-train the BERT_BASE model
(a)
Wikipedia + BookCorpus
BERT_BASE
BERT_BASE+SNO
BERT_BASE+SNO+CLF

Figure 6.1 Flowchart of training and testing an IS-A relationship classifier model with summarization data from Area Taxonomy of SNOMED CT (CLF = Classifier). Pre-train the BERT model (blue arrows), fine-tune the BERT model with area taxonomy (black arrows), and test with data from the new SNOMED CT release (red arrows).

6.1 Pre-training BERT Model with Concept-Level Documents

As a general language representation model, BERT was trained with Wikipedia and BookCorpus data, which do not provide a domain specific (i.e., medical) orientation for the ontology enrichment task. Chapter 5 demonstrated that pre-training BERT with a task-related corpus can improve the model's classification performance over directly fine-tuning the BERT model. Thus, an improved methodology was used by running additional steps of pre-training the BERT model with *Clinical Finding* hierarchy data, prior to training an IS-A relationship classifier using BERT. This section elaborates on the pre-training setup and process (Figure 6.1(a)) of how the data was extracted from the *Clinical Finding* hierarchy of SNOMED CT, and converted it into a format that is compatible with BERT for pre-training.

6.1.1 Data Preparation

The original BERT was pre-trained with general English sentences. To pre-train BERT with the knowledge of a hierarchy of a medical ontology is a challenge because BERT is trained only to handle text, while the hierarchy consists of concepts connected through IS-A relationships. Therefore, each concept's name was treated as a "sentence." The concept's hierarchically closely related concepts were considered as part of its definition in the ontology hierarchy. Thus, for a given concept A, A's hierarchically closely related concept(s)' names were also considered as "sentences."

The challenge is to harness the capability of BERT to model these IS-A relationships between concepts in an ontology. The hierarchical relationship from a specific concept to a general concept is expressed utilizing the features of the BERT model.

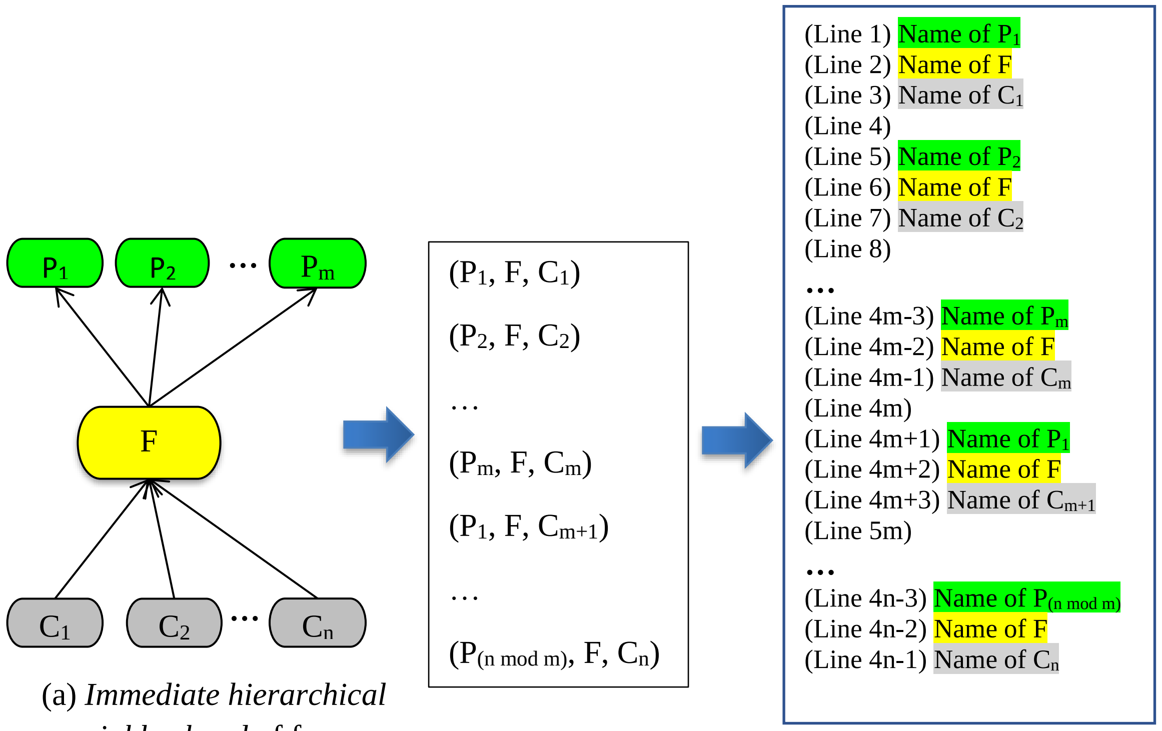
BERT was created for tasks such as Next Sentence Prediction (NSP), in which it needs to predict whether one sentence logically (based on human-like intuition, not on formal logic) follows another sentence in a given text. Given two concepts A and B and a relationship A IS-A B, the BERT model was trained to recognize the sentence of A as the next sentence following the sentence of B.

An ontology-oriented corpus was prepared for pre-training BERT with *Clinical Finding* IS-A relationships. For each concept, a document that consists of the concepts that are hierarchically related to it in a textual form was created. Though the choices for textual representation of hierarchically related concepts for a focus concept can vary, a simple pattern of a triple was preferred to form the document with a Parent – Focus concept – Child in each triple. In this way, two IS-A relationships are embedded, one from the focus concept to the parent concept, the other one from the child concept to the focus concept. Since the parents and children are important contextual knowledge elements of a focus concept, an *immediate neighborhood* [124], contains the concept itself plus all concepts connected to it by a single relationship, either hierarchical or lateral. Derived from this definition, the *immediate hierarchical neighborhood* of a concept is defined as follows.

Definition (*Immediate hierarchical neighborhood*): A concept's *immediate hierarchical neighborhood* contains the concept itself plus all concepts connected to it by a hierarchical relationship. That is, the *immediate hierarchical neighborhood* of a concept contains itself and all concepts at a hierarchical distance of one, i.e., its parents and children.

A general configuration of a focus concept with its *immediate hierarchical neighborhood* is illustrate in Figure 6.2(a). Consider a focus concept F (in yellow) with its *m* parents P_1 to P_m (in green) and *n* children C_1 to C_n (in grey). This configuration was represented by triples of the form (Parent, Focus concept, Child) to capture all the ($m + n$) IS-A relationships between the focus concept and its parent(s) and child(ren). Triples were constructed by using the focus concept and matching the parents and children by their indexes, e.g., the second parent matching with the second child (P_2, F, C_2) as shown in Figure 6.2(b).

Assuming *m* is less than *n* (the number of parents is smaller than the number of children), after exhausting all parents, the remaining children are matched with the parents from the beginning, e.g., (P_1, F, C_{m+1}), and ending with ($P_{(n \bmod m)}$, F, C_n) as shown in Figure 6.2(b). In this matching process, for each child with index $n \geq m$, for cases where n mod m = 0 (e.g. n = 2*m), parent P_m is used instead of $P_{(n \bmod m)}$ to get (P_m, F, C_n). In the rare cases when *n* is less than *m*, a similar modification is done to deal with the remaining parents. In Figure 6.2(c) the transition from the ontology dimension to the textual dimension is demonstrated, as it is needed for BERT, by converting each triple into a "paragraph." A paragraph consists of three lines to accommodate the three names, the parent concept, the focus concept, and the child concept, one name per line. The collection of *n* paragraphs for the *n* triples forms the document for the focus concept.



(a) *Immediate hierarchical neighborhood* of focus concept F with the assumption ($n \geq m$)

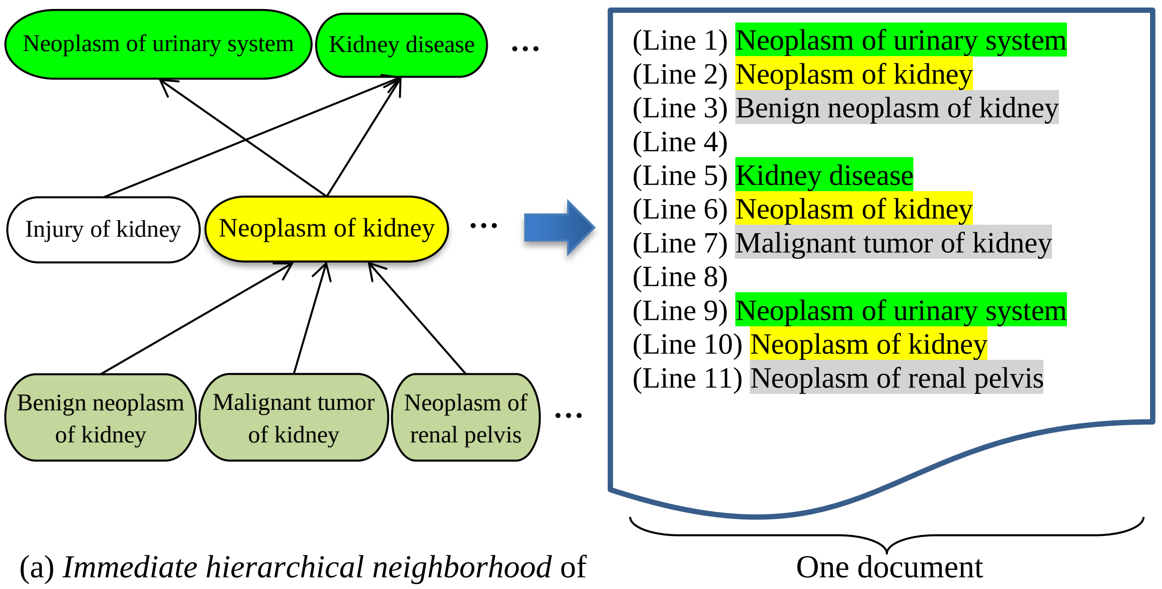
(b) Triples for the excerpt of (a)

(c) One document converted from the triples in (b)

Figure 6.2 Generate a document for a focus concept F: **(a)** the hierarchical structure of a focus concept F with m parent concepts P_1 to P_m and n child concepts C_1 to C_n, assuming $n \geq m$. **(b)** n (P, F, C) triples obtained from (a). **(c)** F's document representation by converting each triple in (b) into a three-line paragraph, with one concept's name per line. Each paragraph is separated from another paragraph by an empty line.

The above transformation is demonstrated with a concrete example with the focus concept *Neoplasm of kidney* in Figure 6.3. Figure 6.3(a) shows the neighborhood network of *Neoplasm of kidney* in yellow with its two parents (*Neoplasm of urinary system*, *Kidney disease* in green), and three children (*Benign neoplasm of kidney*, *Malignant tumor of kidney*, *Neoplasm of renal pelvis* in grey). These concepts are used to construct (Parent, Focus concept, Child) triples in which the focus concept is fixed and the parent concept and child concept are matched by their indexes, e.g., (*Kidney disease*, *Neoplasm of kidney*, *Malignant tumor of kidney*) is a triple matching the second parent with the second child. Generating triples stops when every concept is used in at least one triple. Each triple is converted to a three-line paragraph with one concept's name per line (Figure 6.3(b)). Two paragraphs are separated by an empty line.

For example, starting in Line 5 there are "(Line 5: Parent) *Kidney disease* – (Line 6: Focus concept) *Neoplasm of kidney* – (Line 7: Child) *Malignant tumor of kidney* – (Line 8) EMPTY LINE …". Thus, a list of ontology-oriented documents was generated by creating one "document" per concept. For simplicity, all the generated documents are concatenated in one text file, with pairs of documents separated by two empty lines, as the input to pre-train the BERT$_{\text{BASE}}$ model.



(Line 1) Neoplasm of urinary system
(Line 2) Neoplasm of kidney
(Line 3) Benign neoplasm of kidney
(Line 4)
(Line 5) Kidney disease
(Line 6) Neoplasm of kidney
(Line 7) Malignant tumor of kidney
(Line 8)
(Line 9) Neoplasm of urinary system
(Line 10) Neoplasm of kidney
(Line 11) Neoplasm of renal pelvis

(a) *Immediate hierarchical neighborhood* of *Neoplasm of kidney*

(b) One document for the excerpt of (a)

Figure 6.3 Pre-training data: Serializing **(a)** the hierarchical structure of Neoplasm of kidney into **(b)** one document.

6.1.2 Data Preprocessing

After the list of ontology-oriented documents was obtained, prior to training BERT, the samples were preprocessed in three steps: 1) Text normalization (e.g., Fournier's gangrene, → fournier's gangrene), 2) Punctuation splitting (e.g., fournier's gangrene, → fournier ' s gangrene), and 3) WordPiece tokenization (fournier ' s gangrene, → four ##nier ' s gang ##ren ##e). BERT$_{BASE}$ then converted the preprocessed samples into input embeddings, which are the sum of the token embeddings, the segmentation embeddings and the position embeddings [86]. The same input embedding methods from the BERT paper [86] were adopted. The operations that are essential for pre-training in this study are as follows:

In Figure 6.4, an example of how a training instance is formed from the concept-level document is demonstrated. For example, *Formestane allergy* is the child of *Estrogen antagonist* in the SNOMED CT *Clinical Finding* hierarchy. The input sequence will be "1 *Estrogen antagonist (\t) Formestane allergy (\t)*". The first token of the sequence is the classification embedding ([CLS]), representing a classification label. It is "1" for positive instances and "0" for negative instances in the training data. A special token ([SEP]) is used to separate sentences. Out-of-vocabulary words are split into word pieces and denoted with ##. For example, "estrogen" is denoted as two items "est" and "##rogen." This input will be converted into one training instance as "[CLS] est ##rogen antagonist all ##ergy [SEP] form ##est ##ane all ##ergy [SEP]" as shown in Figure 6.4(a). Similarly, *Main spoken language Turkmen* is <u>not</u> a child of *Born in Scotland*. The input sequence "*0 Born in Scotland (\t) Main spoken language Turkmen (\t)*" will be converted to "[CLS] born in scotland [SEP] main spoken language turk ##men [SEP]" in Figure 6.4(b).

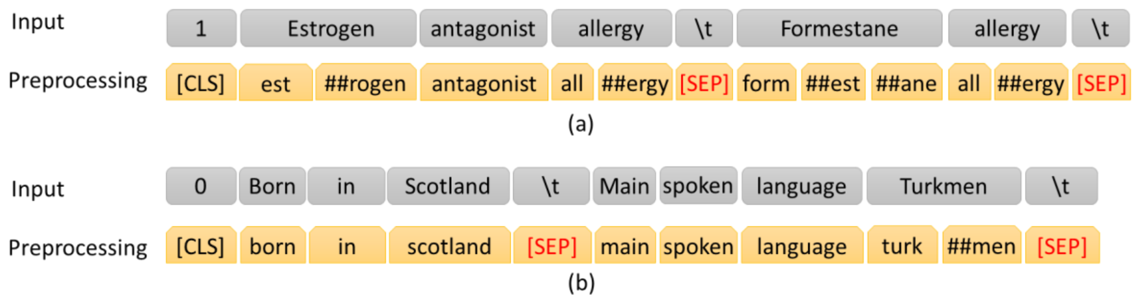


Figure 6.4 Preprocessing **(a)** IS-A and **(b)** non-IS-A concept pairs.

