Analysis of Deficiencies in Drug Applications Using Contextualized Feature Extraction

Satish Kumar Keshri

Department of Computer Science
Ramakrishna Mission Vivekananda Educational and Research Institute
Belur Math, Howrah
Pin - 711 202, West Bengal

A thesis submitted to
Ramakrishna Mission Vivekananda Educational and Research Institute
in partial fulfillment of the requirements for the degree of
MSc in Big Data Analytics
2020

Dedicated to *my parents* for their unconditional love and support

Acknowledgements

I express my sincere appreciation to those who have contributed to this thesis and supported me in one way or the other during this amazing journey for without any of them, this research work would not have been possible. I gratefully acknowledge the stipend received towards my master's thesis from Dr. Reddy's Laboratories Ltd (DRL). In addition to that DRL also extended help by providing me their internal dataset for my thesis.

I would like to express my deep gratitude to my thesis advisor Mr. Nishit Mittal for his guidance and constant supervision as well as for providing necessary information regarding the project and also for his support in completing it. I sincerely appreciate his sage advice, and patient encouragement in innumerable ways that led me through the various stages of this thesis. I am also grateful to Mr. Saurabh Biswas, Head DPE - R&D (DRL) for his insightful criticisms and encouragement.

My sincere gratitude is reserved for Br. Mrinmay for his invaluable insights and suggestions. I really appreciate his willingness to help whenever I needed some clarification even during his busy schedule. The useful discussion and comments that he suggested me widens my knowledge and helped me throughout the journey. I also thank our HOD, Swami Dhyanagamyananda for extending his help needed from time to time to complete this thesis. I am also grateful to all the departmental staff for helping me during these years of my study both academically and officially.

Last but not the least important, I owe more than thanks to my parents for their emotional and financial support and encouragement throughout my life. I would like to mention that without their love and guidance it would not have been possible to complete my thesis work.

Ramakrishna Mission Vivekananda Educational
and Research Institute, Belur Math, West Bengal

Satish Kumar keshri

June 25, 2020

CERTIFICATE FROM THE SUPERVISOR

This is to certify that the thesis entitled *'Analysis of Deficiencies in Drug Applications Using Contextualized Feature Extraction'* submitted by *Mr. Satish Kumar Keshri*, who has been registered for the award of MSc in Big Data Analytics degree of Ramakrishna Mission Vivekananda Educational and Research Institute, Belur Math, Howrah, West Bengal is absolutely based upon his own work under the supervision of *Mr. Nishit Mittal*, Engagement Lead, Digital Process and Excellence, Dr. Reddy's Laboratories Ltd., Hyderabad and that neither his thesis nor any part of the thesis has been submitted for any degree/diploma or any other academic award anywhere before.

Nishit Mittal
Engagement Lead - Data Science
Digital Process and Excellence
Dr. Reddys Laboratories Ltd
Hyderabad, 502325, Telengana

Saurabh Biswas
Head DPE - R&D
Digital Process and Excellence
Dr. Reddys Laboratories Ltd
Hyderabad, 502325, Telengana

Abstract

Texts written in natural language are an unstructured data source that is hard for machines to comprehend. In recent years there has been an exponential growth in the number of complex documents and it requires a deeper understanding of the text contents to be able to accurately classify them. This thesis presents a domain specific contextualized embedding based approach to text classification. Many techniques have been proposed to infuse word embeddings into the statistical vector space model (VSM). However, previous methods mainly use traditional static embedding, but static word embeddings could not deal with the problem of polysemy. For this reason we propose to utilize contextualized word embeddings to effectively encode the text. In this study, we introduce a new vector space model called Contextualized Semantically Augmented Statistical Vector Space model (C-SAS-VSM). Our approach is an extension of existing Semantically Augmented Statistical Vector Space model (SAS-VSM) in which we leverage the power of both contextualized and static embeddings to generate more feature rich text representation. Experimental results show that our proposed method outperforms the existing SAS-VSM approach in domain specific text classification task.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In pharmaceuticals developing a new drug and hitting the market is a very long process. This involves from acquiring the drug APIs, developing new drug, doing bio-studies, scaling up the production from lab scale to manufacturing scale while keeping the overall quality intact. Before getting any marketing approvals the drug manufacturers have to get approval from the corresponding regulatory authorities and this process varies not only from market to market but also from drug to drug. For getting approvals companies need to do a regulatory filing which is called as the ‘Dossier’. Filing a dossier is a very challenging as well as time consuming task. A typical dossier length varies from around 2000 to 10000 pages and even some minor errors in the application can get the approval delayed. So, companies need to be very careful while filing the dossier.

Drug development and filing process can be of two types depending on whether it is an innovator drug or a generic drug. An innovator drug is the first drug created containing its specific active ingredients to receive approval for use. It is usually the product for which efficacy, safety and quality have been fully established. When a new drug is first made, drug patent usually will be acquired by the founding company. Most drug patents are protected for certain period of time and during the patent period, other companies cannot make or sell the same drug until the patent expires. In contrast to this a generic drug is made of the same active ingredients as its innovator drug. An active ingredient is the chemical contained inside a drug that makes it work. In other words, the pharmacological effect of a generic drug is exactly the same as those of its innovator counterpart. Other companies can manufacture the generic drugs when patent expires [7].

At Dr. Reddy's Laboratories Ltd. primary focus is on generic drugs and US is their major market. In the US market, the regulatory authority is USFDA (United States Food and Drug Administration). In generic drugs active ingredients will be the same from it's innovator drugs and the development of the drug is being done keeping in mind that both the innovator and the developed drugs are 'pharmaceutically equivalent' although excipients (e.g., binding agents, preservatives, flavorants, dyes) may differ, they must exist in similar proportion to the active ingredients in the reference product. For generic drugs the application is called ANDA (Abbreviated New Drug Application). It is termed as 'abbreviated' because they are generally not required to include pre-clinical (animal) and clinical (human) data to establish safety and effectiveness. Instead, generic applicants must scientifically demonstrate that their product exhibits similar performance and efficacy as the innovator drug. One way applicants demonstrate that a generic product performs in the same way as the innovator drug is to measure the time a generic drug takes to reach the bloodstream in healthy volunteers. This demonstration of "bioequivalence" gives the rate of absorption, or bio availability, of the generic drug, which can then be compared to that of the innovator drug. To be approved by FDA, the generic version must deliver the same amount of active ingredients into a patient's bloodstream in the same amount of time as the innovator drug. Apart from that the generic must have to contain comparable amount of active ingredients and pass all the stability tests. In broad terms to get an approval for a generic drug, ANDA applicants have to cover details from mainly 7 different categories to establish the sameness with the innovator drug. These are:

1. Drug Product
2. Drug Substance
3. Biologics
4. Container
5. Stability
6. Process and Development
7. Dose and Administration (Labeling)

Dossier filing is an integrated work between all the departments. Different departments conduct different kind of tests and develops a drug. After that they send detailed analysis reports along with supporting results to the regulatory affairs for final submission. The regulatory team meticulously collates all the data and submits the whole application to the USFDA. Upon submission of any new drug application the agency (USFDA) analyses the whole dossier and either approves it or send letters containing the deficiencies in the filing. The main problem is if they detect any deficiency then the whole approval process gets delayed and companies lose a significant amount of time in terms of market share to their competitors. As currently USA is the main market for Dr. Reddy's Laboratories, my thesis is more concerned about that particular market. In the response to a generic drug application (ANDA), USFDA issues three types of deficiency letters: i) Complete Response Letter (CRL), ii) Deficiency Review Letter (DRL) and iii) Administrative. Among these three type of responses the major area of concern is CRL as it can lead to a complete rejection of an application. In CRL, USFDA tags a deficiency as 'Minor', 'Major' or IR (Information Request). A major deficiency means direct rejection, whereas upto 9 minor deficiency can be resolved with addition data submission to the regulatory within a stipulated time-frame. So, upon getting any Complete Response Letter having minor deficiencies the first task is to understand what steps needed to be taken to resolve the issue. For that the authority tries to tag/classify the deficiency and send it to the corresponding Business unit (BU) for required changes. The other task is to understand whether is there any past deficiency related to the current one so that they can quickly access it and check what kind of response lead to a resolution in the past.

So, drug manufacturers can avert these deficiencies by being more cautious while filing the dossier. But in the dossier filling process the major challenge is to compile a large amount of documents from all the units and all the laboratories. In this process they usually forget to conduct some specific tests according to the regulatory guidelines and/or provide test results while compiling it. The average size of an ANDA is around 6000 pages whereas some of them exceeds even 10000 pages and it becomes very difficult at the end moment to understand which are the core areas of concern. So, after getting any CRL their first concern is to understand the deficiency is related to which unit so that they can tag and send it directly to the corresponding unit or laboratory to resolve the issue as soon as possible.

Currently, the Regulatory Affairs team analyzes each USFDA observations one by one and tag them under different sub-categories. The problem in this approach is this is a very time consuming task and USFDA gives a very limited time for resolving the issue and submitting the updated version otherwise they will reject the application. Also the dossier contents change not only for different verticals (Oral, Solid, Injectables etc.) but from drug to drug as each drug has its own tests and guidelines to follow up with. So, they can't lose valuable time just for tagging them and sending the issue to their source unit. Thus, the primary concern of this work is to classify a new deficiency under one of the units with the help of machine learning algorithms. One of the major challenges in classification these deficiencies is the text data contains various chemical, pharmaceutical and scientific methods related words and extracting important features is quite different than extracting from plain English text.

In recent years many approaches have been proposed for extracting features from text data. Han et al. [8] have proposed a class centroid vector based approach for text classification. Galke et al. [9] tried to use IDF re-weighted Word Centroid Similarity in information retrieval domain. Sohrab et al [10] used a method to infuse statistical vector space model with semantic vector space and called it Semantically-Augmented Statistical Vector Space Model (SAS-VSM). In this work we have proposed a new method for classifying the deficiencies based on both statistical and semantic knowledge. We have infused ideas from Han et al.'s [8] centroid based text classification and Semantically Augmented Statistical Vector Space Model. We call it as Contextualized SAS-VSM. Sohrab et al. used Global Vectors (GloVe) [11] in their approach as they were dealing with plain English text but to better understand the contexts of the deficiencies which are mainly chemical and biological text, we have used Word2Vec [12] model pre-trained on Wikipedia and PubMed articles. To get the contextual semantic understanding of the documents we have also used pre-trained BERT [6] embeddings fine-tuned on domain specific corpus in our approach. In addition to that we have observed that by integrating the Han's class centroid based classification on top of this contextualized semantic approach has given us better results. The intuition behind using centroid based classification is mainly the different contexts in which the deficiencies are needed to be classified. Context of some classes are intertwined whereas for other classes they are quite different. For example, stability related deficiencies are quite different from the labeling related deficiencies. In these cases angle between the class centroid vectors will be large enough to correctly classify them. On the other hand, the overall context of container related deficiencies is a subset of the process and development related deficiencies. So, the class centroid vectors of these classes are expected to be close enough. So to distinguish between these contextually comparable classes we have used SAS-VSM [10] and contextual embeddings. The combined discriminative power of this Contextualized SAS-VSM and class-centroid leads to better classification results.

Chapter 2

Literature Review

2.1 Introduction

Dealing with text data is problematic, since our computers, scripts and machine learning models can't read and understand text in any human sense. But computers can handle numerical input well. A very common practice while working with unannotataed texts is to transform the text into vectors. The motive behind this approach is to numerically represent the words so that machines can ingest the data. Transformation of text to vectors can be done in multiple levels: word-level, sentence-level or paragraph level. In text classification tasks one major step is the feature extraction. Feature extraction is the task to extract meaningful features by converting the text into vectors in such a way so that vectors of related texts should end up closer to each other. In general, the text classification system contains three different levels of scope that can be applied:

1. Document level: In the document level, the text classification algorithm obtains the relevant categories of a full document [13].

2. Paragraph level: In the paragraph level, the text classification algorithm obtains the relevant categories of a single paragraph (a portion of a document) [14, 15].

3. Sentence level: In the sentence level, the algorithm obtains the relevant categories of a single sentence [16, 17].

The problem of classification has been widely studied in the database, data mining, and information retrieval communities. The problem of classification is defined as follows. We have a set of training records $D = \{d_1, d_2, \ldots, d_n\}$, such that each record is labeled with a class value drawn from a set of k different discrete values indexed by $\{1, 2, \ldots, k\}$. The training data is used in order to construct a classification model, which relates the features in the underlying record to one of the class labels. For a given test instance for which the class is unknown, the training model is used to predict a class label for this instance. Based on the classification technique mainly two different kinds of versions are there : hard version and soft version. In hard version of the classification problem, a particular label is explicitly assigned to the instance, whereas in the soft version of the classification problem, a probability value is assigned to the test instance. Other variations of the classification problem allow ranking of different class choices for a test instance, or allow the assignment of multiple labels [18] to a test instance. The area of text categorization is so vast that it is impossible to cover all the different approaches in detail in a single chapter. Therefore, our goal is to provide an overview of the most important techniques and the ones which are related to our methodology.

