SEMANTICS ENHANCED DEEP LEARNING MEDICAL TEXT CLASSIFIER

by

Jose David Posada Aguilar

MS, University of Pittsburgh, 2016

Submitted to the Graduate Faculty of

School of Medicine in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

University of Pittsburgh

2018

UNIVERSITY OF PITTSBURGH

SCHOOL OF MEDICINE

This dissertation was presented

by

Jose David Posada Aguilar

It was defended on

July 23, 2018

and approved by

Dr. Harry Hochheiser, Associate Professor, Department of Biomedical Informatics

Dr. Chakra Chennubhotla, Associate Professor, Department of Computational and Systems Biology

Dr. Henk Harkema, Adjunct Assistant Professor, Department of Biomedical Informatics

Dissertation Director: Dr. Fuchiang (Rich) Tsui, Associate Professor, Department of Biomedical Informatics

SEMANTICS ENHANCED DEEP LEARNING MEDICAL TEXT CLASSIFIER

Jose David Posada Aguilar, PhD

University of Pittsburgh, 2018

Electronic health records (EHR) contain a vast amount of data with the potential to leverage applications that improve patient outcomes and enhance the work of health care providers. A major portion of this data is inside unstructured text in the form of clinical narratives. To effectively use clinical text, NLP tools have been developed and applied to numerous problems involving clinical decision support systems, cohort identification, and phenotyping among others.

However, one of the main problems that face the development of NLP tools for the clinical domain is the lack of large annotated data sets. Clinical language and report style variations are another major problem for clinical NLP. These variations lead to problems where NLP systems created with data from one institution exhibit significantly different performance when tested in a different institution.

One way to address the lack of large annotated datasets and variations in clinical language is the explicit incorporation of semantics into the development of clinical NLP tools. Semantics allow us to know that the meaning of words, and thus help us account for language variations. In this work, we incorporate the semantics from ontologies into a loss function of a deep learning text classifier. Also, to specifically address the problem of the lack of large annotated datasets we used a large unannotated or unlabeled dataset, increasing the sample size as a result. To properly use such unlabeled data, we adapted a semi-supervised binary approach that uses the unlabeled dataset during training.

To the best of our knowledge we are the first to do so, and for that reason, this is one of the main theoretical contributions of this work. Also, by reducing the need for extensive

annotations, we believe this work could enable researchers in clinical settings to embrace and leverage the full potential of clinical NLP tools given the reduced effort required to achieve the desired performance. Furthermore, all the methods in this work are designed as reproducible and extensible software tools that aid further biomedical informatics research in this area.

Keywords: Deep Learning, Ontology, Natural Language Processing, Psychiatric Reports.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

For the LORD gives wisdom; from his mouth come knowledge and understanding

Proverbs 2:6

The fear of the Lord is the beginning of wisdom, and the knowledge of the Holy One is insight.

Proverbs 9:10

1.0 INTRODUCTION

Electronic health records (EHR) contain a vast amount of data with the potential to leverage applications that improve patient outcomes, reduce healthcare expenditure and enhance the work of health care providers [1, 2, 3]. It is estimated that roughly 25,000 petabytes of healthcare data will be available by 2020 [1, 3]. A major portion of this data is inside unstructured text in the form of radiology reports, operative notes and discharge summaries among other types of clinical narratives [4]. This clinical text is dictated and transcribed or directly entered by healthcare professionals to communicate the status and history of a single patient to other health care professionals or themselves [5]. This text is what best tells us the patients story and describes the physician thought processes and decision making [6]. Free text continues to be used despite many attempts to encode such information in the form of drop-down menus offered by commercial EHR applications [6].

To effectively use clinical text, natural language processing (NLP) tools have been developed and applied to numerous problems involving clinical decision support systems [4], cohort identification [7, 8], syndromic surveillance systems [9], information extraction [10] and phenotyping [11], among others. Many of these applications depend on some form of text classification. In clinical text classification, the interest is to classify a document into a set of predefined labels that usually represent a disease or condition of interest that is not easily captured by ICD or CPT codes.

Recently, for clinical text classification, deep learning (DL) strategies have gained increased attention. Different applications of deep learning applied to clinical text have shown an increase or at least a comparable performance to strategies that rely on handcrafted features [12, 13, 14, 15, 16, 17, 18, 19, 20, 21]. Usually, these DL text classification strategies rely on pre-trained distributional semantic models for text representation. Distributional

semantic models are models that usually represent word or documents in a vector space. In that space vectors that are close to each other are assumed to have similar meaning, this is being semantically similar. These models are trained with the assumption that words used in similar context have a similar meaning. With that assumption, the models are trained to capture semantics by using the "distribution" of words in the document.

In consequence, words that rarely occur in a similar context but are related will not be close in the vector space. As an example lets, consider the sentences *continued problems in school with frequent suspensions* and *received an undergraduate degree followed by a master's in Zoology*. These sentences are related because both are describing situations related to education. However, it is unlikely that the words in the sentences and the sentences itself occur in a similar context within a clinical text. Because of that distributional semantic models will hardly capture the semantic relationship from sentences like those. To overcome this issue, we can supply an external knowledge source containing relationships usually not present on distributional models. Such knowledge source in the biomedical domain is usually presented in the form of ontologies. In the biomedical domain, ontologies have been integrated into DL strategies for word sense disambiguation [22, 23], text classification [24], representation learning for predictive modeling [25, 18]. Outside the biomedical domain, they have been combined for short text classification [26] and to enhance distributional semantic models [27].

1.1 THE PROBLEM

A well-known problem in clinical NLP research is the lack of large annotated data sets. Training samples for clinical NLP are collected through annotation of clinical reports. Annotations are usually performed by clinicians who read through a clinical report and highlight the portion of text that is relevant. Because a clinician performs the task, the cost of annotating a large corpus is prohibitive for most research projects [28]. That is the reason that when we compare the size of annotated samples for non-clinical text data sets such as bAbI (20,000 samples) [29] or the Imbd movie reviews (50,000 samples) [30], with one of

the most recent annotated clinical data sets (816 samples) [31], we see a notable difference. Any method applied to text classification in the clinical domain needs to account for this problem.

To understand how this problem is solved we need to first describe in more detail the characteristics of those annotations. Clinicians annotate only the portion of text that explicitly contain the relevant information; those annotations are the *positive* samples. All the other portion of text it is assumed to be non-relevant if the clinician read the entire report, and those annotations become the *negative* samples. The majority of the developed clinical NLP exclusively make use of positive and negative samples, requiring fully annotation of the reports. We will name those approaches PN, for positive and negative.

However, if the clinician does not read the entire report, the report is partially annotated, and it cannot be assumed that all the non-annotated text is non-relevant. In this case, those portions of text are considered *unlabeled* samples because they can be either positive or negative. Unlabeled samples can also be collected through clinical reports the annotator never saw but possibly contain the information we are looking. Partial annotation is done in cases where the time required to read the entire report could prevent the collection of enough samples, or when the annotations are done at a document level or when the information for a given patient in a report is just repetitive and does not contribute to the diversity of the samples. To exemplify this let's consider the report in Figure 1 and let's say we would like to annotate information pertaining to living conditions. This particular patient is homeless and the way is described in both of the highlighted sentences is almost the same. Given that the particular living condition for this patient is homelessness, all the information mentioned in the report will be pointing to that. If we want a variety of living conditions, we would have to read a clinical report from a different patient. So instead of spending the precious and costly time of a clinician annotating the same information over and over again, it could be better to ask them to partially annotate this report and move to the next. A report annotated in this form gave way to applications in clinical NLP that only used positive samples [32] or that used a combination of positive, negative and unlabeled samples (PNU) [33, 34]. No attempt has been made in clinical NLP to train a system with only positive and unlabeled samples (PU), and this is one of the focuses of this dissertation.

MEDICATION COMPLIANCE: The patient has not taken these for the last month-and-one-half as he has been homeless. Medication compliance is poor. He left all of his medications at Harbor Lights when he left there.

DIET: No special diet.

ALLERGIES: NO KNOWN DRUG ALLERGIES.

NONALLERGIC DRUG REACTIONS: NONE.

HABITS (current and past):

Tobacco: A one pack per day for one-and-one-half years, quit 9-11 years before that.

Alcohol: Positive for a case of beer daily lately.

Drugs: Marijuana and cocaine "when available."

Withdrawal symptoms: He only states withdrawal seizures in the past. He denies hallucinations, tachycardia, and anxiety.

SIGNIFICANT SOCIAL HISTORY:

He is homeless. He had been at Harbor Lights. He is currently unemployed. He previously worked in the labor unions and done roofing. He is a high school graduate.

Figure 1: Psychiatric Report Excerpt. Highlighted in green the relevant information annotated

The other major problem with clinical text is that clinical language and clinical report styles vary across institutions [35, 36]. Even for the same institution, there is variation across different practices [37]. Prior research shows a significant difference in performance of NLP systems created with data from one institution and tested on data from another [38]. Given this cross-institutional difference it is has been shown that tuning is usually needed when moving a system from one institution to another [39, 40].

One of the ways to address the problem of variability in clinical language and report style is to train with data from multiple institutions [41]. However, this is not always feasible given regulatory restrictions and commercial interests from the healthcare institutions. Another way is to incorporate semantics into the construction of clinical NLP systems [40]. Such semantics can be incorporated through ontologies [42] or distributional semantics models [19]. By incorporating semantics, we can enable NLP systems to understand what is written [43] and consequently improve their performance. As an example, let's consider the sentences *I live by myself*, and *I live alone*. Both sentences are expressing the same reality with different words. An NLP tool trained using only one of the two examples may not generalize well for the second example, without knowing that the meaning of the words *by myself* and *alone* is the same in this context.

Several methods have been investigated to incorporate semantics using ontologies. One of the common approaches is to map words or phrases into concepts inside an ontology such as SNOMED [44]. In this approach, instead of using the raw text in a sentence like this *he was homeless and living in his car*, the mapped equivalent *homeless (C0237154), car owner (C0425252)* is used to train the models. This approach has been used extensively in different applications that use rule-based [45, 46] or machine learning systems [47, 48, 49, 50, 51]. These studies used the ontological concepts as input features. Most of these applications use a dictionary lookup tool such as cTakes [52], MetaMap [53] or NOBLE Coder [54] to accomplish a mapping.

However, none of these approaches take into account the relationships of those concepts in their ontologies. To account for those relationships some approaches incorporate semantic similarity metrics derived from them. Those similarity metrics are computed by using the hierarchical structure of the ontologies, measuring how close two concepts are from each other.

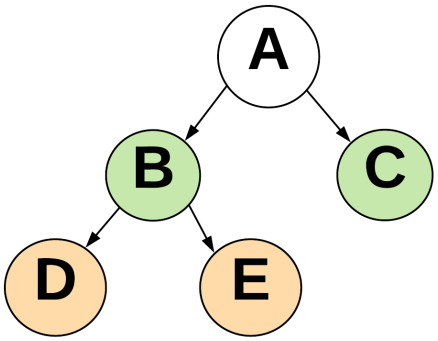


Figure 2: Example of an Ontology

The closeness is measured by accounting for the path between two concepts. Concepts that are closer in the hierarchy (e.g., D and E in Figure 2) are assumed to be more semantically similar than those that are farther apart (e.g., D and C in Figure 2). One of the ways to incorporate this metric in biomedical text is to use support vector machine (SVM) with a semantic kernel [55, 56, 57]. The semantic kernel is a matrix that contains the concept's pairwise semantic similarities derived from the ontologies. This matrix is used as the kernel in the SVM. This approach does not take advantage of possibly defining the class as a set of concepts, but keep the class defined as a simple categorical label. This is indeed attempted to solve by [58] where the class is defined as a set of concepts. In this approach, the distance between the set of concepts that form the document and the set that form the class is measured. The document is assigned to the class with lower distance, resembling a k-nearest neighborhood(kNN) approach. In this approach the distance computed does not fully take into account the hierarchy, suffer from the same weaknesses of kNN(e.g., lack of model and computational cost for prediction) and was only applied to biomedical journals but not clinical text. In both approaches (i.e., semantic kernel and class-based metrics) semantics that could be derived from the words themselves are not considered, and consequently lost.

Besides ontologies, we can also incorporate semantics through distributional semantic models. Those models are usually trained with unlabeled clinical notes using word2vec [59]

or Glove [60]. Two main approaches have been used to obtain those models. The first one used only the raw text to train such models using simple NLP processes such as tokenization or stop word removal on the raw text [61, 62, 63, 19, 17, 14, 64, 28, 15]. The second use dictionary lookup tool to transform the text into concepts and obtain a model for the mapped concepts [65, 66, 13]. Therefore instead of having vector representations for words, the representations are obtained for each concept. In this way, some knowledge from the ontology is transferred by means of word normalization, but the hierarchical relationship of the concepts is lost, and the semantics that could be derived from keeping the raw text representation are also lost.

To overcome this issue, distributional semantic models are improved by using ontologies in a process called *retrofitting* [67, 27, 68] or by modifying the loss function used to train those models to incorporate knowledge constraints [69, 70]. Retrofitting is a process by which those models are modified after training by forcing concepts that are related in the ontology to be related in the models [68, 71]. As an example, let's assume the vector representation in the distributional semantic model for the words *university* and *midterms* are not close to each other. If those words are indeed close in the ontology, the vectors are modified in such a way that the distance between them is reduced. Once these distributional semantic models are obtained, the semantics embedded in those models are leveraged in several works [63, 21, 12, 19, 14, 15, 13] as input features of the machine learning algorithms. However, any of those works merge the representation of the text as concepts, the ontologies hierarchical relationships and the distributional semantics from raw text into the training of the classifiers.

A recent work applied to biomedical journals [24], attempts to use the controlled vocabulary in the ontology, hierarchical structure, and the distributional semantics. In this work, words that belong to a given group in the ontology share parameters in the model. Groups are formed by merging concepts that share common parents. By merging concepts that share common parents the hierarchical relationships are lost. To illustrate this let's consider the Figure 2 where according to this approach concepts B, C and D, E will be merged to groups g_1 and g_2 respectively. Within those groups model parameters for B, C will be shared but no sharing will occur between g_1 and g_2 effectively losing the relationship between B and D. In this work we preserve those relationships, leverage the concept representation for the text

and also focus on the loss function of the classifier rather than enhancing the representation of text, as a markedly different factor from all the previous research that solely focus on modifying the text representation.

1.2 THE APPROACH

We propose a novel approach that combines distributional semantic models with ontologies in a single framework. First, we trained a distributional semantic model from a large corpus of unlabeled clinical reports. The distributional semantic model is in the form of word embeddings. Second, we used an ontology to compute a semantic distance between concepts in the ontology. This distance is incorporated into the loss function of a text classification strategy. The strategy has two main components. The first component is a semantic deep network (SDN) that predicts which concepts are inside a clinical text, similar to the work done by a dictionary lookup tool. The second component is a classification deep network (CDN) that given the probability of each concept in the input text outputs the class label for the text. The unlabeled samples are used in a semi-supervised binary strategy integrated with the SDN and the CDN. The semi-supervised strategy consists of a recursive elimination method that attempts to remove probable positive samples from the unlabeled set of samples. A diagram with the approach is shown in Figure 3.

1.2.1 Theses

In this dissertation, we tested the following hypotheses: (1) the proposed strategy will improve the generalization performance of clinical text semi-supervised classification algorithms by incorporating semantics into the training strategy. (2) The proposed strategy will improve the generalization performance of clinical text supervised classification algorithms by incorporating semantics into the training strategy

From our experiments, we can conclude the following theses: (1) a semi-supervised clinical text classification strategy that integrated a semantic loss function with a recursive

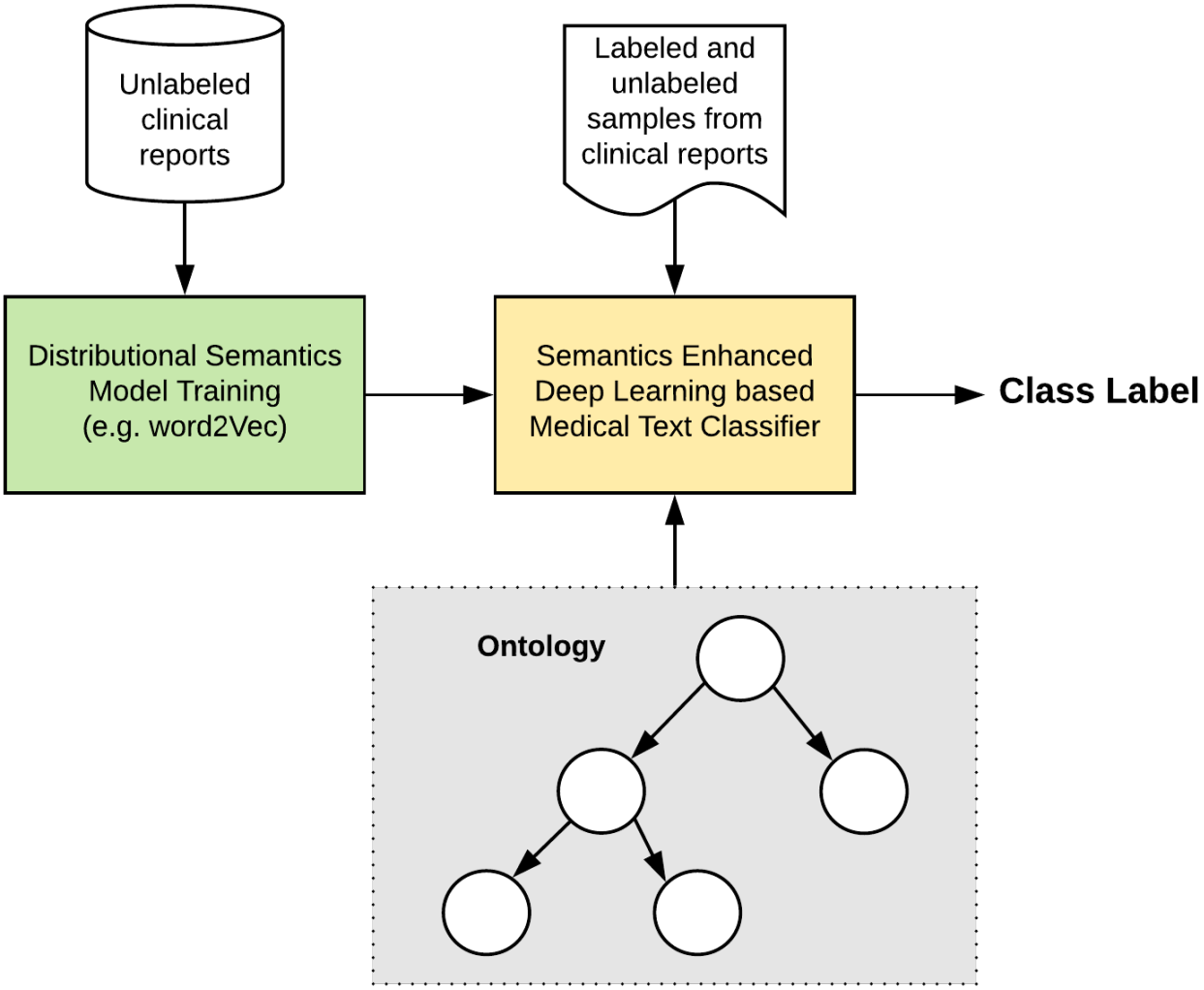


Figure 3: Overview of the approach

elimination method obtained a better generalization performance than strong clinical text classification baseline methods. (2) Adding a semantic loss function did not improve the generalization performance over strong supervised clinical text classification methods. Based on the experiments performed the following strong claims can be made:

1. Only positive labeled samples are needed to achieve good performance in clinical text classification.
2. To train a semi-supervised strategy that only uses positive and unlabeled samples a non-negative risk loss function is sufficient to evaluate model candidates during hyper-parameter optimization.
3. It is feasible to incorporate prior knowledge into a deep learning strategy in the form of a derivable semantic loss function.

The following weak claim can also be made

- A set based semantic distance function can be directly used to identify probable positive samples to be used for training a clinical NLP tool.

1.3 SIGNIFICANCE AND INNOVATION

To the best of our knowledge, this is the first time that ontologies have been integrated with deep learning to explicitly incorporate semantics into a text classification algorithm applied to clinical text.

Also, to the best of our knowledge, this is the first time that a semi-supervised strategy applied to the classification of clinical text has been trained by only using positive and unlabeled samples. Moreover, our approach is one of the first being applied to a corpus of psychiatric notes.

The principal contribution to informatics is the novelty of hierarchical relationships between ontological concepts to inject semantic relationships into a loss function used to train a text classifier. This process is uniquely integrated into the loss function during classifier training as opposed to preprocessing.

Capturing semantic information reduced the amount of data required to achieve a good level of performance when classifying clinical text. This is advantageous for clinical translational research since the cost and time required to annotate clinical text is prohibitive for most research projects [28]. Our strategy is designed to fill this need, with the primary objective to use less training data. By using less data this work has the potential to enable applications that were restricted for the lack of resources to annotate the number of required clinical reports. Also, applications such as cohort discovery [8] and screening patient's eligibility in clinical trials from clinical notes [72] can also benefit from this.

Finally, all the methods developed in this work are designed to be reproducible, transparent, accessible and generalizable for future clinical natural language processing applications. We believe this work is important in both biomedical informatics and applied clinical research.