6.1.3 Pre-train BERT Model

The goal of pre-training is to embed ontology knowledge into BERT's language model. Therefore, training BERT$_{BASE}$ was refined with concept-based documents (prepared in Section 6.1.2) from SNOMED CT. To ensure the obtained model is compatible with the original BERT model, the same two training tasks were adopted, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). These are the two tasks that BERT was originally pre-trained for, since BERT is intended to process text. The training objective of the MLM task is to predict only the masked words. The training objective of NSP is to learn relationships between sentences (concepts) for any given sentence-pair. For the MLM task, 15% of the words are randomly masked in all the concept-based documents, and for each document, an upper bound for the number of masks is set. Then the BERT$_{BASE}$ model is trained to output the masked words rather than other possible words.

For the NSP task, the training objective is to learn the IS-A relationships between concepts: Given two concepts A and B, is B a child of A, or not. In Figure 6.5, two "sentences" *Skin finding*, and *Centrifugal rash* were extracted from the document for the focus concept *Centrifugal rash*. After preprocessing these two concepts (treated as two "sentences") as shown in the middle level, two token – "##ri" and "rash" were masked out. The BERT$_{BASE}$ model was trained to raise the probabilities of two correct tokens "##ri" and “rash” over other tokens in the vocabulary. In addition, as *Centrifugal rash* IS-A *Skin finding*, the BERT$_{\text{BASE}}$ model was also trained to raise the probability for the correct classification label “IsNext.” The obtained model is denoted as BERT$_{\text{BASE+SNO}}$ (SNO=SNOMED CT).

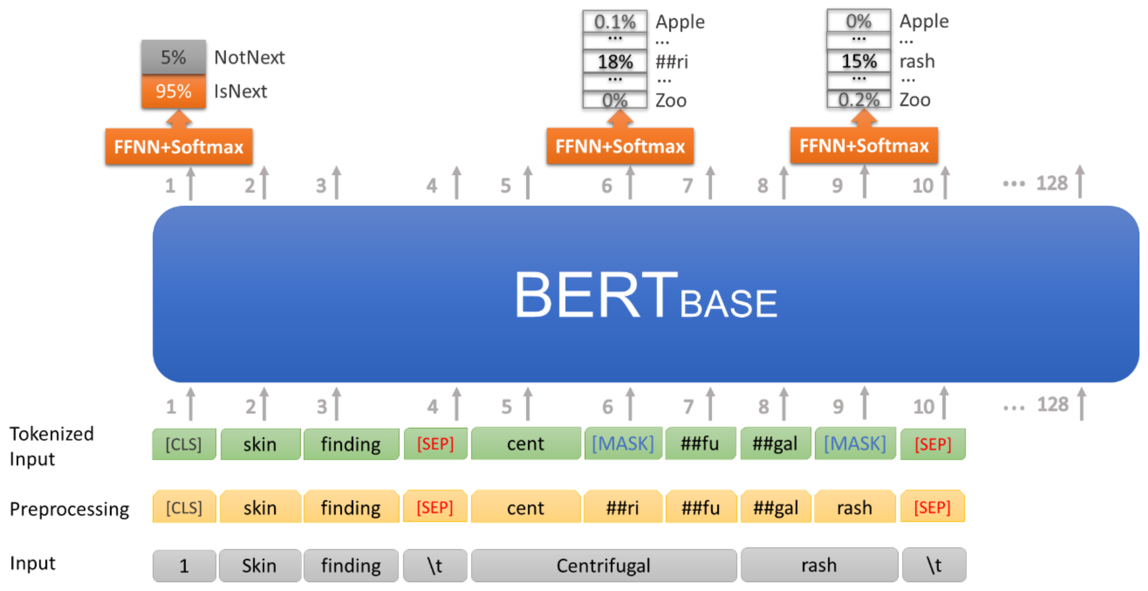


Figure 6.5 Pre-training the BERT$_{\text{BASE}}$ model with concept-based documents to obtain BERT$_{\text{BASE+SNO}}$ model. FFNN is short for Feedforward neural network.

The training parameters used for Pre-training are as follows: batch size = 64, sequence length =128, training steps = 200,000, learning rate =2e^{-5}, dropout rate = 0.1, and activation function = gelu (Gaussian error linear unit).

6.2 Fine-tuning two BERT Models as IS-A Relationship Classifiers

6.2.1 Data Preparation

To fine-tune a BERT model into an IS-A relationship classifier, the model needs to be trained with both IS-A connected concept pairs (positive instances) and concept pairs with no IS-A connections (non-IS-A concept pairs, in short, i.e., negative instances). The IS-A connected concept pairs are explicitly defined in the ontology’s hierarchy. Thus, the positive training data consists of all IS-A concept pairs in the *Clinical Finding* hierarchy. However, the selection of negative training data (non-IS-A concept pairs) is critical for the accuracy of the model. To compare the performance of models trained with and without the area taxonomy-based summarization technique, two sets of negative training data were prepared using the following two methods:

6.2.1.1 Negative training data from Hierarchy. In the previous study [121], for a CNN model, the non-IS-A pairs were limited only to nephew-uncle pairs in the same hierarchy. The rational was that a non-IS-A pair formed by two randomly sampled concepts that are likely completely unrelated concepts, will result in negative examples with a large semantic distance, that do not contribute to learning the “border surface” between positive and negative instances. Thus, for a given IS-A pair A *IS-A* B, it is more useful to learn differences between IS-A and non-IS-A pairs from a non-IS-A pair (A, C), where C is a “near miss,” close to the border surface. An uncle concept was used, i.e., a sibling of B, rather than an arbitrary concept D, which is likely not relevant to concept A, but still semantically close (see Figure 6.6). The training data including all IS-A pairs and nephew-uncle pairs from the same hierarchy will be referred to as **Hierarchy data**.

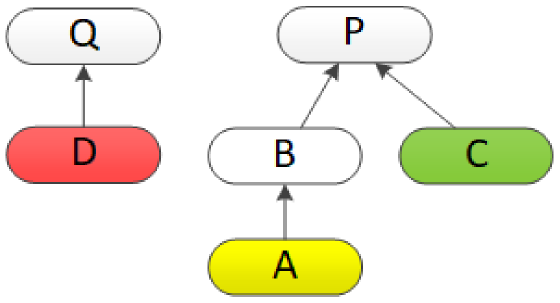


Figure 6.6 The nephew-uncle pair (A, C) (repeated from Figure 4.2 (a)).

6.2.1.2 Negative training data from Area Taxonomy. Utilizing the area taxonomy, the nephew-uncle pairs can be further divided into two types: uncle and nephew concepts are from the same area or uncle and nephew concepts are from different areas. A classification model can benefit more from training with nephew-uncle non-IS-A pairs from the same area, because in a pair from the same area the two concepts are more closely related than in a pair where the two concepts are in different areas. Thus, the classification model can better learn the features representing the subtle differences between IS-A pairs and nearby non-IS-A pairs. As a result, we hypothesized that the extracted features will enable the classification model to better verify whether a concept pair should be connected by an IS-A link or not, achieving better testing performance.

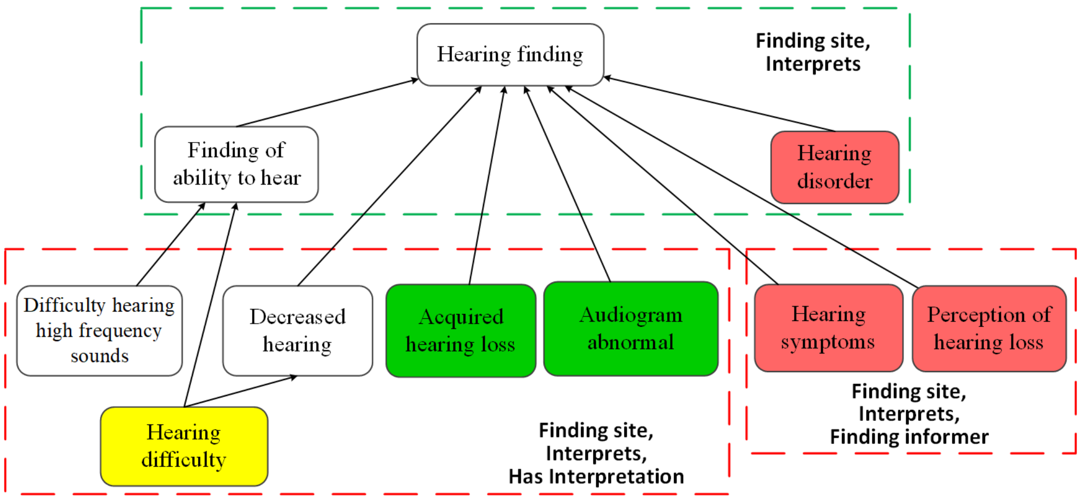

Hearing finding
Finding site, Interprets
Finding of ability to hear
Hearing disorder
Difficulty hearing high frequency sounds
Decreased hearing
Acquired hearing loss
Audiogram abnormal
Hearing symptoms
Perception of hearing loss
Finding site, Interprets, Finding informer
Hearing difficulty
Finding site, Interprets, Has Interpretation

Figure 6.7 Nephew-uncle pairs within/without the same area.

The above distinction is demonstrated with a concrete example (Figure 6.7) from the *Clinical Finding* hierarchy. Let the nephew concept be *Hearing difficulty* (in yellow). Its five uncle concepts are *Acquired hearing loss*, *Audiogram abnormal*, *Hearing symptoms*, *Perception of hearing loss*, and *Hearing disorder*. The first two concepts are the siblings of the concept *Decreased hearing*, which is a parent of *Hearing difficulty*, in the same area. The other three uncle concepts are siblings of the concept *Decreased hearing*, because they are all children of *Hearing finding* in different areas from *Hearing Difficulty*. As Figure 6.7 shows, the first two uncle concepts (in green) are from the area {*Finding site*, *Interprets*} that contains *Hearing difficulty*. In contrast, two uncle concepts *Hearing symptoms* and *Perception of hearing loss* (in red) are in the area {*Finding site*, *Interprets*, *Finding informer*}, while the other uncle concept *Hearing disorder* is in the area {*Finding site*, *Interprets*}. Both are in different areas than the nephew concept *Hearing difficulty*. *Hearing difficulty* is semantically more similar to *Acquired hearing loss,* and *Audiogram abnormal* from the same area as they are all various kinds of hearing findings similar to *Hearing difficulty*.

Hearing difficulty is less similar to *Hearing symptoms* and *Perception of hearing loss* in a different area, since they represent symptoms and the perception of hearing. Similarly, *Hearing difficulty* is also less similar to *Hearing disorder* in another area, which is a more general concept that is the root of a subhierarchy consisting of hearing disease concepts (not shown in the diagram). The training data including all IS-A pairs and nephew-uncle pairs from the same area is referred to as **Area Taxonomy data**, in contrast to the previously introduced Hierarchy data.

6.2.2 Training an IS-A Relationship Classifier

In each experiment, two independent classifiers (models) were trained using different data sets prepared with the above two techniques for performance comparison. In an ontology, there are more non-IS-A concept pairs (negative pairs) than IS-A concept pairs (positive pairs). To avoid an imbalanced training data issue, after the positive and negative pairs were extracted, the collection of negative pairs was randomly downsampled to the size of the positive pairs in each training round for both models. Then the dataset was divided according to a 90:10 ratio for training and validation, respectively.

Thus, the BERT$_{BASE+SNO}$ model was fine-tuned in the training phase to predict the correct labels for the IS-A concept pairs and the non-IS-A concept pairs, utilizing the NSP binary sentence-pair classification task. The sentence prediction capability of BERT$_{BASE+SNO}$ was employed and a *softmax* layer with categorical cross-entropy was added on top of it. The obtained model is denoted as BERT$_{BASE+SNO+CLF}$ (CLF = classifier), the model after fine-tuning.

To achieve this, the model and the classifier were trained at the same time to predict IS-A links between pairs of ontology concepts, i.e., the parameters of BERT$_{BASE+SNO}$ and the classifier were fine-tuned to maximize the log-probability of the correct label (IS-A or non-IS-A). This process is illustrated with the concept *Edema of wrist* as an example in Figure 6.8. The input "1 *Finding of wrist region (\t) Edema of wrist (\t)*" was converted as one training instance to "[CLS] finding of wrist region [SEP] ed ##ema of wrist [SEP]" with Class label = 1. Class 1 means that there should be an IS-A link between the two concepts, and Class 0 means that there shouldn't be such a link.

The BERT$_{BASE+SNO+CLF}$ model computes the probabilities for Class 0 and Class 1 and records the result as a 2-element vector. The label of the class with the higher probability is reported as the prediction output. The error between the predicted label and the true label was backpropagated through the model to improve the model's parameters. In the training, the default model hyperparameters were used in pre-trained BERT$_{BASE+SNO}$, with one exception, the number of training epochs was set to 6. This is an empirical value, based on the fact that the authors of BERT recommend 3 epochs for light fine-tuning with a relatively small dataset. We chose to double this number as our dataset is larger than the dataset they refer to.