2.2 Feature Selection and Transformation

2.2.1 Traditional Approaches

Feature selection is an important problem for text classification. In feature selection, we attempt to determine the features which are most relevant to the classification process. This is because some of the words are much more likely to be correlated to the class distribution than others. Therefore, a wide variety of methods have been proposed in the literature in order to determine the most important features for the purpose of classification. While feature selection is also desirable in other classification tasks, it is especially important in text classification due to the high dimensionality of text features and the existence of irrelevant (noisy) features. In general, text can be represented in two separate ways. The first is as a bag of words [19], in which a document is represented as a set of words, together with their associated frequency in the document. Such a representation is essentially independent of the sequence of words in the collection. The second method is to represent text directly as strings, in which each document is a sequence of words. Most text classification methods use the bag-of-words representation because of its simplicity for classification purposes. In the case of the classification problem, it makes sense to supervise the feature selection process with the use of the class labels. This kind of selection process ensures that those features which are highly skewed towards the presence of a particular class label are picked for the learning process. A wide variety of feature selection methods are discussed in [20, 21]. One of the most common methods for quantifying the discrimination level of a feature is the use of a measure known as the gini-index[22] in which it tries to compute the conditional probability that a document d belongs to class i, given the fact that it contains the word w. Another related measure which is commonly used for text feature selection is that of entropy or information gain [23]. Mutual information [24] is yet another measure which provides a formal way to model the mutual information between the features and the classes. The point-wise mutual information between the word w and the class i is defined on the basis of the level of co-occurrence between the class i and word w.

While feature selection attempts to reduce the dimensionality of the data by picking from the original set of attributes, feature transformation methods create a new set of features as a function of the original set of features. A typical example of such a feature transformation method is Latent Semantic Indexing (LSI) [25], and its probabilistic variant PLSA [26]. The LSI method transforms the vector space of text space of hundreds of word features to a significantly lower dimensional vector space with the help of PCA [27]. The main disadvantage of using techniques such as LSI is that these are unsupervised techniques which are blind to the underlying class-distribution. Thus, the features found by LSI are not necessarily the directions along which the class-distribution of the underlying documents can be best separated. A modest level of success has been obtained in improving classification accuracy by using boosting techniques in conjunction with the conceptual features obtained from unsupervised PLSA [26] method. A method called sprinkling [28] has been proposed in which artificial terms are added to (or "sprinkled" into) the documents, which correspond to the class labels. It tries to create a term corresponding to the class label, and add it to the document. LSI is then performed on the document collection with these added terms. The sprinkled terms can then be removed from the representation, once the eigen vectors have been determined. The sprinkled terms help in making the LSI more sensitive to the class distribution during the reduction process.

2.2.2 Deep Learning Based Approaches

With the recent advancement in digital technologies, the size of data sets has become too large in which traditional data processing and machine learning techniques are not able to cope with effectively. However, analyzing complex, high dimensional, and noise-contaminated data sets is a huge challenge, and it is crucial to develop novel algorithms that are able to summarize, classify, extract important information and convert them into an understandable form. To undertake these problems, deep learning models have shown outstanding performances in the recent decade. In deep learning models a kind of deep nonlinear network structures are used which helps in complex function approximation, according to the characterization of the input data distributed, and in the case of sample set, the essence characteristic of the data set is seldom studied. The major difference between deep learning and traditional pattern recognition methods is that deep learning automatically learns features from big data, instead of adopting handcrafted features [29]. Deep learning technology is applied in common NLP (natural language processing) tasks, such as semantic parsing [30], information retrieval [31, 32], semantic role labeling [33, 34], sentimental analysis [35], question answering [36, 37], machine translation [38, 39], text classification [40], summarization [41, 42], and text generation [43], as well as information extraction, including named entity recognition [44, 45] and relation extraction [46, 47]. Convolution neural network and recurrent neural network are two popular models employed by this work [48]. Next, We will briefly discuss about some deep learning based models.

2.2.2.1 Autoencoder

An autoencoder is a type of neural network that is trained to attempt to copy its input to its output [49]. The autoencoder has achieved great success as a dimensionality reduction method via the powerful reprehensibility of neural networks [50]. The first version of autoencoder was introduced by D.E. Rumelhart et al. [51] in 1985. The main idea is that one hidden layer between input and output layers has fewer units [52] and could thus be used to reduce the dimensions of a feature space. Especially for texts, documents, and sequences that contain many features, using an autoencoder could help allow for faster, more efficient data processing. We will discuss on two specific frameworks along with one general framework.

1. General Framework: As shown in Figure 2.1, the input and output layers of an autoencoder contain n units where $x \in \mathbb{R}^n$, and hidden layer Z contains p units with respect to $p < n$ [53]. For this technique of dimensionality reduction, the dimensions of the final feature space are reduced from $n \rightarrow p$. The encoder representation involves a sum of the representation of all words (for bag-of-words), reflecting the relative frequency of each word [54]:

$$sigm(\eta) = \frac{1}{1 - e^{-\eta}} = \frac{e^{\eta}}{1 - e^{\eta}} \tag{2.1}$$

$$a(x) = c + \sum_{i=1}^{|x|} W_{.,x_i}, \phi(x) = h(a(x)) \tag{2.2}$$

where $h(.)$ is an element-wise non-linearity such as the sigmoid (Equation 2.1).

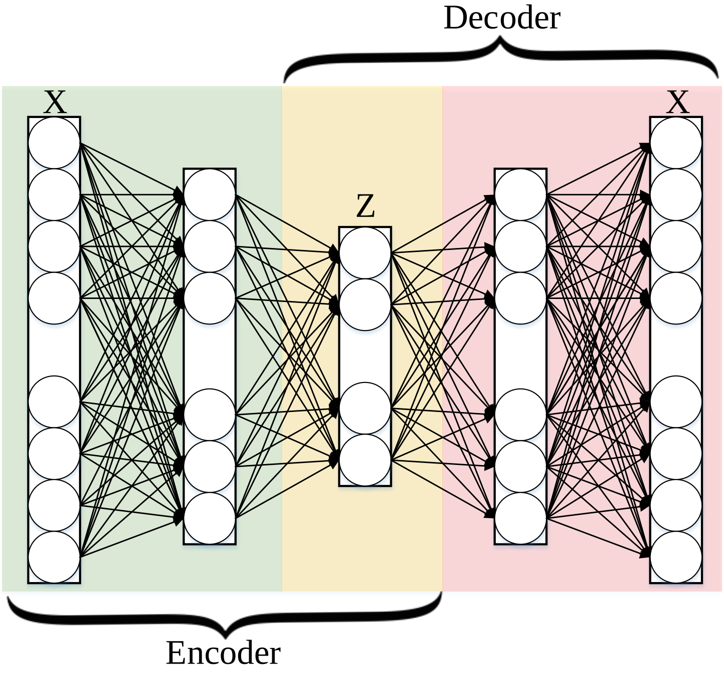


Figure 2.1: Illustration of a Simple Autoencoder

2. Conventional Autoencoder Architecture: A convolutional neural network (CNN)-based autoencoder can be divided into two main steps [55] (encoding and decoding).

$$O_m(i,j) = a\Bigg(\sum_{d=1}^{D}\sum_{u=-2k-1}^{2k+1}\sum_{v=-2k-1}^{2k+1} F^{(1)}_{m_d}(u,v)I_d(i-u,j-v)\Bigg) \quad \forall m = 1,\cdots,n \tag{2.3}$$

where $F \in \{F^{(1)}_1, F^{(1)}_2, \ldots, F^{(1)}_n,\}$ which is a convolutional filter, with convolution among an input volume defined by $I = \{I_1, \cdots, I_D\}$ which learns to represent input combining non-linear functions:

$$z_m = O_m = a(I * F^{(1)}_m + b^{(1)}_m) \quad m = 1,\cdots,n \tag{2.4}$$

where $b^{(1)}_m$ is the bias, and the number of zeros we want to pad the input with is such that: dim(I) = dim(decode(encode(I))). Finally, the encoding convolution is equal to:

$$\begin{aligned} O_w = O_h &= (I_w + 2(2k+1) - 2) - (2k+1) + 1 \\ &= I_w + (2k+1) - 1 \end{aligned} \tag{2.5}$$

The decoding convolution step produces n feature maps $z_{m=1,\ldots,n}$. The reconstructed results $\hat{I}$ is the result of the convolution between the volume of feature maps $Z = \{z_{i=1}\}^n$ and this convolutional filters volume $F^{(2)}$ [56, 57, 55].

$$\tilde{I} = a(Z * F^{(2)}_m + b^{(2)}) \tag{2.6}$$

$$\begin{aligned} O_w = O_h =& (I_w + (2k+1) - 1) - \\ & (2k+1) + 1 = I_w = I_h \end{aligned} \tag{2.7}$$

where Equation 2.7 shows the decoding convolution with I dimensions. Input's dimensions are equal to the output's dimensions.

3. Recurrent Autoencoder Architecture: A recurrent neural network (RNN) is a natural generalization of feed forward neural networks to sequences [58]. Figure 2.2 illustrate recurrent autoencoder architecture. A standard RNN compute the econding as a sequences of output by iteration:

$$h_t = sigm(W^{hx}x_t + W^{hh}h_{t-1}) \tag{2.8}$$

$$y_t = W^{y^h}h_t \tag{2.9}$$

where x is inputs $(x_1, ..., x_T)$ and y refers to output $(y_1, ..., y_T)$. A multinomial distribution (1-of-K coding) can be output using a softmax activation function [59]:

$$p(x_{t,j} = 1|x_{t-1,...,x_1}) = \frac{\exp(w_j h_t)}{\sum_{j'=1}^{K} \exp(w_{j'} h_t)} \tag{2.10}$$

By combining these probabilities, we can compute the probability of the sequence x as:

$$p(x) = \prod_{t=1}^{T} p(x_t|x_{t-1}, ..., x_1) \tag{2.11}$$

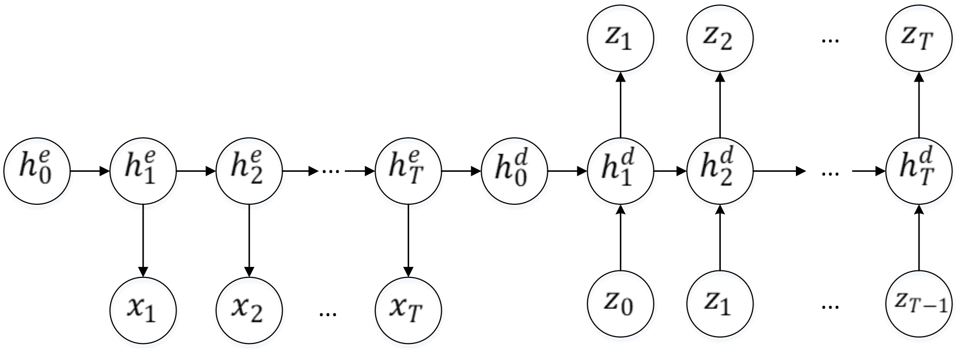


Figure 2.2: A Recurrent Autoencoder Structure.

2.2.2.2 Feed-Forward Neural Networks

Feed-forward networks are among the simplest deep learning models for text representation. Yet, they have achieved high accuracy on many text classification benchmarks. These models view a text as a bag of words. For each word, they learn a vector representation using an embedding model such as word2vec [60] or Glove [11], take the vector sum or average of the embeddings as the representation of the text, pass it through one or more feed-forward layers, known as Multi-Layer Perceptrons (MLPs), and then perform classification on the final layer's representation using a classifier such as logistic regression [61], Naïve Bayes [62], or

SVM [63]. An example of these models is the Deep Average Network (DAN) [1], whose architecture is shown in Figure 2.3. Despite its simplicity, DAN outperforms other more sophisticated models which are designed to explicitly learn the compositionality of texts. For example, DAN outperforms syntactic models on datasets with high syntactic variance. Joulin et al. [64] propose a simple and efficient text classifier called fastText. Like DAN, fastText views a text as a bag of words. Unlike DAN, fastText uses a bag of n-grams as additional features to capture local word order information. This turns out to be very efficient in practice while achieving comparable results to the methods that explicitly use the word order [65].