1.4 THESIS OVERVIEW

In Chapter 2, a review of the relevant literature regarding clinical natural language processing, deep learning, and semantic similarity measures in biomedical text is presented. In Chapter 3, all the developed methods are described in detail, along with the development of the dataset used to evaluate the proposed strategy. In Chapter 4, a detailed description of the experiments performed is shown, along with evaluation results on the datasets. Finally, in Chapter 5, a discussion of the results is presented along with the limitations, how to address those and possible future work.

2.0 BACKGROUND

In this chapter, we will discuss relevant concepts regarding clinical text classification, traditional algorithms, and features used for this task. Later we will explain the basic concepts of deep learning pertinent to this work. Finally, we finish with a brief description of semantic similarity measures applied to biomedical text using ontologies.

2.1 CLINICAL TEXT CLASSIFICATION

Text classification is a task that consist in assigning a document to a predefined set of classes or labels [73]. Given an input document d and a set of n_C class labels $\mathcal{C}_\mathcal{L} \in \{1, \ldots, n_C\}$ we wish to learn a classification function $f : \mathsf{d} \rightarrow \mathcal{C}_\mathcal{L}$ that maps document to classes. For supervised and semi-supervised learning the class labels are generally known during training, contrary to unsupervised learning where the labels are completely unknown during training. For clinical text, the most common supervised methods are Naive Bayes, support vector machines (SVM), hidden Markov models and conditional random fields [74]. More recently convolutional neural networks (CNN) [21, 17] and long short-term memory networks (LSTM) [14] are becoming increasingly popular given their good results in general text. SVM will be described in this section while CNN and LSTM will be described later in Section 2.2.

2.1.1 Support Vector Machines

Support vector machines are a binary discriminative learning approach that aims to learn a hyperplane that separates two classes while maximizing the *margin* between the decision

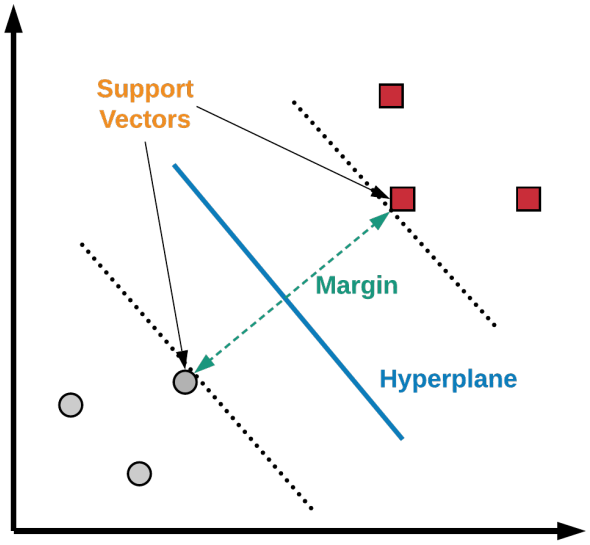


Figure 4: Linear SVM decision boundary and Margin

boundary. The *margin* is defined as the distance between the closest data point to the hyperplane that composes the decision boundary between the two classes [73]. Support vectors are called to the data points that are closest to the hyperplane. In Figure 4, we can see an example of a hyperplane, the margin and the support vectors in two dimensions for a linearly separable case.

In general, the decision boundary is non-linear. Thus a nonlinear function is needed to separate the classes. This non-linear function is embedded into SVM in the form of a kernel. The idea of a kernel is to map the input space into a high dimensional space where the data is separable. For this general case, the decision function for a new sample $\mathbf{x}$ of an SVM is defined by

$$sign(\sum_{i}^{l} y_i \alpha_i K(\mathbf{x}^i, \mathbf{x}) + b) \tag{2.1}$$

where K is the kernel function, $\mathbf{x}^i \in \mathbb{R}^N$, $\forall\, i = 1, \ldots, l$ is the input vector that represents d, with l training samples. $y_i \in \{1, -1\}$ are the labels of each class. α_i and b are the parameters learn by the algorithm [75, 76, 77].

Traditionally to obtain a representation of d suitable for the use with SVM a *tf-idf* vector space model [78] is obtained from the training corpus. To compute *tf-idf* vector we need to compute three quantities: the term frequency, the document frequency, and the inverse document frequency. First let's represent a document d_i in a corpus $\mathcal{D}$ as a sequence of words $\mathsf{d} = (w_1, \ldots, w_n),\ \forall\ \mathsf{d}_i \in \mathcal{D}$. Then we can compute the term frequency tf_{w_j,d_i} as the number of times w_j occur in d_i. Usually, instead of raw counts, we can use a log normalization $1 + log(tf_{w_j,\mathsf{d}_i})$ to increase the weight of rare words that usually are more informative than common words. Now, to compute the document frequency of a word df_{w_j} we only need to count the number of d_i that contains w_j. The inverse document frequency idf_{w_j} is simply the inverse of the document frequency that is usually log scaled and computed by $idf_{w_j} = 1 + log(l/df_{w_j})$. Finally, the *tf-idf* weight for w_j in d_i can be computed using the equation 2.2.

$$\textit{tf-idf}_{w_j,\mathsf{d}_i} = tf_{w_j,\mathsf{d}_i} * idf_{w_j} \tag{2.2}$$

Finally, the input feature vector $\mathbf{x}^i \in \mathbb{R}^N$ with N as the vocabulary size is formed by the *tf-idf* weights of each word such that $\mathbf{x}^i_j = \textit{tf-idf}_{w_j,\mathsf{d}_i}$.

2.2 DEEP LEARNING ARCHITECTURES FOR NATURAL LANGUAGE PROCESSING

In the following sections, I describe a neural a network as well as the main architectures used in the present work. I start by describing Deep Feedforward networks (FFN) to later move to word embeddings as an application of this architecture to obtain a distributional semantic model. In Section 2.2.3 I describe convolutional neural networks (CNN) which were initially successfully applied in computer vision applications but has recently gained traction in the NLP community. I then finalize explaining how the networks are trained.

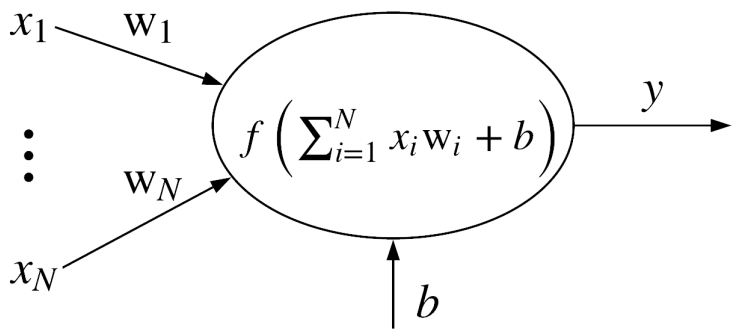


Figure 5: Representation of a basic neuron

2.2.1 Deep Feedforward Networks

Neural networks are a collection of units called *neurons* that aim to approximate any non-linear function. Originally neural networks are biologically inspired, however in this work we will approach them from the mathematical point of view.

A graphical representation of a basic neuron can be seen in Figure 5. This neuron can be seen as a weighted sum of an input $\mathbf{x} \in \mathbb{R}^N$ with weights $w_i \in \mathbb{R},\ \forall\ i, \ldots, N$ and a bias term $b \in \mathbb{R}$. Here f is a nonlinear activation function applied to the result of the weighted sum.

There are several options for nonlinear activation functions. In Table 1 we summarize the most common ones.

A collection of neurons is called a *layer*. A layer is a collection of neurons that is applied to the same input. In Figure 6, we can see the graphical representation. This layer can be represented in matrix notation as

$$\mathbf{y} = f\left(\mathbf{x}\mathbf{W} + \mathbf{b}\right) \tag{2.3}$$

where $\mathbf{W} \in \mathbb{R}^{N \times n_{\text{out}}}$ is the weight matrix, and the bias and output are now vectors $\mathbf{b}, \mathbf{y} \in \mathbb{R}^{n_{\text{out}}}$.

Now deep feedforward networks (FFN) are typically a composition of multiple of such layers one after another, such that the information flows from the input through the output

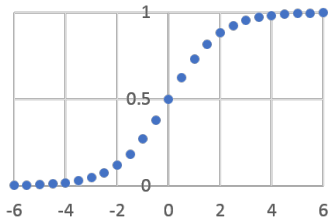
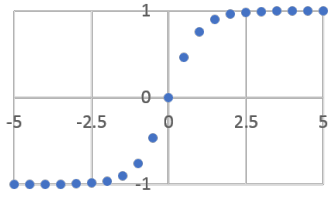
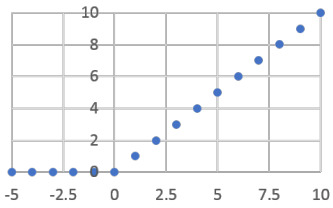
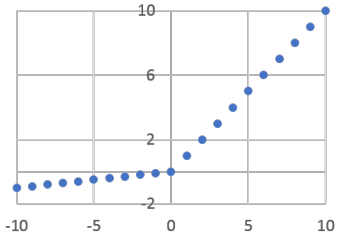
Function	Equation	Plot
Sigmoid	$\frac{1}{1+e^{-x}}$	
Tanh	$\frac{2}{1+e^{-2x}} - 1$	
ReLU [79]	$max(0, x)$	
Leaky ReLU [80]	$max(\beta x, x), \beta \in \mathbb{R}$	

Table 1: Common nonlinear activation functions

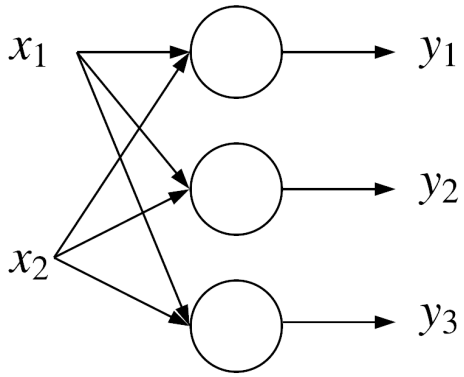


Figure 6: Representation of a basic Neural Network layer with two inputs and three neurons

in one direction with no feedback loops or recursion [81] exist. In this networks, there are three main types of layers: an input layer, a hidden layer, and an output layer. An input layer typically is the one that is in direct contact with the input. The hidden layers are the ones that process information as a result of the output of other layers including the input layer. The output layer is the one that produces the final output and is the one where typically the results are observed. In Figure 7, we can see a graphical representation of a prototypical deep feedforward network.

If the classification problem is multi-label multi-class usually a sigmoid function is used in the output layer. In a multi-class problem the classifier is asked to differentiate between a plural number of classes(i.e., more than two). In a multi-label classification problem a classifier is asked to assign more than one label to each sample. On the other hand, if we have a multi-class problem a *softmax* function is usually applied to the last layer. *softmax* normalize the output of the network between zero and one, thus allowing a probabilistic interpretation of the output. For a vector $\mathbf{x} \in \mathbb{R}^N$ *softmax* can be defined as:

$$softmax\left(\mathbf{x}\right) = \frac{1}{\sum_{i=1}^{N} e^{x_i}} \begin{bmatrix} e^{x_1} \\ \vdots \\ e^{x_N} \end{bmatrix} \tag{2.4}$$

Usually, in NLP, these networks are used in combination with three types of different

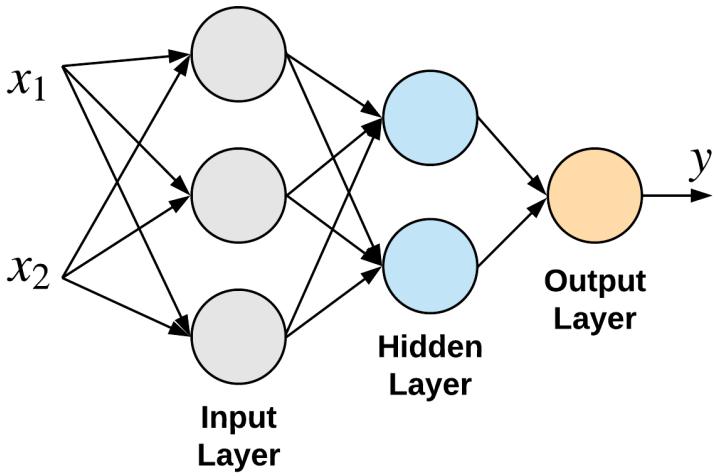


Figure 7: Representation of a typical Feedforward network with one input layer, one hidden layer and one output layer

feature representation for text: *one-hot*, *tf-idf* and word embeddings. These feature representations are used as inputs in the first layer of the network. In *one-hot* representations, each word w_j is represented by a **0** if it does not occur on document d and **1** if it does occur. In this representation, we have a binary feature vector $\mathbf{x} \in \mathbb{Z}^N$ with N as the vocabulary size. In *tf-idf* each document is represented by a real valued feature vector $\mathbf{x} \in \mathbb{R}^N$.

For word embeddings [59] we have a matrix $\mathbf{WE} \in \mathbb{R}^{N \times a}$ with a as the size of the embeddings and N the vocabulary size. Here each w_i is represented as a row vector in $\mathbf{WE}$. The vectors for individual words occurring in a document are aggregated using *sum*, *mean* or other similar operation [61], to represent a full document. Using this approach the feature vector is $\mathbf{x} \in \mathbb{R}^a$. Word embeddings will be discussed in detail in Section 2.2.2.

2.2.2 Word Embeddings

Word embeddings are a distributional semantics representation of words usually obtained through unsupervised training in a large corpus [82]. The most common algorithm to obtain word embeddings is called word2vec [59]. The most common architecture for word2vec is

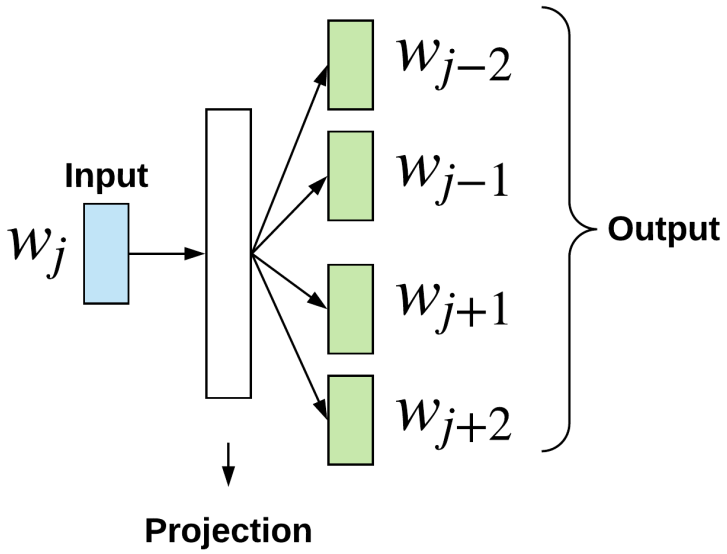


Figure 8: Skip-gram model architecture for word2vec

called Skip-gram (see Figure 8).

In this architecture, the goal is to predict the context words $C(w_j) = (w_{j-k}, \ldots, w_{j+k})$ given an input word w_j in a window of length k. The model attempts to maximize the average log probability of the corpus $\mathcal{D}$ [83]:

$$\arg\max_{\boldsymbol{\theta}} \sum_{(w,cw)\in\mathcal{D}_p} \log(p(cw|w;\boldsymbol{\theta})) \tag{2.5}$$

where $\boldsymbol{\theta}$ are a set of model parameters, and $\mathcal{D}_p \subset \mathcal{D}$ is a set composed of all word and context pairs that occur in the corpus. $\mathcal{D}_p$ is a subset because it is regulated by the size of the context window, so if the context window is infinite, the subset is exactly equal to the set, but this will be computationally intractable. For simplicity, we drop the j index in w, and any word in $C(w_j)$ is represented now as cw. The proposed architecture to compute this probability is an FFN with one input layer $\mathbf{W} \in \mathbb{R}^{N\times a}$, one output layer $\mathbf{W'} \in \mathbb{R}^{a\times N}$ with *softmax* output, where a is the size of the embeddings. However, the *softmax* is not entirely computed but approximated using a method called negative sampling [59]. Negative sampling is a method composed by a mathematical approximation followed

by a series of heuristics where the initial objective proposed in Equation 2.5 is changed by a binary classification problem. In this binary classification problem, the objective is to predict whether a pair of words occur in the corpus or not. In the following, we will describe in detail how the algorithm works.

Let $\mathbf{x} \in \mathbb{Z}^N$ be the *one-hot* vector representing the word w. The output of the FFN can be computed as $\mathbf{y} = softmax\,(x^T\mathbf{W}\mathbf{W}'), \mathbf{y} \in \mathbb{R}^N$. As we can see from the previous equation, the only non-linearity comes from the computation of the *softmax* . Computing the *softmax* is expensive when the vocabulary has millions of words. Instead, the computation is reframed by using rows and column vectors from the input and output weights.

Let $\mathbf{v}_w \in \mathbb{R}^a$ be a row vector from the matrix $\mathbf{W}$ and $\mathbf{v}_{cw} \in \mathbb{R}^a$ be a column vector from the matrix $\mathbf{W}'$. Also, let z be a dummy variable that indicates whether a pair (w, cw) is in the corpus or not and defined by:

$$\begin{aligned} z &= 1 \;\forall\; (w, cw) \in \mathcal{D}_p \\ z &= 0 \;\forall\; (w, cw) \notin \mathcal{D}_p \end{aligned} \tag{2.6}$$

Now let's propose two logistic functions to define the probability of z as:

$$\begin{aligned} P(z = 1|(w, cw)) &= \sigma(\mathbf{v}_w \cdot \mathbf{v}_{cw}) \\ P(z = 0|(w, cw)) &= \sigma(-\mathbf{v}_w \cdot \mathbf{v}_{cw}) \end{aligned} \tag{2.7}$$

where $\sigma = \frac{1}{1+\exp(-x)}$. With this proposition, the new objective is to minimize [83]:

$$\arg\max_{\boldsymbol{\theta}} \sum_{(w,cw)\in\mathcal{D}_p} \log \sigma(\mathbf{v}_w \cdot \mathbf{v}_{cw}) + \sum_{(w,cw)\notin\mathcal{D}_p} \log \sigma(-\mathbf{v}_w \cdot \mathbf{v}_{cw}) \tag{2.8}$$

However, to compute the second term, this means the probability for all the samples that do not occur in the corpus will be computationally infeasible. To overcome this instead of computing for all $(w, cw) \notin \mathcal{D}_p$ the sum is computed for ns negative samples which become a hyper-parameter of the algorithm. Negative sample pairs are formed such that $((w, cw_1), \ldots, (w, cw_{ns}))$. The objective in Equation 2.8 is more computationally efficient than a *softmax* because it is not necessary to obtain the output for each one of the N words in the vocabulary. Instead, this costly procedure is replaced by computing the probability

only for the words in the window k plus ns negative samples. To select the negative samples a heuristic is implemented. The heuristic consists in randomly drawing words from the vocabulary with a probability:

$$p_{negative-sample}(cw_i) = \frac{(tf_{cw_i})^{3/4}}{\sum_{j=1}^{ns}(tf_{cw_j})^{3/4}} \tag{2.9}$$

Also, to compensate for too frequent words (e.g. *the*), each word in a sequence is assigned a probability for keeping such word. If the computed probability is below a predefined threshold, the word is discarded. The probability according to their published code is computed by:

$$p_{keeping}(w) = (\sqrt{\frac{U(w)}{0.001}} + 1) \cdot \frac{0.001}{U(w)} \tag{2.10}$$

where $U(w)$ is the unigram probability of the word w. Finally, instead of keeping the entire network we only keep the input weights matrix $\mathbf{W}$. That matrix ($\mathbf{W}$) is what we called the embeddings.

2.2.3 Convolutional Neural Networks

Convolutional neural networks (CNN) are neural networks that use a *convolution* operation along with matrix multiplication operations to compute their output [81]. A convolution is an operation typical defined in the signal processing world as a real value operation between time-dependent signals. For NLP a convolution is a binary operation that involves segments of text that are represented as real values [84] and a *filter* or kernel, whose purpose is to extract useful features from the text. One of the advantages of CNN vs. FFN is that we do not need to aggregate the embeddings of words occurring in a document.