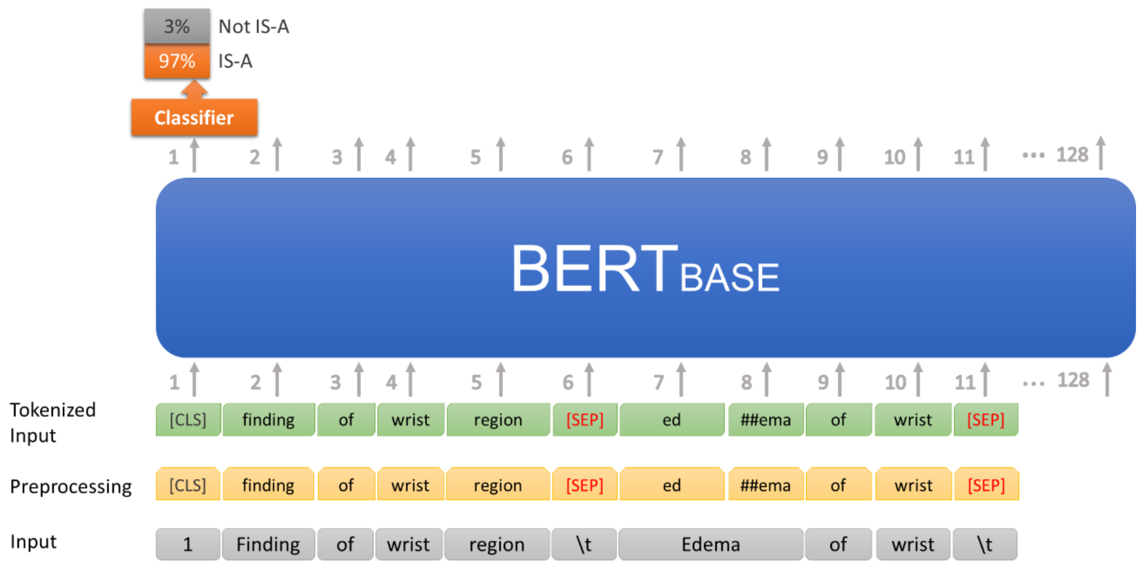


Figure 6.8 Fine-tuning the BERT$_{\text{BASE+SNO}}$ model with concept pairs to obtain BERT$_{\text{BASE+SNO+CLF}}$ model.

6.3 Performance Comparison between two Classifiers

6.3.1 Test with New Clinical Finding Data

To evaluate the BERT$_{\text{BASE+SNO+CLF}}$ models on previously unseen data, separate test tasks were created, using new concepts from the *Clinical finding* hierarchy of the January 2018 release. For each new concept that was added to the *Clinical finding* hierarchy in this release, both positive and negative samples were prepared for testing. To obtain positive testing sample instances, each new concept and its parents were extracted as IS-A concept pairs. For example, *Lesion of left ear* has two parents *Disorder of left ear* and *Ear lesion*. The corresponding positive testing instances are "*Disorder of left ear* (\t) *Lesion of left ear*" and "*Ear lesion* (\t) *Lesion of left ear*" with the class label = 1 (true).

The negative sample instances are not limited to nephew-uncle pairs, but general combinations of non-IS-A concept pairs. Thus, to obtain negative testing instances, each new concept was paired with a randomly chosen concept from the other new concepts' parents as non-IS-A concept pairs. For example, *Disorder of soft tissue of upper limb*, which is the parent of *Congenital trigger finger of right hand*, was selected and paired with *Lesion of left ear* to form a testing instance "*Disorder of soft tissue of upper limb* (\t) *Lesion of left ear*" with the label = 0.

All the testing concept pairs were randomly shuffled, divided into batches and sent to the trained $\text{BERT}_{\text{BASE+SNO+CLF}}$ models for prediction. The tested models use the previously learned weights to process each input pair and return a class label (0 or 1) as prediction result. Label 1 is correct for a positive testing instance, indicating the existence of an IS-A link in the new SNOMED CT release. In other words, the existence of an IS-A link in the new release of SNOMED CT is correctly predicted. For a negative testing instance, there should be no IS-A link between these two concepts, so label 0 would be correct. The prediction accuracy was calculated in terms of Precision, Recall, F1 and F2 scores by comparing the labels predicted by the tested model with the ground-truth labels.

6.3.2 Results

The prediction results of the two models, one trained with hierarchy data (referred to as Hierarchy model) and the other one trained with area taxonomy data (referred to as Area Taxonomy model), will be reported in this subsection. The testing instances (concept pairs) were extracted from the *Clinical Finding* hierarchy of the SNOMED CT 2018 January release, which were not included in the training data set. In each experiment, the two trained models were tested using the same testing sample. Besides the typical metrics Precision, Recall and F1, another metric called F2 was used. The F2 score is calculated from the generalized F score F_β where

$$F_\beta = (1 + \beta^2) * \frac{\text{precision} * \text{recall}}{(\beta^2 * \text{precision}) + \text{recall}} \quad \textbf{(6-1)}$$

with $\beta = 2$. The value of β is set to 2, to emphasize that recall is considered more important than precision in this task. This experiment was repeated ten times and the Precision (P), Recall (R), F1, and F2 scores for the corresponding ten tests are presented in Table 6.1. Each model was tested against 8,574 pairs (4,287 positives and 4,287 negatives). For example, in Test 5 for IS-A classification, the Precision is 0.83, Recall is 0.93, the F1 score is 0.88, and the F2 score is 0.91 for the Hierarchy model. When testing the Area Taxonomy model, Precision is 0.81, Recall is 0.96, F1 is 0.88, and F2 is 0.93. The Precision score dropped from 0.83 to 0.81 while the Recall score increased from 0.93 to 0.96, improving by 3%. For Non-IS-A tests, the Recall score dropped from 0.81 to 0.78 while the Precision score increased from 0.93 to 0.95.

Table 6.1 Precision (P), Recall (R), F1, and F2 Scores for Ten Experiments of *Clinical Finding* Hierarchy

Test	**IS-A Classification**								**Non-IS-A Classification**							
	Hierarchy				**Area Taxonomy**				**Hierarchy**				**Area Taxonomy**			
No.	P	R	F1	F2	P	R	F1	F2	P	R	F1	F2	P	R	F1	F2
1	0.83	0.94	0.88	0.92	0.79	0.95	0.87	0.91	0.93	0.81	0.87	0.83	0.94	0.75	0.84	0.78
2	0.84	0.93	0.88	0.91	0.8	0.96	0.87	0.92	0.93	0.82	0.87	0.84	0.95	0.77	0.85	0.80
3	0.85	0.94	0.9	0.92	0.8	0.96	0.87	0.92	0.94	0.84	0.88	0.86	0.95	0.76	0.84	0.79
4	0.87	0.94	0.9	0.93	0.8	0.96	0.87	0.92	0.93	0.86	0.9	0.87	0.95	0.76	0.85	0.79
5	0.83	0.93	0.88	0.91	0.81	0.96	0.88	0.93	0.93	0.81	0.87	0.83	0.95	0.78	0.85	0.81
6	0.86	0.94	0.9	0.92	0.81	0.96	0.88	0.93	0.93	0.85	0.89	0.86	0.95	0.78	0.86	0.81
7	0.85	0.94	0.89	0.92	0.79	0.96	0.87	0.92	0.93	0.84	0.88	0.86	0.95	0.75	0.84	0.78
8	0.84	0.94	0.89	0.92	0.8	0.96	0.87	0.92	0.93	0.83	0.88	0.85	0.95	0.76	0.84	0.79
9	0.83	0.94	0.89	0.92	0.8	0.96	0.87	0.92	0.94	0.81	0.87	0.83	0.95	0.76	0.84	0.79
10	0.87	0.94	0.9	0.93	0.8	0.96	0.87	0.92	0.93	0.85	0.89	0.86	0.95	0.75	0.84	0.78
Average	0.85	**0.94**	0.89	0.92	0.80	**0.96**	0.87	0.92	0.93	0.83	0.88	0.85	0.95	0.76	0.85	0.79
Standard Deviation	0.02	0.00	0.01	0.01	0.01	0.00	0.00	0.00	0.00	0.02	0.01	0.02	0.00	0.01	0.01	0.01

Comparing the average of ten experiments between the Hierarchy model and the Area Taxonomy model shows that the Area Taxonomy model improves the Recall from 0.94 to 0.96 at the cost of the Precision dropping from 0.85 to 0.80, and the F1 score drops from 0.89 to 0.87, while the F2 score remains the same at 0.92.

Regarding the prediction of IS-A links for new concepts, ten examples of the two models' prediction results (Table 6.2) were shown for ten pairs for which the second concept was newly added to SNOMED CT's *Clinical finding* hierarchy in the 2018 January release. The first five examples are IS-A connected concept pairs, which is indicated by the value 1 in the True Label column. The other five examples are synthesized non-IS-A concept pairs, indicated in the True Label column by 0.

Table 6.2 Prediction Results of Two Models on Five IS-A & Five non-IS-A Examples from *Clinical Finding* Hierarchy. Green fill indicates that the model correctly predicted the True Label

Index	**Test Concept**	**New Concept**	**True Label**	**Hierarchy model prediction**	**Area Taxonomy prediction**
1	*Visual cortex injury*	*Injury of right visual cortex*	1	🟩 1	🟩 1
2	*Drug therapy finding*	*Has supply of rescue medication*	1	🟩 1	🟩 1
3	*Cerebrovascular disease*	*Occlusion of left pontine artery*	1	🟩 1	🟩 1
4	*Decreased hearing*	*Congenital conductive hearing loss*	1	🟥 0	🟩 1
5	*Congenital anomaly of fetus*	*Malformation of central nervous system of fetus*	1	🟩 1	🟩 1
6	*Disorder of bilateral ulnar nerves*	*Loss of tissue of right eye co-occurrent with laceration*	0	🟩 0	🟩 0
7	*Gastric ulcer*	*Complex burn of wrist*	0	🟩 0	🟩 0
8	*Occlusion of left cerebellar artery*	*Dissection of basilar artery*	0	🟩 0	🟩 0
9	*Osteomyelitis of right ankle*	*Bone cyst of right foot*	0	🟥 1	🟥 1
10	*Injury of toe*	*Open wound of left foot*	0	🟥 1	🟩 0

For each test, one *Test Concept* was paired with one *New Concept* as one test instance, then we let the BERT model predict IS-A links between them. For instance, for

Example 3, *Cerebrovascular disease* was chosen as the test concept and paired it with the new concept *Occlusion of left pontine artery*. Then the task became to predict whether there is an IS-A link between the two concepts. Both the Hierarchy model and the Area Taxonomy model returned the correct label (=1). Correct predictions are marked in green. In Example 4, the Hierarchy model is wrong, and the Area Taxonomy model is correct that *Congenital conductive hearing loss* IS-A *Decreased hearing*. Both models are wrong about *Bone cyst of right foot,* because it is not an *Osteomyelitis of right ankle* (Example 9), thus they are marked in red.

The procedure to reproduce the results of this chapter is described in the APPENDIX.

CHAPTER 7

USING EMBEDDING SIMILARITY FOR TERMINOLOGY PLACEMENT

This chapter introduces and tests a similarity-based approach to identify parent(s) for a new concept in order to properly place it into a terminology. We rank a list of parent candidate concepts for placing a new concept, based on the similarity score of the vector representation of the concept names without manual feature engineering. Each of the candidate parents is paired with the new concept. The similarity score is calculated using the cosine distance between the vector representations of the two concepts of the pair. In addition, syntax rules are utilized to filter out similar concepts that have no potential of being parents of the new concept. Curators will need to manually scan the algorithmically prepared list of candidates, easily eliminating the improper candidates, to obtain all or most parents of each new concept.

Using the concept level embeddings and a post-processing method, we validated our approach with the 913 new concepts added to the *Procedure* hierarchy of SNOMED CT in the 2018 January release. The result shows that for new concepts, our method found 43.35% of all the parents. The Machine Learning Doc2vec [63] technology with the similarity measure enables to automatically provide a short candidate list, capturing a sizable percentage of the parents of the new concepts.

7.1 Method

We previously observed that the names of parent and a child concepts are often similar. This similarity is reflected in the Doc2vec embeddings. Calculating the similarity of vectors of concept names can be utilized to identify linguistically similar concepts. In general, we expect the parents of a new concept to be concepts from the previous release (in brief: old concepts). However, some new concepts have parents that are also new concepts (Figure 7.1). This happens when a new release contains a batch of new concepts on the same subject.

In this chapter, we harness the vector representations of the concept names for recommending candidate parents for a new concept. New concepts can be divided into three categories based on the distribution of a new concept's parent(s) across two consecutive releases (Figure 7.1), the previous release and the new release. A new concept that only has parent(s) in the old release is assigned to Category I. A new concept that only has parent(s) in the new release is assigned to Category II. A new concept that has parents in both the old release and the new release is assigned to Category III. We list examples for all three categories from the SNOMED 2018 January release (new release).

The new concept *Open biopsy of adrenal gland* belongs to Category I, because all of its parents—*Surgical biopsy of adrenal gland* and *Open biopsy* are in the old release. The new concept *Revision of left total hip arthroplasty* belongs to Category II since its parent *Revision of total hip arthroplasty* is a new concept as well. The new concept *Magnetic resonance imaging of bilateral hands* belongs to Category III, because it has two parents *Magnetic resonance imaging of left hand* and *Magnetic resonance imaging of right hand* in the old release and one parent *Magnetic resonance imaging of bilateral upper limbs* in the new release. Among 913 new concepts added into the *Procedure* hierarchy of the new release, there are 624, 92, and 197 concepts in Categories I, II, and III, respectively. The total number of actual child-parent concept pairs is 1954.

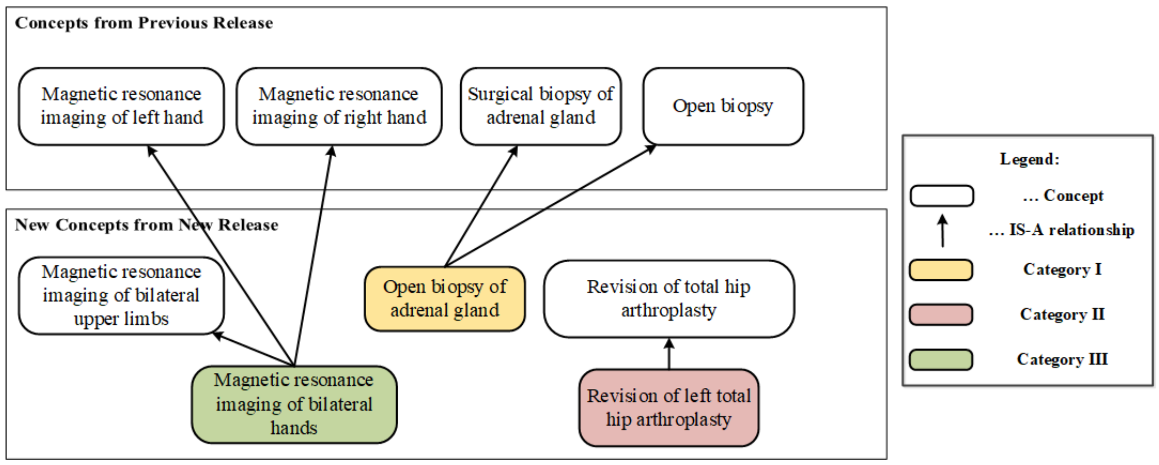

Concepts from Previous Release
Magnetic resonance imaging of left hand
Magnetic resonance imaging of right hand
Surgical biopsy of adrenal gland
Open biopsy
New Concepts from New Release
Magnetic resonance imaging of bilateral upper limbs
Open biopsy of adrenal gland
Revision of total hip arthroplasty
Magnetic resonance imaging of bilateral hands
Revision of left total hip arthroplasty
Legend:
... Concept
... IS-A relationship
Category I
Category II
Category III

Figure 7.1 Examples of new concepts that belong to the three categories.