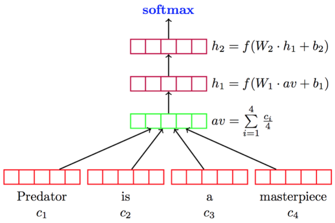

softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator
c_1
is
c_2
a
c_3
masterpiece
c_4

Figure 2.3: The Architecture of the Deep Average Network (DAN) [1].

Le and Mikolov [2] propose doc2vec, which uses an unsupervised algorithm to learn fixed-length feature representations of variable-length pieces of texts, such as sentences, paragraphs, and documents. As shown in Figure 2.4, the architecture of doc2vec is similar to that of the Continuous Bag of Words (CBOW) model [60, 66].

The only difference is the additional paragraph token that is mapped to a paragraph vector via matrix D. In doc2vec, the concatenation or average of this vector with a context of three words is used to predict the fourth word. The paragraph vector represents the missing information from the current context and can act as a memory of the topic of the paragraph. After being trained, the paragraph vector is used as features for the paragraph (e.g., in lieu of or in addition to BoW), and fed to a classifier for prediction. Doc2vec achieved new state-of-the-art results on several text classification and sentiment analysis tasks when it was published.

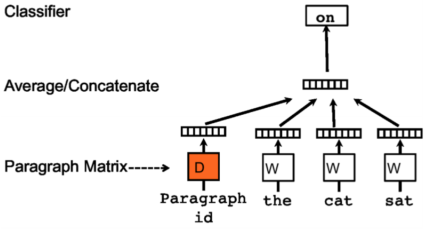


Figure 2.4: The doc2vec Model [2].

2.2.2.3 RNN-Based Models

RNN-based models view text as a sequence of words, and are intended to capture word dependencies and text structures for text classification. However, vanilla RNN models do not work well, and often underperform feed-forward neural networks. Among many variants of RNNs, Long Short-Term Memory (LSTM) is the most popular architecture, which is designed to better capture long term dependencies. LSTM addresses the gradient vanishing or exploding problems suffered by the vanilla RNNs by introducing a memory cell to remember values over arbitrary time intervals, and three gates (input gate, output gate, forget gate) to regulate the flow of information into and out of the cell. There have been works on improving RNNs and LSTM models for text classification by capturing richer information, such as tree structures of natural language, long-span word relations in text, document topics, and so on.

Tai et al. [3] have developed a Tree-LSTM model, a generalization of LSTM to tree-structured network typologies, to learn rich semantic representations. The authors argue that Tree-LSTM is a better model than chain-structured LSTM for NLP tasks because natural language exhibits syntactic properties that would naturally combine words to phrases. They validate the effectiveness of Tree-LSTM on two tasks: sentiment classification and predicting the semantic relatedness of two sentences. The architectures of these models are shown in Figure 2.5. Zhu et al. [67] also extend the chain-structured LSTM to tree structures, using a memory cell to store the history of multiple child cells or multiple descendant cells in a recursive process. They argue that the new model provides a principled way of considering long-distance interaction over hierarchies, e.g., language or image parse structures.

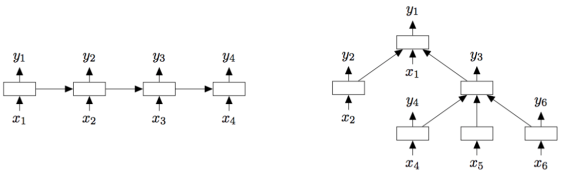


Figure 2.5: (Left) A Chain-structured LSTM Network and (right) a Tree-structured LSTM Network with Arbitrary Branching Factor [3].

To model long-span word relations for machine reading, Cheng et al. [68] augment the LSTM architecture with a memory network in place of a single memory cell. This enables adaptive memory usage during recurrence with neural attention, offering a way to weakly induce relations among tokens. This model achieves promising results on language modeling, sentiment analysis, and Natural Language Inference (NLI).

The Multi-Timescale LSTM (MT-LSTM) neural network [69] is also designed to model long texts, such as sentences and documents, by capturing valuable information with different timescales. MT-LSTM partitions the hidden states of a standard LSTM model into several groups. Each group is activated and updated at different time periods. Thus, MT-LSTM can model very long documents. MT-LSTM has been reported to outperform a set of baselines, including the models based on LSTM and RNN, on text classification.

RNNs are good at capturing the local structure of a word sequence, but face difficulties remembering long-range dependencies. In contrast, latent topic models are able to capture the global semantic structure of a document but do not account for word ordering. Bieng et al. [70] propose a TopicRNN model to integrate the merits of RNNs and latent topic models. It captures local (syntactic) dependencies using RNNs and global (semantic) dependencies using latent topics. TopicRNN has been reported to outperform RNN baselines for sentiment analysis.

2.2.2.4 CNN-Based Models

RNNs are trained to recognize patterns across time, whereas CNNs learn to recognize patterns across space [71]. RNNs work well for NLP tasks such as POS tagging or QA where the comprehension of long-range semantics is required, while CNNs work well where detecting local and position-invariant patterns is important.

These patterns could be key phrases that express a particular sentiment like “I like” or a topic like ”endangered species”. Thus, CNNs have become one of the most popular model architectures for text classification.

One of the first CNN-based models for text classification is proposed by Kalchbrenner et al. [4]. This model uses dynamic k-max pooling, and is called the Dynamic CNN (DCNN). As illustrated in Figure 2.6, the first layer of DCNN constructs a sentence matrix using the embedding for each word in the sentence. Then a convolutional architecture that alternates wide convolutional layers with dynamic pooling layers given by dynamic k-max pooling is used to generate a feature map over the sentence that is capable of explicitly capturing short and long-range relations of words and phrases. The pooling parameter k can be dynamically chosen depending on the sentence size and the level in the convolution hierarchy.

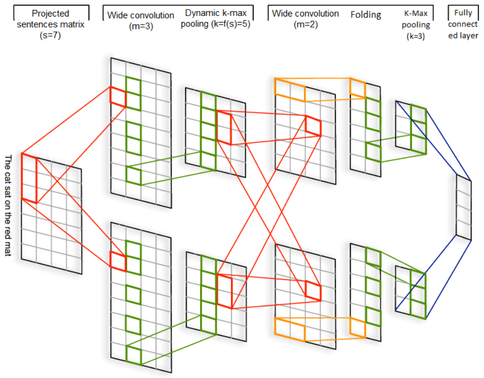


Figure 2.6: The Architecture of DCNN Model [4].

Later, Kim [5] proposed a much simpler CNN-based model than DCNN for text classification. As shown in Figure 2.7, Kim’s model uses only one layer of convolution on top of word vectors obtained from an unsupervised neural language model i.e., word2vec. Kim also compared four different approaches to learning word embeddings: (1) CNN-rand, where all word embeddings are randomly initialized and then modified during training; (2) CNN-static, where the pre-trained word2vec embeddings are used and stay fixed during model training; (3) CNN-non-static, where the word2vec embeddings are fine-tuned during training for each task; and (4) CNN-multi-channel, where two sets of word embedding vectors are used, both are initialized using word2vec, with One updated during model training while the other fixed. These CNN-based models were reported to improve upon the state-of-the-art on sentiment analysis and question classification.

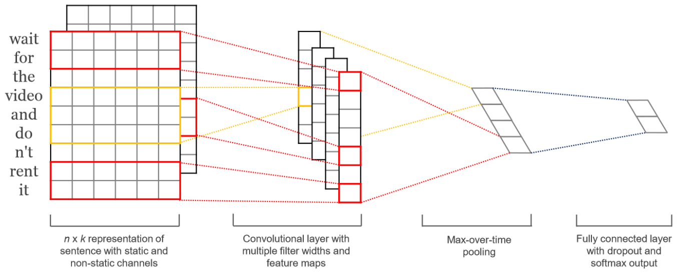

wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 2.7: The Architecture of a Simple CNN Model for Text Classification. Courtesy of Yoon Kim [5].

There have been efforts of improving the architectures of CNN-based models of [4, 5]. Liu et al. [72] propose a new CNN-based model that makes two modifications to the architecture of Kim-CNN [5]. First, a dynamic max-pooling scheme is adopted to captures more fine-grained features from different regions of the document. Second, a hidden bottleneck layer is inserted between pooling and output layer to learn compact document representations to reduce model size and boosts model performance. In [73, 74], instead of using pre-trained low-dimensional word vectors as input to CNNs, the authors directly apply CNNs to high-dimensional text data to learn the embeddings of small text regions for classification.

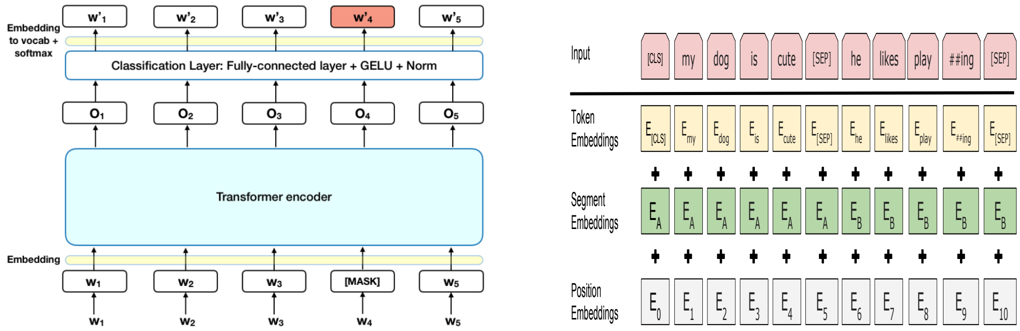
2.2.2.5 Transformers

One of the computational bottlenecks suffered by RNNs is the sequential processing of text. Although CNNs are less sequential than RNNs, the computational cost to capture relationships between words in a sentence also grows with increasing length of the sentence, similar to RNNs. Transformers [75] overcome this limitation by applying self-attention to compute in parallel for every word in a sentence or document an “attention score” to model the influence each word has on another. Due to this feature, Transformers allow for much more parallelization than CNNs and RNNs, which makes it possible to efficiently train very big models on large amounts of data on GPU clusters.

Since 2018 we have seen the rise of a set of large-scale Transformer-based Pre-trained Language Models (PLMs). Compared to earlier contextualized embedding models based on CNNs [76] or LSTMs [77], Transformer-based PLMs use much deeper network architectures (e.g., 48-layer Transformers [78]), and are *pre-trained* on much larger amounts of text corpora to learn contextual text representations by predicting words conditioned on their context. These PLMs have been *fine-tuned* using task-specific labels, and created new state-of-the-art in many downstream NLP tasks, including text classification. Although pre-training is unsupervised, fine-tuning is supervised learning.

PLMs can be grouped into two categories, autoregressive and autoencoding PLMs. One of the earliest autoregressive PLMs is OpenGPT [79, 78], a unidirectional model which predicts a text sequence word by word from left to right (or right to left), with each word prediction depending on previous predictions. OpenGPT can be adapted to downstream tasks such as text classification by adding task-specific linear classifiers and fine-tuning using task-specific labels.

One of the most widely used autoencoding PLMs is BERT [6]. Unlike OpenGPT which predicts words based on previous predictions, BERT is trained using the masked language modeling task that randomly masks some tokens in a text sequence, and then independently recovers the masked tokens by conditioning on the encoding vectors obtained by a bidirectional Transformer. BERT is trained on the BooksCorpus dataset (800M words) [available here] and text passages of English Wikipedia. Two pre-trained model sizes for BERT are available: BERT-Base and BERT-Large. BERT can be used directly from the pre-trained model on un-annotated data, or fine-tuned on one's task-specific data. Figure 2.8 illustrates the architecture and the input representation of BERT. There have been numerous works on improving BERT. BioBERT [80] is initialized with the general BERT model and pre-trained on PubMed abstracts and PMC full-text articles. It is further fine-tuned for biomedical text mining tasks such as named entity recognition (NER), question answering, and relation extraction. RoBERTa [81] is more robust than BERT, and is trained using much more training data. ALBERT [82] lowers the memory consumption and increases the training speed of BERT. DistillBERT [83] utilizes knowledge distillation during pre-training to reduce the size of BERT by 40% while retaining 99% of its original capabilities and making the inference 60% faster. BERT and its variants have been fine-tuned for various NLP tasks, including QA [84] and text classification [85].