To show how embeddings are used with a CNN lets start by defining a document as an ordered sequence of words $\mathsf{d} = (w_1, \ldots, w_n)$. Each word is represented as a row vector in $\mathbf{WE}$ and thus an input feature matrix $\mathbf{x} \in \mathbb{R}^{n \times a}$ is formed by the concatenation of each row vector that represents each w_j. Next, a convolution *filter* is applied to the feature matrix. A convolution filter is defined as a matrix of weights $\mathbf{W} \in \mathbb{R}^{h \times a}$ where $h <= n$ is the size of the sliding window in which the convolution is applied. Before defining the convolution,

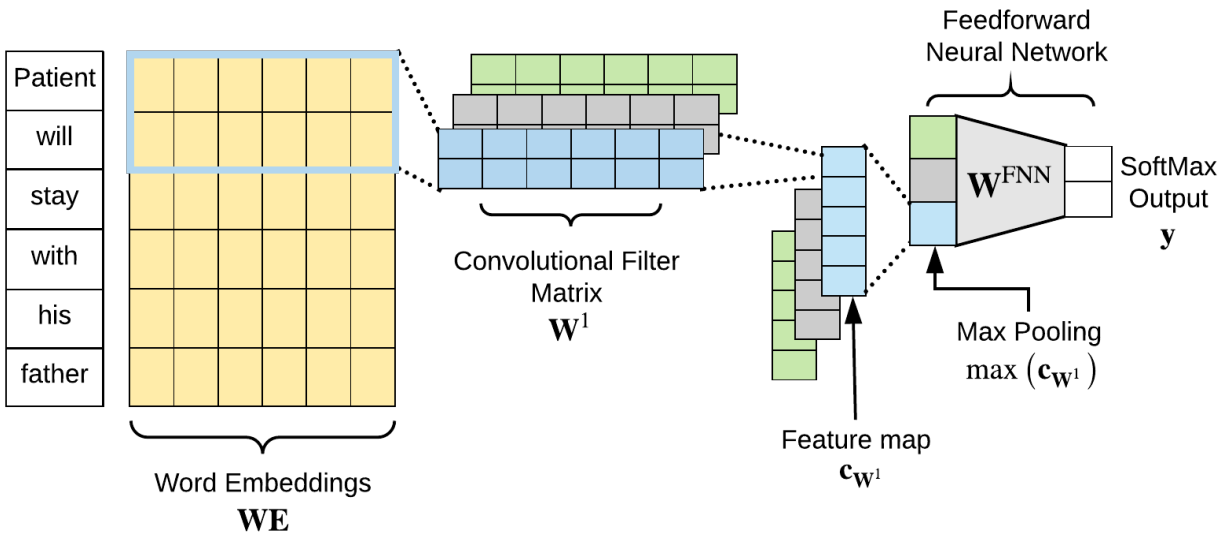


Figure 9: Convolutional Neural Network with Word Embeddings

let us define first the concatenation of h consecutive row vectors as $\mathbf{x}_{j:j+h-1}$. Then we can define the result of the convolution $c_k \in \mathbb{R}$ as:

$$c_k = f\left(\mathbf{W} \cdot \mathbf{x}_{k:k+h-1} + b\right), \ \forall \ k = 1, \ldots, n-h \tag{2.11}$$

where $\cdot$ is the dot product, $b \in \mathbb{R}$ is a bias term, and f is a nonlinear function (see Section 2.2.1). After doing the convolution over the entire document we have a feature map $\mathbf{c_W} = [c_1, \ldots, c_{n-h}] \in \mathbb{R}^{n-h}$ [85]. Over this feature map we apply a max-pooling operation $\hat{c}_\mathbf{W} = \max\left(\mathbf{c_W}\right), \hat{c}_\mathbf{W} \in \mathbb{R}$ [86]. Given that we can have multiple q filters $\mathcal{W} = \{\mathbf{W}^1, \ldots, \mathbf{W}^q\}$ we can form a feature pooled vector $\hat{\mathbf{c}}_\mathcal{W} = [\hat{c}_{\mathbf{W}^1}, \ldots, \hat{c}_{\mathbf{W}^q}], \hat{\mathbf{c}}_\mathcal{W} \in \mathbb{R}^q$ [84]. After this, we use an FFN with at least one layer with $\hat{\mathbf{c}}_\mathcal{W}$ as the input and $\mathbf{y} \in \mathbb{R}^{n_C}$ as the output vector for n_C class labels. Without hidden layers, we will have a weight matrix $\mathbf{W}^{\text{FFN}} \in \mathbb{R}^{q \times n_C}$. The final output is obtained by using a *softmax* function over $\mathbf{y}$. A graphical representation of a CNN for NLP can be seen in Figure 9.

2.2.4 Long Short Term Memory Networks

Long short-term memory networks (LSTM) are a type of recurrent neural networks (RNN) were the order of the input sequence is taken into account. To make predictions the input sequence does not need to have a fixed length [82]. Specifically, an LSTM was designed to address the vanishing gradient problem with long-term dependencies, allowing predictions with longer sequences than a regular RNN [87]. An RNN is called recurrent because part of their output is feedback to the input. As well as CNN, LSTM can use the word embeddings directly without the need for an aggregation operator.

In an LSTM there are five main components which can be defined as vectors in $\mathbb{R}^{h_u}$: an input gate $\mathbf{i}_j$, an output gate $\mathbf{o}_j$, a forget gate $\mathbf{f}_j$, a cell state $\mathbf{c}_j$, and an output state $\mathbf{h}_j$. In all of those vectors h_u represent the number of hidden units. The input of this network is again a feature matrix $\mathbf{x} \in \mathbb{R}^{n \times a}$ produced as the concatenation of the row vectors from the word embeddings that represent the document $\mathtt{d}$. $\mathbf{x}_j$, $\forall\ j = 1, \ldots, n$ is a single row in the feature matrix representing the word w_j in the ordered sequence. Also, we will represent the concatenation of two row vectors with [**;**]. The graphical representation of the required computation for an LSTM is shown in Figure 10. To compute the output $\mathbf{h}_j$ we need the following equations [82, 14]

$$
\begin{aligned}
\mathbf{o}_j &= \sigma(\mathbf{W}^{io}\mathbf{x}_j + \mathbf{b}^{io} + \mathbf{W}^{ho}\mathbf{h}_{j-1} + \mathbf{b}^{ho}) \\
\mathbf{i}_j &= \sigma\left[\mathbf{W}^{ii}\left(\mathbf{x}_j\right)^T + \mathbf{b}^{ii} + \mathbf{W}^{hi}\mathbf{h}_{j-1} + \mathbf{b}^{hi}\right] \\
\mathbf{g}_j &= \tanh(\mathbf{W}^{ig}\mathbf{x}_j + \mathbf{b}^{ig} + \mathbf{W}^{hc}\mathbf{h}_{j-1} + \mathbf{b}^{hg}) \\
\mathbf{f}_j &= \sigma(\mathbf{W}^{if}\mathbf{x}_j + \mathbf{b}^{if} + \mathbf{W}^{hf}\mathbf{h}_{j-1} + \mathbf{b}^{hf}) \\
\mathbf{c}_j &= \mathbf{f}_j \odot \mathbf{c}_{j-1} + \mathbf{i}_j \odot \mathbf{g}_j \\
\mathbf{h}_j &= \mathbf{o}_j \odot \tanh(\mathbf{c}_j)
\end{aligned}
\tag{2.12}
$$

$$
\begin{aligned}
\left\{\mathbf{W}^{ii}, \mathbf{W}^{if}, \mathbf{W}^{ig}, \mathbf{W}^{io}\right\} &\in \mathbb{R}^{h_u \times a} \\
\left\{\mathbf{W}^{hi}, \mathbf{W}^{hf}, \mathbf{W}^{hc}, \mathbf{W}^{ho}\right\} &\in \mathbb{R}^{h_u \times h_u} \\
\left\{\mathbf{b}^{ii}, \mathbf{b}^{hi}, \mathbf{b}^{if}, \mathbf{b}^{hf}, \mathbf{b}^{ig}, \mathbf{b}^{hg}, \mathbf{b}^{io}, \mathbf{b}^{ho}\right\} &\in \mathbb{R}^{h_u} \\
\sigma : sigmoid, \odot &: \text{component wise product}
\end{aligned}
$$

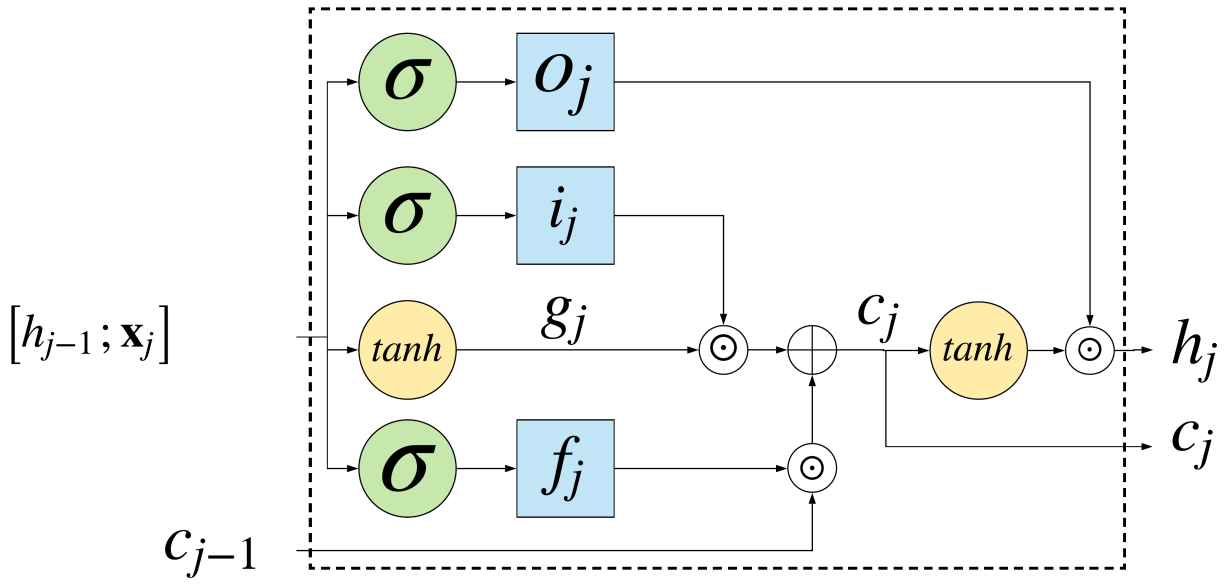


Figure 10: Long Short-Term Memory Network with Word Embeddings

All the gates outputs are computed by using the previous output state $\mathbf{h}_{j-1}$ and the current input $\mathbf{x}_j$. $\mathbf{h}_{j-1}$ is also used to update candidate $\mathbf{g}_j$. The forget gate $\mathbf{f}_j$ controls how much of the previous cell state $\mathbf{c}_{j-1}$ is considered to compute the new cell state $\mathbf{c}_j$, and the input cell $\mathbf{i}_j$ controls how much of the proposed update to keep. Finally, the output gate $\mathbf{o}_j$ controls how much of the current state $\mathbf{c}_j$ is used to compute the output state $\mathbf{h}_j$. To compute a final output we could use $\mathbf{h}_n$ directly or use this output state in combination with an FFN layer and *softmax* or sigmoid output.

2.3 DEEP LEARNING TRAINING

To train a neural network, we need to usually specify three main things: the architecture, the loss function, and the training algorithm. We will go over each one of them in the following sections.

2.3.1 Loss Functions

Loss functions are an essential component of the training since they specify how good a given classifier is doing. They define what is good and what is bad, so for a given dataset, they are the eyes that look at how well the strategy is doing. The two most common loss functions used in neural network training for NLP are Hinge and cross entropy loss. These losses are computed for each training sample j using the predicted output $\hat{\mathbf{y}}_i \in \mathbb{R}^{n_C}$ and the true output $\mathbf{y}_i \in \mathbb{R}^{n_C}$ where n_C is the number of classes. Let us represent a loss function as $\mathsf{L}\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) \in \mathbb{R}$ and the total loss for all the samples is usually computed as the sum of each of the sample loss:

$$\mathsf{L}_T = \sum_{i=1}^{l} \mathsf{L}\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) \tag{2.13}$$

Usually, the loss should be zero if a perfect classification was made. Now, we will show explicitly how the different losses are computed. Without loss of generality, in the following equations, we will drop the sample index i to simplify the equations when possible.

Hinge Loss

For binary classification problems, the output of the network is usually a scalar $\hat{y} \in \mathbb{R}$ and the class labels $y \in \{-1, 1\}$. This is similar to what is done in an SVM and, like in that case, the decision function is $sign\left(\hat{y}\right)$. A correct classification is produced if $y * \hat{y} > 0$. Is important to consider that the output for the network when using this loss function is usually between $(-\infty, +\infty)$ and linear. Finally, for a single sample the loss can be formulated as [82]:

$$\mathsf{L}_{Hinge}\left(\hat{y}, y\right) = \max\left(0, 1 - y * \hat{y}\right) \tag{2.14}$$

This loss attempts to have a correct classification with a margin of at least one

Multiclass Hinge Loss

The multiclass extension of the hinge loss was proposed by [88] where now $\hat{\mathbf{y}} \in \mathbb{R}^{n_C}$ is the predicted output vector, and the true class labels are a one-hot vector $\mathbf{y} \in \mathbb{R}^{n_C}$. It is important to notice that here the output is not a probability but a *score* that in general is

not bounded. To classify a sample, we need to compute:

$$prediction = \arg\max(\hat{\mathbf{y}}) \tag{2.15}$$

The loss can be computed in two ways. The first one [82] only considers the prediction for the highest scoring class as shown in the following equation:

$$\mathsf{L}_{Hinge}\left(\hat{\mathbf{y}}, \mathbf{y}\right) = \begin{Bmatrix} \max\left(0, \hat{y}_j - \hat{y}_k + \Delta\right) & \text{if } \arg\max(\hat{\mathbf{y}}) \neq \arg\max(\mathbf{y}) \\ 0 & otherwise \end{Bmatrix} \tag{2.16}$$

where $\hat{y}_j = \max(\hat{\mathbf{y}})$ and $\hat{y}_k$ is the output for the true class with $k = \arg\max(\mathbf{y})$ being the index for the true class. This loss function wants the predicted score for the correct class being larger than the score for any other class at least for a margin Δ. Classically, this margin is static and have the value of one, but in general is a hyperparameter to be tuned.

The second way is to consider all the other scores for all the other classes into the computation of the loss, thus this time attempting to produce not just a high score for the true class but the lowest possible score for all the other classes that are not the true one. This function can be defined as:

$$\mathsf{L}_{Hinge}\left(\hat{\mathbf{y}}, \mathbf{y}\right) = \sum_j \max\left(0, \hat{y}_j - \hat{y}_k + \Delta\right) \tag{2.17}$$

$$\hat{y}_j \in \hat{\mathbf{y}}, \quad \forall j (j \in \{1, \ldots, n_C\} \wedge j \neq k), \quad k = \arg\max(\mathbf{y})$$

The first one is more appealing since a prediction is not needed for all the classes and only for the one that has the maximum predicted value.

Cross Entropy Loss

When the outputs of the neural network can be interpreted as probabilities a cross entropy loss can be used. The output of the network is usually normalized using a *softmax* , and in consequence, we could interpret such output as a probability of each class. This time let $\mathbf{y} = \{y_1, \ldots, y_{n_C}\}$ be set representing the true multinomial distribution over n_C classes and let $\hat{\mathbf{y}} = \{\hat{y}_1, \ldots, \hat{y}_{n_C}\}$ be the network's output. Because we are using *softmax* , the output can be interpreted as $\hat{y}_j = P(y_j|\mathbf{x})$ where $\mathbf{x}$ is the input of the network. Thus, the categorical

cross entropy is measuring the dissimilarity between the two probability distribution and is defined as:

$$\mathsf{L}_{cE} = (\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{j} y_j \log \hat{y}_j \tag{2.18}$$

2.3.2 Regularization

Usually to avoid overfitting several strategies need to be used for training a deep neural network. Due to the vast amount of parameters, the network can easily overfit the training data and perform poorly on unseen data. Three types of regularization methods are discussed in this section. The first one imposes penalties on the learned weights, the second attempts to regularize the intermediate layer outputs, and the third one uses a developing set to stop the training process.

Parameter Norm Penalties

In this family of regularization, the two most used methods are L_1 and L_2 regularization [81]. Both methods impose an additional term to the loss function that depends exclusively on the norm of the weights. With this regularization a loss function takes the following form:

$$\mathsf{L}_T = \sum_{i=1}^{l} \mathsf{L}\left(\hat{\mathbf{y}}_i, \mathbf{y}_i\right) + \alpha\Omega(\boldsymbol{\theta}) \tag{2.19}$$

where $\boldsymbol{\theta}$ is the set of model parameters and $\alpha \in [0, \infty)$ is a hyperparameter that regulates the contribution of the penalty term to the total loss function. In L_1 regularization $\Omega(\boldsymbol{\theta})$ is substituted by the L_1 norm. In the L_2 regularization $\Omega(\boldsymbol{\theta})$ is substituted by L_2 norm. Each of the norms is computed by:

$$\mathsf{L}_1 = \|\boldsymbol{\theta}\|_1 = |\mathbf{W}| \tag{2.20}$$

$$\mathsf{L}_2 = \frac{1}{2}\|\boldsymbol{\theta}\|_2 = \frac{1}{2}\mathbf{W}^T\mathbf{W} \tag{2.21}$$

where $\mathbf{W}$ are the weights of each one of the layers.

Layer's Output Regularization

For layer's output regularization two main techniques are presented here: *dropout* [89] and *batch normalization* [90]. Dropout is a technique that randomly drops components of the input vector of any layer using a Bernoulli distribution. Formally let $\mathbf{x} \in \mathbb{R}^N$ be an input vector and a random vector $\mathbf{r} \sim \text{Bernoulli}(pr), \mathbf{r} \in \mathbb{R}^N$ with probability pr, a dropout operation is defined as:

$$\tilde{\mathbf{x}} = \frac{1}{1 - pr} \mathbf{x} \odot \mathbf{r} \tag{2.22}$$

where $\odot$ is the component-wise product and $\tilde{\mathbf{x}}$ is the vector used now as the input for the layer. During test time $\mathbf{r}$ is not generated and thus $\tilde{\mathbf{x}} = \mathbf{x}$. This technique attempts to prevent co-adaptation by relying only on some specific weights to produce an output. In [91] is shown that dropout has a strong connection with L_2 regularization.

The second technique, batch normalization, attempts to reduce the amount of change in the distribution of network activations. This change is called internal covariate shift. Through the use of batch normalization, the optimizer is less likely to get stuck in saturated regime and training would accelerate [90]. Let $x_{ij} \in \mathbf{x}_i$ be the values of the variable j in sample i. Let us define a batch for a variable j as a series of consecutive m samples as $\mathcal{B}_j = \{x_{1j}, \ldots, x_{mj}\}$. First, to compute the normalized output for the variable j $\tilde{x}_{ij}$ we need to compute the mean and variance of the batch as

$$\begin{aligned} \mu_{\mathcal{B}_j} &= \frac{1}{m} \sum_{i=1}^{m} x_{ij} \\ \sigma^2_{\mathcal{B}_j} &= \frac{1}{m} \sum_{i=1}^{m} \left(x_{ij} - \mu_{\mathcal{B}_j} \right) \end{aligned} \tag{2.23}$$

then we standardize the input

$$\hat{x}_{ij} = \frac{x_{ij} - \mu_{\mathcal{B}_j}}{\sqrt{\sigma^2_{\mathcal{B}_j} + \epsilon}} \tag{2.24}$$

where ϵ is a constant added for numeric stability. To finally produce the output:

$$\tilde{x}_{ij} = \gamma \hat{x}_{ij} + \beta, \ \forall \ \gamma, \beta \in \mathbb{R} \tag{2.25}$$

where γ and β are parameters to be estimated. As we can see, batch normalization becomes another layer of the neural network. At test time, means and the variances estimated during training are used for testing. In consequence, γ, β, $\mu_{\mathcal{B}_j}$, $\sigma^2_{\mathcal{B}_j}$ are constants during testing.

Early Stopping

Early stopping is a regularization technique that stops the training process before the overfitting happens [81]. For this technique, a separate developing set is required. This developing set is used to compute the same loss being optimized or another relevant performance metric to the task (e.g., precision). Using the result of this computation a decision is taken to whether continue training or stop. The decision is taken in two different ways. The first one is checking if the validation loss is going up instead of going down. The second is checking if the validation loss is not changing at all. For both strategies two parameters are defined: patience and $\delta \in \mathbb{R}$. Patience is the number of iterations/epochs to wait while the condition is met before to stop training. δ is the threshold used to decide whether a change in the metric being watched is taken into account or not. In Figure 11 a flowchart with the strategy is shown. The meaning of epochs and iterations is clarified in the Section 2.3.3.