Initially we considered identifying parents in the old release (Category 1 and part of Category 3) by training with the old concepts and identifying parents in the new release (Category 2) by training with the new concepts. We would obtain two sorted similarity lists containing old and new candidate parents, respectively, and then merge the two lists by similarity score in a descending order. However, we then realized that we could instead train a model, in an integrated algorithm, with the concepts in the new release, which contains both the new and old concepts, obtaining just one similarity list of candidate parents. Two algorithms were tested.

7.1.1 Algorithm 1

Algorithm 1 (Figure 7.2) relies exclusively on concept similarity (cosine similarity between embeddings) to obtain the N most similar child-parent pairs. We first preprocess the names of all concepts from the new release. (At this stage, before inserting the new concepts, the new release does not exist, but the *list* of new concepts does exist and together with all the concepts of the old release, the combined list consists of all the concepts of the new release).

Each concept's preprocessed name is treated as a document. We train a Doc2vec embedding model using all those documents to obtain the embeddings for all concepts.

For the new concept X, we query the pre-trained Doc2vec embedding model to retrieve its vector. A new concept's vector is used to calculate its cosine similarity to each of the other concepts' vectors. We select N candidate concepts that are most similar to the new concept, ranking them by their similarity scores in descending order. We select the top M concepts with the highest similarity scores. Each candidate concept of the M concepts is paired as a candidate parent with the new concept X. These concept pairs constitute the result.

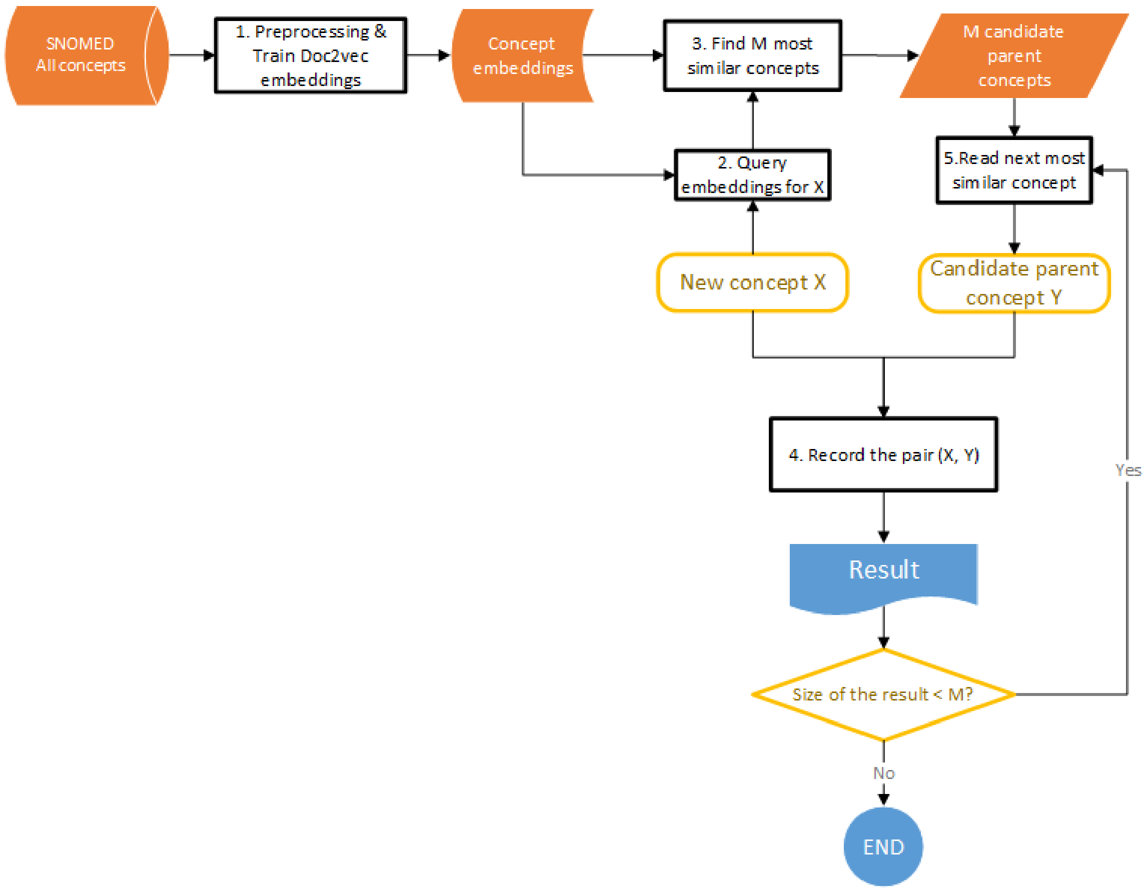

SNOMED All concepts
1. Preprocessing & Train Doc2vec embeddings
Concept embeddings
3. Find M most similar concepts
M candidate parent concepts
2. Query embeddings for X
5.Read next most similar concept
New concept X
Candidate parent concept Y
4. Record the pair (X, Y)
Yes
Result
Size of the result < M?
No
END

Figure 7.2 Flowchart of Algorithm 1 to generate M candidate child-parent pairs for a new concept X.

7.1.2 Algorithm 2

To improve the performance of Algorithm 1, we included two rules that modify it, resulting in Algorithm 2. We observed that when the child and parent's multi-word names are similar, the child's name is typically more detailed than the parent's, making the parent name's set of words a subset of the set of words in the name of the child. For example, *Repair of paraumbilical hernia* is a parent of *Laparoscopic repair of paraumbilical hernia with suture*. We utilize this observation as follows:

Rule 1: If the set of words of the candidate parent concept is not a proper subset of the set of words of the new concept, then discard this candidate parent.

Sometimes the most similar concept of a new concept is a **bilateral sibling**. For example, the most similar concept for *Magnetic resonance imaging of left knee with contrast* is *Magnetic resonance imaging of right knee with contrast*. Such concepts will be in the first positions of the result list but are not parents. Rule 1 eliminates such concepts. However, it may also delete legitimate parents from the candidate list. For example, *Ultrasonography of bilateral popliteal fossa* is a child of *Ultrasonography of left popliteal fossa*. In this case, the names are identical except that "bilateral" in the child is replaced by "left" in the parent. To override such exclusions, we apply Rule 2 before applying Rule 1.

Rule 2: If the word "bilateral" is in the set of words of the new concept and "left" or "right" is in the set of words of a candidate parent, then Rule 1 is applied only after removing "left" or "right" from the set of words of the parent.

An example of Rule 2 is as follows: The concept *Magnetic resonance imaging of bilateral elbows* is converted to Set 1 {"magnetic", "resonance", "image", "bilateral", “elbow”}. One of its parents, *Magnetic resonance imaging of left elbows*, after being converted to Set 2 is {“magnetic”, “resonance”, “image”, “left”, “elbow”}. Before applying Rule 1, we remove “left” from Set 2. Then Set 2 is a subset of Set 1, so Rule 1 is not activated.

To obtain an accurate word set for a concept, we use the Natural Language Toolkit (NLTK) [125]. The concept’s name is first tokenized by word_tokenize() provided by NLTK to get a list of words. Then any stop word is removed from the list. The WordNet lemmatizer then transforms each word into its normal form. Next, duplicate words are removed from the list. Finally, we check whether Set 2 is a subset of Set 1.

Figure 7.3 shows the flowchart of Algorithm 2, which modifies Algorithm 1 to include Rules 1 and 2 to accumulate M candidate pairs after discarding some improper candidates. We concentrate on the roles of the two rules skipping the explanations of components that are identical to Algorithm 1. After we obtain N most similar concepts as a candidate list for a new concept X, we test each concept in that candidate list. For each candidate concept, we pair it with the new concept. Set 1 (Set 2) is a word set of the name of the new concept (candidate parent concept). The flowchart applies Rule 2, which is an exception check, before applying Rule 1. If Set 2 is a subset of Set 1, this concept pair is kept in the result list. If not, we remove this concept pair. If the result size is smaller than M, where M is a pre-selected threshold, we move on to the next candidate concept in the list. This process is repeated until M child-parent pairs have been found; they are returned. However, if there are fewer than M concept pairs after iterating through all N candidate parent concepts, we return the recorded pairs.

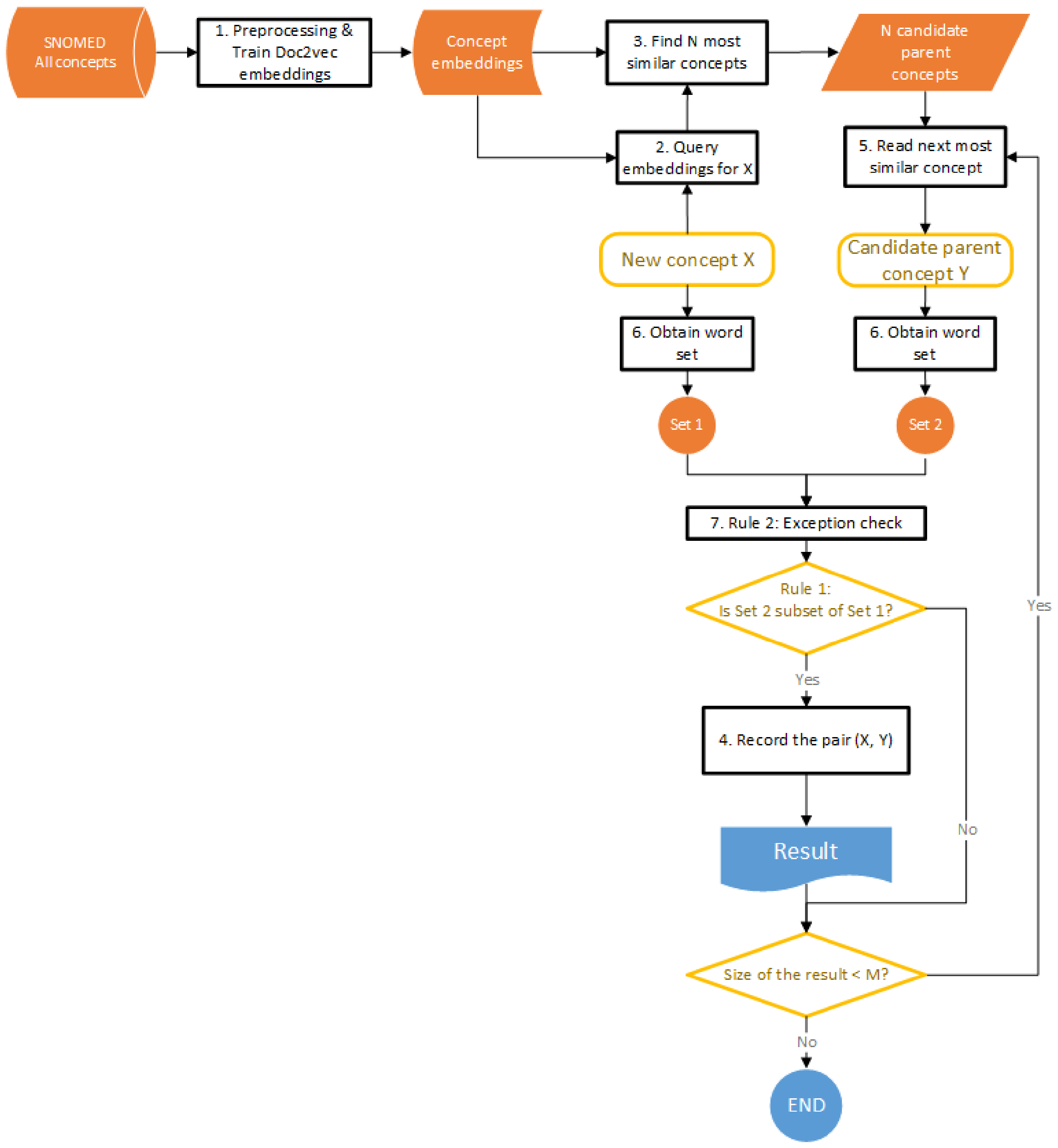

SNOMED
All concepts
1. Preprocessing & Train Doc2vec embeddings
Concept embeddings
3. Find N most similar concepts
N candidate parent concepts
2. Query embeddings for X
5. Read next most similar concept
New concept X
Candidate parent concept Y
6. Obtain word set
6. Obtain word set
Set 1
Set 2
7. Rule 2: Exception check
Rule 1: Is Set 2 subset of Set 1?
Yes
4. Record the pair (X, Y)
Result
No
Yes
Size of the result < M?
No
END

Figure 7.3 Flowchart of Algorithm 2 to generate up to M child-parent pairs for new concept X.

7.1.3 Evaluation Procedure

The child-parent pairs of our methods are compared with the child-parent pairs in the new release, which serve as a gold standard. The performance of our approach is evaluated by the number of child-parent pairs in the new release identified by our method. The recall is the ratio of the new release child-parent pairs identified by the algorithms. The precision is the ratio of identified child-parent pairs in the new release out of the candidate pairs submitted by the algorithm. We use a parameter N for the size of the list of concepts similar to the new concept and M for the maximum number of recommended parents. Since the recall grows and the precision declines with M, the choice of an optimal M can be made by maximizing the F1 measure combining recall and precision.

7.1.4 Preprocessing

We apply three preprocessing steps to each concept: Text normalization (e.g., Excision of Bartholin's gland → excision of bartholin's gland), punctuation splitting (e.g., excision of bartholin's gland, → excision of bartholin ' s gland), and word piece tokenization (excision of bartholin ' s gland, → ex ##cision of bart ##hol ##in ' s gland). This mechanism is adopted from the BERT model [86]. It employs WordPiece embeddings [126] with a 30,522 token vocabulary. Tokens are from a fixed size vocabulary that Google used for training, namely the concatenation of the BooksCorpus [123] and English Wikipedia.

7.1.5 Train Doc2vec Embeddings

Doc2vec is trained with all the concepts in the *Procedure* hierarchy in combination with the distributed bag of words (PV-DBOW) model, which has the advantage of embedding similarity in the vector representations of terms. This is desirable since similar biological entities are often named in similar ways. We employed Gensim [116] to train Doc2vec with the following hyperparameters: Vector size = 512, window size = 5, epochs = 100, workers = 4 cores, alpha=0.025, min alpha=0.00001, negative sampling = 10.

7.1.6 Similarity Score

The similarity between two vectors X and Y, produced by Doc2Vec, is computed with the cosine similarity measure as $Cos_{sim}(X,Y) = \frac{X \cdot Y}{\|X\|\|Y\|}$, where X · Y is the scalar product of X and Y, and $\|X\|$ is the Euclidean norm of vector X.