(a) 1

(b) 2

Figure 2.8: a. BERT Architecture and b. Input layer Representation of BERT. Courtesy of Devlin et al. [6]

2.3 Neural Language Models and Word Embedding

Word embedding is a feature learning technique in which each word or phrase from the vocabulary is mapped to a N dimension vector of real numbers. Language modeling adopts data-driven approaches to capture salient statistical properties of text sequences in natural language, which can later be used to predict future words in a sequence, or to perform slot-filling in related tasks. The main idea behind this is to create a vector representation of the words with the hope that related words will end up closer to each other. Many embedding methods have been developed so far. The n-gram technique is a set of n-word which occurs "in that order" in a text set. This is not a representation of a text, but it could be used as a feature to represent a text. BOW is a representation of a text using its words (1-gram) which loses their order (syntactic). This model is very easy to obtain and the text can be represented through a vector, generally of a manageable size of the text. On the other hand, n-gram is a feature of BOW for a representation of a text using 1-gram. It is very common to use 2-gram and 3-gram. However, these models cannot capture long-distance dependence of tokens which often encodes semantic relations [86]. Therefore, there have been a lot of efforts of developing richer language models, among which one of the most successful is the neural language model [87].

Neural language models learn to represent textual-tokens (such as words) as dense vectors, referred as to word embeddings, in a self-supervised fashion. These learned representations can then be used for various NLP applications.
Vector-based representations have established their effectiveness in NLP tasks such as information extraction [46, 47], semantic role labeling [33, 34], word similarity [88] or word sense disambiguation [89] to name a few. One of the main drawbacks of the conventional Vector Space Models(VSM) approaches is the high dimensionality of the produced vectors. Since the dimensions correspond to words in the vocabulary, this number could easily reach hundreds of thousands or even millions, depending on the underlying corpus. To overcome this problem recently researchers are trying to directly learn low-dimensional word representations.

Learning low-dimensional vectors from text corpora can alternatively be achieved by exploiting neural networks. These models are commonly known as word embeddings and have been shown to provide valuable prior knowledge thanks to their generalization power ([90]). This property has proved to be decisive for achieving state-of-the-art performance in many NLP tasks. Recently, neural network based approaches which process massive amounts of textual data to embed word's semantics into low-dimensional vectors, the so-called word embeddings, have garnered a lot of attention [11, 12]. Word embeddings have demonstrated their effectiveness in storing valuable syntactic and semantic information [91]. In fact, they have been shown to be beneficial to many Natural Language Processing (NLP) tasks. A wide range of applications have reported improvements upon integrating word embeddings, including machine translation [92], syntactic parsing [93], text classification [5] and question answering [94], to name a few. However, despite their flexibility and success in capturing semantic properties of words, the effectiveness of word embeddings is generally hampered by an important limitation : the inability to discriminate among different meanings of a word. Therefore, accurately capturing the semantics of ambiguous words plays a crucial role in the language understanding of NLP systems. Depending on the textual units these models deal with embedding can be grouped into the following categories

- Static Embedding or Context-free

- Contextual Embedding

2.3.1 Static Embedding

After training a static word embedding model maps each word to a fixed vector irrespective of the context. For the past few decades static embeddings are gaining quite a lot of popularity. T. Mikolov et al. [60, 66] presented *'word to vector'* (Word2Vec [12]) representation as an improved word embedding architecture. The Word2Vec approach uses shallow neural networks with two hidden layers, continuous bag-of-words (CBOW), and the Skip-gram model to create a vector for each word. The Skip-gram model dives a corpus of words w and context c [95]. The goal is to maximize the probability:

$$arg \max_{\theta} \prod_{w \in T} \left[\prod_{c \in c(w)} p(c|w;\theta) \right] \tag{2.12}$$

where T refers to Text, and θ is parameter of $p(c|w;\theta)$.

Figure 2.9 shows a simple CBOW model which tries to find the word based on previous words, while Skip-gram tries to find words that might come in the vicinity of each word. The weights between the input layer and output layer represent $v \times N$ [96] as a matrix of w.

$$h = W^T c = W^T_{k,.} := v^T_{wI} \tag{2.13}$$

Continuous Bag-of-Words Model

The continuous bag-of-words model is represented by multiple words for a given target of words. For example, the word 'doctor' and 'hospital' as context words for 'patient' as the target word. This consists of replicating the input to hidden layer connections β times which is the number of context words [60]. Thus, the bag-of-words model is mostly used to represent an unordered collection of words as a vector. The first thing to do is create a vocabulary, which means all the unique words in the corpus. The output of the shallow neural network will be w_i that the task as *predicting the word given its context.* The number of words used depends on the setting for the window size (common size is 4–5 words).

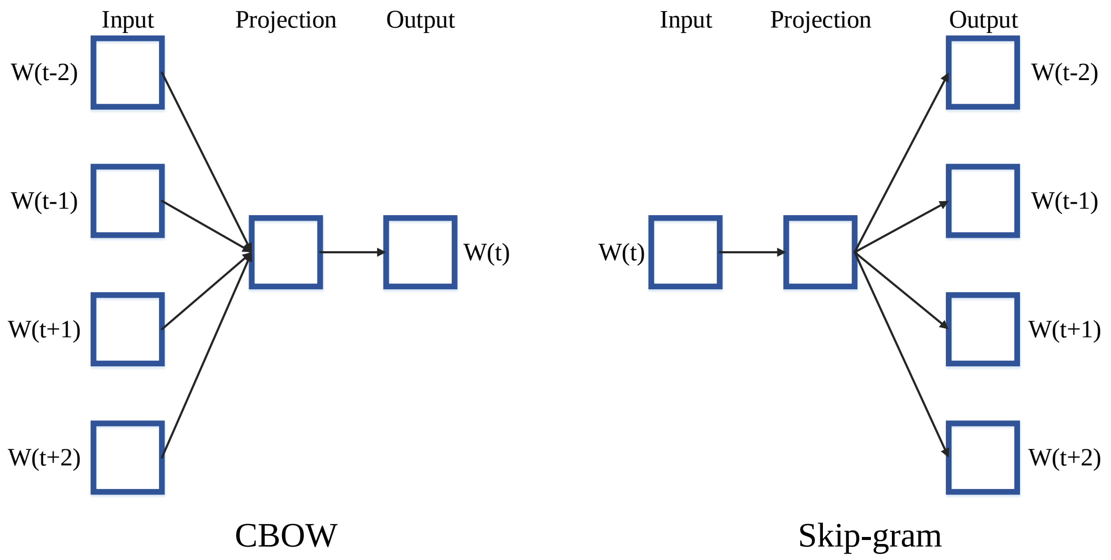


Figure 2.9: The CBOW and Skip-gram Architecture

Continuous Skip-Gram Model

Another model architecture which is very similar to CBOW [60] is the continuous Skip-gram model, however this model, instead of predicting the current word based on the context, tries to maximize classification of a word based on another word in the same sentence. The continuous bag-of-words model and continuous Skip-gram model are used to keep syntactic and semantic information of sentences for machine learning algorithms.

Global Vectors for Word Representation (GloVe)

Another powerful word embedding technique that has been used for text classification is Global Vectors (GloVe) [11]. The approach is very similar to the Word2Vec method, where each word is presented by a high dimension vector and trained based on the surrounding words over a huge corpus. The pre-trained word embedding used in many works is based on 400,000 vocabularies trained over Wikipedia 2014 and Gigaword 5 as the corpus and 50 dimensions for word presentation. GloVe also provides other pre-trained word vectorizations with 100, 200, 300 dimensions which are trained over even bigger corpora, including Twitter content. Figure 2.10 shows a visualization of the word distances over a sample data set. The objective function is as follows:

$$f(w_i - w_j, \widetilde{w}_k) = \frac{P_{ik}}{P_{jk}} \tag{2.14}$$

where w_i refers to the word vector of word i, and P_{ik} denotes to the probability of word k to occur in the context of word i.

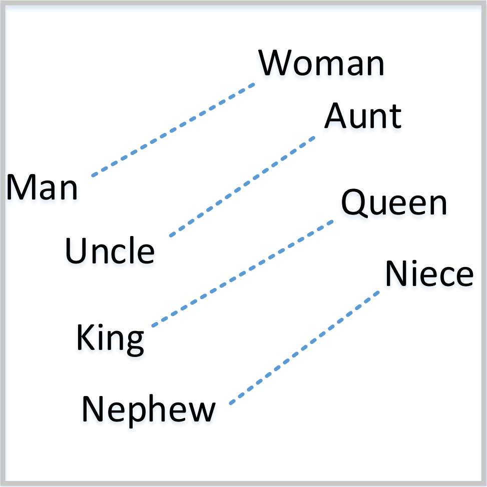


Figure 2.10: GloVe: Global Vectors for Word Representation.

These trained vectors can also be used to initialize the input representations for NLP models. We call these embeddings static embeddings because the embedding tables are invariant across different sentences. These pre-trained word vectors exhibit interesting properties and provide a performance gain over randomly initialized word vectors, but words rarely appear in isolation. Models that use pre-trained word vectors must learn how to use them. In glove and Word2vec the pre-trained embeddings specify a function which maps elements v in a (word) vocabulary V to vectors $h \in \mathbb{R}^d$ that is $f_{vocab} : v \rightarrow h$.

2.3.2 Contextual Embedding

Contextualized word embeddings have started gaining popularity in recent years. These are deep learning based language models and they are showing very promising results in various tasks like NER, sentiment analysis etc. A language modelling component is responsible for analyzing the context of the target word and generating its dynamic embedding. Unlike context-independent word embeddings, which have static representations, contextualized embeddings have dynamic representations that are sensitive to their context. Contextualized words embeddings aim at capturing word semantics in different contexts to address the issue of polysemous and the context-dependent nature of words. The sequence tagger of Li and McCallum [97] is one of the pioneering works that employ contextualized representations. Since 2011, with the introduction of word embeddings [60, 76] and the efficacy of neural networks, and in the light of meaning conflation deficiency of word embeddings, context-sensitive models have once again garnered research attention. In recent years, language-model-based contextualized word representation, such as ELMo [98], BERT [6], XLNet [99] and RoBERTa [81], has been proposed to replace static word embeddings as input representations. Pre-trained from large raw text, contextualized representations have been shown to improve the performance in many downstream tasks, such as question answering, reading comprehension, and co-reference resolution. Such embeddings are different from static embeddings in their parameterization. In addition to a lookup table, a sequence encoding neural network such as a multi-layer self attention network is used to represent global context information, giving them more representation power. One characteristic of contextualized representation is that the word vector differs under the different sentences, we thus call them dynamic embeddings. In these type of embedding models contextual representations of language are functions that have yet a more expressive type signature; they leverage the intuition that the meaning of a particular word in a particular text depends not only on the identity of a word, but also on the words that surround it at that moment. Let $(w_1, w_2, \cdots, w_N)$ be a text (say, a sentence) where each $w_i \in V$ is a word. A contextual representation of language is a function on the whole text, which assigns a vector representation to each word in the sequence:

$$f_{contextual} : (w_1, w_2, \cdots, w_N) \rightarrow (h_1, h_2, \cdots, h_N)$$

Contextualized word representations are mainly based on the context2vec [100], the method which uses bidirectional long short-term memory (LSTM). M.E. Peters et al. [77] built upon this technique to create the deep contextualized word representations technique. This technique contains both the main feature of word representation: (1) complex characteristics of word use (e.g., syntax and semantics) and (2) how these uses vary across linguistic contexts (e.g., to model polysemy) [77]. The main idea behind these word embedding techniques is that the resulting word vectors are learned from a bidirectional language model (biLM), which consist of both forward and backward LMs.

Chapter 3

Methodology

3.1 Introduction

In early text classification algorithms researchers used to leverage the statistical features to represent a document in feature space where they usually used the number of terms in vocabulary as the dimension of the vector space [101, 102, 103]. In those approaches term weighting plays a crucial role. Usually two steps are required after pre-processing the raw text for converting it to vectors: Indexing and term-weighting. Indexing is the job to assign the indexing terms for the documents whereas, Term weighting is the job to assign the weights of terms which measure the importance of terms in documents. Currently, there are many term weighting methods which are derived from the different assumptions of 'terms' characteristics or behaviors in texts. One very popular approach is IDF (inverse document frequency) [104] which assumes that the importance of a term is inversely proportional to the frequency of occurrence of this term in all the documents. IDF for a term j is defined as

$$idf(j) = log\frac{1+n}{1+df(D,j)} \tag{3.1}$$

The document frequency $df(D,j)$ is the number of documents in the corpus D that contain word j. On the other hand in recent years researchers are focusing more on the semantic understanding of texts. It is very important to extract the semantically important features from the text. Semantic features can be of two types: contextual and context-free. To extract the semantically important features from the data we have used both the approaches.