2.3.3 Optimization Algorithms

Stochastic gradient descent (SGD) and their variants are the most widely used algorithms for deep learning training [81]. Stochastic gradient descent is a modification of the original gradient descent [92, 93]. The most significant difference of SDG vs. traditional gradient descent in the use of the *batch* $\mathcal{B}$ (sometimes referred as minibatch). The batch is no more than a set composed by a sample of the training data. Rather than computing the gradient after passing all the data through the network, SGD computes the gradient and modify the weights by only using the batch (see Figure 12). The batch size $|\mathcal{B}|$ is usually greater than one and less than the training sample size. However, smaller sizes are currently preferred since

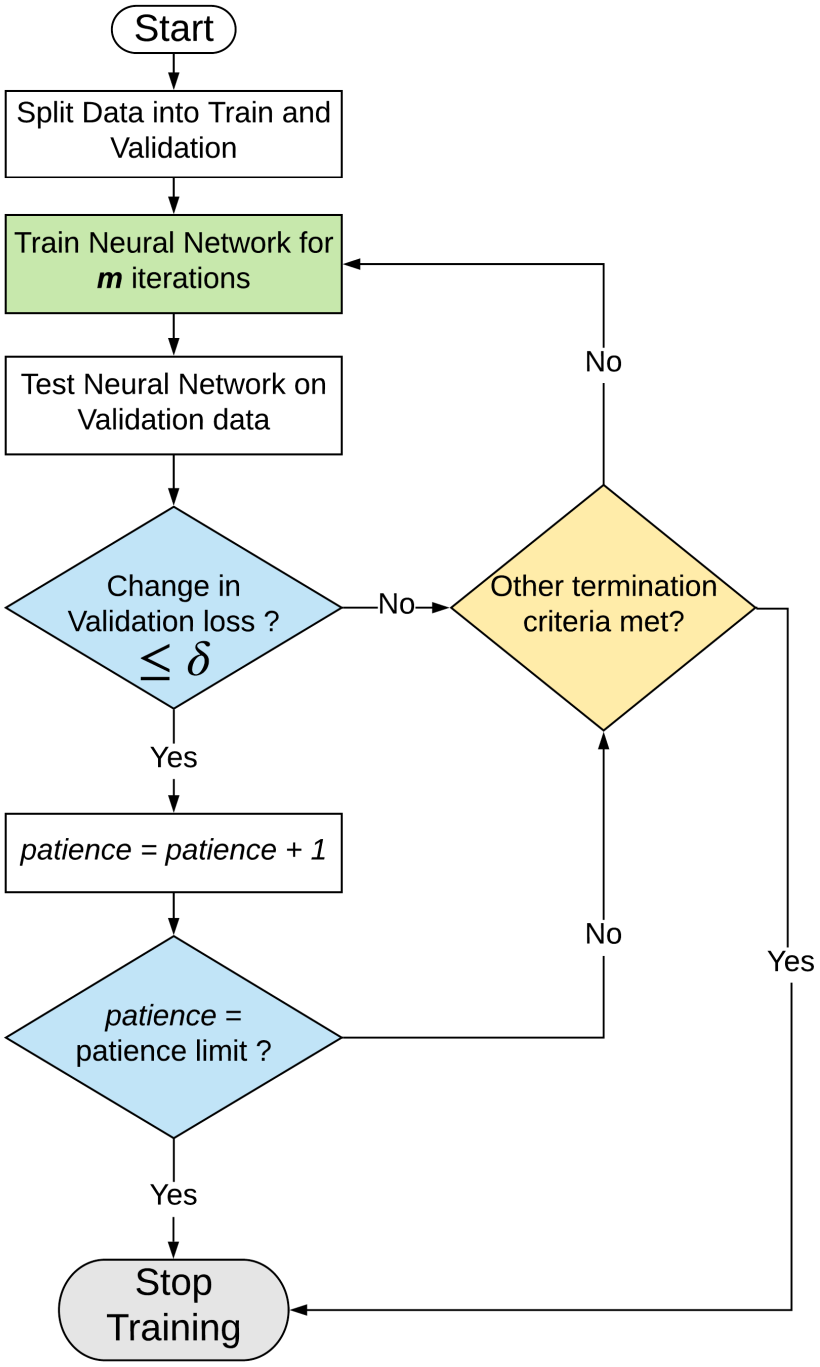


Figure 11: Early Stopping using a Validation Set. Other stopping criteria includes number of epochs, number of iterations and minimum required loss.

Algorithm Stochastic Gradient Descent

Require: Training set formed by $\{\mathbf{x}_1, \ldots, \mathbf{x}_l\}$ inputs and $\{\mathbf{y}_1, \ldots, \mathbf{y}_l\}$ outputs

Require: A function $f(\mathbf{x}_i; \boldsymbol{\theta})$ with parameters $\boldsymbol{\theta}$

Require: Loss Function L

Require: Learning rate η

1: **while** stopping criterion not met **do**

2: Sample without replacement $\mathcal{B}$ with size m from the training set such that $\mathcal{B} = \{(\mathbf{x}_1, \mathbf{y}_1), \ldots, ((\mathbf{x}_m, \mathbf{y}_m))\}$

3: Compute the Loss $\mathsf{L}_{\mathcal{B}} \leftarrow \sum_i \mathsf{L}\left(f(\mathbf{x}_i; \boldsymbol{\theta}), \mathbf{y}_i\right)$

4: Compute the gradient $\hat{\mathbf{g}} \leftarrow \frac{1}{m} \nabla_{\boldsymbol{\theta}} \mathsf{L}_{\mathcal{B}}$

5: update the parameters $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \hat{\mathbf{g}}$

6: **end while**

7: **return** $\boldsymbol{\theta}$

Figure 12: Stochastic Gradient Descent Algorithm with batches

they produce faster convergence [82]. They also improve training efficiency by computing the steps in lines 3 and 4 of Figure 12 in parallel. Given that now with have batches an *epoch* is defined as the moment where all the training data has been used for modifying the weights in the network. In consequence, an epoch is composed of several *iterations*. If the sample taken from the batch is without replacement, we can compute the number of iterations as $\lfloor l/|\mathcal{B}| \rfloor$ where l is the number of samples, and $\lfloor / \rfloor$ is the integer division. If the division is not exact, there are two possible options. In the first option, the last batch is bigger. In the second the samples in the last batch are discarded. To compute the gradient several automatic differentiation techniques have been proposed [94], and they are currently used by popular frameworks such as TensorFlow [95] and PyTorch [94]. The later is the one being used in the present work. Those techniques allow the computation of the gradient without the need to define it explicitly. The only requirement is that the loss function must always be positive and differentiable.

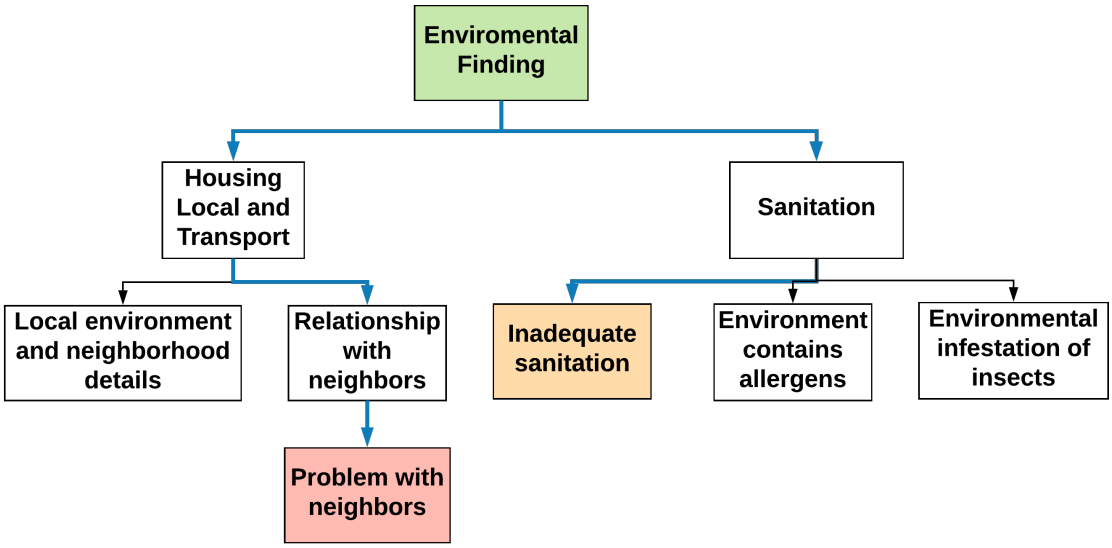


Figure 13: Snapshot of SNOMED CT Ontology. Path between two concepts (blue). For this case the path has a value of five. In green, we can see the least common subsummer of the two highlighted concepts

2.4 SEMANTIC SIMILARITY IN CLINICAL TEXT

Semantic similarity is a metric of how similar are the meanings of two words. This similarity can be measured using distributional methods or ontologies. With distributional methods, we rely on the hypothesis that words occurring in the same context tend to have the same meaning [96]. With ontologies, we rely on the taxonomic structure to define the similarity [97]. In ontologies word of phrases are seen as concepts. This has the advantage that ontologies can be used as a source for word normalization or as a controlled terminology. Those concepts have relationships to other concepts as part of the definition of the ontology. One of those relationships ins the *is-a* relationship that define a hierarchy between the given concepts, e.g., Failed Exams *is-a* Academic Problem. In the biomedical domain, other useful relationships are *part-of* and *treated-by*. In the present work, we will limit to the *is-a* relationship. In the following, metrics to measure the semantic similarity between two concepts in an ontology are presented.

2.4.1 Semantic Similarity using Ontologies

An ontology $\mathcal{O}$ is a set of concepts c_i that belong to a controlled terminology usually organized in a tree-like structure (see Figure 13). The simplest way to compute the semantic similarity between two concepts in the ontology is using the path p between them. The path is defined as the shortest path between two concepts in the tree as seen in Figure 13. Thus, the similarity between two concepts c_1 and c_2 is defined by [98]:

$$sim_{path}(\mathsf{c}_1, \mathsf{c}_2) = \frac{1}{p} \tag{2.26}$$

However, the main drawback of this measure is that does not take into account how big is the ontology. Tho address this issue a new measure was proposed [99] and later normalized to the unit scale [100]. This measure is usually called LCH because of the authors *Leacock & Chodorow*. The following equation defines the metric:

$$sim_{LCH}(\mathsf{c}_1, \mathsf{c}_2) = 1 - \frac{\log p}{\log 2d} \tag{2.27}$$

where d is the maximum number of nodes from the root to any concept. Later on, a new metric was proposed [101] to take into account the granularity of the least common subsumer of two concepts $lcs(\mathsf{c}_1, \mathsf{c}_2)$. *lcs* is the closest common parent of any two given concepts (see Figure 13). The normalized version as available in the NLTK Toolkit [102] is:

$$sim_{wp}(\mathsf{c}_1, \mathsf{c}_2) = \frac{2 * depth(lcs(\mathsf{c}_1, \mathsf{c}_2))}{p - 1 + 2 * depth(lcs(\mathsf{c}_1, \mathsf{c}_2))} \tag{2.28}$$

where the *depth* of a concept is the number of nodes in the path between the root and the concept. Besides paths, we can also use the information content $IC(\mathsf{c}_i)$ to measure the semantic similarity. This metric attempts to quantify information by measuring a ratio between the number of *leaves* and the number of *subsumers*. The *leaves* are a set formed by the descendants of a concept that have no descendants. The *subsumers* is a set formed by all the ancestors of a concept included the concept itself. The information content is defined

by [103]:

$$IC(\mathsf{c}_i) = -\log\left(\frac{\frac{|leaves(\mathsf{c}_i)|}{|subsumers(\mathsf{c}_i)|}+1}{maxLeaves+1}\right) \tag{2.29}$$

where *maxLeaves* is the total number of leaves in the ontology. Using this metric, we can measure the semantic similarity between two concepts by [100]:

$$sim_{LIN}(\mathsf{c}_1,\mathsf{c}_2) = \left(\frac{2 * IC(lcs(\mathsf{c}_1,\mathsf{c}_2))}{IC(\mathsf{c}_1)+IC(\mathsf{c}_2)}\right) \tag{2.30}$$

3.0 SEMANTICS ENHANCED DEEP LEARNING BASED MEDICAL TEXT CLASSIFIER

In this dissertation, we developed a supervised and semi-supervised deep learning strategy. Our strategy aims to classify clinical text incorporating a semantic similarity loss as part of the training process. A corpus of social context sentences inside psychiatric reports was used to assess the performance of the proposed strategy. The main contributions of the proposed strategy are: (1) incorporate a medical ontology as prior knowledge into a deep learning strategy, (2) train a deep learning strategy using such ontology to compute a semantic loss that is used during training, (3) use unlabeled data during semi-supervised training.

3.1 OVERVIEW OF THE METHODS

Deep knowledge-based Semantics Medical Text Classifier is a deep learning strategy for classifying medical text that is composed of two components: a semantic deep network (SDN) and a classification deep network (CDN). The primary task of the SDN is to predict the most probable concepts present in the text. The CDN uses the prediction from SDN to assign the most probable label to the input text. Formally, given a tokenized input document d and a set of n_C class labels $\mathcal{C}_{\mathcal{L}} \in \{1, \ldots, n_C\}$, the strategy will predict $P(\mathcal{C}_{\mathcal{L}}|\mathsf{d})$.

Both networks are trained simultaneously by backpropagation using a loss function that incorporates a semantic loss and a classification loss. The semantic loss function is computed using the output from SDN and a medical ontology (e.g., SNOMED). For the classification loss, we used cross-entropy or a modified hinge loss. Both losses are described in Section 2.3. Details of both networks and how the losses are computed are given in the following

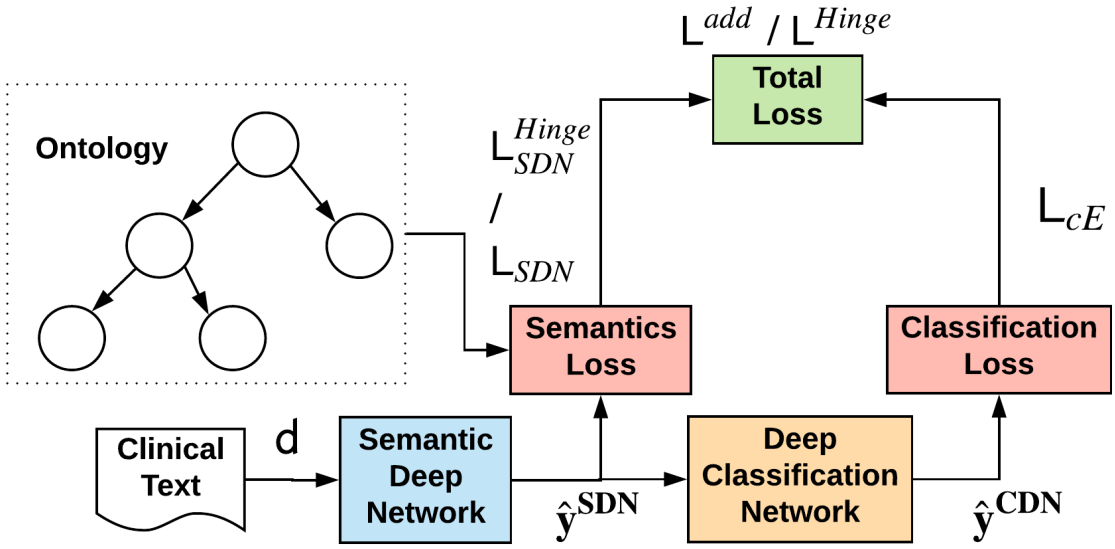


Figure 14: Overall design of the deep learning strategy

sections. In Figure 14, we can see a graphical representation of the strategy.

3.2 SEMANTIC DEEP NETWORK

Given a tokenized input document d as an ordered sequence of n words w_i such that $\mathsf{d} = (w_1, \ldots, w_n)$ the task of SDN is to predict the most probable m ($m <= n$) set of concepts $\mathcal{SC} = \{\mathsf{c}_1, \ldots, \mathsf{c}_j, \ldots, \mathsf{c}_m, \ \forall \ \mathsf{c}_j \in \mathcal{O}\}$ inside d, with $\mathcal{O}$ as the ontology. More formally, given d, SDN predicts $P(\mathcal{SC} = \mathsf{c}_j | \mathsf{d}), \ \forall \ \mathsf{c}_j \in \mathcal{SC}$.

SDN will have two possible architectures CNN and LSTM (see Section 2.2). For both architectures, the first component is the word embeddings. For embeddings constructed with algorithms that consider only entire with words(e.g., word2Vec [59]) they are represented as $\mathbf{WE} \in \mathbb{R}^{N \times a}$ where N is the size of the Vocabulary $\mathbf{V}$, and a is the size of the embeddings. $\mathbf{WE}$ transform d into feature matrix $\mathbf{x} \in \mathbb{R}^{n \times a}$. In the following, the result of the embedding operation over d will be represented as $\mathbf{x} \in \mathbb{R}^{n \times a}$.

As the output of this network regardless of which architecture is used a *sigmoid* activation function is used in the output layer. The *sigmoid* function has some desirable properties such as bounded output $(0, 1)$ and smooth gradient. If we interpret each output as a probability they are mutually exclusive, so in general they not sum up to 1, because d can contain multiple concepts. Each node in the output represents a concept in the ontology, and it is assigned before the training process starts and its keep consistent during all training and evaluation.

However, the issue that emerges here is that the number of possible concepts to predict is the combination $\binom{|\mathcal{O}|}{l}$. To give an example, an ontology such as SNOMED has more than 3×10^5 concepts, if we consider only 1000 and a sequence of length 10 we have 2.63×10^{23} possible predictions. To solve this problem, we will only consider the top k outputs using *k-max* pooling [104]. Formally, given a predefined value k and a sequence $\mathbf{p} \in \mathbb{R}^p$ where $p \geq k$, *k-max* pooling selects a subsequence $\mathbf{p}^k_{max}$ of the k highest values of $\mathbf{p}$ [104]. For the rest of the values, a zero is assigned to nullify the effects of further operations that feed on those nodes. This is similar to what occurs in a dropout layer. In consequence, we can then represent the output of SDN as $\hat{\mathbf{y}}^{\mathbf{SDN}} \in \mathbb{R}^k$ after the *k-max* pooling layer.

The next aspect to consider is how this network is trained. Given that the input is a sequence of words, initially, we do not have the actual concepts that occur within the sequence. To solve this problem I used MetaMap [53] to obtain $\mathcal{SC}$ and use those are labels. The output of MetaMap is only used in conjunction with the loss function in Equation 3.4.

Now given a set of mapped concepts $\mathcal{SC}$, we formulated a supervised loss function using a structured hinge loss [105] as

$$\mathsf{L}^{Hinge}_{SDN} = max(0, s(\mathcal{SC}|\mathbf{x}) - s(\mathcal{SC}_{true}|\mathbf{x}) + \mathsf{S}_\mathsf{L}(\hat{\mathbf{y}}^{\mathbf{SDN}})) \tag{3.1}$$

Where S_L is the semantic loss. The computation of S_L is explained in detail in Section 3.2.1. In this equation $s\left(\mathcal{SC}_{true}|\mathbf{x}\right)$ is computed for the set of true concepts regardless of they being the top k and $s(\mathcal{SC}|\mathbf{x})$ is computed for the top k obtained after the *k-max* pooling layer.

In general the score s is computed as

$$s(\mathcal{SC}|\mathbf{x}) = \sum_{j=1}^{k} \hat{\mathbf{y}}_j^{\mathbf{SDN}} = \sum_{j=1}^{k} P(\mathcal{SC} = \mathsf{c}_j|\mathbf{x}) \tag{3.2}$$

3.2.1 Semantic Loss Function

In this function, we wanted to capture how semantically similar is an input document to a target class defined as a set of concepts in an ontology. We wanted this loss function to have a $[0, 1]$ range. This loss is defined as an inverse measure of a semantic similarity because we want the most similar elements to have zero loss. With this bounded range, it resembles a distance function. In the following, we will treat this loss as a distance function.

A target class is usually defined as an integer label which does not adequately represent everything that a class comprises. Here we attempt to extend the definition of a target class. In this work we defined a target class as a set of concepts in an ontology. Formally let us represent a target class as $\mathcal{C}_\mathcal{O} \subset \mathcal{O}$. The elements in $\mathcal{C}_\mathcal{O}$ are selected by an expert clinician and should represent his knowledge about a particular condition of interest. In this work, we limit the expressiveness to a set of concepts without further using first-order logic (FOL). By doing this, we exclude properties on the concepts such as negation.

Given $\mathcal{C}_\mathcal{O}$ and $\mathcal{SC}$, we used a distance [106] that takes into account the hierarchy of the ontology. The loss can be computed using the following equation:

$$\mathsf{S}_\mathsf{L}(\hat{\mathbf{y}}^{\mathbf{SDN}}) = \frac{1}{|\mathcal{SC} \cup \mathcal{C}_\mathcal{O}|} \left[\sum_{\mathsf{c}_j \in \mathcal{SC} \setminus \mathcal{C}_\mathcal{O}} \frac{1}{|\mathcal{C}_\mathcal{O}|} \sum_{\mathsf{c}_i \in \mathcal{C}_\mathcal{O}} d(\mathsf{c}_i, \mathsf{c}_j) + \sum_{c_i \in \mathcal{C}_\mathcal{O} \setminus \mathcal{SC}} \frac{1}{|\mathcal{SC}|} \sum_{\mathsf{c}_j \in \mathcal{SC}} d(\mathsf{c}_i, \mathsf{c}_j) \right] \tag{3.3}$$

Here $d(\mathsf{c}_i, \mathsf{c}_j)$ is the distance between two concepts which is computed using the metric presented in equation 2.30.

Incorporating the hierarchy in the semantic loss is used to inform about similarities that otherwise may be hindered. Let's use as an example the following sentences *she will now not be able to go to college*, and *She did not graduate from special education at the high school level*. The mapped concepts from those sentences are *[College (C0557806), Able (C1299581)]*, and *[Special Education (C0013649), High School Level (C0683862)]*. If we use a metric such

as the Jaccard distance, that does not take into account the hierarchy, there is no similarity between those sentences because they do not share common concepts. The concepts in both texts are different, but it is obvious that they are talking about academic problems. What is needed to know this is a hierarchy that reflects this similarity, because without them there is no relationship between those sentences because there is no intersection. So this semantic loss is not directly aiding the prediction of the concepts made by the SDN but the classification task by informing about similarities in the input text.

3.3 CLASSIFICATION DEEP NETWORK AND TOTAL LOSS FUNCTION

Given the output of the SDN as $\hat{\mathbf{y}}^{\mathbf{SDN}}$ the classification network implements a non-linear function $f : \mathbb{R}^k \rightarrow \mathbb{R}^{n_C}$ to classify the input text in one of each n_C classes. The architecture for this network is a Feed Forward Network as described in Section 2.2.1. The output of this network is a *softmax* .