7.2 Results

We tested our methodology with the *Procedure* hierarchy of the SNOMED July 2017 and January 2018 releases. There were 913 new concepts in the latter. Of these, 624 have parents only in the previous release (Category I); 92 concepts have parents only in the new release (Category II); and 197 concepts have parents in both releases (Category III). Out of the 1954 new child-parent pairs, 454 pairs are internal pairs.

Table 7.1 Precision, Recall, and F1 of Algorithm 1 and Algorithm 2 for Ten Testes

	M	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
ALGORITHM 1	***Pairs found***	184	440	591	674	754	798	830	864	889	914
	Out of	913	1826	2739	3652	4565	5478	6391	7304	8217	9130
	Actual child-parent pairs	1954	1954	1954	1954	1954	1954	1954	1954	1954	1954
	Precision	0.202	0.241	0.216	0.185	0.165	0.146	0.130	0.118	0.108	0.100
	Recall	0.094	0.225	0.302	0.345	0.386	0.408	0.425	0.442	0.455	0.468
	F1	0.128	0.233	0.252	0.240	0.231	0.215	0.199	0.187	0.175	0.165
ALGORITHM 2	***Pairs found***	476	652	765	819	847	853	853	853	853	853
	Out of	782	1322	1699	1961	2124	2205	2254	2277	2290	2297
	Actual child-parent pairs	1954	1954	1954	1954	1954	1954	1954	1954	1954	1954
	Precision	0.609	0.493	0.450	0.418	0.399	0.387	0.378	0.375	0.372	0.371
	Recall	0.244	0.334	0.392	0.419	0.433	0.437	0.437	0.437	0.437	0.437
	F1	0.348	0.398	0.419	0.418	0.415	0.410	0.405	0.403	0.402	0.401

We employed Algorithms 1 and 2 to determine candidate parent concepts. The resulting pairs were compared with the actual child-parent pairs in the new release to calculate precision, recall, and F1 for both algorithms. For each concept, M is the number of parent candidates to be viewed by curators, and N is the number of most similar concepts in the list considered. We set N = 100. We executed Algorithm 1 and Algorithm 2 in ten tests with the value of M varying from 1 to 10 (Table 7.1).

Table 7.1 shows that when we applied Algorithm 1 with M=5, there were 4,565 (=913 * 5) child-parent pairs. Out of these pairs, 754 pairs are actual child-parent pairs. The precision, recall, and F1 are 0.165, 0.386, and 0.231, respectively. The maximum F1 score is achieved for M=3 (with precision = 0.216, recall = 0.302, F1 = 0.252), and the graphs (omitted) for precision and recall intersect between M=2 and M=3. When we applied Algorithm 2 with M=4, there were 1,961 child-parent pairs. Out of these pairs, 819 pairs are actual child-parent pairs. The precision, recall, and F1 are 0.418, 0.419, and 0.418, respectively. As shown in Figure 7.4, M=4 is where all three lines intersect, representing the optimal balance of precision and recall. The maximum F1 score is achieved at M=3 (with precision = 0.450, recall = 0.392, F1 = 0.419). Thus, either 3 or 4 can be used as the optimal value of M.

In our scenario, we believe that curators would be willing to pay the price of a slightly lower precision to identify substantially more child-parent pairs, because their priority is to automatically obtain the most likely child-parent pairs for reviewing while limiting their effort. Thus, we prefer the M value where the recall is high, and we still maintain a relatively high F1 score at the same time. Thus, we preferred M=5 over M=3 because 82 (= 847-765) more child-parent pairs were identified for the price of losing only

0.051 (=0.450-0.399) in precision and losing 0.004 (=0.419-0.415) in F1. In contrast for M=6, to gain 6 (=853-847) more child-parent pairs requires reviewing 81 (=2205-2124) more parent candidates. The 0.004 (=0.437-0.433) improvement in recall will not justify the additional effort required to achieve it.

Table 7.2 Result Comparison between Algorithm 1 and Algorithm 2 for M=5

M = 5, N = 100	Algorithm 1	Algorithm 2
	Integrated Search	Integrated Search
Pairs found	754	847
Out of	4565	2124
Real child-parent pairs	1954	1954
Precision	0.1652	0.3988
Recall	0.3859	0.4335
F1	0.2313	0.4154

To summarize, we compare in Table 7.2 the results of Algorithm 1 and Algorithm 2 for M=5, where the preferred Algorithm 2 is optimized. For Algorithm 2 we use N=100 and M=5, since this algorithm considers more candidates from the similarity list to replace improper candidates. We prefer a small M to save curators' time and effort, and an N which is large enough to include most of the actual parents with names that are similar to the new concept name. For fair comparison, allowing the same number of M recorded candidates to be reviewed by the curators, for Algorithm 1 N=M=5. For instance, using Algorithm 1 we recorded 913*5=4565 candidate pairs since no candidates are discarded and we found 754 actual child-parent pairs. Similarly, we identified 847 actual child-parent pairs out of 2124 recorded by Algorithm 2 (for some new concepts, fewer than five candidates are identified even when considering the 100 most similar concepts).

Comparing precision, recall and F1 in Table 7.2, Algorithm 2 is preferable, due to the savings in reviewing improper candidates. Thus, from now on we will discuss only Algorithm 2. Figure 7.4 shows precision, recall, and F1 of the ten results of Algorithm 2.

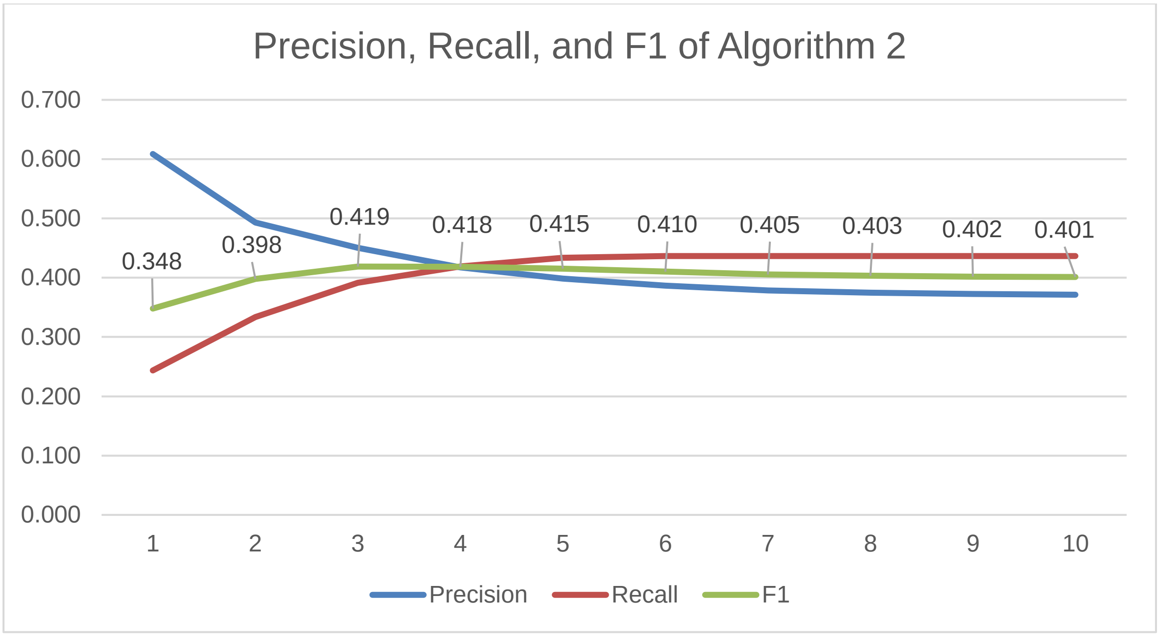


Figure 7.4 Precision, recall, and F1 of Algorithm 2 for M = 1 to 10. The F1 scores for the ten tests are labeled. At M=3 and M=4, the F1 scores are 0.419 and 0.418, the highest two scores among ten tests. Note that after M=5 the tangent of the recall chart declines.

Up to this point we considered the new release as a gold standard regarding the parents of the new concepts. However, in our extensive research on Quality Assurance of biomedical terminologies in general and of SNOMED CT in particular [17, 34, 35, 43, 127, 128], we have shown that SNOMED modeling is not perfect and modeling errors including IS-A relationship errors occur. Thus, domain expert (GE) studied a random sample of parents of 91 new concepts (10%). We were looking for two kinds of errors for the parents of new concepts in the 2018 release. One kind is missing parents, where Algorithm 2 identifies candidates that, by our judgement, are missing parents overlooked by the SNOMED curators. Hereby we refer to an IS-A relationship missing between the new concept and an existing concept in the July 2017 release. For example, our domain expert found that *screening for abuse* should be considered a parent of *screening for solvent abuse*. Thus, the existing parent *Screening procedure* should instead be a grandparent.

The second kind is wrong parents, where an “attributive phrase” in the child concept name is listed as parent, while the parent should be part of the “essential phrase” of the child name. We hypothesize that such mistakes might have occurred due to using a classifier algorithm without a subsequent manual review. *Left salpingo-oophorectomy* is a parent of *Total hysterectomy with left salpingo-oophorectomy* in the 2018 release, which we consider a mistake, since *with left salpingo-oophorectomy* is an attribute of the essential phrase *total hysterectomy*.

Out of the 188 pairs for the sample of 91 new concepts in the January 2018 release, the domain expert found 28 wrong parents, 6 of which should be ancestors rather than parents, due to more refined parents. Out of the 257 candidate pairs obtained by Algorithm 2 for the same 91 concepts, 23 are judged missing parents and should be added to SNOMED. We show a sample of review results in Table 7.3. For the new concept *Intralesional cryotherapy of lesion of skin*, *Cryotherapy to skin lesion* is its actual parent, while *Cryotherapy of skin* is one of its ancestors. Our expert found that *Biopsy of adrenal gland* should be a parent of *Open biopsy of adrenal gland,* missing in the new release. An example for a wrong parent is that *Biopsy of abdominal mass* should be an ancestor of *Fine needle aspiration biopsy of pseudocyst of pancreas using computed tomography guidance*, instead of a parent. Its proper parent should be another concept which is *Needle biopsy of mass of retroperitoneum using computed tomography guidance* in the same release, as it is modeled in the SNOMED July 2019 release. The reviewer further found that 95 candidate parent concepts are actual parents in the new release, 104 are ancestors, 29 are non-relevant concepts, and 6 are wrong parents among the 28 mentioned above.

Table 7.3 Sample of Review Results for Randomly Picked 91 New Concepts from *Procedure* Hierarchy

New concept	**Candidate parent**	**Parent (P)**	**Ancestor (A)**	**Missing parent (M)**	**Wrong parent (W)**
Intralesional cryotherapy of lesion of skin	Cryotherapy to skin lesion	**P**			
Intralesional cryotherapy of lesion of skin	Cryotherapy of skin		**A**		
Open biopsy of adrenal gland	Biopsy of adrenal gland			**M**	
Fine needle aspiration biopsy of pseudocyst of pancreas using computed tomography guidance	Biopsy of abdominal mass				**W**

Algorithm 2 fails to identify some correct parents, even though they appear in the similarity list. In Table 7.4, we listed ten examples of such errors. The reasons for their rejection fall into five categories: *Anatomy*, *Linguistic*, *Synonym*, *Ancestor*, and *Not Subset*. In the Anatomy error category, Algorithm 2 fails due to the anatomical variation between “elbows” and “upper limbs” (row 1), “tibia and fibula” and “lower limbs” (row 2).

For linguistic errors, the phrase differences between “of fetus” and “fetal” (row 3), and between “parent” and “parenting” (row 4) are the causes of the failures. The frequently interchanged use of synonyms also leads to problems such as “neoplasm” vs. “tumor” (row 5), and “ultrasonographic” vs. “ultrasonic” (row 6). Algorithm 2 failed to utilize the inherited hierarchical information between phrases. Thus, “computed tomography,” is a descendant of “imaging” (row 7), and “Fluoroscopy” subsumes “Fluoroscopic arthrography” (row 8). The last two errors are due to the phrase variations between the names of new concepts and candidate parent concepts, making them not comply with Rule 2 in Algorithm 2.

Table 7.4 Categorization of Parent Concepts Which Algorithm 2 Fails to Identify

	Category	**New concept**	**Missed Candidate Parent concept**
1	*Anatomy*	Computed tomography of bilateral elbows with contrast	Computed tomography of bilateral upper limbs with contrast
2	*Anatomy*	Computed tomography of bilateral tibia and fibula with contrast	Computed tomography of bilateral lower limbs with contrast
3	*Linguistic*	Two dimensional echocardiography of fetus	Fetal echocardiography
4	*Linguistic*	Parent education about sibling rivalry	Parenting education
5	*Synonym*	Single photon emission computed tomography using octreotide for localization of neoplasm	Single photon emission computed tomography of tumor
6	*Synonym*	Percutaneous renal needle biopsy using ultrasonographic guidance	Ultrasonic guidance for needle biopsy
7	*Ancestor*	Single photon emission computed tomography of cerebrospinal fluid flow	Cerebrospinal fluid flow imaging
8	*Ancestor & Anatomy*	Fluoroscopic arthrography of left elbow	Fluoroscopy of left upper limb
9	*Not Subset*	Insertion of cerebral ventriculoatrial shunt using fluoroscopic guidance	Creation of cerebral ventriculo-atrial shunt
10	*Not Subset*	Mammary ductography of bilateral breasts	Bilateral mammography

7.3 Discussion

Placement of new concepts into an ontology is difficult. It requires domain knowledge as well as ontology engineering [129] fluency. Curators utilize classifiers such as Snorocket [9] or HermiT [10], but the accuracy of the placement by such classifiers depends on accurate modeling of the attribute relationships of the new concepts and of existing similar concepts.

Thus, curators will likely embrace computational techniques offering accurate placement of a sizable ratio of new concepts. They will be left with only placing the remainder of the new concepts, as well as considering more parents for those placed. Furthermore, even if the computational technique will produce a short list of parent candidates for a new concept, a curator will prefer to review such a list and easily delete the concepts which are not proper parents, to end up with all or almost all actual parents of the new concepts. We offer such a technique in this chapter.

A limitation of this method is that for many child-parent pairs in biomedical ontologies the names of the two concepts are substantially different. *Amyotrophic lateral sclerosis* has no word in common with *Lou Gehrig's Disease*. Another challenge for our technique is constituted by cases where the names of pairs are similar, but their similarities are beyond the capability of Doc2vec to identify them by vector computation. For instance, *Complete axillary lymphadenectomy* is the parent of *Complete excision of lymph node group of left axilla*.