So in text classification, we are mainly concerned about two properties: semantic quality and statistical quality. Semantic quality is related with the meaning of the terms the document contains, i.e., to how much extent the terms represent the document. Statistical quality is related with the discriminative (resolving) power of the terms present in the document to positively discriminate it from other texts in the corpus. For statistical vector space model we have used $TF-IDF$ which is proposed by Sparck Jones [105] and has evolved from IDF with the heuristic intuition that a query term which occurs in many documents is not a good discriminator, and should be given less weight than one which occurs in few documents. TF-IDF is a classical way for term weighting which is defined as

$$TF.IDF(t_i, d) = tf_{t_i,d} \times idf(t_i) \tag{3.2}$$

where $tf_{t_i,d}$ is the number of occurrences of term t_i in document d and $idf(t_i)$ is the inverse document frequency of term t_i.

Sohrab et al. [10] used a different approach for automatic text classification (ATC). They have proposed Semantically-Augmented Statistical Vector Space Model (SAS-VSM) to infuse Word embeddings into existing statistical vector space models (e.g., term weighting with GloVe [11] vectors). They tried to infuse the continuous word embedding weights into discrete weights of term vectors. After transforming the vectors they have used SVM [63] for text classification and we will discuss on it in section 3.2. Han et. al [8] used centroid-based classification algorithm. In this approach they have represented each document d_i using TF-IDF vector representation

$$d_{tfidf} = \left(tf_1 log(\frac{N}{df_1}), tf_2 log(\frac{N}{df_2}), \cdots, tf_n log(\frac{N}{df_n}) \right)$$

where tf_i is the frequency of the ith term in the document. After that for each set of documents belonging to the same class, they have computed their centroid vectors. If there are k classes in the training set, this leads to k centroid vectors $C_1, C_2, \cdots, C_K$ where each C_i is the centroid for the i th class. The centroid vector for a class i, C_i is defined as

$$C_i = \frac{1}{|S|} \sum_{d \in S} d \tag{3.3}$$

where set S is the set of documents belonging to class i and d denotes the vector representation of the document. Given a new document t, first the TF-IDF weighted vector-space representation of t will be computed. Then the similarity between vector representation of t to all k centroids will be computed using the cosine measure

$$cos(t, C_i) = \frac{t.C_i}{\|t\|_2 * \|C_i\|_2} \tag{3.4}$$

Finally, based on these similarities, the document t will be assigned to the class corresponding to the most similar centroid. That is class of t will be determined by

$$\arg \max_{j=1,2,\cdots,k} cos(t, C_j) \tag{3.5}$$

Recently NLP researchers moved their focus towards context based vector space modeling of documents for text classification. Google's Bidirectional Encoder Representations from Transformers (BERT) [6] have recently gained much attention. Many BERT based models has emerged in recent years and showing state-of-the-art results. X-BERT [106] has tried to leverage the power of BERT in multi-label text classification tasks. Similarly many other models like Bio-BERT [80], DocBERT [107], SciBERT [16], Conversational BERT [108], Sentence Bert (S-BERT) [109], ALBERT [82] have been developed and are doing quite well in a wide range of NLP domains. Though these models have been trained to do specific tasks it is possible to extract embeddings out of their hidden layers. In fact the authors of BERT have demonstrated that token representations from the top four hidden layers can be used to get the embedding vectors. To get a contextual embedding of the documents we are also proposing to use the average token representations from the top four hidden layers of any domain specific appropriate BERT based model.

In this work we are proposing to infuse SAS-VSM with contextual embeddings to get even more feature rich vector space model. Our aim is to capture both the context-free and contextual semantic knowledge in the vector space model. We call it Contextualized Semantically-Augmented Statistical Vector Space Model

(C-SAS-VSM). So, in this approach we are first infusing the statistical vector space with context-free semantic vector space to give it continuous weights. This will generate the SAS-VSM representaion. After that we will merge the contextual semantic information to it. Finally for classification authors of SAS-VSM have used SVM [63] classifier but instead of SVM we will use class centroid vector based approach. This idea comes from the success of Galke et. al's [9] IDF re-weighted word centroid similarity (IWCS) approach which they have used for similarity scoring in practical information retrieval. In that article they have shown that if a term is only present in the test set but its synonym(s) present in the training set then the document would be scored by the TF-IDF retrieval model relatively low since the term does not occur frequently in the training set. To overcome this issue they have suggested IWCS as it will score the document higher because the vector representations for semantically similar words are close to each other in the embedding space. As we are also using TF-IDF in C-SAS-VSM so we will be using the class centroid based similarity measures as a text classifier algorithm.

3.2 Theory Behind Our Approach

In SAS-VSM the authors have tried to give context-free continuous word embeddings to the statistical vector space model. Statistical features play a very crucial role in text classification algorithms but semantic features are also very helpful in these tasks. In our understanding, to extract meaningful information from a text we need both the contextual and context-free embeddings. We are proposing a Contextualized SAS-VSM which is an extension of Sohrab's SAS-VSM [10] and leveraging the power of class centroid vectors for text classification.

Suppose there are n documents in the document space $D = \{d_1, d_2, \ldots d_n\}$ and they belong to k classes. A document d consists of sequence of terms or words $t_i = t_1, t_2, \ldots t_n$. Let V is the vocabulary set of corpus D having M elements. The context-free word embedding matrix W has dimension $M \times N$ where N is the dimension of each word embedding. That means each term t_i is represented in an N dimensional feature space. First we will use tf-idf to construct a statistical vector space of the documents as follows:

$$X^{stat} = \begin{bmatrix} f(t_1,d_1) & f(t_2,d_1) & \dots & f(t_M,d_1) \\ \vdots & \ddots & & \\ f(t_1,d_n) & f(t_2,d_n) & \dots & f(t_M,d_n) \end{bmatrix} \tag{3.6}$$

where $f(t_i, d_j)$ is the tf-idf weight of term t_i in document d_j. On the other hand the semantic vector space model (VSM) for the corpus D is defined as

$$X^{sem} = X^{stat} * W = \begin{bmatrix} f(t_1,d_1) & f(t_2,d_1) & \dots & f(t_M,d_1) \\ \vdots & \ddots & & \\ f(t_1,d_n) & f(t_2,d_n) & \dots & f(t_M,d_n) \end{bmatrix} * \begin{bmatrix} w_1 \\ \vdots \\ w_M \end{bmatrix} \tag{3.7}$$

where each row w_i of W represents the embedding vector for term t_i. This augmented features, X^{sem} for corpus D are incorporated with discrete and continuous weights. However the tf-idf weights are remained free from getting continuous weights. To infuse continuous weights into the existing discrete weights sum centroid embedding (SCE) has been used. For any document d, SCE is the sum of all continuous embedding weights for the column vectors of W that correspond to word embedding of terms in d. That is SCE for the corpus D is defined as

$$SCE = \begin{bmatrix} I(t_1,d_1) & I(t_2,d_1) & \dots & I(t_M,d_1) \\ \vdots & \ddots & & \\ I(t_1,d_n) & I(t_2,d_n) & \dots & I(t_M,d_n) \end{bmatrix} * W \tag{3.8}$$

where $I(t_i, d_j) = \begin{cases} 1, & if\ t_i \in d_j \\ 0, & otherwise \end{cases}$

So, SCE tries to capture the overall semantic sense of each document present in the corpus. Then the mean of the SCE which is called as centroid-means-embedding (CME) is computed for the corpus D using

$$CME = \frac{1}{N} \begin{bmatrix} \sum_{i=1}^{N} SCE_{1i} \\ \sum_{i=1}^{N} SCE_{2i} \\ \vdots \\ \sum_{i=1}^{N} SCE_{ni} \end{bmatrix} \tag{3.9}$$

where $CME \in \mathbb{R}^{n \times 1}$. Then the discrete weights of statistical VSM are updated using this CME based scalars as

$$X^{stat'} = X^{stat} \otimes CME \tag{3.10}$$

where $\otimes$ denotes the element-wise right-multiplication with each column. While computing X^{sem}, to avoid over-fitting of the newly generated weights scaling of the embedding vectors has been done by setting a hyper parameter. It is a Gaussian distribution based scaling function to scale each new generated weight for SAS-VSM. The hyperparameter $\lambda(d) = (\lambda_1, ..., \lambda_N)$ for document d is defined as:

$$\lambda_i(d) = \frac{1}{\sigma_d\sqrt{2\pi}} exp(-\frac{(x_i^{sem}(d) - \mu_d)^2}{2\sigma_d^2}) \tag{3.11}$$

where the mean μ_d and standard deviation σ_d for a document d are calculated as

$$\mu_d = \frac{1}{M}\sum_{i=1}^{M} f(t_i, d) \tag{3.12}$$

and

$$\sigma_d = \sqrt{\frac{1}{M}\sum_{i=1}^{M}(f(t_i) - \mu_d)^2} \tag{3.13}$$

Finally, the Semantically augmented statistical VSM is computed as

$$X^{sem'} = \Lambda \odot (X^{stat'}W) \tag{3.14}$$

where $\odot$ denotes the element-wise multiplication and $\Lambda = \begin{bmatrix} \lambda_{d_1} \\ \lambda_{d_2} \\ \vdots \\ \lambda_{d_n} \end{bmatrix}$.

Till now we have used the SAS-VSM approach. To infuse the contextualized meaning in the vector representations of the documents we are proposing to use any domain specific pre-trained BERT [6] model. As we are interested in biomedical domain we will use BioBERT [80] in our use case but the the same approach can be adopted in different domains by using a different embedding model.

Given a text document d, we will feed it to the embedding model and the model will first tokenize the text. Then as proposed by the authors of BERT we will use the average representations of the top four layers as embedding vectors for each token. Then we will take an average over all the vectorized tokens to get a document level embedding as follows:

$$Cont_d = \frac{1}{r}\sum_{i=1}^{r} encoded(t_i) \tag{3.15}$$

where the document d has r tokens $t_1, t_2, \ldots, t_r$ and $encoded(t_i)$ is the embedding of the i th token. The contextual embedding of the whole corpus D is denoted as

$$Cont_D = \begin{bmatrix} Cont_{d_1} \\ Cont_{d_2} \\ \vdots \\ Cont_{d_n} \end{bmatrix} \tag{3.16}$$

To merge these vectors with the existing SAS-VSM we have taken the average of these two representations as

$$X^{csem} = \frac{1}{2}\left(X^{sem'} + Cont_D\right) \tag{3.17}$$

and call it Contextualized Semantically-Augmented Statistical Vector Space Model (C-SAS-VSM).

After that we will use the Han et al [8] proposed Centroid-Based Document Classification technique to compute the centroid vectors for each class in the corpus D. The only difference is instead of using Han et al.'s original tf-idf vector representations we will be using the newly computed X^{csem} for classifying a new document t using (3.4) and Eq. (3.5). Our proposed method is illustrated in Figure 3.1.

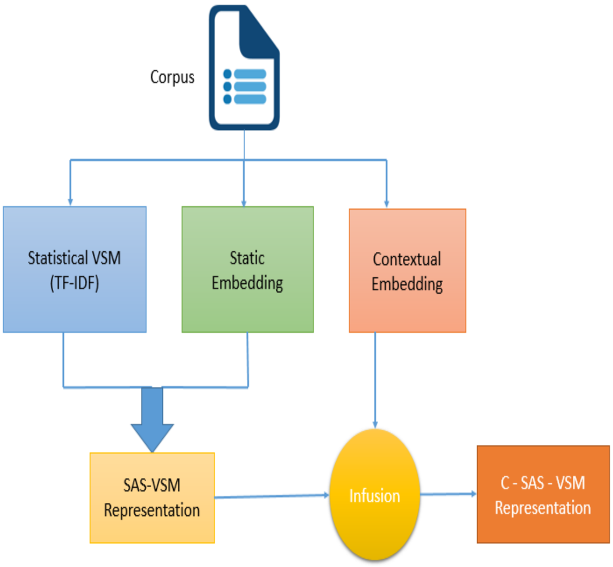


Figure 3.1: Illustration of C-SAS-VSM Approach

3.3 Algorithm

In this section we will briefly discuss about the algorithms used in our approach. First for the document corpus D we will compute the SAS-VSM representation. Then we will compute the contextual embeddings for each of the text documents $d \in D$. After that we will compute our proposed C-SAS-VSM using the average of SAS-VSM and contextual embeddings. Finally we will compute the class centroid vectors for text classification. The variables used in the algorithms can be found at Table 3.1.