For training, we proposed two functions: an additive loss and a hinge loss. The additive loss is computed using the following equation:

$$\mathsf{L}^{add} = \mathsf{L}_{cE}(\hat{\mathbf{y}}^{\mathbf{CDN}}, \mathbf{y}) + \lambda * \mathsf{L}^{Hinge}_{SDN} \tag{3.4}$$

where L_{cE} is the cross-entropy loss (Equation 2.18), λ is a scalar, $\hat{\mathbf{y}}^{\mathbf{CDN}}$ is the predicted output and $\mathbf{y}$ is the true output. The hinge loss is defined by:

$$\mathsf{L}^{Hinge} = max(0, s(\hat{\mathbf{y}}^{\mathbf{CDN}}|\mathbf{x}) - s(\mathbf{y}|\mathbf{x}) + \mathsf{S}_{\mathsf{L}}(\hat{\mathbf{y}}^{\mathbf{SDN}})) \tag{3.5}$$

where $s(\hat{\mathbf{y}}^{\mathbf{CDN}}|\mathbf{x})$ is the score for the output with the maximum predicted value and $s(\mathbf{y}|\mathbf{x})$ is the score for the predicted true output. In this case, the score is the unnormalized output of CDN. As an example, let us say we have two outputs and the true label is 1. If the output of the network is $[10, 20]$ the maximum value is in index 2. In this case, the predicted score is 20, and the score for the correct output is 10. The scores can be computed using the unnormalized output of the network. This means the score is computed before applying the *softmax* .

3.4 SEMI-SUPERVISED STRATEGY

To use unlabeled samples to improve generalization we used to three methods. The first one was training word embeddings using the method described in Section 2.2.2. The second has to do with hyperparameters search and will be explained in Section 3.5. The third is presented in this section, and it is a strategy that directly deals with unlabeled samples while training a text classifier. The semi-supervised strategy we used belongs to a subset of binary classification strategies called positive and unlabeled (PU) learning [107]. In this mode of learning, we have two sets available for training: positive and unlabeled. The positive set contains samples adequately labeled as belonging to a class of interest. The unlabeled set contains samples that could be either positives or negatives. By negatives, we mean samples that do not belong to the class of interest. To evaluate the overall performance of the strategy a small subset of positive and negative samples are usually obtained.

In this work, we modified a recently proposed strategy [108] for PU learning that uses text data. In this strategy, an iterative process is proposed where probable positive samples are removed from the unlabeled set. In the previous work, they used an FFN in combination with part of speech tags to make predictions for single words. In this work, we used a CNN and the proposed strategy as the classifiers. The algorithm for the strategy is shown in Figure 15.

In step 5 a binary CNN is trained or a binary version of the proposed strategy. The threshold in step 8 is computed by assuming a Gaussian distribution of the predicted probabilities for the spy samples. Thus ϕ is computed by using the maximum likelihood estimation of the mean μ and the standard deviation sd as shown in the following equations:

$$\begin{aligned} \phi &= \mu + \text{sd} \\ \mu &= \frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} s_{\mathcal{S}_i} \\ \text{sd} &= \left(\frac{1}{|\mathcal{S}|} \sum_{i=1}^{|\mathcal{S}|} \left(s_{\mathcal{S}_i} - \mu \right)^2 \right)^{1/2} \end{aligned} \tag{3.6}$$

Algorithm SE-CNN

Require: Training set formed by $(\mathbf{x}_i, \mathbf{y}_i),\ \forall\ \mathbf{x}_i \in \mathbf{X} \wedge \mathbf{y}_i \in \mathbf{Y}$ input-output pairs

$\mathcal{P} \subset \mathbf{X}, \mathbf{Y},\ \forall\ \mathbf{y}_i = +1$ ▷ Positive Set

$\mathcal{U} \subset \mathbf{X}, \mathbf{Y},\ \forall\ \mathbf{y}_i = -1$ ▷ Unlabeled Set

$\mathcal{S} \leftarrow \mathit{RandomSample}(\mathcal{P}, \gamma)$ ▷ $\mathcal{S}$: Spy Set, γ: Proportion of $\mathcal{P}$ to sample

Assign $\mathbf{y}_i \leftarrow -1,\ \forall\ \mathbf{y}_i \in \mathcal{S}$

while $|\mathcal{S}| \geq \delta * \gamma * |\mathcal{P}|$ **do** ▷ $\delta \in (0, 1)$, $\gamma * |\mathcal{P}|$ is the original size of $\mathcal{S}$

  $\mathcal{U}_s \leftarrow \mathcal{U} \cup \mathcal{S}$ ▷ Unlabeled Set plus the Spy set

  $\boldsymbol{M} \leftarrow CNN(\mathcal{P}, \mathcal{U}_s)$ ▷ Train a CNN

  $s_{\mathcal{S}} = s(\boldsymbol{M}, \mathcal{S})$ ▷ Predicted probabilities $p(\mathbf{y}_i = 1|\mathbf{x}_i),\ \forall\ \mathbf{x}_i \in \mathcal{S}$

  $s_{\mathcal{U}} = s(\boldsymbol{M}, \mathcal{U})$ ▷ Predicted probabilities $p(\mathbf{y}_i = 1|\mathbf{x}_i),\ \forall\ \mathbf{x}_i \in \mathcal{U}$

  $\phi = \text{ComputeThreshold}(\mathcal{S})$

  for $\mathbf{x}_i \in \mathcal{U}$ **do**

    if $p(\mathbf{y}_i = 1|\mathbf{x}_i) > \phi$ **then**

      $\mathcal{U} \leftarrow \mathcal{U} - (\mathbf{x}_i, \mathbf{y}_i)$

    end if

  end for

  for $\mathbf{x}_i \in \mathcal{S}$ **do**

    if $p(\mathbf{y}_i = 1|\mathbf{x}_i) > \phi$ **then**

      $\mathcal{S} \leftarrow \mathcal{S} - (\mathbf{x}_i, \mathbf{y}_i)$

    end if

  end for

end while

Figure 15: Spy-based elimination of positive class instances using CNN

3.5 HYPERPARAMETERS SEARCH

Given the extreme importance of the hyperparameters in the performance of deep learning strategies [109, 110, 111] we performed a random search [110] to find the best set of hyperparameters using the developing set.

For the Binary dataset, we used the F_1 score as the criteria to maximize given a set of hyperparameters. However, for the PU dataset, the *true negatives* are unknown during training. For this reason, a loss or a risk function is needed to guide the search by only using positive and unlabeled examples. Recently a non-negative risk estimator for positive and unlabeled learning (NNPU) was developed [112]. This risk guarantees a reduction in the mean squared error. We selected the hyperparameters with the minimum risk. The risk can

be estimated using the following equation:

$$\widetilde{R}_{\mathrm{pu}}(g) = \pi_{\mathrm{p}}\widehat{R}_{\mathrm{p}}^{+}(g) + \max\left\{0, \widehat{R}_{\mathrm{u}}^{-}(g) - \pi_{\mathrm{p}}\widehat{R}_{\mathrm{p}}^{-}(g)\right\}. \tag{3.7}$$

In this equation π_{p} is the positive class prior or prevalence, and $\widehat{R}_{\mathrm{p}}^{+}(g)$, $\widehat{R}_{\mathrm{u}}^{-}(g)$, and $\widehat{R}_{\mathrm{p}}^{-}(g)$ can be estimated by:

$$\widehat{R}_{\mathrm{p}}^{+}(g) = \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}}\ell(g(\mathbf{x}_i^{\mathrm{p}}), +1) \tag{3.8}$$

$$\widehat{R}_{\mathrm{u}}^{-}(g) = \frac{1}{n_{\mathrm{u}}}\sum_{i=1}^{n_{\mathrm{u}}}\ell(g(\mathbf{x}_i^{\mathrm{u}}), -1) \tag{3.9}$$

$$\widehat{R}_{\mathrm{p}}^{-}(g) = \frac{1}{n_{\mathrm{p}}}\sum_{i=1}^{n_{\mathrm{p}}}\ell(g(\mathbf{x}_i^{\mathrm{p}}), -1) \tag{3.10}$$

Here $\mathbf{x}_i^{\mathrm{p}}$ and $\mathbf{x}_i^{\mathrm{u}}$ are the positive and unlabeled input samples respectively, g is a decision function for binary classification. In other words, g is the function that applies out proposed text classifier to the input feature vector. Finally, $\ell : \mathbb{R} \times \{\pm 1\} \to \mathbb{R}$ is a loss function that for our case is a sigmoid of the form

$$\ell_{\mathrm{sig}}(y, t) = \frac{1}{1 + \exp(yt)} \tag{3.11}$$

where $y = 2 * \arg\max(\hat{\mathbf{y}}^{\mathbf{CDN}}) - 1$ and $t = 2 * \arg\max(\mathbf{y}) - 1$. These transformations are done because those equations need an -1/1 type output.

3.6 SOCIAL CONTEXT SENTENCE CORPUS FROM PSYCHIATRIC REPORTS

To assess the performance of the proposed strategy we used a previously created corpus of social context sentences from psychiatric reports. The goal of this corpus is to train text classifiers capable of classifying sentences in psychiatric reports that contain social context information. To create this corpus, we developed the description for social context using

SNOMED and DSM-IV. From DSM-IV, we used the axis IV description of psychosocial and environmental problems. We excluded patient psychiatric symptoms, psychiatric disorders and behaviors strongly linked to psychiatric disorders which might otherwise be considered social context (e.g., violence and drug abuse) to avoid overlapping with psychiatric clinical findings. Our description extends the DSM-IV axis IV because it includes positive and negative stressors without explicitly considering their impact on the diagnosis, treatment, and prognosis of mental disorders.

We used SNOMED CT US version present on the unified medical language system (UMLS) 2017AA release [113]. To build the description we considered two places inside the SNOMED hierarchy: social context, and clinical findingfinding by methodhistory findingsocial and personal history finding. Below those two places in the hierarchy, there are 6463 concepts. We used SNOMED to explicitly find concepts semantically similar to categories defined in DSM-IV axis IV, but without ruling out concepts that are not problems. We finally defined eleven types (see Table 2) that relate to some SNOMED concepts and DSM-IV categories defined in axis IV.

The list in Table 2 is not comprehensive but instead provides examples of some of the concepts considered in SNOMED. In summary, we described a persons social context as situations, conditions and environment of a patient and his relationship with the society which may influence peoples health outcomes [114, 115, 116].

Later we developed a guideline to annotate sentences inside psychiatric inpatient discharge summaries containing social context information according to our description. The guideline includes specifics about each type and several examples taken from a subset of reports. We refined the guideline using an iterative process. In the training phase, the two annotators independently applied the guideline to 4 reports using the software e-Host [117]. After that, the annotators and the research team met to review the disagreements. In this agreement meeting, a consensus was reached related to the changes needed to the guideline. The process was repeated in phase one with 18 more discharge summaries, and the inter-annotation agreement (IAA) was measured over these 18 reports. After reaching a final consensus, phase II started with the refined guidelines. The corpus was constructed using the annotations from phase two to phase six as shown in Figure 16.

Social context type	DSM-IV axis IV Category	Related SNOMED concept (CUI code)
Education	Educational problems	Education and/or schooling finding (C1287119)
Other	Other psychosocial and environmental problems	Prisoner of war, life event (C0425175)
Occupational	Occupational problems	Employment finding (C0578791)
Housing	Housing problems	Residence and accommodation circumstances (C1268616)
Interaction with legal system	Problems related to interaction with legal system/crime	Criminal Life Style (C0524399)
		Legal affairs and legal constraints (C1287222)
Health care	Problems with access to health care services	Person in the healthcare environment (C0580210)
		Inadequate community resources (C1828305)
Social environment	Problems related to social environment	Person in family of subject (C2732431)
		Family (C0015576)
Support circumstances and network	Problems with primary support group	Partner in relationship with subject (C2733152)
		Finding related to care and support circumstances and networks (C1287142)
Economic	Economic problems	Economic status (C0337781)
		Financial circumstances (C1287207)
Spiritual life	*	Person categorized by affiliation with belief system (C2585518)
		Religion AND/OR philosophy (C0574831)
Transportation	*	Access to transport and related findings (C0578828)

Table 2: Social context types with related DSM-IV axis IV categories and SNOMED concepts

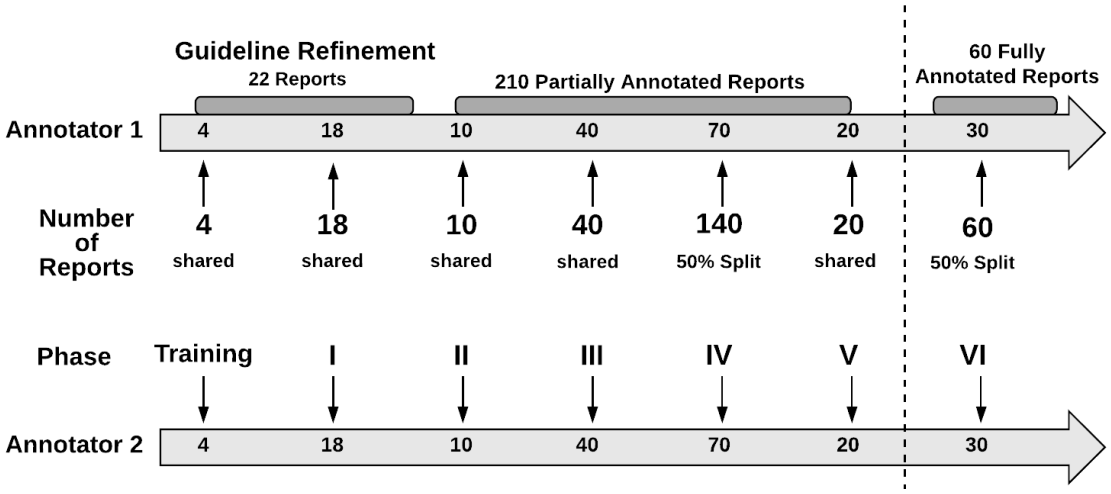


Figure 16: Summary of Annotation of social context sentences

For the training corpus, only the first sentence containing a mention indicative of a type of social context information was annotated in each report. For example, for a given report, the first sentence containing the word mother is annotated; subsequent sentences in the report mentioning mother are not annotated as sentences containing social context information. We decided this with the intent to speed up the annotation process and collect a wider variety of samples. In our case variety means collecting information from possibly more patients or different conditions from the same patient. When collecting information from more patients, the assumption is that a wider variety of social contexts could occur more between patients than within patients.

As a result, the reports from phase 2 to 5 contain two sets of annotations that we define as positive and unlabeled. Positive annotations are sentences annotators explicitly labeled as containing a social context. Unlabeled are sentences annotators did not annotate as containing social context. These unlabeled sentences could have social context information.

The test corpus was fully annotated, this means the annotator read and annotate the entire report. This test corpus contains two sets of annotations: positive and negative. As before, positive annotations contain sentences annotators explicitly labeled as containing a social context. Negative annotations contain sentences annotators did not annotate as

	Phases II to V	Phase VI
Number of clinical reports	210	60
Number of sentences	6879	2249
Number of social context sentences	816	435
Number of **non**-social context sentences	N/A	1814
Number of unique words	1435	-

Table 3: Characteristic of Social Context Sentence Corpus

having social context information.

Two board-certified psychiatrists annotated 270 reports. The IAA for phases I, II, III was 47.3%, 71%, and 62.1% respectively. The final IAA measured on phase V was 75.4%. In the 270 discharge summaries, 817 sentences containing social context information were annotated. Finally, for the test set, 60 discharge summaries were annotated giving 30 to each one of the annotators. In this part, each report was fully annotated. In the test set, 436 sentences containing social context information were annotated.

A common psychiatric discharge summary is formed by different sections and a set of structured questions that depend on of each patient condition. Because of this, we used a customized developed sectionizer to obtain the unlabeled/negative sentences from all the phases. Is essential to identify sections given that social context information is more likely to be found in sections such as *reason for referral*, and *narrative summary of treatment course*. Some section titles include information that is considered social; however, the title itself does not provide information about the patient. For example,a section named *legal services*, if naively parsed, could lead to affirm that the patient has some legal services in place when the content of the section could be completely empty. We found a total of 91 possible sections that a report could contain. The average number of sections per report was 71. Using those identified sections alone with the developed sectionizer we extracted the sentences from reports to use them as unlabeled/negative samples.

From this corpus, we created two different datasets: **PU** and **Binary**. The PU dataset was formed by using the samples from phases II to V as part of the training and developing

sets. The test dataset was formed by using only the samples from phase 6. The Binary dataset was formed by combining the positive samples from phases II to VI and using only the negative samples from phase 6. Those were later randomly partitioned into train, develop, and test sets.

4.0 EVALUATION

In the following chapter, we describe the evaluation steps taken to assess the performance and gain insights into the functioning of the proposed algorithm. The focus of the evaluation is to show the viability of the proposed loss function and their usefulness, particularly in the semi-supervised scenario. First, we describe the different experiments and their purpose along with the metrics used. Following the description of the experiments, the results of each one of the main components of this work are presented.

4.1 EXPERIMENTS AND METRICS

To evaluate the model we used two datasets (PU and Binary) described in Section 3.6. For both datasets we split the data into training, developing and testing sets. For the PU dataset, the developing set is 20% of the total training data. For the Binary dataset, the developing set is 10%, and the test set is 20%. In both datasets, the developing set is used to tune the hyperparameters and also perform early stopping.

To compare the performance of the proposed strategy in the semi-supervised setting we choose four baselines, with two of them used previously in text classification tasks [118]. The first one is Biased PrTFIDF (B-Pr) [119] which is a probability-based algorithm. The second is biased-SVM (bSVM) [107]. For bSVM we used the traditional *tf-idf* feature matrix, and for B-Pr we used word counts. The third strategy that we called *dictionary lookup*, directly used the output from MetaMap. In the dictionary lookup strategy, any sentence that contained at least one of the concepts defined as a social context class is assigned to this class. This strategy does not need a training corpus. The fourth algorithm, Spy-elimination

CNN (SeCNN) is a strategy we developed, but it does not use the novel semantic loss function. This strategy uses the recursive elimination strategy presented in Section 3.4 in combination with a CNN presented in Section 2.2.3. All the hyperparameters were optimized using the NNPU loss function and the strategy described in Section 3.5.

For the supervised setting three baselines were used. A standard SVM with a *tf-idf* feature matrix and CNN and a dictionary lookup that works in the same way than for the semi-supervised setting. The hyperparameter search is also performed with random search, but the optimized metric is the F_1 score instead of the NNPU loss. Once the hyperparameters for each model is selected the final performance of all the strategies is evaluated in the correspondent test set, which is worth mentioning again, is different for the PU and Binary datasets. To evaluate the model, we focused on the F_1 score given that the prevalence of the class is low, so a balanced metric such as the AUC is usually not adequate for text classification tasks. For the best performing model of the proposed strategy, we run experiments were training is performed with only a portion of the available data, and tested using the same test dataset. We used 25, 50 and 80 percent of the training data to quantify the impact of the reduction. For the PU dataset, only positive samples are reduced since those are the ones costly to produce. The number of unlabeled samples was kept constant. For the Binary dataset, both positive and negative samples were reduced. The complete set of experiments is shown in Table 4.

The experiments were run in a desktop computer located at the Department of Biomedical Informatics at the University of Pittsburgh with an Intel i7 @3.6GHz, 32GB of RAM and an NVIDIA 1070Ti graphics card. The majority of the code was written in Python using Scikit Learn [120] and Pandas [121] for machine learning and data manipulation, PyTorch [94] for deep learning, gensim [122] for obtaining the embeddings, and NLTK [102] for natural language processing standard tasks. The semantic similarity was computed using the semantic measures library and toolkit [123] written in Java.

Method	Dataset
BPr	PU
bSVM	PU
Dictionary Lookup	PU and Binary
Se-CNN	PU
SVM	Binary
DMT + Add + LSTM	PU and Binary
DMT + Hinge + LSTM	PU and Binary
DMT + Hinge + CNN	PU and Binary
DMT + Add + CNN	PU and Binary

Table 4: Full set of experiments performed to evaluate the proposed strategy. DMT: Proposed Strategy. Add: Additive Loss from Equation 3.4. Hinge: Hinge loss from Equation 3.5. LSTM/CNN: Architecture for the Semantic Deep Network.

4.2 RESULTS

4.2.1 Word Embeddings Training Results

We trained word embeddings from MIMIC III [124], a freely available clinical database that contains more than 2 millions clinical notes that are properly de-identified. Those have been used previously [17] showing a better performance than embeddings trained with general text. To train the word embeddings we used word2vec, an algorithm described in Section 2.2.2. The word2vec version used was the one available in the python package gensim [122]. The reports were preprocessed using a custom pipeline based on NLTLK [102]. At the string level we lowercased all characters, removed newline characters and double spacings and substituted all numbers with the ♯character. After this procedures, we split the reports into sentences and tokenized the text. The sentence split was performed in an attempt to reduce unrelated words to be related by being within the context window. We used almost all the defaults parameters of the algorithm and only changed the minimum word counts to 10 and the size of the embeddings to 200. Properly chose the hyperparameters of the

	Query Word		
	homeless	**divorce**	**bankruptcy**
Nearest Words	shelter	bankruptcy	foreclosure
	shelters	eviction	eviction
	estranged	unemployment	evicted
	sophomore	probation	freelance
	widowed	murder	budget
	banned	evicted	employement
	divorced	marriage	profession
	roommates	teenagers	victims
	freshman	teachers	airline
	alcohol	economy	parole

Table 5: Example for three words and their closest words in the vocabulary

algorithm is not an easy task given that a proper evaluation of the embeddings is a task that requires domain knowledge, an in this case physicians. A recent study does this evaluation with a corpus obtained in Mayo Clinic [125]. This embeddings evaluation is not the focus of this work. Our evaluation of the embeddings will be limited to an inspection of similarity between words of interest.

71,825 unique words compose the resulting word embeddings. As an inspection of the semantics captured, it is useful to choose some words of interest and see what the closest to those are. In Table 5 are shown words with their top 10 closest words by using cosine similarity between the word vectors.