As mentioned before, for a random sample of 91 new concepts in 257 pairs obtained by Algorithm 2, our domain expert found 6 wrong child-parent pairs and 23 missing-parents. Assuming that the ratio of these errors is proportional for 913 new concepts in 2124 pairs, we should expect 50 wrong child-parent pairs and 190 missing-parents for Algorithm 2. This assumption would change the recall from 0.4335 to 0.5326 and the precision from 0.3988 to 0.4647.

Applying Rules 1 and 2 comes with a tradeoff. While eliminating many improper candidates from the candidate list, it also deletes a few proper pairs (Table 7.4). Algorithm 2 is able to compensate for the loss of such pairs by yielding 847 pairs versus the 754 of

Algorithm 1. When improper pairs are deleted from the list, the lower pairs in the list are used by Algorithm 2 to achieve M candidates. Even though Algorithm 2 considers candidates out of the 100 most similar concepts, we end up with an average of much fewer than M candidates, because most similar concepts are improper parents.

An interesting observation is that in the candidate lists produced by both algorithms, there are many pairs that are actually proper in the sense of being more general than the new concept. The reason they are improper *parents* is that they are not parents but grandparents or even great grandparents. Typically, a parent is more similar to the new concept and thus appears earlier in the list.

In general, we expect the parents of a new concept to be concepts in the previous release. However (Figure 7.1), some new concepts have parents that are also new. This is a common phenomenon when a group of new concepts are all about a specific subject. A set of such concepts typically has a tree or DAG structure with a unique root. This root may be the only concept with a parent in the previous release. Another option is that members of the set belong to Category 3.

CHAPTER 8

FUTURE WORK

The IS-A relationship classification between two concepts is equivalent to a binary classification problem. The multiple relationship classification between two concepts can be considered a multi-class classification problem. For ontology enrichment, multiple-relationship classification is defined as assigning relationship labels (including IS-A relationship) to pairs of concepts. For example, there are 17 lateral relationships in the *Clinical Finding* hierarchy, such as “finding site,” “associated morphology,” and “interprets,” and 29 lateral relationships in the *Procedure* hierarchy of SNOMED 2020 January release, such as “method,” “procedure site – direct,” and “component.” In the *Clinical Finding* hierarchy, there are 85,849 pairs of concepts that are connected through the “finding site” relationship, 60,091 pairs of “associated morphology,” and 36,939 pairs of “interprets.” In the *Procedure* hierarchy, there are 59,786 pairs of concepts that are connected through the “method” relationship, 33,278 pairs connected by “procedure site – direct,” and 8,414 pairs connected by “component.” For each lateral relationship, if the number of concept pairs connected by this relationship is sufficiently large for training, then this relationship can be included as one class label in the training data for a multi-class classifier. Both the CNN and BERT models used in this dissertation can be modified and trained to classify multiple relationships between concept pairs.

In this dissertation, fine-tuning the BERT model for IS-A relationship classification is based on the idea of Transfer Learning. The essence of transfer learning is to exploit knowledge gained from a pre-trained model and apply it to solve another problem. Hence, the quality, richness, and bias of the pre-trained model determines its compatibility with a downstream task. This chapter discuss three future ideas of conducting transfer learning research for BERT.

In this dissertation, the $\text{BERT}_{\text{BASE}}$ model (due to hardware limitations) was fine-tuned for an IS-A relationship classification task. In the future, the work will be extended to the $\text{BERT}_{\text{LARGE}}$ model, which has more parameters than the $\text{BERT}_{\text{BASE}}$ model and was proven to be more powerful in various NLP benchmark tests.

The $\text{BERT}_{\text{BASE}}$ model was trained with the concatenation of the BooksCorpus (800M words) [123] and English Wikipedia (2,500M words). It employs the WordPiece embeddings [126] with a 30,522 tokens vocabulary, which does not include most medical terms. Thus, medical terms that are not in the WordPiece vocabulary are split into word pieces denoted by ##. For example, “adenoid” is split into “aden” and “##oid.” The lack of medical terms in BERT’s vocabulary limited its applicability to support insertion of new concepts into a medical terminology such as SNOMED CT, and would probably impair other NLP tasks within the medical domain. In future work, the vocabulary will be expanded to include common medical terms selected from terminologies such as SNOMED CT. The BERT model pre-trained with medical terms can potentially improve its performance in some common NLP tasks in the medical domain, such as tagging and named entity recognition in EHRs.

Another possibility is to compare and provide insights into the different existing BERT models that are pre-trained with rich medical knowledge. For instance, BERT’s variants in the biomedical or clinical domains, BioBert [130], ClinicalBERT [131] and

NCBI BlueBERT [132], will be fine-tuned to compare their performance for different ontology-related tasks, including enrichment of medical ontologies.

CHAPTER 9

CONCLUSIONS

The comprehensive modeling and hierarchical positioning of a new concept in an ontology heavily relies on its set of proper subsumption relationships (IS-As) to other concepts. However, identifying a concept's IS-A relationships is a laborious task requiring curators to have both domain knowledge and terminology skills. This dissertation conducted four studies based on Machine Learning (ML), Natural Language Processing (NLP) models, and Abstraction Networks of ontologies to address the challenge of inserting new concepts into their proper positions in an ontology. The studies in this dissertation can be summarized under the following three main subjects:

1. The knowledge built into an ontology can be exploited to provide features for machine learning. For instance, concept embeddings can be learned directly from the names of a concept and its hierarchically related concepts. In addition, by converting the neighborhood network of a concept into "sentences," certain NLP models' capability of predicting the adjacency of two sentences (BERT) can be exploited for relationship classification between concepts.
2. Ontology summarization network, i.e., Abstraction Networks (AbNs), can be used to prepare high-quality training data because the restrictions imposed by AbNs can be used to better distinguish between IS-A connected concept pairs and non IS-A concept pairs, i.e., concept pairs that are not connected by IS-A relationships.
3. Some neural network-based ML models or transformer-based NLP models can be trained to verify and predict an IS-A relationship between a new concept and an existing concept, which will recommend the proper location of the new concept in the hierarchy.

To perform ontology enrichment automatically, the CNN model's classification capability and BERT's Next Sentence Prediction (NSP) capability were utilized to predict the presence of IS-A relationships between a new concept and existing concepts. The obtained automatic models can not only identify potential parents of a new concept, but also filter out irrelevant candidate concepts, reducing the number of improper placement choices for a concept. To augment these models' performance, the quality of training data matters and can be improved by various techniques including ontology summarization.

Chapter 3 presented a study which trained a Convolutional Neural Network (CNN) model to learn the likely locations of new IS-A links for the whole SNOMED CT. This approach was based on document embeddings for each concept in the SNOMED CT. The model was trained with data from the SNOMED CT 2017 July release and tested to ascertain its ability to verify IS-A relationships for new concepts added to the SNOMED CT 2018 January release. In testing, the trained CNN model achieved an average F1 score of 0.70, which demonstrates the utility of using the classification capability of a Machine Learning model to verify IS-A relationships between new concepts and pre-existing concepts.

The study in Chapter 4 suggested an improvement on Chapter 3 by using high quality, task-oriented training data. The CNN model, trained with data obtained from the area taxonomy, learned the subtle distinctions between IS-A pairs and closely related non-IS-A pairs. As a result, the derived CNN model was better at verifying whether a concept pair should be connected by an IS-A link or not, achieving better accuracy.

The study in Chapter 5 investigated whether an NLP model can be utilized instead of a CNN model for the automatic enrichment of terminologies. The study showed that a BERT model can be fine-tuned to verify the parent(s) of new concepts added to a terminology, employing its "next sentence prediction" capability. The test results of BERT models for the *Procedure* and *Clinical finding* hierarchies confirm that this technique is indeed able to verify the IS-A relationships from new concepts to their parents with a 0.80

F1 average value. When enhancing fine-tuning of BERT with pre-training, the average F1 score improved to 0.85. This work shows that it is possible to harness the power of the BERT model to differentiate between IS-A pairs and pairs not linked by IS-A relationship in an ontology.

To further improve the BERT model's performance, the study in Chapter 6 experimented with two approaches to refine both the data used for pre-training and for fine-tuning in Chapter 5. The data used for fine-tuning was prepared using the area taxonomy, employing the same method as in Chapter 4. The data used for pre-training followed a three-concept triple configuration, which was shown to be a superior way of modelling a concept's hierarchical information. The BERT model trained with these two enhancements achieved the same average *recall* score as in Chapter 5, while improving the average *precision* score from 0.79 to 0.85. This improvement shows the importance of accurate concept modeling through transformation from a concept's immediate hierarchical neighborhood to its corresponding text format.

Chapter 7 proposes a novel, unsupervised approach to recommend a set of concepts in a terminology as candidate parents for each new concept, based on concept name similarities. The similarity score is calculated using the cosine distance between the vector representations of the two concepts of the pair. In addition, syntax rules are utilized to filter out similar concepts that have no potential of being parents of a new concept. The approach was validated with the 913 new concepts added to the *Procedure* hierarchy of SNOMED CT in the January 2018 release. The result shows that for new concepts, our method found 43.35% of all the parents. In practice, curators can manually scan the short list of candidates prepared by our algorithm, easily eliminating any improper candidates, to obtain a large percentage of the correct parents of a new concept.

In this dissertation, two types of neural network models, i.e., a convolutional neural network (CNN) model and a BERT model, were trained to classify IS-A relationships between concept pairs. In the data preparation, an ontology summarization technique was successfully used to specify what types of concept pairs would provide better data for training. The system performance showed a significant improvement when an Abstraction Network was employed to constrain the training data.

From the performance reported in this dissertation, we conclude that the BERT model is superior to the CNN model for ontological relationship classification. Some potential reasons are as follows: first, the advantage of CNN is its automatic feature engineering, enabled by the convolution operations on the input data. It achieves better convolved results on image format data (2D convolution), since different features can be captured as the number of filters and layers increases. While text is often represented as embeddings (1-d, i.e., vectors), the features that can be identified through 1-d convolution are to a high degree determined by the feature window size, and are not proportional to the number of filters or layers as in 2-d or 3-d input data. Therefore, it is very difficult to capture the long-term dependencies that are frequently expressed in language or text. In addition, the quality of the embeddings determines the number of semantic or grammatical features that are preserved in the vector representations. Therefore, the performance of CNN is upper bounded by the quality of input embeddings and the efficacy of convolution operations.

BERT, on the contrary, is pre-trained with a large English corpus. It is already incorporating rich semantic and grammatical knowledge from general text, making it a more proper model for NLP related tasks, compared to a CNN model. Moreover, BERT's attention-based structure can maintain various granularities of attentions by controlling the number of attention heads and transformers in training. The attentions are computed using the whole input without losing the long-term dependencies in the context. Therefore, most knowledge and information from the original input is preserved with these attentions. The downstream applications, with simple fine-tuning, can easily achieve good performance by taking advantage of these attentions for classification or regression tasks, including the ontological relationship classification task discussed in this dissertation.

A trained CNN or BERT model can be used to predict IS-A relationships between new concepts and pre-existing concepts. These models can not only identify potential parents of a new concept, but also filter out unlikely parent concepts, reducing the number of improper placement choices for a concept. Ontology curators can benefit from such a model, since it will propose a higher ratio of proper parents for a given concept. Therefore, the work in this dissertation can save curators time and effort that would be needed to search for those parent candidates manually.

APPENDIX

TRANSFER LEARNING FROM BERT

This appendix describes the technical details of pre-training and fine-tuning BERT$_{BASE}$ model as an IS-A relationship classifier in order to reproduce the results reported in this dissertation.

1. Environment configuration:

    1.1. Hardware requirement (Recommended)

    ```
    NVIDIA RTX Titan with 24GB GDDR6 memory
    Driver version == 418.56
    CUDA version == 10.1
    ```

    1.2. Software requirement

    ```
    Python 3.6
    Keras==2.2.4
    matplotlib==3.0.2
    nltk==3.4.1
    numpy==1.16.4
    pandas==0.23.0
    scikit-learn==0.21.2
    tensorflow-gpu==1.13.1
    ```

2. Source code

    The source code is maintained on GitHub repository with the link below:

    https://github.com/hl395/Bert_Ontology

    The pre-trained BERT model, released by Google can be found at:

    https://github.com/google-research/bert

3. Dataset format

    3.1. Pre-training data format

In our task, the pre-training data is concept triples. Each triple contains three concepts, with one concept per line. An empty line is used to separate triples. In cases a focus concept has no parent or child, there are only two concepts in the corresponding triple. An example of pre-training data is as follows:

```
“
congenital anomaly of aorta
congenital stenosis of aorta
congenital supravalvular aortic stenosis

genodermatosis
inherited cutaneous hyperpigmentation
naegeli-franceschetti-jadassohn syndrome

hyperpigmentation of skin
inherited cutaneous hyperpigmentation
terminal osseous dysplasia and pigmentary defect syndrome
”
```

A short version of the example file for the pre-training data can be found at /data/pre_training_data_example.txt in the GitHub repository.

3.2. Fine-tuning data format

For fine-tuning, we need three files: “train.tsv” for training, “dev.tsv” for validation, and “test.tsv” for prediction. The “train.tsv” and “dev.tsv” files share the same format while the “test.tsv” is different by hiding the true labels.

To fine-tune BERT as IS-A relationship classifier, we extract IS-A connected concept pairs as positive training sample, and concept pairs that are not connected as negative training sample. Each concept pair is recorded as one string in one line, with the two concepts’ ids and names, and the IS-A label of this pair. The information is organized into five columns:

- Column “Quality” indicates the IS-A label between the two concepts.
- Column “#1 ID” represents the SNOMED ID of the first concept.
- Column “#2 ID” represents the SNOMED ID of the second concept.

- Column “#1 String” represents the SNOMED name of the first concept.
- Column “#2 String” represents the SNOMED name of the second concept.

Columns are separated using Tab as the delimiter. An example of the fine-tuning data is as follows:

```
“
Quality#1 ID   #2 ID   #1 String        #2 String
1        366054000     301976001     finding of fluorescein tear drainage
         fluorescein tear drainage impaired
0        295116004     295019008     allergy to chymotrypsin        allergy
to mannitol
”
```

A short version of the three example files, “train.tsv”, “dev.tsv”, and “test.tsv” for the pre-training data can be found at the “data” directory in the GitHub repository.

3.3. Test data format

To test the trained IS-A relationship classifier, we extract both IS-A connected concept pairs as positive testing sample, and concept pairs that are not connected as negative testing sample. Each concept pair is recorded as one string in one line, with the two concepts’ ids and names. The information is organized into five columns:

Column “Quality” indicates the IS-A label between the two concepts.
Column “#1 ID” represents the SNOMED ID of the first concept.
Column “#2 ID” represents the SNOMED ID of the second concept.
Column “#1 String” represents the SNOMED name of the first concept.
Column “#2 String” represents the SNOMED name of the second concept.