Variable	**Description**
D	An Array of Text Documents (corpus)
V	An array of Vocabulary of the Corpus
W	Static Word Embedding Matrix
$embedding\_model$	The Model Used for Contextualized Embedding
$X^{sem'}$	The SAS-VSM Representation
$Class\_Labels$	An array of Class Labels
$Cont_D$	Contextualized Embedding Matrix
X^{csem}	Our proposed C-SAS-VSM Representation

Table 3.1: Glossary of the Variables Used in Algorithms

The algorithms are as follows:

Algorithm 1 *Compute_CME*

```
Input: $D$, $V$, $W$
Output: $CME$
  $num\_doc \leftarrow length(D)$
  $vocab\_size \leftarrow length(V)$
  $A \leftarrow$ an empty matrix of size $num\_doc \times vocab\_size$
  for $i$ in $range(num\_doc)$ do
    for $j$ in $range(vocab\_size)$ do
      if the $j$ th term of the vocabulary V belongs to text document $i$ then
        $A[i][j] \leftarrow 1$
      else
        $A[i][j] \leftarrow 0$
      end if
    end for
  end for
  $SCE \leftarrow AW$
  $CME \leftarrow$ row means of $SCE$
```

The algorithm *Compute_CME* [Alg. 1] computes the centroid-means-embedding (CME) using the Equation 3.8 and Equation 3.9.

Algorithm 2 Compute_SAS-VSM

```
Input: $D$, $V$, $W$
Output: $X^{sem'}$
  $X^{stat} \leftarrow TF - IDF(D)$ //creates the TF-IDF representation of D
  $X^{sem} \leftarrow X^{stat}(D).W$
  $CME \leftarrow$ Compute_CME(D,V,W)
  $X^{stat'} \leftarrow X^{stat} \otimes CME$                 // using Eq. (3.10)
  $\Lambda =$ Compute_lambda($X^{sem}, X^{stat}$)       // using Eq. (3.11)-(3.13)
  $X^{sem'} \leftarrow \Lambda \odot (X^{stat'}W)$
```

Then the algorithm $Compute\_SAS - VSM$ [Alg. 2] computes the SAS-VSM representation of the corpus. We have used the same methods used by Sohrab [10]. The hyperparameter Λ are calculated using Eq. (3.11)-(3.13). It is used to overcome the fitting as suggested by the authors of SAS-VSM.

Algorithm 3 Contextual_Embedding

```
Input: $D$, $embedding\_model$
Output: $Cont_D$
  $num\_doc \leftarrow length(D)$
  $Cont_D \leftarrow$ an empty matrix of size $num\_doc \times$ vector dimension
  for $i$ in $range(num\_doc)$ do
    feed the $i$ th text document to the $embedding\_model$ and take the average
        of the top four layers and assign it to the $i$ th row of $Cont_D$ matrix
  end for
```

The algorithm *Contextual_Embedding* [Alg. 3] computes the contextualized embeddings of the whole corpus D using a pre-trained model. Next, we will use this embeddings to update the existing $X^{sem'}$ representation of corpus D.

Algorithm 4 C-SAS-VSM

```
Input: $X^{sem'}$, $Cont_D$
Output: $X^{csem}$
  $X^{csem} \leftarrow \frac{1}{2}\big(X^{sem'} + Cont_D\big)$
```

Finally, we merge the contextual embeddings with the SAS-VSM embeddings using the $C - SAS - VSM$ algorithm [Alg. 4]. This algorithm infuses both contextual and context-free semantic features in the statistical feature vector space (i.e. TF-IDF representation).

Algorithm 5 Compute_Centroid

```
Input: $X^{csem}$, $Class\_Labels$
Output: $C$
  $Uniq\_label \leftarrow$ array of unique $Class\_labels$
  $C \leftarrow$ an empty matrix of size $length(Uniq\_label)\times$ number of columns in $X^{csem}$
  for each $i$ in $Uniq\_label$ do
    compute the mean over all the vector representations of the documents which
    belong to class $i$ and assign it to the $i$ th row of C
  end for
```

At last we compute the class centroid vectors by averaging out all the vector representations of any particular class using the algorithm *Compute_Centroid* [Alg. 5]. Now, for any new document t we will first calculate its C-SAS-VSM representation and then classify it to one of the classes using Eq. (3.4) and (3.5).

Chapter 4

Experimental Evaluation

In this section, we will briefly explain the dataset and the experimental setup. Then we will compare and analyze the different outputs obtained using different approaches. We will also demonstrate how our proposed method is an improvement over the existing one. We have evaluated our proposed method with three different setups.

4.1 Description of Data

We are using past USFDA deficiencies to classify them under different sub-categories. The dataset is confidential and it is property of Dr. Reddy's Laboratories Ltd. Due to a non-diclosure agreement with the organization we won't be able to show the exact data or any statistics of it. To better understand the context we will only give an overview of the dataset.

The dataset consists of USFDA deficiencies against ANDAs filed by DR. Reddys Laboratories ranging from year 1999 to 2019. The exact number of deficiencies is confidential. The dataset is mainly divided into two parts:

1. Deficiency text

2. Subcategory

The deficiencies are English text containing mainly drug chemistry and pharmaceuticals related terms. Each deficiency is a USFDA action point on which the agency (USFDA) wants to get more clarification and/or submission of additional supporting evidence(s). The deficiencies are short texts and the mean sentence count is 6. There are 7 sub-categories in the dataset each denoting a class. The dataset is slightly imbalanced. Each sub-category is handled by one business unit(BU). To get a better understanding we have provided a brief idea about the deficiencies for each of the class labels.

1. **Drug Substance**: The ANDA must contain the details of the chemicals used to produce the drug. This section is popularly known as Drug API. APIs are the active ingredients of any drug. These APIs have to be chemically same as of the innovator drug. The drug manufacturer must have to use USFDA approved drug APIs for approvals. Many pharmaceutical companies use their in-house drug APIs. In that case they have to report all the tests and procedure for manufacturing the API. On the other hand there many companies which procure these drug APIs from third-party vendors. In that case they can only acquire these APIs from FDA approved manufacturers along with they need to submit a certificate of the approval from the procuring company. If any deficiency comes due to problems in the third-party APIs then the submission will be delayed until the API manufacturer obtains the approval from USFDA.

2. **Drug Product**: Drug product is the final product which the pharmaceutical company intendeds to sell in the market. To get approval the companies must specify all the binding agents, preservatives, flavorants, dyes, colour and strength of the drugs. Sometimes there are fixed strength wise colour codings of the drugs and the applicant has to follow it strictly. There are many cases where the generic drug strength and colour coding mismatches with the innovator and that leads to discarding entire manufactured lots. The generic drug should be in the same physical form as of its innovator counterpart. That is if an innovator drug is in liquid form then the generic drug should also be in liquid form; if the innovator drug has some specific coating on the tablet then the same should be there in the generic drug also. These minor details are very important as after developing and testing the drug efficacies if they fail in physical comparison then sometimes the whole drug development needs to be carried agian from the scratch.

3. **Stability**: To access the stability of any drug USFDA requires many

stability tests under different conditions which varies based on the drug specific guidelines. The manufacturer needs to submit all prescribed analytical results for the stability of the drug. The ANDA must include all long-term stability data in various temperatures and humidity levels. These results are very critical as if the applicant fails to justify the results then it can lead to a major deficiency and that means direct rejection of the application. In addition to that to correct the stability results the whole drug development and R&D process needs to be revised which is very time as well as cost-effective.

4. **Process and Development**: The USFDA needs each and every details starting from the lab level drug development to the factory level manufacturing process. The whole process needs to be documented properly along with justification of various tests and their results to assess the drug integrity over all the batches. One of the major concerns in this area is the applicant needs to provide blend uniformity of the chemicals throughout different batches. In addition to that an applicant also needs to submit the exact amount of chemical ingredients used in the whole process as that needs to be consistent between the lab -scale and factory scale production. Drug manufacturer has to submit all the process parameters and various process control methods. They also need to quantify all the known and unknown impurities in the product and try to contain them within acceptable limits.

5. **Biologics**: Biologics of a drug broadly contains the bio-pharmaceutics and bio equivalence of a drug product. Drug manufacturers need to establish suitable drug delivery systems like in-vitro release of the drug. Currently release is assessed using a variety of methods including sample and separate (SS), continuous flow (CF), dialysis membrane (DM) methods. They have to provide acceptable release specification results to USFDA. In case of bioequivalence the generic drug and the innovator drug should exhibit comparable release specifications. This includes Comparative Mean Nebulization Time (MNT), Mean Delivered Dose, sameness of polymorphic form of the drug substance based on X-ray diffraction, sameness of shape (crystalline habit) of the drug substance, comparative Unit Dose Content (UDC) of drug. This tests are very critical as the generic drug is supposed to replicate the innovator drug in terms of dosage and efficacy.

6. **Container**: The applicant must provide all the drug container related details. Drug manufacturer needs to come up with a suitable sampling plan. Each lot of components, drug product containers, and closures shall be withheld from use until the lot has been sampled, tested, or examined, as appropriate, and released for use by the quality control unit. The containers shall be opened, sampled, and resealed in a manner designed to prevent contamination of their contents and contamination of other components, drug product containers, or closures. Sterile equipment and aseptic sampling techniques shall be used when necessary. While submitting to the USFDA containers from which samples have been taken shall be marked to show that samples have been removed from them. While doing the inspection each drug component shall be tested for conformity with all appropriate written specifications for purity, strength, and quality. In addition to that they need to test drug product containers and closures so that they shall not be reactive, additive, or absorptive so as to alter the safety, identity, strength, quality, or purity of the drug beyond the official or established requirements.

7. **Dose and Administration (Labeling)**: In the submitted ANDA the applicant has to provide details of intended marketing strategies like different drug strengths and dosages. In addition to that the medication guide and all other drug related information should be properly documented in the ANDA. According to USFDA guidelines the generic drug's labeling should be in-line with the innovator drug. The proposed medication guide and routes of drug administration have to be same as the innovator drug. Drug labeling also should be strictly matching with the innovator's one. This includes the position, description and even the font sizes used in various parts of the drug label.

Now, with keeping all these FDA guidelines in mind the task is to automatically classify any new deficiency to one of the above mentioned classes so that the regulatory team can redirect it to the concerned BU to get it resolved at the earliest.

4.2 Evaluation Techniques

The goal of any text classification algorithm is to correctly classify texts under different classes with high accuracy score. Accuracy is the proportion of correct classifications among all classifications. But relying only on accuracy can lead to poor results. When we use accuracy, we assign equal cost to false positives and false negatives. Measuring only accuracy can be very dangerous in case of highly imbalanced datasets. In those cases if almost all the data points belong to one class then even if the accuracy is high the model will be unable to learn the patterns of the other Small classes. To correctly measure one model's learning ability we should focus on two major things: (1) the model should be able to learn patterns present in a class and ultimately detect it in text datasets and (2) when it is able to detect a particular class with how much accuracy it classifies data point to that particular class. The first point is popularly known as recall and the second is known as precision score. In literature, while choosing the best model to balance out these two scores one usually combine both the recall and precision scores and generates f1-measure. As our data is slightly imbalanced we will take into account both accuracy and f1-measure to choose the best model.

4.3 Experimental Setup

To generate the proposed C-SAS-SVM representations we need to get the context-free and contextual embeddings. In SAS-VSM approach the authors have infused the statistical vector space with context-free semantic vector space to give it continuous weights. In our case the static embedding vectors are generated using Word2Vec [12] model. The original Word2Vec model is trained on Google News corpus but as we are working with pharmaceutical related biomedical domain we have used a model whose vectors were induced from PubMed and PMC texts. In this model each embedding vector belongs to a 200 dimensional feature space. The pre-trained word vectors are generated by the NLP group of Turku University, Finland which is publicly available **here**.

In addition to this to get more feature rich vector space model we also need to get contextual embeddings. To generate these contextual embeddings we are leveraging the representation power of Bio-BERT [80], which has been first trained on BERT-BASE (available **here**) model and then trained on English Wikipedia and BooksCorpus data and then further trained on PubMed and PMC corpora. In this model each embedding vector belongs to a 768 dimensional feature space. This pre-trained Bio-BERT model is publicly available **here**. To combine the contextual embeddings in the SAS-VSM we need to have same dimensionality and for that we have used PCA [27] for dimension reduction and transformed the contextual vectors into 200 dimensional vectors. Finally, for classification we have used class Centroid based document classification technique [8].

We have done a comparative analysis of our method with two other approaches. As we are using Sohrab et al.'s [10] SAS-VSM approach we have tabulated the results using their method in the section 4.4. In their paper they have used SVM [63] classifier for classification on top of the SAS-VSM. As our dataset is different we have tuned using grid-search approach to get the best hyperparameters for SVM. We have used linear kernel and got the best result using $C = 50$ (C is the cost parameter). In this approach we have generated the embedding vectors using pre-trained Word2Vec [12] vectors which are trained on Wikipedia and PubMed articles to better capture the bio-medical context.