To visualize the embeddings we used principal component analysis to reduce the dimensionality of the vectors to 2. To visualize how related words form clusters, we plotted the embeddings for the words homeless and diabetes and their 600 closest words in the original embedding space. In Figure 17 we can see the separation between the clusters of the related words with some minor overlap. We can also zoom to their ten closest words in the bottom of Figure 17. This inspection suggests the embeddings are actually capturing the semantics of such words and consequently showing their usefulness for clinical text classification.

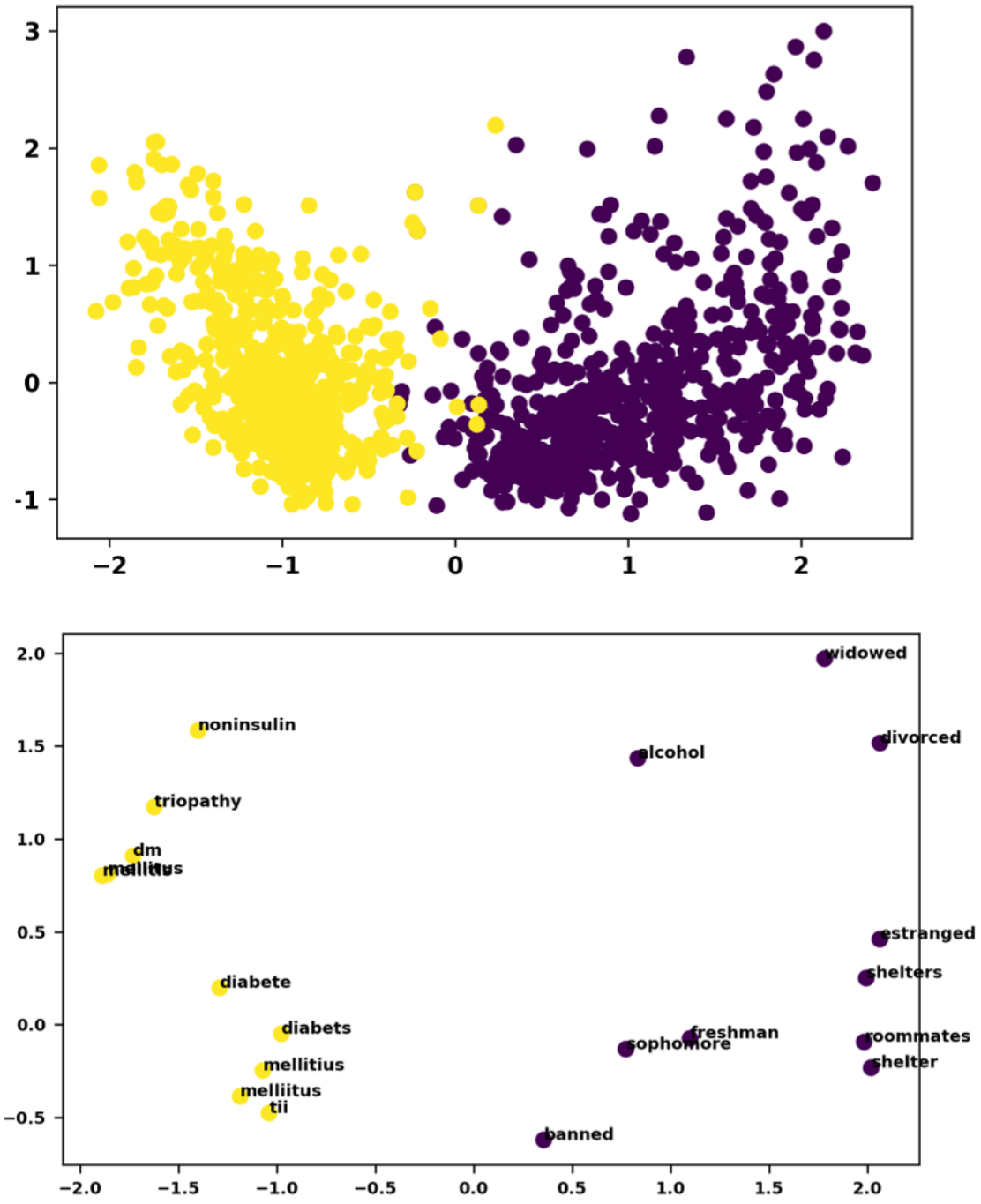


Figure 17: Visualization of the word embeddings in a reduced dimensionality space for the 600 closest words to homeless(purple) and diabetes(yellow). In the bottom we can observe a closeup for the closest 10 words

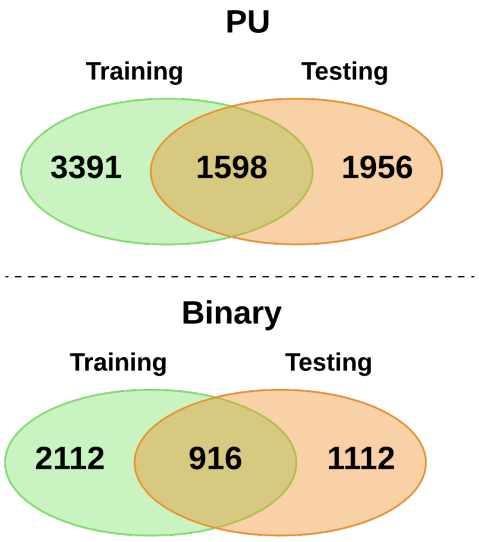


Figure 18: Number of concepts in training and testing for PU and binary datasets

4.2.2 Semantic Distance Results

Metamap Mapping

We used Metamap [53] to map each one of the sentences to CUIs for both datasets. In Figure 18 we can see the distribution of concepts in training and testing for both datasets. The main difference between the absolute number of unique concepts is the larger sample size of the PU dataset. For the PU dataset we a had a total of 3749 CUIs, and for the Binary we had 2308 unique CUIs.

If we consider the entire SNOMED, the social context class contains 6462 possible CUIs. However, in our case, only 209 and 170 occurred in the PU and binary datasets respectively.

For both datasets, in Figure 19 we can see that the distribution of CUIs is heavily skewed with some CUIs occurring the majority of the time. However, to better observe the impact of this phenomenon on the entire corpus in Figure 20 it can be seen that less than 1% have a prevalence higher than 5%.

Finally, in Figure 6 we can appreciate the output of MetaMap for some of the sentences. In general, some of the concepts related to words like *mother*, *boyfriend*, and *family* are

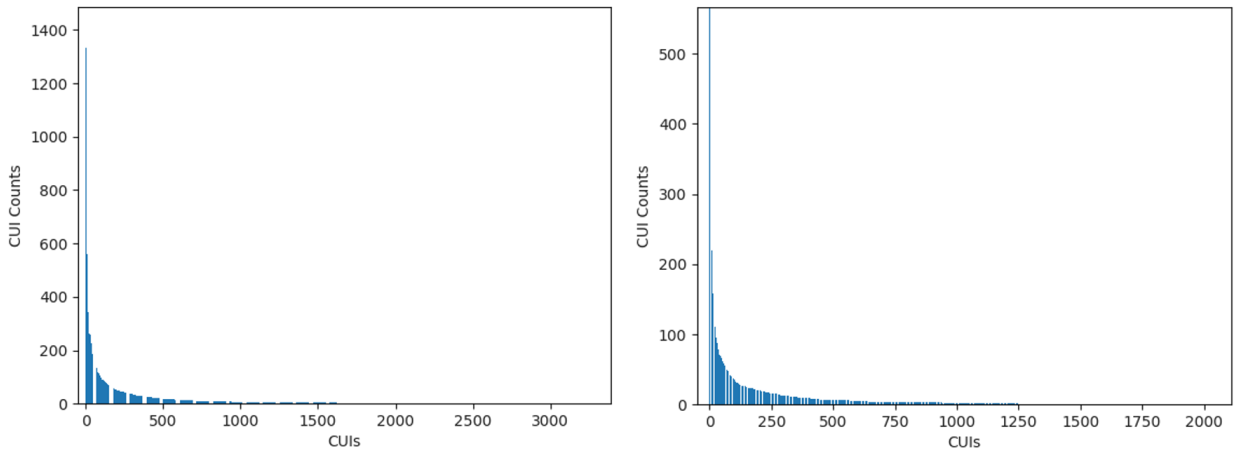


Figure 19: Histograms for the distribution of CUIs in PU(left) and binary(right) datasets for the training set. IN the x axe we have the CUI index instead of the full code to facilitate the display

appropriately mapped most of the time. Some others like the abbreviation for patient *pt* and the words *stated* and *to* are mapped to possible expansions that are clearly not relevant in this context.

Semantic Distance

With this mapping, we proceed to asses whether the semantic distance metric from Equation 3.3 has the ability to distinguish between the different classes. To do this, we computed the semantic distance of each sample in the training set to the social context class. The results for both datasets are shown in Figure 21.

With those distances computed, we then proceeded to use a non-parametric test to see whether the two distributions are different. We used two well-known tests available in the python package SciPy [126]: The Kolmogorov-Smirnoff test and the Mann-Whitney rank test. For both datasets, p-values were zero.

Finally, given that only a small subset of the social context class occurred in both datasets we decided to investigate the semantic relationship between those. In this case, we used the pairwise similarity between each of the concepts measured using equation 2.30. Those similarities can be seen in Figure 22. The big block of blue suggest that most of the CUIs

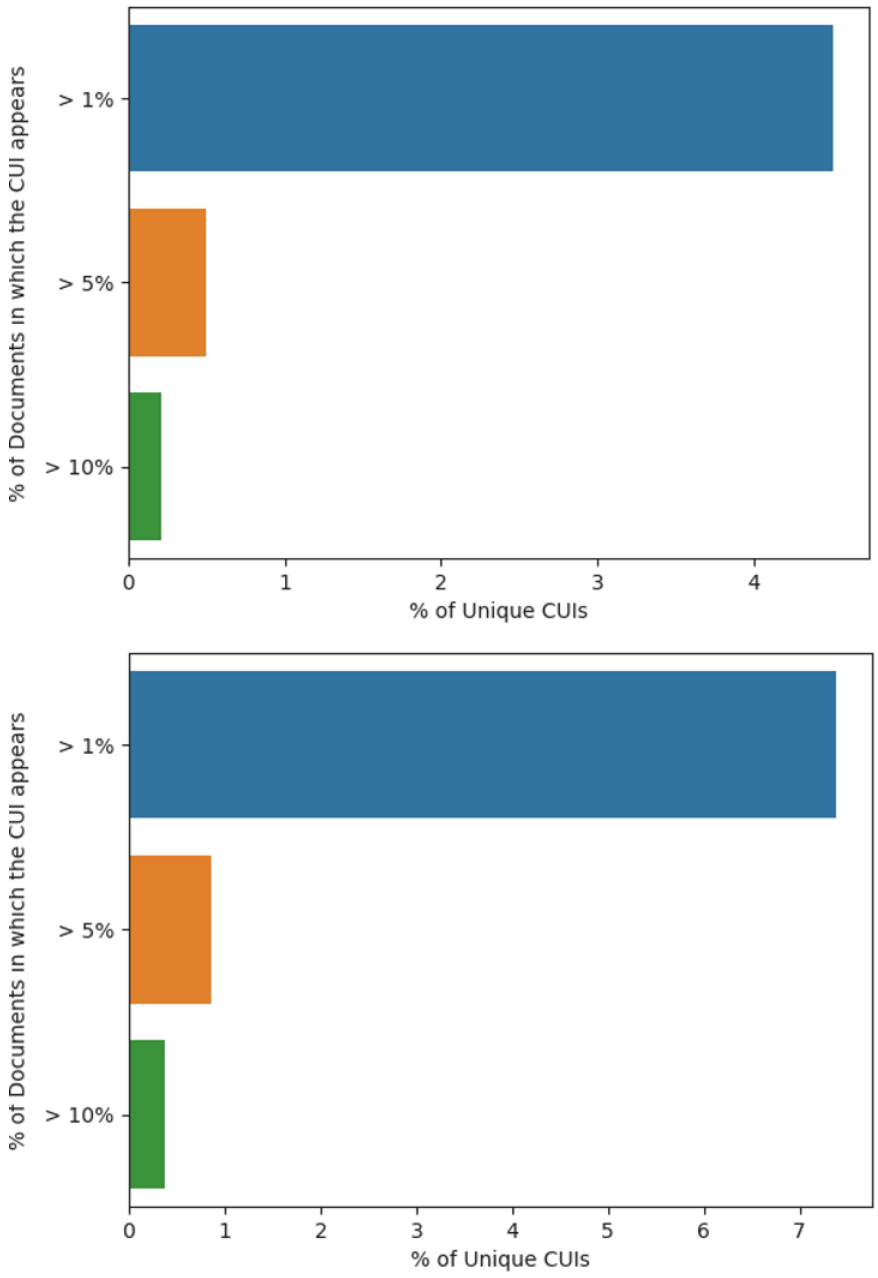


Figure 20: Prevalence of CUIs in PU(up) and binary(bottom) datasets

Sentence	Mapped Concepts
lives with mother and mother's boyfriend	Mother, Lives with mother, Boyfriend
Pt currently lives with his grandmother and reports problems trying to get medicaid	Medicaid, Positron-Emission Tomography, Togo, Tryptophanase, Current, Physical therapy, Report, Reporting, Problem, Lives with grandmother
In terms of current symptoms mother notes that pt has been hearing voices and holding conversations with nonexistent voices and people	Positron-Emission Tomography, Mother, Persons, Physical therapy, Verbal auditory hallucinations, Current, Symptoms, Hold - dosing instruction fragment, vocal
He stated that he is estranged from his family and his mother	Mother, Family, Geographic state
We collaborated with her family, mother reports gradual decline over a 5 year interval, with loss of employment as nurse and strained relations among family and friends as her paranoia persisted.	Family, Mother, Nurses, Paranoia, friend, Report (document), Reporting, Relationships, Muscle strain, Gradual, Interval, per year, year

Table 6: Examples from the mapping performed by MetaMap

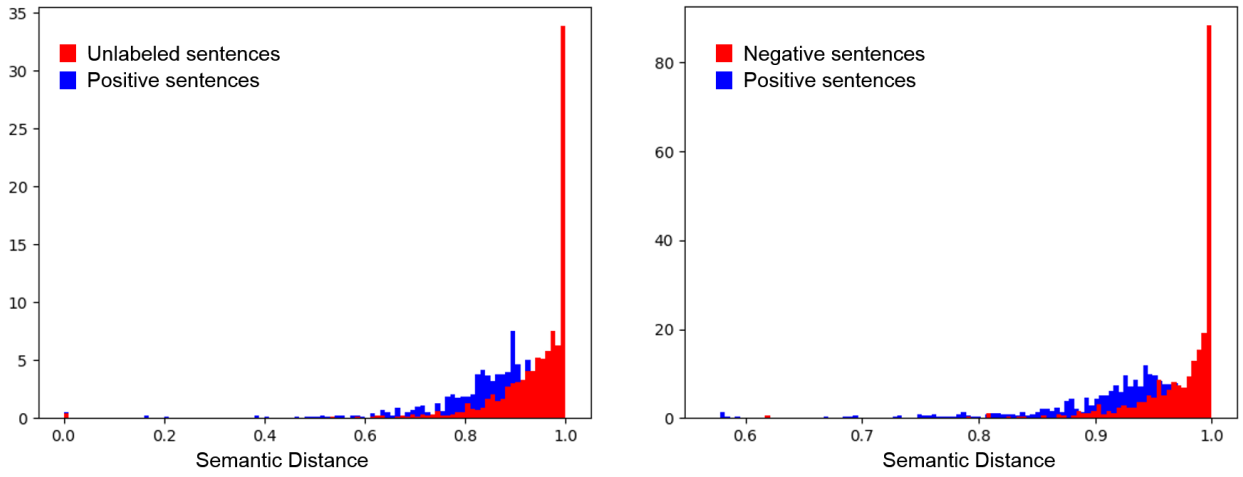


Figure 21: Histogram of the semantic distances to the social context class in PU(left) and binary(right) datasets. The area of histograms is normalized to account for the differences between the number positive(blue) and unlabeled/negative (red) samples

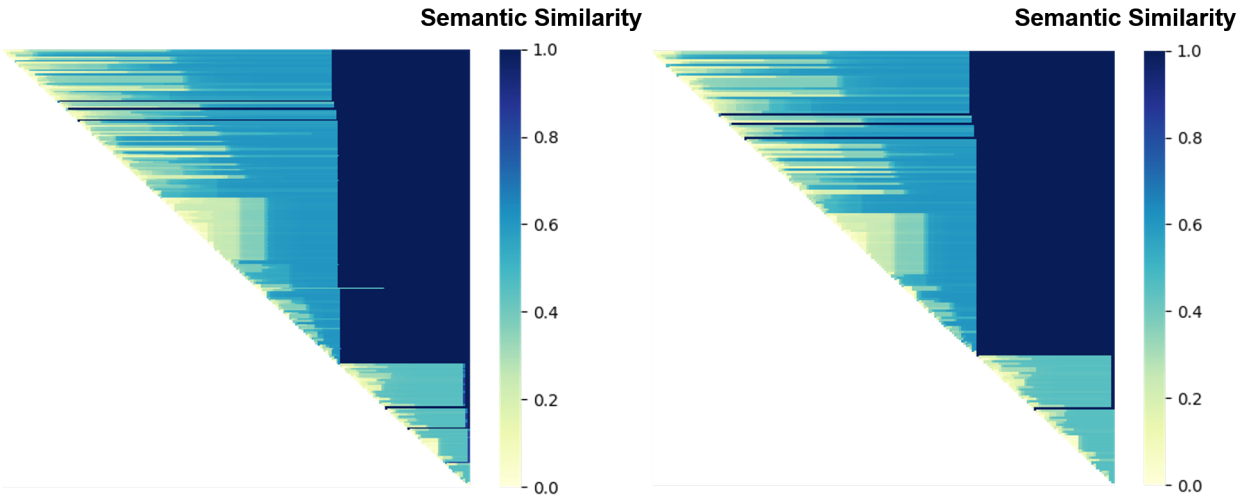


Figure 22: Pairwise semantic similarity of CUIs in the social context class in PU(left) and binary(right) datasets.

have the same distance with the least common ancestor and have the same number of leaves. This frequently happens with concepts that have multiple descendants with no children themselves. This plot also suggests that most of the concepts are occurring in a similar place in the ontology.

4.2.3 Classification Results

For the semi-supervised setting, the proposed strategy outperformed all the others except when the dataset is very limited. Taking a closer look at the performance of the winning strategy when trained with 100% of the available data, the performances observed are 0.73, 0.76, 0.78, 0.79, and 0.82 for iterations one to five respectively. We can also observe, by looking at the F_1 scores that the elimination procedure is working as expected given that the performance of the last iteration is better than the performance of the first.

For the supervised setting, the proposed strategy had a better performance than the SVM but not better than the CNN. In most of the cases, the performance of the proposed strategy degraded with less data, while the performance of the CNN stayed almost the same

except when 25% of training data was used.

Finally, the behavior of the proposed loss function can be seen in Figure 23. In the figure both the total and the semantic loss function decrease with each epoch. This decrease is empirical evidence that the proposed loss function is derivable.