Columns are separated using Tab as the delimiter.

Note that the IS-A label of this pair is also included for evaluation simplicity. The true label is not visible or used in testing our classifier. An example of the fine-tuning data is as follows:

```
“
index   #1 ID   #2 ID   #1 String       #2 String
0       400186008       773629001       neoplasm of integumentary system
        onychomatricoma       1
1       298756009       316561000119102      finding of bone of upper limb
        osteophyte of left elbow        1
……
6734    51868009        735563002       duodenal ulcer disease          cicatrix
of middle ear   0
6735    762366009       735906001       prolapse of left eye co-occurrent
with lacerationeffects of water pressure        0
……
”
```

A short version of the example file for the pre-training data can be found at /data/test.tsv in the GitHub repository.

4. Execute program

4.1. Preparation

a) The program repository can be cloned using command:

“git clone https://github.com/hl395/Bert_Ontology.git”

The hardware compatibility and software requirement should be verified before executing the program.

b) Download the pre-trained BERT model, e.g. BERT$_{\text{BASE}}$ uncased model BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters from: https://storage.googleapis.com/bert_models/2018_10_18/uncased_L-12_H-768_A-12.zip

The downloaded BERT model should include the “vocab.txt” file and “bert_config.json” and three bert checkpoint files, “bert_model.ckpt.meta”, “bert_model.ckpt.index”, and “bert_model.ckpt.data-00000-of-00001”.

4.2. Pre-training

4.2.1. Create pre-training data

The parameters used to control this data creation are specified in the "creating_pretraining_data.py". The required parameters include:

```
FLAGS.input_file = "/path/to/pre-training_data_example.txt"
FLAGS.output_file = "/path/to/tf_examples.tfrecord"
FLAGS.vocab_file =
"/path/to/downloaded_BERT_model/vocab.txt"
FLAGS.do_lower_case = True
FLAGS.max_seq_length = 128
FLAGS.max_predictions_per_seq = 20
FLAGS.masked_lm_prob = 0.15
FLAGS.random_seed = 12345
FLAGS.dupe_factor = 5
```

The usage of parameters can be referred at the vanilla BERT GitHub page. To generate pre-training data, run "python creating_pretraining_data.py". After the pre-training data is generated, it is wrote to the output directory named by "tf_examples.tfrecord."

4.2.2. Run pre-training

The parameters used to control pre-training are specified in the "run_pretraining.py". The required parameters include:

FLAGS.input_file = "/path/to/**<u>tf_examples.tfrecord</u>**" (from 4.2.1)
FLAGS.output_dir = "/path/to/**<u>pre_trained_model_directory</u>**"
FLAGS.vocab_file =
"/path/to/downloaded_BERT_model/vocab.txt" (the same as in 4.2.1)
FLAGS.do_train = True (perform training)
FLAGS.do_eval = True (perform evaluation/validation)
FLAGS.bert_config_file =
"/path/to/downloaded_BERT_model/bert_config.json"
FLAGS.init_checkpoint =
"/path/to/downloaded_BERT_model/bert_model.ckpt"
FLAGS.train_batch_size = 64
FLAGS.max_seq_length = 128

```
FLAGS.max_predictions_per_seq = 20
FLAGS.num_train_steps = 100000
FLAGS.num_warmup_steps = 5000
FLAGS.save_checkpoints_steps = 20000
FLAGS.learning_rate = 2e-5
```

The usage of parameters can be referred at the vanilla BERT GitHub page.

To pre-training the downloaded BERT with our own corpus, run "python run_pretraining.py".

After pre-training the BERT model, the obtained new model is saved to the output directory in "/path/to/**<u>pre_trained_model_directory</u>**" where the value of "FLAGS.output_dir." Note that the obtained new model's name could vary depends on the number of training steps are used. However, the model still consists of three files and checkpoint file. An example using the parameters above will generate a model with three files as follows: "model.ckpt-100000.meta", "model.ckpt-100000.index", and "model.ckpt-100000.data-00000-of-00001" with the same number 100000 in their names as it is the value used as the number of training steps.

4.3. Fine-tuning

To fine-tune the obtained model from 4.2, we run "run_classifier_hao.py" with specifying the following required parameters:

FLAGS.bert_config_file = "/path/to/downloaded_BERT_model/bert_config.json"
FLAGS.vocab_file = "/path/to/downloaded_BERT_model/vocab.txt"
FLAGS.init_checkpoint = "/path/to/**pre_trained_model_directory**" (in 4.2)
FLAGS.data_dir = "/path/to/**fine_tuning_data_directory**/" (the directory that contains the both fine-tuning training, evaluation, and testing data)
FLAGS.output_dir = "/path/to/**fine_tuned_model_directory**/"
FLAGS.train_batch_size = 64
FLAGS.max_seq_length = 128
FLAGS.num_train_epochs = 3

```
FLAGS.do_train = True (perform training)
FLAGS.do_eval = True (perform evaluation/validation)
FLAGS.do_predict = True (perform prediction)
FLAGS.task_name = "MRPC"
```

The usage of parameters can be referred at the vanilla BERT GitHub page.

To fine-tune the pre-trained BERT with concept pairs, run "python run_classifier_hao.py".

After fine-tuning the BERT model as an IS-A relationship classifier, the obtained classifier is saved to the output directory in "/path/to/**fine_tuned_model_directory**" where the value of "FLAGS.output_dir" is specified.

4.4. Testing

The testing is performed after the model is fine-tuned in 4.3, because we turn on the flag for prediction by setting "FLAGS.do_predict = True." The fine-tuned model's prediction is saved in the "test_results.tsv" file in the directory of "/path/to/**fine_tuned_model_directory.**" Note that we only need to fine-tune the model once, and then use the obtained model to test on different testing data. This can be achieved by update the testing sample, i.e. "test.tsv" file with new testing data, and set the "FLAGS.do_train = False" and "FLAGS.do_eval = False."

4.5. Evaluation

For each testing concept pair, our model predicts the probabilities that the two concepts should be connected by IS-A and not, respectively. The results are recorded in the "test_results.tsv" file. Use "output/read_results.py" to read the true results and prediction results to evaluate the model's performance. The metrics used including

Precision, Recall, F1 and F2 scores, the micro average and macro average of these metrics are also calculated.

REFERENCES

[1] Rector AL, Qamar R, Marley T. Binding ontologies and coding systems to electronic health records and messages. Applied Ontology. 2009;4(1):51-69.

[2] Eichelberg M, Aden T, Riesmeier J, Dogac A, Laleci GB. A survey and analysis of electronic healthcare record standards. ACM Computing Surveys (CSUR). 2005;37(4):277-315.

[3] Giannangelo K, Berkowitz L. SNOMED CT helps drive EHR success. Journal of American Health Information Management Association. 2005;76(4):66-7.

[4] Rubin DL, Shah NH, Noy NF. Biomedical ontologies: a functional perspective. Briefings in Bioinformatics. 2008;9(1):75-90.

[5] Bodenreider O. Biomedical ontologies in action: role in knowledge management, data integration and decision support. Yearb Med Inform. 2008:67-79.

[6] Hastings J, Owen G, Dekker A, Ennis M, Kale N, Muthukrishnan V, *et al*. ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic Acids Res. 2016;44(D1):D1214-9.

[7] Whetzel PL, Noy NF, Shah NH, Alexander PR, Nyulas C, Tudorache T, *et al*. BioPortal: enhanced functionality via new Web services from the National Center for Biomedical Ontology to access and use ontologies in software applications. Nucleic Acids Res. 2011;39(Web Server issue):W541-5.

[8] de Coronado S, Haber MW, Sioutos N, Tuttle MS, Wright LW. NCI Thesaurus: using science-based terminology to integrate cancer research results. Stud Health Technol Inform. 2004;107(Pt 1):33-7.

[9] Jimenez AM, Lawley MJ. Snorocket 2.0: Concrete domains and concurrent classification. OWL Reasoner Evaluation Workshop (ORE). 2013.

[10] Shearer R, Motik B, Horrocks I. HermiT: A highly-efficient OWL reasoner. Owled; 2008;(p. 91).

[11] SNOMED CT [11/17/2019]. Available from: https://www.snomed.org/. (accessed on 4/2/2020).

[12] Arguello Casteleiro M, Maseda Fernandez D, Demetriou G, Read W, Fernandez-Prieto M, Des Diz J, *et al*. A case study on sepsis using PubMed and Deep Learning for ontology learning. Informatics for Health: Connected Citizen-Led Wellness and Population Health. 2017;235:516-20.

[13] Minarro-Giménez JA, Marin-Alonso O, Samwald M. Exploring the application of deep learning techniques on medical text corpora. Stud Health Technol Inform. 2014;205:584-8.

[14] Pembeci İ. Using word embeddings for ontology enrichment. International Journal of Intelligent Systems and Applications in Engineering. 2016;4(3):49-56.

[15] Hartel FW, de Coronado S, Dionne R, Fragoso G, Golbeck J. Modeling a description logic vocabulary for cancer research. Journal of biomedical informatics. 2005;38(2):114-29.

[16] Halper M, Gu H, Perl Y, Ochs C. Abstraction networks for terminologies: Supporting management of “Big Knowledge”. Artificial intelligence in medicine. 2015;64(1):1-16.

[17] Wang Y, Halper M, Min H, Perl Y, Chen Y, Spackman KA. Structural methodologies for auditing SNOMED. Journal of biomedical informatics. 2007;40(5):561-81.

[18] Min H, Perl Y, Chen Y, Halper M, Geller J, Wang Y. Auditing as part of the terminology design life cycle. J Am Med Inform Assoc. 2006;13(6):676-90.

[19] Ceci F, Pietrobon R, Gonçalves AL. Turning text into research networks: information retrieval and computational ontologies in the creation of scientific databases. PloS one. 2012;7(1):e27499.

[20] Asiaee AH, Minning T, Doshi P, Tarleton RL. A framework for ontology-based question answering with application to parasite immunology. Journal of biomedical semantics. 2015;6(1):31.

[21] Shen F, Lee Y. Knowledge discovery from biomedical ontologies in cross domains. PloS one. 2016;11(8):e0160005.

[22] Chepelev LL, Hastings J, Ennis M, Steinbeck C, Dumontier M. Self-organizing ontology of biochemically relevant small molecules. BMC bioinformatics. 2012;13(1):3.

[23] Manda P, Ozkan S, Wang H, McCarthy F, Bridges SM. Cross-ontology multi-level association rule mining in the gene ontology. PloS one. 2012;7(10):e47411.

[24] Lussier YA. Ontologies for natural language processing. Encyclopedia of Genetics, Genomics, Proteomics and Bioinformatics. 2005.

[25] Estival D, Nowak C, Zschorn A. Towards ontology-based natural language processing. Proceeedings of the Workshop on NLP and XML (NLPXML-2004). 2004:59-66.

[26] Ovchinnikova E. Integration of world knowledge for natural language understanding: Springer Science & Business Media; 2012.

[27] Brown PF, Desouza PV, Mercer RL, Pietra VJD, Lai JC. Class-based n-gram models of natural language. Computational linguistics. 1992;18(4):467-79.

[28] Zheng L, Perl Y, Elhanan G, Ochs C, Geller J, Halper M. Summarizing an Ontology: A" Big Knowledge" Coverage Approach. Stud Health Technol Inform. 2017;245:978-82.

[29] Ceusters W, Smith B, Kumar A, Dhaen C. Ontology-based error detection in SNOMED-CT. Stud Health Technol Inform. 2004;107(Pt 1):482-6.

[30] Ceusters W. Applying evolutionary terminology auditing to SNOMED CT. AMIA Annu Symp Proc; 2010;(p. 96).

[31] Zhang GQ, Bodenreider O. Large-scale, Exhaustive Lattice-based Structural Auditing of SNOMED CT. AMIA Annu Symp Proc. 2010;2010:922-6.

[32] Elhanan G, Perl Y, Geller J. A survey of SNOMED CT direct users, 2010: impressions and preferences regarding content and quality. J Am Med Inform Assoc. 2011;18 Suppl 1:i36-44.

[33] Agrawal A, Perl Y, Elhanan G. Identifying problematic concepts in SNOMED CT using a lexical approach. Stud Health Technol Inform. 2013;192:773-7.

[34] Wang Y, Halper M, Wei D, Gu H, Perl Y, Xu J, *et al*. Auditing complex concepts of SNOMED using a refined hierarchical abstraction network. Journal of biomedical informatics. 2012;45(1):1-14.

[35] Ochs C, Geller J, Perl Y, Chen Y, Xu J, Min H, *et al*. Scalable quality assurance for large SNOMED CT hierarchies using subject-based subtaxonomies. J Am Med Inform Assoc. 2014;22(3):507-18.

[36] Zhu X, Fan JW, Baorto DM, Weng C, Cimino JJ. A review of auditing methods applied to the content of controlled biomedical terminologies. Journal of biomedical informatics. 2009;42(3):413-25.

[37] Geller J, Perl Y, Halper M, Cornet R. Special issue on auditing of terminologies. Journal of biomedical informatics. 2009;42(3):407-11.

[38] Structural Analysis of Biomedical Ontologies Center (SABOC) [Available from: http://saboc.njit.edu/. (accessed on 4/2/2020).

[39] Ochs C, Perl Y, Halper M, Geller J, Lomax J. Quality assurance of the gene ontology using abstraction networks. Journal of bioinformatics and computational biology. 2016;14(3):1642001.

[40] Stearns MQ, Price C, Spackman KA, Wang AY. SNOMED clinical terms: overview of the development process and project status. AMIA Annu Symp Proc. 2001:662-6.

[41] Ashburner M, Ball CA, Blake JA, Botstein D, Butler H, Cherry JM, *et a*l. Gene ontology: tool for the unification of biology. The Gene Ontology Consortium. Nature genetics. 2000;25(1):25-9.

[42] Wang Y, Halper M, Wei D, Perl Y, Geller J. Abstraction of complex concepts with a refined partial-area taxonomy of SNOMED. Journal of biomedical informatics. 2012;45(1):15-29.

[43] Ochs C, Geller J, Perl Y, Chen Y, Agrawal A, Case JT, *et al*. A tribal abstraction network for SNOMED CT hierarchies without attribute relationships. J Am Med Inform Assoc. 2014;22(3):628-39.

[44] Ochs C, Agrawal A, Perl Y, Halper M, Tu SW, Carini S, *et al*. Deriving an abstraction network to support quality assurance in OCRe. AMIA Annu Symp Proc. 2012;2012:681-9.

[45] He Z, Ochs C, Agrawal A, Perl Y, Zeginis D, Tarabanis K, *et al*. A family-based framework for supporting quality assurance of biomedical ontologies in BioPortal. AMIA Annu Symp Proc. 2013;2013:581-90.