We have also used the pre-trained Bio-BERT [80] representations as contextualized embeddings to generate classification results using SVM with linear kernel. Here also we did a grid search and got the best value of the cost hyperparameter as $C = 30$.

After that we have generated results for our proposed approach and classified the deficiencies using SVM (linear kernel) to get a better idea of how infusing both context-free and contextual embeddings together can help us in text classification tasks. In this case we obtained the hyperparameter value as $C = 15$. All these results have been tabulated in the next section 4.4.

Finally we tested our approach which is based on SAS-VSM and Bio-BERT embedding vectors and used the class centroids vectors as the classifier algorithm to classify the texts into different classes.

4.4 Analysis of Results

In this section we have done a detailed comparison between the four approaches. To have a better understanding and comparison between the approaches we have summarized the overall accuracy and the f1-score in Table 4.1.

Metric	**SAS-VSM+SVM**	**Bio-BERT+SVM**	**Our+SVM**	**Our+Centroids**
Accuracy	0.69	0.71	0.85	0.91
f1-score	0.58	0.61	0.79	0.86

Table 4.1: Comparison Between Different Approaches

To have a detailed analysis we have also provided class-wise precision, recall and f1-scores to better analyze the performances. As the data is confidential we are not providing any kind of statistics like confusion matrix which can lead to breaching of the non-disclosure agreement.
The results with analysis for the four approaches are as follows:

1. SAS-VSM Using SVM:

Class	**Precision**	**Recall**	**f1-score**
Biologics	0.78	0.80	0.79
Container	0	0	0
Dose and Administration(Labeling)	0.80	0.69	0.74
Drug Product	0.59	0.72	0.65
Drug Substance	0.60	0.43	0.5
Process and Development	0.63	0.76	0.69
Stability	0.78	0.69	0.73

Table 4.2: Classification Report Using SAS-VSM and SVM

In this approach we can see that the model is unable to detect the Container class. The overall performance is also very poor. One interesting pattern is prominent which is the classes whose context are somewhat similar have suffered more than contextually unrelated classes. For example, context of container class and Process and Development class are quite similar. Development of a suitable container also falls under process unit. On the other hand biologics and stability are two completely different classes so their f1-scores are on the higher side. One possible explanation of this phenomenon can be SAS-VSM approach uses tf-idf internally to generate the statistical vector space model and class-wise vocabulary of related classes will also be very much related. So data points from similar classes will get similar kind of vector representations and the model fails to draw a good decision boundary between them.

2. Bio-BERT Embeddings Using SVM:

Class	**Precision**	**Recall**	**f1-score**
Biologics	0.80	0.81	0.81
Container	0.23	0.07	0.1
Dose and Administration(Labeling)	0.75	0.76	0.75
Drug Product	0.63	0.65	0.64
Drug Substance	0.53	0.52	0.52
Process and Development	0.66	0.69	0.67
Stability	0.66	0.68	0.67

Table 4.3: Classification Report for Bio-Bert Embeddings Using SVM

In class-wise comparison contextual embeddings have done better job than the SAS-VSM approach. Here, the model has done slightly better in case of 'container' class. We can see that overall performance of the model in stability class and the precision score in the drug substance class have decreased. As drug stability is directly related to the drug substances used so they are quite related. Maybe that is a reason for this kind of behaviour of this model. But SAS-VSM based and Bio-BERT based models have comparable overall accuracy and f1-score.

3. C-SAS-VSM Using SVM:

Class	**Precision**	**Recall**	**f1-score**
Biologics	0.90	0.90	0.90
Container	0.76	0.32	0.45
Dose and Administration(Labeling)	0.89	0.88	0.88
Drug Product	0.80	0.83	0.82
Drug Substance	0.77	0.78	0.78
Process and Development	0.83	0.86	0.84
Stability	0.80	0.86	0.83

Table 4.4: Classification Report of Our Approach Using SVM

In this approach we can see that the recall for the container class is low but the precision is quite good. That means this model usually fails to detect the class but when it does it identifies it with high accuracy. Here the model has outperformed both the above two approaches in terms of class-wise f1-score as well as overall accuracy and f1-score.

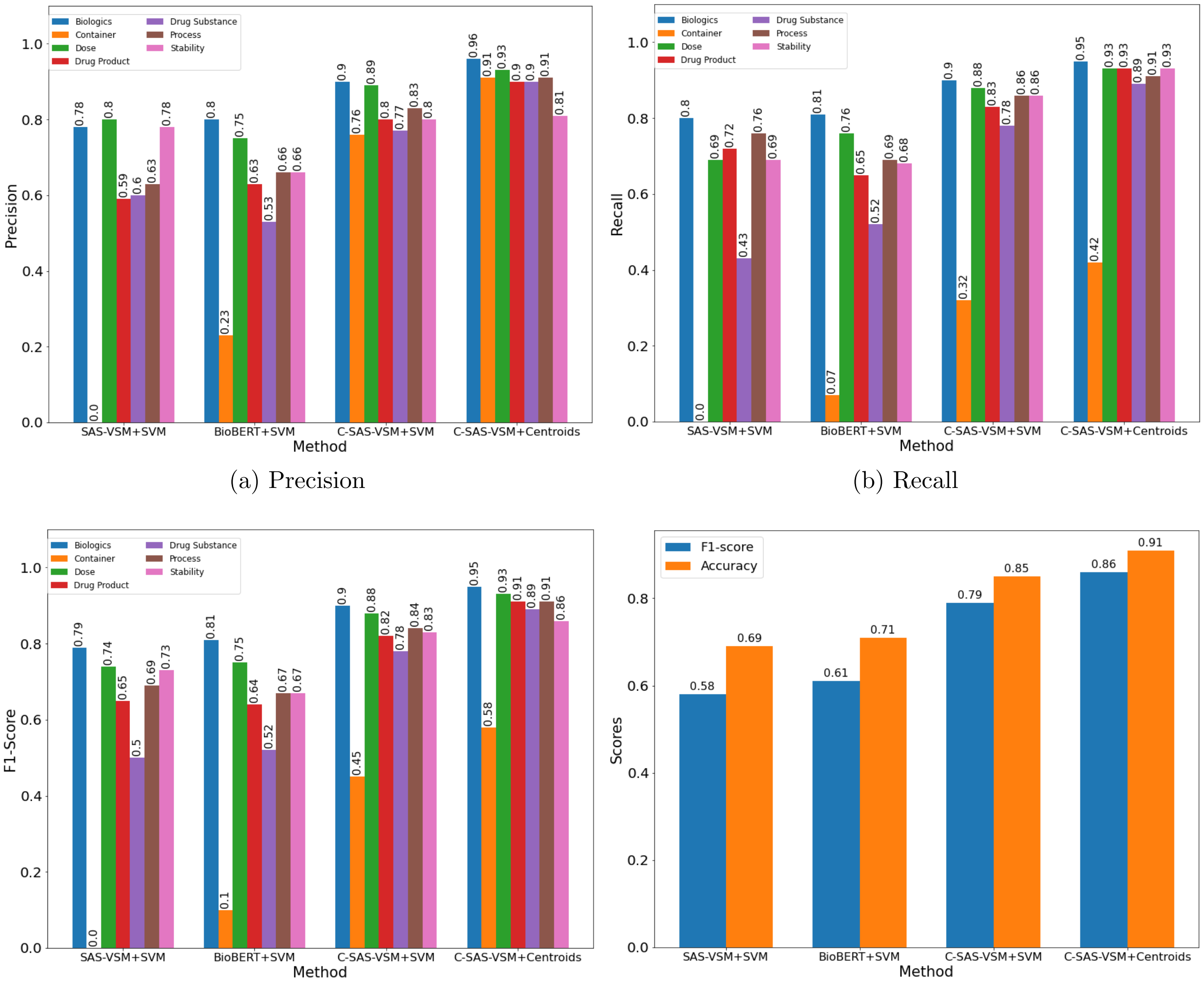
4. C-SAS-VSM Using Class Centroid Vectors:

Class	**Precision**	**Recall**	**f1-score**
Biologics	0.96	0.95	0.95
Container	0.91	0.42	0.58
Dose and Administration(Labeling)	0.93	0.93	0.93
Drug Product	0.90	0.93	0.91
Drug Substance	0.90	0.89	0.89
Process and Development	0.91	0.92	0.91
Stability	0.81	0.93	0.86

Table 4.5: Classification Report for Our Approach Using Centroid Vectors

Our approach to merge context-free embeddings with the contextual vectors and using a class centroid vector approach has outperformed all other approaches on top of which it is built on. This model clearly beats them in terms of class-wise f1-score as well as the overall f1 and accuracy score. The class centroid vectors play a very crucial role here. They tries to capture the overall sense through a single data point over the feature space. As we have transformed the vectors with statistical, context-free embeddings and contextual embeddings this helps the model to extract better features out of the data and that directly effects the performance of the model.

To enhance the readability of the various results and have a better comparison between class-wise performances of the models we have added various plots below.



(a) Precision

(b) Recall

(c) F1-Score

(d) Overall Score

Figure 4.1: **(a) - (c)**: Class-wise Precision, Recall and F1-Score, **(d)**: Overall F1-Score and Accuracy

Chapter 5

Discussions

Today in the age of digitization everyday we generate a huge amount of data and a significant portion of which are in the form of text. To exploit the knowledge out of these texts we need to have better understanding of the context. In this thesis we have worked on bio-medical and pharmaceutical related texts. The dataset on which we are working is very different from what NLP researchers usually work on. We demonstrated that to extract better features it is better to leverage the power of statistical as well as semantic features of the text.

In semantic feature extraction from a text we have to keep in mind both the context-free and contextual features. Context-free features help in understanding the overall context of a document. It also helps in general to extract the summary of the text. But in the semantic feature extraction contextual features play an equally important role. The contextual features help us in understanding the document specific context. In SAS-VSM [10], the approach was to infuse the static continuous weights into the discrete weights of statistical vector space model. We tried to extend the same thing by infusing contextual continuous weights into them. We also tried to centralize the overall concept of a single class into one data point with the use of class centroid vectors. The overall approach works well in case of our dataset. To better capture the meaning we used models and vector representations pre-trained on bio-medical corpora. The contribution of these models are very significant in our approach.

Chapter 6

Conclusions and Scope of Further Research

Humans are generating a lot of data in form of text. To utilize the true power of these data researchers are continuously working on bringing new techniques. In this work we have just shown that using domain-specific semantic knowledge can improve the understanding of complex documents. There is a lot of scope to improve our approach.

1. **Generation of Contextualized Embeddings**: Contextualized embeddings can help in extracting the underlying features in a better way in comparison to just using static embeddings. We have generated the contextual embeddings using the sum of the top four layers as suggested by the authors of the BERT model. To further increase the performance of our model we can try with other approaches for generating the contextual embeddings by taking different combinations of the hidden layer representations.

2. **Pre-training on Classification Task**: To generate the embeddings we have used the pre-trained models in our case due unavailability of large datasets at hand. In future we would like to update the embeddings by pre–training on domain specific classification tasks using large datasets.

3. **Infusing Contextual Embedding with Statistical Model**: In this work we have taken the average over the two different vector space models. Our aim was to show that leveraging the power of domain specific contextualized embeddings can help in extracting good features. In future we would like to

extend this approach by experimenting with other combinations of infusing the two different kinds of weights to improve the performance of the model.

4. **Classification Algorithm**: We have used the class centroid based approach to classify the documents. In future we would like to try deep-learning based approaches to understand and utilize the true potential of this approach.

Bibliography

[1] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, Beijing, China, July 2015. Association for Computational Linguistics.

[2] Quoc Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, page II–1188–II–1196. JMLR.org, 2014.

[3] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566, Beijing, China, July 2015. Association for Computational Linguistics.

[4] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[5] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, Doha, Qatar, October 2014. Association for Computational Linguistics.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[7] Karan B. Thakkar and Gauri Billa. The concept of: Generic drugs and patented drugs vs. brand name drugs and non-proprietary (generic) name drugs. *Frontiers in Pharmacology*, 4, 2013.

[8] Eui-Hong Han and George Karypis. Centroid-based document classification: Analysis and experimental results. In *Principles of Data Mining and Knowledge Discovery*, pages 424–431. Springer Berlin Heidelberg, 2000.

[9] *Evaluating the Impact of Word Embeddings on Similarity Scoring in Practical Information Retrieval*. Zenodo, September 2017.

[10] Mohammad Golam Sohrab, Makoto Miwa, and Yutaka Sasaki. Centroid-means-embedding: An approach to infusing word embeddings into features for text classification. In *Advances in Knowledge Discovery and Data Mining*, pages 289–300. Springer International Publishing, 2015.

[11] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.