PU Dataset

	F1 Score			
% of Training Data	*25*	*50*	*80*	*100*
Metamap	-	-	-	0.50
BSVM	0.24	0.43	0.48	0.54
B-Pr	0.09	0.21	0.46	0.59
Se-CNN	**0.73**	0.76	0.80	0.79
DMT + Add + LSTM	-	-	-	0.39
DMT + Hinge + LSTM	-	-	-	0.39
DMT + Hinge + CNN	-	-	-	0.65
DMT + Add + CNN	0.71	**0.77**	**0.82**	**0.82**

Binary Dataset

	F1 Score			
% of Training Data	*25*	*50*	*80*	*100*
Metamap	-	-	-	0.69
SVM	0.76	0.82	0.84	0.85
CNN	**0.84**	**0.88**	**0.88**	**0.89**
DMT + Add + LSTM	-	-	-	0.64
DMT + Hinge + LSTM	-	-	-	0.62
DMT + Hinge + CNN	-	-	-	0.88
DMT + Add + CNN	0.81	0.84	0.86	0.88

Table 7: Semi-Supervised and Supervised Classification results on the test set. Best results for each partition of training data in bold

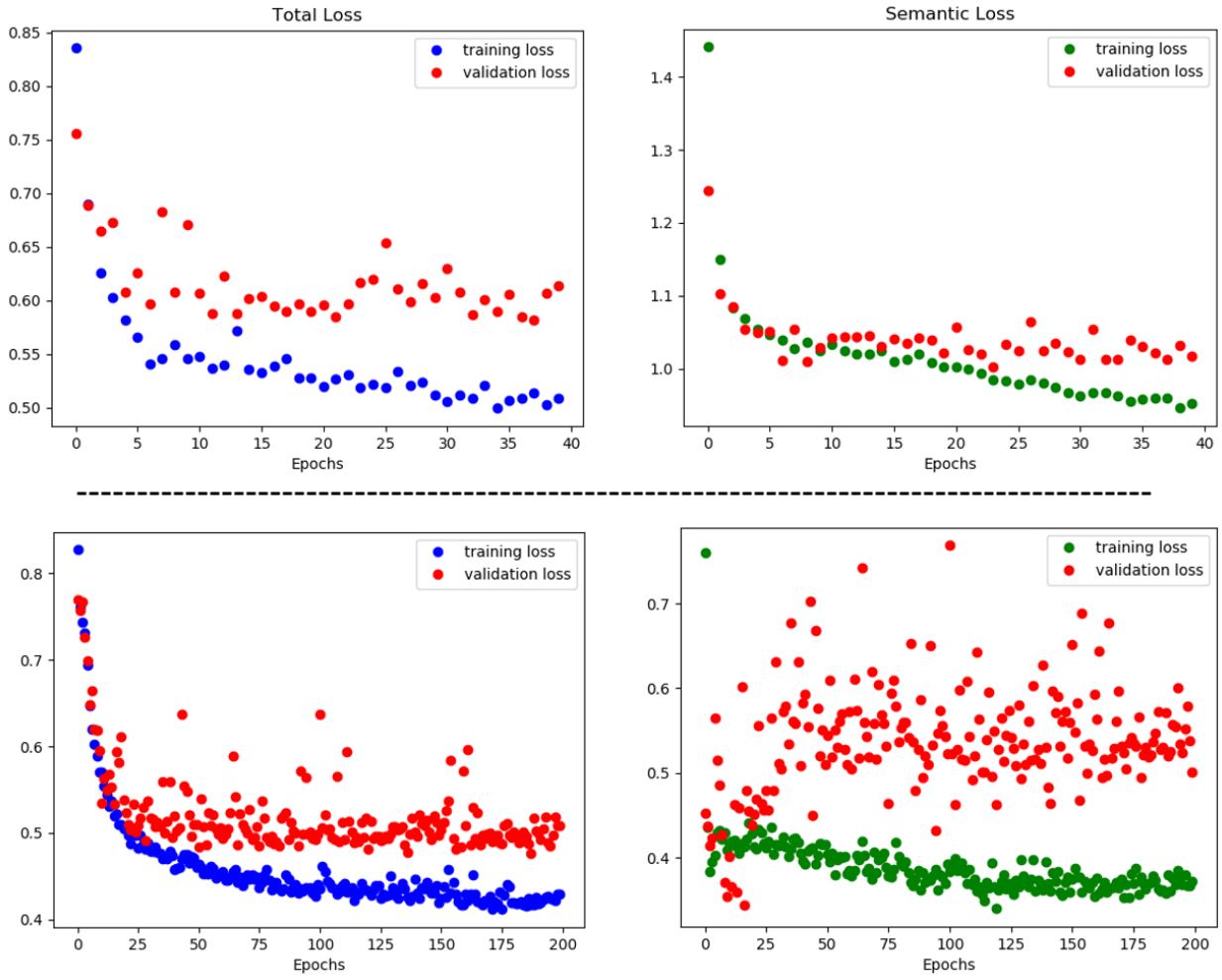


Figure 23: Loss function behavior during training for PU(up) and binary(down) datasets. Each point is the average loss over all the batches in the epoch.

5.0 DISCUSSION

In this dissertation, we tested a hypothesis of to whether the incorporation of a semantic loss function into the training of a deep learning supervised and semi-supervised medical text classifier could improve their generalization performance when compared with strong baselines. For the semi-supervised case, we can evidence from the results shown in Table 7, where the performance of the proposed strategy outperformed the baselines when training with 50, 80 and 100% of the data. These results also support the claim that only positive and unlabeled samples are needed to achieve good performance on clinical text classification.

To the best of our knowledge, we believe this is the first time in the biomedical informatics domain that a semantic loss function for training a deep learning classifier is proposed. Also, to the best of our knowledge, this is the first time that a semi-supervised strategy applied to clinical text is trained by only using positive and unlabeled samples, more even so applied to a corpus of psychiatric notes.

To evaluate the behavior of the semantic distance used in the loss function we observed the distributions of the distances for samples in the positive and unlabeled/negative set shown in Figure 21. These distances were computed using Equation 3.3. The first insight is that the distributions from the positive and the unlabeled/negative class does not look markedly different. For the case of positive vs. unlabeled, this may be because some of the unlabeled samples are positive. However, for the case of positive vs. negative, we could have expected a more appreciable difference. Given that this is not the case, a possible explanation is that given that the IAA was not 100% there is room for some for those sentences in the negative class to be actually positive and vice versa. This cannot be easily corroborated without going through those overlapping examples with the annotators. However, what we can see is that the positive sentences distribution has a longer tail that extend to lower

distances, while the negative sentences distribution is skewed towards the 1 as it is expected, given that they do not belong to the social context class.

Given that visually is not possible to conclusively assess whether there is a difference between the distributions we test the difference between them using two nonparametric statistical tests. The results showed on Section 4.2.2 evidence that the distributions are indeed different given that for both comparisons (i.e., positive vs. unlabeled and positive vs. negative) p-values were zero in both tests. This is an interesting result in itself given that such metrics were used in the past to phenotype but not to classify portions of clinical text. This opens up an avenue to explore whether such metrics can be enhanced by using linguistic modifiers among the concepts derived from an algorithm such as ConText [127]. Also, the use of set metrics derived from ontologies is recommended as input features to clinical text classifiers. This departs from previous works [56] that used the semantic similarity as a feature selection method and was only used at the individual concept level and not at the set level. Also in an active learning scenario [49] where instead of a cosine distance to compute a similarity, a set based similarity used here may prove more useful to select candidates for labeling.

We can also see that the recursive elimination strategy in the semi-supervised setting played an important role in the final performance. This is evidenced when the performance of the models at each iteration increase even when the number of samples used for training is reduced (see Section 4.2.3). This may be to the fact that models trained in this strategy are forced to assign a label 0 for unlabeled samples that may be very similar to samples in the positive set which we want to assign a label of 1. Two very similar sentences from the training data can be used to exemplify this. The sentence *"lives with mother and mother's boyfriend"* belongs to the unlabeled set, and the sentence *"He lives with mother, step-father and 4 siblings"* belongs to the positive set. It can be seen directly the usefulness of remove sentences like the first one from the unlabeled set. By removing samples like this, the classifier can better generalize by not forcing it to assign a label to sample that does not truly belong to that class.

Along with the recursive strategy, the NNPU risk also played an important role in selecting the best set of hyperparameters for each strategy. The challenge while training a PU

learning strategy is how to compare the performance of two sets of model parameters without actually having negative samples (see Table 7). The risk proposed in [112] allowed us to do that. To the best of our knowledge, we believe this is the first time that a hyperparameters search strategy is done in a truly PU setting where negative samples are not required in any part of the training. The results support the claim that an NNPU risk loss function is sufficient to evaluate model candidates during hyperparameter optimization. This claim is validated by the good performance of the proposed strategy on the test set. This we believe is a relevant result in the biomedical domain where the prevalence and characteristics of diseases make easier to define positive samples than negative samples.

Now putting aside the hyperparameter optimization, from looking at Figure 23 two main conclusions can be drawn. First, in general, the semantic loss proposed in Equation 3.1 decreased at each epoch and the automatic differentiation from PyTorch did the computation of the gradient without explicitly defining it. This is strong evidence that the proposed loss is derivable. Second, the figure reinforce the need for early stopping as a regularization technique. For the semi-supervised setting the validation loss stop decreasing around epoch 10 but the training loss continued its descend as expected. For the supervised setting, overfitting is easier to appreciate when we observe the semantic loss "wandering" around epoch 25 with a tendency to increase. Given that cross entropy contributes more to the total loss, the effect of the wandering is less noticeable in the final additive loss.

Consistently, CNN based strategies with word embeddings outperformed all the other algorithms. It is important to notice that the CNN used in the proposed approach is much larger than the CNN used in the supervised and semi-supervised setting given the need to predict CUIs. Even with a larger set of parameters and not a large amount of data, the proposed approach has a better performance in the semi-supervised case and a competitive one in the supervised one. This may be attributed at a possible regularizing effect performed by the semantic loss, given that to truly capture the semantics it is imperative to generalize better.

Regarding the embeddings is important to note that they were trained using a corpus from a different institution and a different hospital units. The embedding weights were not updated during training. Even with those two conditions the performance of the deep

learning strategies was not presumably impacted given that the final results outperformed all the baselines. The results here are similar to the ones found in [128]. In this particular work, an application using psychiatric notes also benefited from embeddings derived from MIMIC instead of sources outside the clinical domain.

5.1 LIMITATIONS AND FUTURE WORK

The main limitation of this work is the lack of improvement in the supervised setting when compared with the deep learning baseline. Contrary to expectations the performance was lower even for the cases when there was less training data. This unexpected behavior may be triggered by the number of concepts that the SDN needed to predict. As mentioned in Section 4.2.2 the number of concepts to predict for the supervised setting was 2112. Closely analyzing the distribution of those concepts, it can be observed from Figure 20 than less than 7% of the CUIs had a prevalence greater than 1%. This imposes a severe burden on the SDN given than most of the CUIs had less than 70 samples.

We may solve this issue by pre-training the SDN on a larger corpus such as MIMIC. Then with this pre-trained network, we could use a transfer learning approach similar to the one proposed in this work [129], where different components of the network are unfrozen gradually, and fine tunned to prevent forgetting what the network already learned. Also, the architecture for SDN may be changed for a seq2seq network [130, 131]. These networks are particularly good when labels are sparse and the number of classes to predict at each sequence depends on the length of the sequence. Also for a sentence like this: *Pt reports having verbal argument w/ stepfather, then pt pushed stepfather.* when we have multiple occurrences of a single concept *stepfather* seq2seq networks, have proven useful in the past. Also in this architecture recently released CUI embeddings [132] can be used. One interesting aspect of having a network to predict concepts instead of mappings is the possibility to have associated probabilities to each one of the predicted concepts, that may be seen as a degree of certainty in the prediction made. Given the deterministic nature of most of the dictionary mapping tools, this is a desirable feature when using CUIs in downstream tasks like the

one presented in this work. Using the output of a dictionary lookup directly with a neural network will increase the sparsity of the input given the lack of embeddings, loosing also the information contained in the raw text that does not map to any concept and losing the capacity of optimizing for the primary task (i.e., text classification), while also trying to get better at the secondary task (i.e., concept mapping). One possible hypothesis to consider is whether a network like this can perform extrapolation, possibly do unseen mapping and when used full documents help with the very challenging task of co-reference resolution in the clinical NLP.

However, one important aspect to consider is the performance of the dictionary mapping tool. In Table 6 we see the results of a less than successful mappings performed by MetaMap. We can appreciate three main problems by looking at this example. The first is the lack of domain adaptation, reflected in the expansion of abbreviations that are not relevant for this particular set of notes. Second is the lack of coverage by the specific terminology used, observed in the lack of mapping for a possible concept such as *estranged from his family*. The last one is the lack of mapping of concepts that are indeed present in the terminology but where not mapped such as *employment*. We hypothesize that these particular set of problems affected more the supervised than the semi-supervised strategy because of the sample size. With a bigger sample size, there is a chance that we have more samples of concepts with low prevalence but important to the classification. The low prevalence could be caused by either an actual low prevalence in the corpus or by a low success rate in mapping such concept by the dictionary lookup tool. By having more samples, the prediction of such concepts could be better, and the effects of the errors ameliorated. In the future, to overcome the problems associated with the dictionary lookup we could look at several options. The first one is to use a better tool or to attempt tuning an existing one. Tuning a dictionary lookup tool can be very challenging since they depend on a set of rules that may not allow easy customization. On the other hand, finding a better tool depends on what better means and how to measure it. Access to a properly labeled corpus for this task is not trivial, and every tool is usually tested and tuned on a single institution corpus, or a single domain, making the selection of the tools challenging. Instead, an approach previously presented [133] where several tools are combined to produce a silver standard corpus may be more desirable.

Another limitation is that surprisingly LSTM did not perform well for our case. We hypothesize that this occurred due to overfitting, produced by the lack of a bigger sample size required for this architecture. It is important to state that the architecture itself is the one questioned here, instead of the number of parameters. The reasons for this conclusion is that small architectures were part of the search done when looking for best hyperparameters. Also, this architecture is well suited when there is the need to capture long-term dependencies. Since the LSTM was used in the SDN, long-term dependencies are not needed since usually a concept map to a word or series of words that are contiguous.

For three of the four cases where the Hinge loss was used, it did not have a good performance. This may because there are no labels for the prediction in the SDN and in consequence the semantic distance computed is not as good as when the labels are used. For the supervised setting, the close performance with the additive loss may be due to that given that most of the loss comes from predicting the label right it is sufficient to predict concepts that may not be the ones present in the text but have a low semantic distance with the class. Besides the seq2seq architecture explained before another possible solution is the use of weight sharing [24]. In this strategy, a set of words is tied to a concept represented by an embedding. In this way, different words activate different concepts and sparsity is enforced.

One of the most promising avenues for future work is the use of the computation of the semantic loss only on unlabeled samples. This may be feasible only when large available of unlabeled samples are available, and it requires the use of the predicted label. This is similar to the works of [134, 135, 136] where the loss for unlabeled samples is computed different than the loss for labeled samples. It is important to notice here that the SDN is the one enabling us to compute this semantic loss. If as an alternative we would like to use only the word embeddings to compute a semantic loss a very different architecture should be proposed. First such a strategy needs to prevent the embeddings from forgetting what was learned during pre-training. Second, a class definition using raw words may need to be proposed in a way that the incorporation a differentiable loss is feasible. This is not a trivial task, and it could be an interesting route to explore in future works. In the present work, we proposed a solution that departs from only using the word embeddings attempting to leverage the knowledge contained in the manually curated ontologies.

Another future avenue is to explore whether predicting concepts as an intermediate task could help explain overall model predictions. For instance specific predictions current techniques such as LIME [137] construct a linear model with a modified version of the input features. These linear models may not capture the complexities involved in such predictions. Instead, we could use the information provided by an intermediate task (i.e., concept mapping) to explain those predictions. To do so, we could generate synthetic text by sampling sentences from an unlabeled corpus. After sampling, we could modify some portions of the text with synonyms without affecting syntax or grammar. As an example let consider the sentence *discharged to home*, from which we could generate synthetic samples such as *discharged to house*, and *discharged to domicile*. Doing this we could see how the model responds to these changes in the final prediction, whether the concept predicted changed or not and whether the prediction is dependent on this particular word changed or others in the given sentence.

APPENDIX

CONFIDENCE INTERVALS FOR CLASSIFICATION RESULTS

F1 score is a widely used metric in NLP to compare the performance of two systems when the prevalence of the target class is low. This is because this metric does not take into account the true negatives for its computation. However, it is important to note that the F1 score does not have a direct probabilistic interpretation given that is the harmonic mean between two quantities that can be interpreted as probabilities, this is recall and precision [138]. Because of that the computation of the confidence intervals (CI) presented here was done using boostrap, a common non-parametric technique. The intervals computed here are *BCa* intervals. *BCa* is a bias corrected bootstrap that adjust for skewness in the bootstrap distribution [139]. The intervals were computed using the package *boot* from R [140] with 10,000 bootstrap replicas. The final CIs are shown in Table 8

PU Dataset

	F1 Score (95% CI)			
% of Training Data	*25*	*50*	*80*	*100*
Metamap	-	-	-	0.50 (0.46, 0.53)
BSVM	0.24 (0.19, 0.29)	0.43 (0.38, 0.48)	0.48 (0.43, 0.52)	0.54 (0.49, 0.59)
B-Pr	0.09 (0.06, 0.13)	0.21 (0.17, 0.27)	0.46 (0.41, 0.50)	0.59 (0.54, 0.63)
Se-CNN	0.73 (0.69, 0.76)	0.76 (0.73, 0.79)	0.8 (0.77, 0.83)	0.79 (0.76, 0.82)
DMT + Add + LSTM	-	-	-	0.39 (0.36, 0.42)
DMT + Hinge + LSTM	-	-	-	0.39 (0.36, 0.43)
DMT + Hinge + CNN	-	-	-	0.65 (0.62, 0.69)
DMT + Add + CNN	0.71 (0.68, 0.75)	0.77 (0.73, 0.80)	0.82 (0.78, 0.84)	0.82 (0.79, 0.85)

Binary Dataset

	F1 Score (95% CI)			
% of Training Data	*25*	*50*	*80*	*100*
Metamap	-	-	-	0.69 (0.64, 0.73)
SVM	0.76 (0.71, 0.80)	0.82 (0.78, 0.86)	0.84 (0.80, 0.87)	0.85 (0.81, 0.88)
CNN	0.84 (0.80, 0.87)	0.88 (0.84, 0.91)	0.88 (0.84, 0.91)	0.89 (0.86, 0.92)
DMT + Add + LSTM	-	-	-	0.64 (0.60, 0.68)
DMT + Hinge + LSTM	-	-	-	0.62 (0.59, 0.67)
DMT + Hinge + CNN	-	-	-	0.88 (0.84, 0.90)
DMT + Add + CNN	0.81 (0.77, 0.84)	0.84 (0.80, 0.87)	0.86 (0.82, 0.89)	0.88 (0.85, 0.91)

Table 8: Confidence Intervals for Classification Results

BIBLIOGRAPHY

[1] Roski J, Bo-Linn GW, Andrews TA. Creating value in health care through big data: Opportunities and policy implications. Health Affairs. 2014;33(7):1115–1122.

[2] Raghupathi W, Raghupathi V. Big data analytics in healthcare: promise and potential. Health Information Science and Systems. 2014;2(1):3.

[3] Maddox TM, Matheny MA. Natural Language Processing and the Promise of Big Data: Small Step Forward, but Many Miles to Go. Circulation: Cardiovascular Quality and Outcomes. 2015;8(5):463–465.

[4] Demner-Fushman D, Chapman WW, McDonald CJ. What can natural language processing do for clinical decision support? Journal of Biomedical Informatics. 2009;42(5):760–772.

[5] Leaman R, Khare R, Lu Z. Challenges in clinical natural language processing for automated disorder normalization. Journal of Biomedical Informatics. 2015;57:28–37.

[6] Editors A, Harper MB, Hoffman JM, Zorc JJ, Kimia AA, Savova G, et al. An Introduction to Natural Language Processing: How You Can Get More From Those Electronic Notes You Are Generating. Pediatric Emergency Care. 2015;31(7):536–541.

[7] Kreimeyer K, Foster M, Pandey A, Arya N, Halford G, Jones SF, et al. Natural language processing systems for capturing and standardizing unstructured clinical information: A systematic review. Journal of Biomedical Informatics. 2017;73:14–29.

[8] Chen W, Kowatch R, Lin S, Splaingard M, Huang Y. Interactive Cohort Identification of Sleep Disorder Patients Using Natural Language Processing and i2b2. Applied Clinical Informatics. 2015;6(2):345–363.

[9] Tsui FC, Espino JU, Dato VM, Gesteland PH, Hutman J, Wagner MM. Technical description of RODS: A real-time public health surveillance system. Journal of the American Medical Informatics Association. 2003;10(5):399–408.

[10] Crowley RS, Castine M, Mitchell K, Chavan G, McSherry T, Feldman M. caTIES: a grid based system for coding and retrieval of surgical pathology reports and tissue specimens in support of translational research. Journal of the American Medical Informatics Association. 2010 aug;17(3):253–264.

[11] Savova GK, Tseytlin E, Finan S, Castine M, Miller T, Medvedeva O, et al. DeepPhe: A Natural Language Processing System for Extracting Cancer Phenotypes from Clinical Records. Cancer Research. 2017 nov;77(21):e115 LP – e118.

[12] Geraci J, Wilansky P, De Luca V, Roy A, Kennedy JL, Strauss J. Applying deep neural networks to unstructured text notes in electronic medical records for phenotyping youth depression. Evidence-Based Mental Health. 2017;20(3):83–87.

[13] Scheurwegs E, Luyckx K, Luyten L, Goethals B, Daelemans W. Assigning clinical codes with data-driven concept representation on Dutch clinical free text. Journal of Biomedical Informatics. 2017;69:118–127.

[14] Luo Y. Recurrent neural networks for classifying relations in clinical notes. Journal of Biomedical Informatics. 2017;72:85–95.

[15] Gehrmann S, Dernoncourt F, Li Y, Carlson ETE, Wu JT, Welt J, et al. Comparing Rule-Based and Deep Learning Models for Patient Phenotyping; 2017.

[16] Zhu Z, Yin C, Qian B, Cheng Y, Wei J, Wang F. Measuring patient similarities via a deep architecture with medical concept embedding. Proceedings - IEEE International Conference on Data Mining, ICDM. 2017;p. 749–758.

[17] Luo Y, Cheng Y, Uzuner Ö, Szolovits P, Starren J. Segment convolutional neural networks (Seg-CNNs) for classifying relations in clinical notes. Journal of the American Medical Informatics Association. 2017;0(October):1–6.

[18] Choi E, Bahadori MT, Song L, Stewart WF, Sun J. GRAM: Graph-based Attention Model for Healthcare Representation Learning. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. ACM; 2017. p. 787–795.

[19] Bejan CA, Angiolillo J, Conway D, Nash R, Shirey-Rice JK, Lipworth L, et al. Mining 100 million notes to find homelessness and adverse childhood experiences: 2 case studies of rare and severe social determinants of health in electronic health records. Journal of the American Medical Informatics Association. 2017;0(0):1–11.

[20] Hughes M, Li I, Kotoulas S, Suzumura T. Medical Text Classification using Convolutional Neural Networks. 2017;.

[21] Rios A, Kavuluru R. Ordinal Convolutional Neural Networks for Predicting RDoC Positive Valence Psychiatric Symptom Severity Scores. Journal of Biomedical Informatics. 2017;.

[22] Sabbir AKM, Yepes AJ, Kavuluru R. Knowledge-Based Biomedical Word Sense Disambiguation with Neural Concept Embeddings and Distant Supervision; 2016.

[23] Jimeno Yepes A, Berlanga R. Knowledge based word-concept model estimation and refinement for biomedical text mining. Journal of Biomedical Informatics. 2015;53:300–307.

[24] Zhang Y, Lease M, Wallace BC. Exploiting Domain Knowledge via Grouped Weight Sharing with Application to Text Categorization. Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers). 2017;p. 155–160.

[25] Miotto R, Li L, Kidd BA, Dudley JT. Deep Patient: An Unsupervised Representation to Predict the Future of Patients from the Electronic Health Records. Scientific reports. 2016;6(April):26094.

[26] Wang J, Wang Z, Zhang D, Yan J. Combining Knowledge with Deep Convolutional Neural Networks for Short Text Classification. Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17). 2017;p. 2915–2921.

[27] Jauhar SK, Dyer C, Hovy E. Ontologically Grounded Multi-sense Representation Learning for Semantic Vector Space Models. Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-2015. 2015;p. 683–693.

[28] Kholghi M, De Vine L, Sitbon L, Zuccon G, Nguyen A. Clinical information extraction using small data: An active learning approach based on sequence representations and word embeddings. Journal of the Association for Information Science and Technology. 2017;00(00).

[29] Weston J, Bordes A, Chopra S, Mikolov T, Rush AM. Towards AI-Complete Question Answering: A Set of Prerequisite Toy Tasks. arXiv preprint. 2015;273:486–494.

[30] Maas AL, Daly RE, Pham PT, Huang D, Ng AY, Potts C. Learning Word Vectors for Sentiment Analysis. In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies. Portland, Oregon, USA: Association for Computational Linguistics; 2011. p. 142–150.

[31] Filannino M, Stubbs A, Uzuner Ö. Symptom severity prediction from neuropsychiatric clinical records: Overview of 2016 CEGS N-GRID Shared Tasks Track 2. Journal of Biomedical Informatics. 2017;.

[32] Joffe E, Pettigrew EJ, Herskovic JR, Bearden CF, Bernstam EV. Expert guided natural language processing using one-class classification. Journal of the American Medical Informatics Association. 2015;22(5):962–966.

[33] Garla V, Taylor C, Brandt C. Semi-supervised clinical text classification with Laplacian SVMs: An application to cancer case management. Journal of Biomedical Informatics. 2013;46(5):869–875.

[34] Fodeh S, Benin A, Miller P, Lee K, Koss M, Brandt C. Laplacian SVM Based Feature Selection Improves Medical Event Reports Classification. In: 2015 IEEE International Conference on Data Mining Workshop (ICDMW); 2015. p. 449–454.

[35] Tepper M, Capurro D, Xia F. Statistical Section Segmentation in Free-Text Clinical Records. Lrec. 2012;(2002):2001–2008.

[36] Carrell DS, Schoen RE, Leffler DA, Morris M, Rose S, Baer A, et al. Challenges in adapting existing clinical natural language processing systems to multiple, diverse health care settings. Journal of the American Medical Informatics Association. 2017;24(5):986–991.

[37] Patterson O, Hurdle JF. Document Clustering of Clinical Narratives: a Systematic Study of Clinical Sublanguages. AMIA Annual Symposium Proceedings. 2011 oct;2011:1099–1107.

[38] Ferraro JPP, Ye Y, Gesteland PHH, Haug PJJ, Tsui FRR, Cooper GFF, et al. The effects of natural language processing on cross-institutional portability of influenza case detection for disease surveillance. Applied Clinical Informatics. 2017;8(2):560–580.

[39] Liu M, Shah A, Jiang M, Peterson NB, Dai Q, Aldrich MC, et al. A Study of Transportability of an Existing Smoking Status Detection Module across Institutions. AMIA Annual Symposium Proceedings. 2012 nov;2012:577–586.

[40] Sohn S, Wang Y, Wi CII, Krusemark EA, Ryu E, Ali MH, et al. Clinical documentation variations and NLP system portability: a case study in asthma birth cohorts across institutions. Journal of the American Medical Informatics Association. 2017;0(January):1–7.

[41] Patterson O, Igo S, Hurdle JF. Automatic Acquisition of Sublanguage Semantic Schema: Towards the Word Sense Disambiguation of Clinical Narratives. AMIA Annual Symposium Proceedings. 2010 nov;2010:612–616.

[42] Garla VN, Brandt C. Ontology-guided feature engineering for clinical text classification. Journal of Biomedical Informatics. 2012;45(5):992–998.

[43] Johnson SB. A Semantic Lexicon for Medical Language Processing. Journal of the American Medical Informatics Association. 1999;6(3):205–218.

[44] IHTSDO. SNOMED CT® Editorial Guide July 2016 International Release (US English); 2016.

[45] Lu HM, Zeng D, Trujillo L, Komatsu K, Chen H. Ontology-enhanced automatic chief complaint classification for syndromic surveillance. Journal of Biomedical Informatics. 2008;41(2):340–356.

[46] Malhotra A, Gündel M, Rajput AM, Mevissen HT, Saiz A, Pastor X, et al. Knowledge Retrieval from PubMed Abstracts and Electronic Medical Records with the Multiple Sclerosis Ontology. PLOS ONE. 2015 feb;10(2):e0116718.

[47] Moon S, Pakhomov S, Melton GB. Automated disambiguation of acronyms and abbreviations in clinical texts: window and training size considerations. AMIA Annual Symposium proceedings / AMIA Symposium AMIA Symposium. 2012;2012:1310–9.

[48] Lin C, Karlson EW, Canhao H, Miller TA, Dligach D, Chen PJ, et al. Automatic Prediction of Rheumatoid Arthritis Disease Activity from the Electronic Medical Records. PLoS ONE. 2013;8(8).

[49] Kholghi M, Sitbon L, Zuccon G, Nguyen A. Active learning: A step towards automating medical concept extraction. Journal of the American Medical Informatics Association. 2016;23(2):289–296.

[50] Buchan K, Filannino M, Uzuner Ö. Automatic prediction of coronary artery disease from clinical narratives. Journal of Biomedical Informatics. 2017;72:23–32.

[51] Castro SM, Tseytlin E, Medvedeva O, Mitchell K, Visweswaran S, Bekhuis T, et al. Automated annotation and classification of BI-RADS assessment from radiology reports. Journal of Biomedical Informatics. 2017;69:177–187.

[52] Savova GK, Masanz JJ, Ogren PV, Zheng J, Sohn S, Kipper-Schuler KC, et al. Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications. Journal of the American Medical Informatics Association. 2010 sep;17(5):507–513.

[53] Aronson AR, Lang FM. An overview of MetaMap: historical perspective and recent advances. Journal of the American Medical Informatics Association. 2010 jan;17(3):229–236.

[54] Tseytlin E, Mitchell K, Legowski E, Corrigan J, Chavan G, Jacobson RS. NOBLE - Flexible concept recognition for large-scale biomedical natural language processing. BMC Bioinformatics. 2016;17(1):1–15.

[55] Aseervatham S, Bennani Y. Semi-structured document categorization with a semantic kernel. Pattern Recognition. 2009;42(9):2067–2076.

[56] Garla VN, Brandt C. Ontology-guided feature engineering for clinical text classification. Journal of Biomedical Informatics. 2012;45(5):992–998.

[57] Albitar S, Espinasse B, Fournier S. Semantic Enrichments in Text Supervised Classification: Application to Medical Domain. Proceedings of the Twenty-Seventh International Florida Artificial Intelligence Research Society Conference. 2013;(i):425–430.

[58] Calvier FE, Plantié M, Dray G, Ranwez S. Ontology Based Machine Learning for Semantic Multiclass Classification. 2013;100.

[59] Mikolov T, Chen K, Corrado GS, Dean J, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and their Compositionality. Nips. 2013;26:1–9.

[60] Pennington J, Socher R, Manning CD. GloVe: Global Vectors for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. 2014;.

[61] Wu Y, Xu J, Zhang Y, Xu H. Clinical Abbreviation Disambiguation Using Neural Word Embeddings. Proceedings of BioNLP 15. 2015;(BioNLP):171–176.

[62] Pakhomov SVS, Finley G, McEwan R, Wang Y, Melton GB. Corpus domain effects on distributional semantic modeling of medical terms. Bioinformatics. 2016;32(23):3635–3644.

[63] Roberts K. Assessing the Corpus Size vs. Similarity Trade-off for Word Embeddings in Clinical NLP. In: Proceedings of the Clinical Natural Language Processing Workshop; 2016. p. 54–63.

[64] Zhang Y, Zhang O, Wu Y, Lee HJ, Xu J, Xu H, et al. Psychiatric symptom recognition without labeled data using distributional representations of phrases and on-line knowledge. Journal of Biomedical Informatics. 2017;75:S129–S137.

[65] De Vine L, Zuccon G, Koopman B, Sitbon L, Bruza P. Medical Semantic Similarity with a Neural Language Model. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management. CIKM '14. New York, NY, USA: ACM; 2014. p. 1819–1822.

[66] Choi Y, Chiu CYI, Sontag D. Learning Low-Dimensional Representations of Medical Concepts. Proceedings of the 23rd International Conference on World Wide Web. 2014;p. 373–374.

[67] Faruqui M, Dodge J, Jauhar SK, Dyer C, Hovy E, Smith NA. Retrofitting Word Vectors to Semantic Lexicons. 2014;p. 1606–1615.

[68] Yu Z, Wallace BC, Johnson T, Cohen T. Retrofitting Concept Vector Representations of Medical Concepts to Improve Estimates of Semantic Similarity and Relatedness; 2017.

[69] Yu M, Dredze M. Improving Lexical Embeddings with Semantic Knowledge. Acl-2014. 2014;1(1):545–550.

[70] Liu X, Bao H, Zhao D, Cao P. A label distance maximum-based classifier for multi-label learning. Bio-Medical Materials and Engineering. 2015;26(s1):S1969–S1976.

[71] Liu S, Tang B, Chen Q, Wang X. Drug-Drug Interaction Extraction via Convolutional Neural Networks. Computational and Mathematical Methods in Medicine. 2016;2016.

[72] Shivade C, Hebert C, Lopetegui M, de Marneffe MC, Fosler-Lussier E, Lai AM. Textual inference for eligibility criteria resolution in clinical trials. Journal of Biomedical Informatics. 2015;58:S211–S218.

[73] Manning CD, Raghavan P, Schütze H. Introduction to Information Retrieval. Cambridge, UK: Cambridge university press Cambridge; 2008.

[74] Nadkarni PM, Ohno-Machado L, Chapman WW. Natural language processing: an introduction. Journal of the American Medical Informatics Association : JAMIA. 2011;18(5):544–551.

[75] Boser BE, Guyon IM, Vapnik VN. A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory. ACM; 1992. p. 144–152.

[76] Cortes C, Vapnik V. Support-vector networks. Machine learning. 1995;20(3):273–297.

[77] Chang CC, Lin CJ. LIBSVM: A Library for Support Vector Machines. ACM Trans Intell Syst Technol. 2011 may;2(3):27:1—-27:27.

[78] Jurafsky D, Martin JH. Speech and Language Processing (2Nd Edition). Upper Saddle River, NJ, USA: Prentice-Hall, Inc.; 2009.

[79] Nair V, Hinton GE. Rectified linear units improve restricted boltzmann machines. In: Proceedings of the 27th international conference on machine learning (ICML-10); 2010. p. 807–814.

[80] Maas AL, Hannun AY, Ng AY. Rectifier nonlinearities improve neural network acoustic models. In: in ICML Workshop on Deep Learning for Audio, Speech and Language Processing. Citeseer; 2013. .

[81] Goodfellow I, Bengio Y, Courville A. Deep Learning. MIT Press; 2016. Available from: http://www.deeplearningbook.org.

[82] Goldberg Y. A Primer on Neural Network Models for Natural Language Processing. Journal of Artificial Intelligence Research. 2015 oct;57:345–420.

[83] Goldberg Y, Levy O. word2vec Explained: Deriving Mikolov et al.'s Negative-Sampling Word-Embedding Method. arXiv preprint arXiv:14023722. 2014;(2):1–5.

[84] Rios A, Kavuluru R. Convolutional Neural Networks for Biomedical Text Classification: Application in Indexing Biomedical Articles. In: Proceedings of the 6th ACM Conference on Bioinformatics, Computational Biology and Health Informatics. BCB '15. New York, NY, USA: ACM; 2015. p. 258–267.

[85] Kim Y. Convolutional Neural Networks for Sentence Classification. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014). 2014;p. 1746–1751.

[86] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural Language Processing (almost) from Scratch. The Journal of Machine Learning Research. 2011;12(Aug):2493–2537.

[87] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Computation. 1997 nov;9(8):1735–1780.

[88] Crammer K, Singer Y. On The Algorithmic Implementation of Multiclass Kernel-based Vector Machines. Journal of Machine Learning Research (JMLR). 2001;2:265–292.

[89] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research. 2014;15:1929–1958.

[90] Ioffe S, Szegedy C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: Proceedings of the 32Nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15. JMLR.org; 2015. p. 448–456.

[91] Wager S, Wang S, Liang P. Dropout Training as Adaptive Regularization. 2013;p. 1–9.

[92] LeCun Y, Bottou L, Orr GB, Müller KR. Efficient backprop. In: Neural networks: Tricks of the trade. Springer; 1998. p. 9–50.

[93] Bottou L. Stochastic gradient descent tricks. In: Neural networks: Tricks of the trade. Springer; 2012. p. 421–436.

[94] Paszke A, Chanan G, Lin Z, Gross S, Yang E, Antiga L, et al. Automatic differentiation in PyTorch. In: Advances in Neural Information Processing Systems 30. Nips; 2017. p. 1–4.

[95] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A System for Large-Scale Machine Learning. In: OSDI. vol. 16; 2016. p. 265–283.

[96] Harris ZS. Distributional structure. Word. 1954;10.

[97] Sánchez D, Batet M, Isern D, Valls A. Ontology-based semantic similarity: A new feature-based approach. Expert Systems with Applications. 2012;39(9):7718–7728.

[98] Harispe S, Sánchez D, Ranwez S, Janaqi S, Montmain J. A framework for unifying ontology-based semantic similarity measures: A study in the biomedical domain. Journal of Biomedical Informatics. 2014;48:38–53.

[99] Leacock C, Chodorow M. Combining local context with wordnet similarity for word sense identification, in WordNet: A Lexical Reference System and its Application'. MIT Press; 1998.

[100] Garla VN, Brandt C. Semantic similarity in the biomedical domain: an evaluation across knowledge sources. BMC Bioinformatics. 2012;13(1):261.

[101] Wu Z, Palmer M. Verb Semantics and Lexical Selection. In: 32nd Annual Meeting of the Association for Computational Linguistics; 1994. .

[102] Loper E, Bird S. NLTK: The Natural Language Toolkit. In: Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1. ETMTNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics; 2002. p. 63–70.

[103] Sánchez D, Batet M. Semantic similarity estimation in the biomedical domain: An ontology-based information-theoretic perspective. Journal of Biomedical Informatics. 2011;44(5):749–759.

[104] Kalchbrenner N, Grefenstette E, Blunsom P. A Convolutional Neural Network for Modelling Sentences. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). 2014;p. 655–665.

[105] Tsochantaridis I, Joachims T, Hofmann T, Altun Y, Org AC. Large Margin Methods for Structured and Interdependent Output Variables. Journal of Machine Learning Research. 2005;6:1453–1484.

[106] Girardi D, Wartner S, Halmerbauer G, Ehrenmüller M, Kosorus H, Dreiseitl S. Using concept hierarchies to improve calculation of patient similarity. Journal of Biomedical Informatics. 2016;63:66–73.

[107] Liu B, Dai Y, Li X, Lee WS, Yu PS. Building Text Classifiers Using Positive and Unlabeled Examples. In: Third IEEE International Conference on Data Mining, 2003. ICDM 2003.; 2003. p. 179–186.

[108] Wang Y, Zhang Y, Liu B. Sentiment Lexicon Expansion Based on Neural PU Learning, Double Dictionary Lookup, and Polarity Association. In: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing; 2017. p. 564–574.

[109] Snoek J, Larochelle H, Adams RP. Practical Bayesian Optimization of Machine Learning Algorithms. In: Pereira F, Burges CJC, Bottou L, Weinberger KQ, editors. Advances in Neural Information Processing Systems 25. Curran Associates, Inc.; 2012. p. 2951–2959.

[110] Bergstra J, Bengio Y. Random Search for Hyper-Parameter Optimization. Journal of Machine Learning Research. 2012;13:281–305.

[111] Levy O, Goldberg Y, Dagan I. Improving Distributional Similarity with Lessons Learned from Word Embeddings. Transactions of the Association for Computational Linguistics. 2015;3:211–225.

[112] Kiryo R, Niu G, du Plessis MC, Sugiyama M. Positive-Unlabeled Learning with Non-Negative Risk Estimator. In: Advances in Neural Information Processing Systems 30. Nips; 2017. p. 1674–1684.

[113] U S National Library of Medicine. UMLS 2017AA Release Documentation; 2017. Available from: https://www.nlm.nih.gov/pubs/techbull/mj17/mj17_umls_2017aa_release.html.

[114] Marmot M, Friel S, Bell R, Houweling TA, Taylor S. Closing the gap in a generation: health equity through action on the social determinants of health. The Lancet. 2008;372(9650):1661–1669.

[115] Shier G, Ginsburg M, Howell J, Volland P, Golden R. Strong Social Support Services, Such As Transportation And Help For Caregivers, Can Lead To Lower Health Care Use And Costs. Health Affairs. 2013;32(3):544–551.

[116] Marmot M. Social determinants of health inequalities. The Lancet. 2005;365(9464):1099–1104.

[117] South BR, Shen S, Leng J, Forbush TB, DuVall SL, Chapman WW. A prototype tool set to support machine-assisted annotation. In: Proceedings of the 2012 Workshop on Biomedical Natural Language Processing. Association for Computational Linguistics; 2012. p. 130–139.

[118] Khan SS, Madden MG. One-class classification: taxonomy of study and review of techniques. The Knowledge Engineering Review. 2014;293(3):345–374.

[119] Zhang D, Lee WS. A simple probabilistic approach to learning from positive and unlabeled examples. Proceedings of the 5th Annual UK Workshop on Computational Intelligence (UKCI). 2005;p. 83–87.

[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. 2012;12:2825–2830.

[121] McKinney W. pandas: a Foundational Python Library for Data Analysis and Statistics. Proceedings of PyHPC2011. 2011;p. 1–9.

[122] 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. Valletta, Malta: ELRA; 2010. p. 45–50.

[123] Harispe S, Ranwez S, Janaqi S, Montmain J. The semantic measures library and toolkit: Fast computation of semantic similarity and relatedness using biomedical ontologies. Bioinformatics. 2014;30(5):740–742.

[124] Johnson AEW, Pollard TJ, Shen L, Lehman LWH, Feng M, Ghassemi M, et al. MIMIC-III, a freely accessible critical care database. Scientific data. 2016;3:160035.

[125] 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. 2018;p. 1–21.

[126] Jones E, Oliphant T, Peterson P, Others. {SciPy}: Open source scientific tools for {Python};. Available from: http://www.scipy.org/.

[127] Harkema H, Dowling JN, Thornblade T, Chapman WW. ConText: An algorithm for determining negation, experiencer, and temporal status from clinical reports. Journal of Biomedical Informatics. 2009;42(5):839–851.

[128] Zhang Y, Li HJ, Wang J, Cohen T, Roberts K, Xu H. Adapting Word Embeddings from Multiple Domains to Symptom Recognition from Psychiatric Notes. In: AMIA Summits on Translational Science Proceedings. vol. 2017. American Medical Informatics Association; 2018. p. 281–289.

[129] Howard J, Ruder S. Universal Language Model Fine-tuning for Text Classification; 2018.

[130] Neubig G. Neural Machine Translation and Sequence-to-sequence Models: A Tutorial. 2017;p. 1–65.

[131] Sutskever I, Vinyals O, Le QV. Sequence to Sequence Learning with Neural Networks. Nips. 2014;p. 3104–3112.

[132] Finlayson SG, LePendu P, Shah NH. Building the graph of medicine from millions of clinical narratives. Scientific data. 2014 jan;1:140032.

[133] Oellrich A, Collier N, Smedley D, Groza T. Generation of Silver Standard Concept Annotations from Biomedical Texts with Special Relevance to Phenotypes. PloS one. 2015 jan;10(1):1–17.

[134] Miyato T, Dai AM, Goodfellow I. Adversarial Training Methods for Semi-Supervised Text Classification; 2016.

[135] Xu J, Zhang Z, Friedman T, Liang Y, den Broeck GV. A Semantic Loss Function for Deep Learning Under Weak Supervision; 2017.

[136] Bautista MA, Sanakoyeu A, Ömmer B. Deep Unsupervised Similarity Learning using Partially Ordered Sets. arXiv. 2017;p. 7130–7139.

[137] Ribeiro MT, Singh S, Guestrin C. Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. ACM; 2016. p. 1135–1144.

[138] Goutte C, Gaussier E. A probabilistic interpretation of precision, recall and F-score, with implication for evaluation. In: European Conference on Information Retrieval. Springer; 2005. p. 345–359.

[139] Davison AC, Hinkley DV. Bootstrap Methods and Their Applications. Cambridge: Cambridge University Press; 1997.

[140] Canty A, Ripley BD. boot: Bootstrap R (S-Plus) Functions; 2017.