[46] Sim I, Tu SW, Carini S, Lehmann HP, Pollock BH, Peleg M, *et al*. The Ontology of Clinical Research (OCRe): an informatics foundation for the science of clinical research. Journal of biomedical informatics. 2014;52:78-91.

[47] Zeginis D, Hasnain, A., Loutas, N., *et al*. A collaborative methodology for developing a semantic model for interlinking Cancer Chemoprevention linked-data sources. Semantic Web. 2014;5(2):127-42.

[48] Ochs C, He, Z., Perl, Y., *et al*. Choosing the granularity of abstraction networks for orientation and quality assurance of the Sleep Domain Ontology. International Conference on Biomedical Ontology (ICBO). 2013:84-9.

[49] Arabandi S, Ogbuji C, Redline S, Chervin R, Boero J, Benca R, *et al*. Developing a Sleep Domain Ontology. Summit on Clinical Research Informatics. 2010.

[50] He Z, Ochs C, Soldatova L, Perl Y, Arabandi S, Geller J. Auditing redundant import in reuse of a top level ontology for the Drug Discovery Investigations Ontology. Vaccine and Drug Ontology Studies (VDOS) international workshop 2013.

[51] Qi D, King RD, Hopkins AL, Bickerton GR, Soldatova LN. An ontology for description of drug discovery investigations. Journal of integrative bioinformatics. 2010;7(3).

[52] Ochs C, Geller J, Perl Y, Musen MA. A unified software framework for deriving, visualizing, and exploring abstraction networks for ontologies. Journal of biomedical informatics. 2016;62:90-105.

[53] Petasis G, Karkaletsis V, Paliouras G, Krithara A, Zavitsanos E. Ontology population and enrichment: State of the art. Knowledge-driven multimedia information extraction and ontology evolution: Springer; 2011. p. 134-66.

[54] Astrakhantsev N, Turdakov DY. Automatic construction and enrichment of informal ontologies: A survey. Programming and computer software. 2013;39(1):34-42.

[55] Jayawardana V, Lakmal D, de Silva N, Perera AS, Sugathadasa K, Ayesha B. Deriving a representative vector for ontology classes with instance word vector embeddings. Seventh International Conference on Innovative Computing Technology (INTECH); 2017: IEEE;(p. 79-84).

[56] Zhang G-Q, Huang Y, Cui L. Can SNOMED CT changes be used as a surrogate standard for evaluating the performance of its auditing methods? AMIA Annu Symp Proc. 2017:1886-95.

[57] Pakhomov SV, Finley G, McEwan R, Wang Y, Melton GB. Corpus domain effects on distributional semantic modeling of medical terms. Bioinformatics. 2016;32(23):3635-44.

[58] McInnes BT, Pedersen T, Liu Y, Pakhomov SV, Melton GB. Using second-order vectors in a knowledge-based method for acronym disambiguation. Proceedings of the Fifteenth Conference on Computational Natural Language Learning; 2011: Association for Computational Linguistics;(p. 145-53).

[59] Rajani NF, Bornea M, Barker K. Stacking with auxiliary features for entity linking in the medical domain. BioNLP 2017. 2017:39-47.

[60] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:13013781. 2013.

[61] Hersh W, Buckley C, Leone T, Hickam D. OHSUMED: an interactive retrieval evaluation and new large test collection for research. SIGIR'94; 1994: Springer;(p. 192-201).

[62] Sajadi A. Graph-based domain-specific semantic relatedness from Wikipedia. Canadian Conference on Artificial Intelligence; 2014: Springer;(p. 381-6).

[63] Le QV, Mikolov T. Distributed Representations of Sentences and Documents. CoRR. 2014;abs/1405.4053.

[64] LeCun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015;521(7553):436.

[65] Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1; Lake Tahoe, Nevada. 2999257: Curran Associates Inc.; 2012. p. 1097-105.

[66] Graves A, Mohamed A-r, Hinton G. Speech recognition with deep recurrent neural networks. Acoustics, speech and signal processing (icassp); 2013: IEEE;(p. 6645-9).

[67] Lecun Y, Bottou L, Bengio Y, Haffner P. Gradient-based learning applied to document recognition. Proceedings of the IEEE. 1998;86(11):2278-324.

[68] Rawat W, Wang Z. Deep convolutional neural networks for image classification: A comprehensive review. Neural computation. 2017;29(9):2352-449.

[69] Convolutional Neural Networks (CNNs / ConvNets) [Available from: https://cs231n.github.io/convolutional-networks/. (accessed on 4/18/2020).

[70] LeCun Y, Boser BE, Denker JS, Henderson D, Howard RE, Hubbard WE, *et al*. Handwritten digit recognition with a back-propagation network. Advances in neural information processing systems; 1990;(p. 396-404).

[71] LeCun Y, Boser B, Denker JS, Henderson D, Howard RE, Hubbard W, *et al*. Backpropagation applied to handwritten zip code recognition. Neural computation. 1989;1(4):541-51.

[72] Ranzato MA, Huang FJ, Boureau Y-L, LeCun Y. Unsupervised learning of invariant feature hierarchies with applications to object recognition. IEEE conference on computer vision and pattern recognition; 2007;(p. 1-8).

[73] Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:12070580. 2012.

[74] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:14091556. 2014.

[75] Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. European conference on computer vision; 2014;(p. 818-33).

[76] Lin M, Chen Q, Yan S. Network in network. arXiv preprint arXiv:13124400. 2013.

[77] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, *et al*. Going deeper with convolutions2015: Cvpr.

[78] Xu B, Wang N, Chen T, Li M. Empirical evaluation of rectified activations in convolutional network. arXiv preprint arXiv:150500853. 2015.

[79] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. Proceedings of the 27th international conference on machine learning (ICML-10); 2010;(p. 807-14).

[80] Ng AY. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. Proceedings of the twenty-first international conference on Machine learning; 2004;(p. 78).

[81] Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:14085882. 2014.

[82] Lin Y, Shen S, Liu Z, Luan H, Sun M. Neural relation extraction with selective attention over instances. Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers); 2016;(p. 2124-33).

[83] Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems; 2013;(p. 3111-9).

[84] Pennington J, Socher R, Manning C. Glove: Global vectors for word representation. Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP); 2014;(p. 1532-43).

[85] Joulin A, Grave E, Bojanowski P, Douze M, Jégou H, Mikolov T. Fasttext. zip: Compressing text classification models. arXiv preprint arXiv:161203651. 2016.

[86] Devlin J, Chang M-W, Lee K, Toutanova K. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:181004805. 2018.

[87] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, *et al*. Attention is all you need. Advances in neural information processing systems. 2017:5998-6008.

[88] Mozer M. BERT does business: Implementing the BERT model for natural language processing at Wayfair 2019 [updated 11/27/2019; cited 2020 3/11]. Available from: https://tech.wayfair.com/data-science/2019/11/bert-does-business-implementing-the-bert-model-for-natural-language-processing-at-wayfair/. (accessed on 3/11/2020).

[89] Britz D, Goldie A, Luong M-T, Le Q. Massive exploration of neural machine translation architectures. arXiv preprint arXiv:170303906. 2017.

[90] Ba JL, Kiros JR, Hinton GE. Layer normalization. arXiv preprint arXiv:160706450. 2016.

[91] Sang EF, De Meulder F. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. arXiv preprint cs/0306050. 2003.

[92] Rajpurkar P, Zhang J, Lopyrev K, Liang P. Squad: 100,000+ questions for machine comprehension of text. arXiv preprint arXiv:160605250. 2016.

[93] Socher R, Perelygin A, Wu J, Chuang J, Manning CD, Ng A, *et al*. Recursive deep models for semantic compositionality over a sentiment treebank. Proceedings of the 2013 conference on empirical methods in natural language processing; 2013;(p. 1631-42).

[94] Tian F, Dai H, Bian J, Gao B, Zhang R, Chen E, *et al*. A probabilistic model for learning multi-prototype word embeddings. Proceedings of COLING; 2014: the 25th International Conference on Computational Linguistics: Technical Papers;(p. 151-60).

[95] Hedderich MA, Yates A, Klakow D, de Melo G. Using multi-sense vector embeddings for reverse dictionaries. arXiv preprint arXiv:190401451. 2019.

[96] Cer D, Yang Y, Kong S-y, Hua N, Limtiaco N, John RS, *et al*. Universal sentence encoder. arXiv preprint arXiv:180311175. 2018.

[97] Pagliardini M, Gupta P, Jaggi M. Unsupervised learning of sentence embeddings using compositional n-gram features. arXiv preprint arXiv:170302507. 2017.

[98] Radford A, Narasimhan K, Salimans T, Sutskever I. Improving language understanding with unsupervised learning. Technical report, OpenAI; 2018.

[99] Peters ME, Neumann M, Iyyer M, Gardner M, Clark C, Lee K, *et al*. Deep contextualized word representations. arXiv preprint arXiv:180205365. 2018.

[100] Howard J, Ruder S. Universal language model fine-tuning for text classification. arXiv preprint arXiv:180106146. 2018.

[101] Gers FA, Schmidhuber J, Cummins F. Learning to forget: Continual prediction with LSTM. Neural computation. 2000;12:2451-71.

[102] Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Networks. 2005;18(5-6):602-10.

[103] Radford A, Wu J, Child R, Luan D, Amodei D, Sutskever I. Language models are unsupervised multitask learners. OpenAI Blog. 2019;1(8).

[104] Yang Z, Dai Z, Yang Y, Carbonell J, Salakhutdinov R, Le QV. XLNet: Generalized autoregressive pretraining for language understanding. arXiv preprint arXiv:190608237. 2019.

[105] Moen S, Ananiadou TSS. Distributional semantics resources for biomedical text processing. Proceedings of LBM. 2013:39-44.

[106] Beam AL, Kompa B, Fried I, Palmer NP, Shi X, Cai T, *et al*. Clinical concept embeddings learned from massive sources of multimodal medical data. arXiv preprint arXiv:180401486. 2018.

[107] Wang Y, Liu S, Afzal N, Rastegar-Mojarad M, Wang L, Shen F, *et al*. A comparison of word embeddings for the biomedical natural language processing. Journal of biomedical informatics. 2018;87:12-20.

[108] Smaili FZ, Gao X, Hoehndorf R. Onto2vec: joint vector-based representation of biological entities and their ontology-based annotations. Bioinformatics. 2018;34(13):i52-i60.

[109] Pirró G, Euzenat J. A feature and information theoretic framework for semantic similarity and relatedness. International semantic web conference; 2010: Springer;(p. 615-30).

[110] Halper M, Geller J, Perl Y. An OODB part relationship model. Proc of the 1st Int Conf on Information and Knowledge Management, CIKM; 1992: Citeseer;(p. 602-11).

[111] Halper M, Geller J, Perl Y. An OODB part-whole model: Semantics, notation and implementation. Data & Knowledge Engineering. 1998;27(1):59-95.

[112] Banerjee S, Pedersen T. Extended gloss overlaps as a measure of semantic relatedness. Ijcai; 2003;(p. 805-10).

[113] Patwardhan S, Pedersen T. Using WordNet-based context vectors to estimate the semantic relatedness of concepts. Proceedings of the Workshop on Making Sense of Sense: Bringing Psycholinguistics and Computational Linguistics Together; 2006.

[114] Pesaranghader A, Muthaiyah S, Pesaranghader A. Improving gloss vector semantic relatedness measure by integrating pointwise mutual information: Optimizing second-order co-occurrence vectors computed from biomedical corpus and UMLS. International Conference on Informatics and Creative Multimedia; 2013: IEEE;(p. 196-201).

[115] Taieb MAH, Aouicha MB, Hamadou AB. A new semantic relatedness measurement using WordNet features. Knowledge and information systems. 2014;41(2):467-97.

[116] Rehurek R, Sojka P. Software framework for topic modelling with large corpora. In Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks; 2010: Citeseer.

[117] Liang X, Wang G. A Convolutional Neural Network for transportation mode detection based on smartphone platform. 14th International Conference on Mobile Ad Hoc and Sensor Systems (MASS); 2017: IEEE;(p. 338-42).

[118] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, *et al*. TensorFlow: A system for large-Scale machine learning. OSDI; 2016;(p. 265-83).

[119] Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint arXiv:160904747. 2016.

[120] Pedregosa F, Varoquaux G, Gramfort A, Michel V, Thirion B, Grisel O, *et al*. Scikit-learn: Machine learning in Python. Journal of machine learning research. 2011;12(Oct):2825-30.

[121] Liu H, Geller J, Halper M, Perl Y. Using Convolutional Neural Networks to Support Insertion of New Concepts into SNOMED CT. AMIA Annu Symp Proc; 2018: American Medical Informatics Association;(p. 750-9).

[122] Liu H, Perl Y, Geller J. Transfer learning from BERT to support insertion of new concepts into SNOMED CT. AMIA Annu Symp Proc. 2019.

[123] Zhu Y, Kiros R, Zemel R, Salakhutdinov R, Urtasun R, Torralba A, *et al*. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. Proceedings of the IEEE international conference on computer vision; 2015;(p. 19-27).

[124] Morrey CP, Geller J, Halper M, Perl Y. The Neighborhood Auditing Tool: a hybrid interface for auditing the UMLS. Journal of biomedical informatics. 2009;42(3):468-89.

[125] Loper E, Bird S. NLTK: the natural language toolkit. arXiv preprint cs/0205028. 2002.

[126] Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, *et al*. Google's neural machine translation system: Bridging the gap between human and machine translation. arXiv preprint arXiv:160908144. 2016.

[127] Halper M, Wang Y, Min H, Chen Y, Hripcsak G, Perl Y, *et al*. Analysis of error concentrations in SNOMED. AMIA Annu Symp Proc. 2007:314-8.

[128] Zheng L, Chen Y, Elhanan G, Perl Y, Geller J, Ochs C. Complex overlapping concepts: An effective auditing methodology for families of similarly structured BioPortal ontologies. Journal of biomedical informatics. 2018;83:135-49.

[129] Kendall EF, McGuinness DL. Ontology Engineering. Synthesis Lectures on The Semantic Web: Theory and Technology. 2019;9(1):i-102.

[130] Lee J, Yoon W, Kim S, Kim D, Kim S, So CH, *et al*. Biobert: pre-trained biomedical language representation model for biomedical text mining. arXiv preprint arXiv:190108746. 2019.

[131] Huang K, Altosaar J, Ranganath R. ClinicalBERT: Modeling clinical notes and predicting hospital readmission. arXiv preprint arXiv:190405342. 2019.

[132] Peng Y, Yan S, Lu Z. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. arXiv preprint arXiv:190605474. 2019.