[12] Tomas Mikolov, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

[13] M. P. Akhter, Z. Jiangbin, I. R. Naqvi, M. Abdelmajeed, A. Mehmood, and M. T. Sadiq. Document-level text classification using single-layer multisize filters convolutional neural network. *IEEE Access*, 8:42689–42707, 2020.

[14] Paul Ferguson, Neil O'Hare, Michael Davy, Adam Bermingham, Paraic Sheridan, Cathal Gurrin, and Alan F Smeaton. *Exploring the use of paragraph-level annotations for sentiment analysis of financial blogs*. 2009.

[15] Rangareddy Jayashree, K. Srikantamurthy, and Basavaraj S. Anami. Suitability of naïve bayesian methods for paragraph level text classification in the kannada language using dimensionality reduction technique.

International Journal of Artificial Intelligence —& Applications, 4:121–131, 2013.

[16] Iz Beltagy, Kyle Lo, and Arman Cohan. SciBERT: A pretrained language model for scientific text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3615–3620, Hong Kong, China, November 2019. Association for Computational Linguistics.

[17] Plaban Kr. Bhowmick, Anupam Basu, Pabitra Mitra, and Abhishek Prasad. Multi-label text classification approach for sentence level news emotion analysis. In *Lecture Notes in Computer Science*, pages 261–266. Springer Berlin Heidelberg, 2009.

[18] Siddharth Gopal and Yiming Yang. Multilabel classification with meta-level features. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, page 315–322, New York, NY, USA, 2010. Association for Computing Machinery.

[19] Zellig S. Harris. Distributional structure. *WORD*, 10(2-3):146–162, August 1954.

[20] Yiming Yang. Noise reduction in a statistical approach to text categorization. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '95, page 256–263, New York, NY, USA, 1995. Association for Computing Machinery.

[21] Yiming Yang and Jan O. Pedersen. A comparative study on feature selection in text categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*, ICML '97, page 412–420, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc.

[22] Gini index. In *The Concise Encyclopedia of Statistics*, pages 231–233. Springer New York.

[23] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, March 1986.

[24] R. Steuer, J. Kurths, C. O. Daub, J. Weise, and J. Selbig. The mutual information: Detecting and evaluating dependencies between variables. *Bioinformatics*, 18(Suppl 2):S231–S240, October 2002.

[25] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, September 1990.

[26] Thomas Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, page 50–57, New York, NY, USA, 1999. Association for Computing Machinery.

[27] Ian Jolliffe. Principal component analysis. In *International Encyclopedia of Statistical Science*, pages 1094–1096. Springer Berlin Heidelberg, 2011.

[28] Sutanu Chakraborti, Rahman Mukras, Robert Lothian, Nirmalie Wiratunga, Stuart N. K. Watt, and David J. Harper. Supervised latent semantic indexing using adaptive sprinkling. In *IJCAI*, 2007.

[29] Haohan Wang and Bhiksha Raj. On the origin of deep learning. *arXiv preprint arXiv:1702.07800*, 2017.

[30] Wen-tau Yih, Xiaodong He, and Christopher Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[31] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, page 373–374, New York, NY, USA, 2014. Association for Computing Machinery.

[32] Aliaksei Severyn and Alessandro Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in*

Information Retrieval, SIGIR '15, page 373–382, New York, NY, USA, 2015. Association for Computing Machinery.

[33] Jie Zhou and Wei Xu. End-to-end learning of semantic role labeling using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1127–1137, Beijing, China, July 2015. Association for Computational Linguistics.

[34] Alexey Mazalov, Bruno Martins, and David Matos. Spatial role labeling with convolutional neural networks. In *Proceedings of the 9th Workshop on Geographic Information Retrieval*, GIR '15, New York, NY, USA, 2015. Association for Computing Machinery.

[35] Aliaksei Severyn and Alessandro Moschitti. Twitter sentiment analysis with deep convolutional neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, page 959–962, New York, NY, USA, 2015. Association for Computing Machinery.

[36] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A neural network for factoid question answering over paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 633–644, Doha, Qatar, October 2014. Association for Computational Linguistics.

[37] Wenpeng Yin, Sebastian Ebert, and Hinrich Schütze. Attention-based convolutional neural network for machine comprehension. In *Proceedings of the Workshop on Human-Computer Question Answering*, pages 15–21, San Diego, California, June 2016. Association for Computational Linguistics.

[38] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[39] Orhan Firat, Kyunghyun Cho, and Yoshua Bengio. Multi-way, multilingual neural machine translation with a shared attention mechanism. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 866–875, San Diego, California, June 2016. Association for Computational Linguistics.

[40] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2326–2335, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[41] H. Wu, Y. Gu, S. Sun, and X. Gu. Aspect-based opinion summarization with convolutional neural networks. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3157–3163, 2016.

[42] Luís Marujo, Wang Ling, Ricardo Ribeiro, Anatole Gershman, Jaime Carbonell, David Martins de Matos, and João P. Neto. Exploring events and distributed representations of text in multi-document summarization. *Knowledge-Based Systems*, 94:33–42, February 2016.

[43] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[44] Hongzhao Huang, Larry Heck, and Heng Ji. Leveraging deep neural networks and knowledge graphs for entity disambiguation. *arXiv preprint arXiv:1504.07678*, 2015.

[45] Thien Huu Nguyen, Avirup Sil, Georgiana Dinu, and Radu Florian. Toward mention detection robustness with recurrent neural networks. *arXiv preprint arXiv:1602.07749*, 2016.

[46] Thien Huu Nguyen and Ralph Grishman. Combining neural networks and log-linear models to improve relation extraction. *arXiv preprint arXiv:1511.05926*, 2015.

[47] Yan Xu, Lili Mou, Ge Li, Yunchuan Chen, Hao Peng, and Zhi Jin. Classifying relations via long short term memory networks along shortest

dependency paths. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1785–1794, 2015.

[48] Feifan Liu, Jinying Chen, Abhyuday Jagannatha, and Hong Yu. Learning for biomedical information extraction: Methodological review of recent advances. *arXiv preprint arXiv:1606.07993*, 2016.

[49] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[50] Wei Wang, Ying Huang, Yizhou Wang, and Liang Wang. Generalized autoencoder: A neural network framework for dimensionality reduction. *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 496–503, 2014.

[51] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, DTIC Document, 1985.

[52] Hong Liang, Xiao Sun, Yunlei Sun, and Yuan Gao. Text feature extraction based on deep learning: a review. *EURASIP Journal on Wireless Communications and Networking*, 2017(1), December 2017.

[53] Pierre Baldi. Autoencoders, unsupervised learning, and deep architectures. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham Taylor, and Daniel Silver, editors, *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, volume 27 of *Proceedings of Machine Learning Research*, pages 37–49, Bellevue, Washington, USA, 02 Jul 2012. PMLR.

[54] Sarath Chandar A P, Stanislas Lauly, Hugo Larochelle, Mitesh Khapra, Balaraman Ravindran, Vikas C Raykar, and Amrita Saha. An autoencoder approach to learning bilingual word representations. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 1853–1861. Curran Associates, Inc., 2014.

[55] Jonathan Masci, Ueli Meier, Dan Cireşan, and Jürgen Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Proceedings*

of the 21th International Conference on Artificial Neural Networks - Volume Part I, ICANN'11, page 52–59, Berlin, Heidelberg, 2011. Springer-Verlag.

[56] K. Chen, M. Seuret, M. Liwicki, J. Hennebert, and R. Ingold. Page segmentation of historical document images with convolutional autoencoders. In *2015 13th International Conference on Document Analysis and Recognition (ICDAR)*, pages 1011–1015, 2015.

[57] Jie Geng, Jianchao Fan, Hongyu Wang, Xiaorui Ma, Baoming Li, and Fuliang Chen. High-resolution SAR image classification via deep convolutional autoencoders. *IEEE Geoscience and Remote Sensing Letters*, 12(11):2351–2355, November 2015.

[58] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA, 2014. MIT Press.

[59] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar, October 2014. Association for Computational Linguistics.

[60] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[61] Raymond E Wright. Logistic regression. 1995.

[62] David D Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *European conference on machine learning*, pages 4–15. Springer, 1998.

[63] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[64] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[65] Sida Wang and Christopher Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 90–94, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[66] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA, 2013. Curran Associates Inc.

[67] Xiaodan Zhu, Parinaz Sobihani, and Hongyu Guo. Long short-term memory over recursive structures. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 1604–1612, Lille, France, 07–09 Jul 2015. PMLR.

[68] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 551–561, Austin, Texas, November 2016. Association for Computational Linguistics.

[69] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 2326–2335, Lisbon, Portugal, September 2015. Association for Computational Linguistics.

[70] Adji B. Dieng, Chong Wang, Jianfeng Gao, and John W. Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *CoRR*, abs/1611.01702, 2016.

[71] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. 1998.

[72] Jingzhou Liu, Wei-Cheng Chang, Yuexin Wu, and Yiming Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, page 115–124, New York, NY, USA, 2017. Association for Computing Machinery.

[73] Rie Johnson and Tong Zhang. Effective use of word order for text categorization with convolutional neural networks. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 103–112, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[74] Rie Johnson and Tong Zhang. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 562–570, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[75] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, undefinedukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc.

[76] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[77] Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[78] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[79] Alec Radford. Improving language understanding by generative pre-training. 2018.

[80] Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *CoRR*, abs/1901.08746, 2019.

[81] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[82] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*, 2020.

[83] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC—$\hat{2}$ Workshop*, 2019.

[84] Siddhant Garg, Thuy Vu, and Alessandro Moschitti. Tanda: Transfer and adapt pre-trained transformer models for answer sentence selection. 2019.

[85] Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. How to fine-tune BERT for text classification? In *Lecture Notes in Computer Science*, pages 194–206. Springer International Publishing, 2019.

[86] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition.* Prentice Hall PTR, USA, 1st edition, 2000.

[87] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[88] Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich, and Shaul Markovitch. A word at a time: Computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, page 337–346, New York, NY, USA, 2011. Association for Computing Machinery.

[89] Roberto Navigli. Word sense disambiguation: A survey. *ACM Comput. Surv.*, 41(2), February 2009.

[90] Yoav Goldberg. A primer on neural network models for natural language processing. *Journal of Artificial Intelligence Research*, 57:345–420, 2016.

[91] Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[92] Will Y Zou, Richard Socher, Daniel Cer, and Christopher D Manning. Bilingual word embeddings for phrase-based machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1393–1398, 2013.

[93] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, 2015.

[94] Antoine Bordes, Sumit Chopra, and Jason Weston. Question answering with subgraph embeddings. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 615–620, Doha, Qatar, October 2014. Association for Computational Linguistics.

[95] Yoav Goldberg and Omer Levy. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *CoRR*, abs/1402.3722, 2014.

[96] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[97] Wei Li and Andrew McCallum. Semi-supervised sequence modeling with syntactic topic models. In *Proceedings of the 20th National Conference on Artificial Intelligence - Volume 2*, AAAI'05, page 813–818. AAAI Press, 2005.

[98] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proc. of NAACL*, 2018.

[99] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5754–5764, 2019.

[100] Oren Melamud, Jacob Goldberger, and Ido Dagan. context2vec: Learning generic context embedding with bidirectional LSTM. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 51–61, Berlin, Germany, August 2016. Association for Computational Linguistics.

[101] Franca Debole and Fabrizio Sebastiani. Supervised term weighting for automated text categorization. In *Text Mining and its Applications*, pages 81–97. Springer Berlin Heidelberg, 2004.

[102] Flora S. Tsai and Agus T. Kwee. Experiments in term weighting for novelty mining. *Expert Systems with Applications*, May 2011.

[103] Zhong Tang, Wenqiang Li, and Yan Li. An improved term weighting scheme for text classification. *Concurrency and Computation: Practice and Experience*, 32(9), May 2020.

[104] Iadh Ounis. Inverse document frequency. In *Encyclopedia of Database Systems*, pages 1570–1571. Springer US, 2009.

[105] Karen Spärck Jones. IDF term weighting and IR research lessons. *Journal of Documentation*, 60(5):521–523, October 2004.

[106] Wei-Cheng Chang, Hsiang-Fu Yu, Kai Zhong, Yiming Yang, and Inderjit S. Dhillon. A modular deep learning approach for extreme multi-label text classification. *CoRR*, abs/1905.02331, 2019.

[107] Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*, 2019.

[108] Yasuhito Ohsugi, Itsumi Saito, Kyosuke Nishida, Hisako Asano, and Junji Tomita. A simple but effective method to incorporate multi-turn context with BERT for conversational machine comprehension. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 11–17, Florence, Italy, August 2019. Association for Computational Linguistics.

[109] Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics.