Text Classification With Deep Neural Networks

Thesis

Text Classification With Deep Neural Networks

Trung Huynh

Knowledge Media Institute

The Open University

Doctor of Philosophy

July 2019

Abstract

The thesis explores different extensions of Deep Neural Networks in learning underlying natural language representations and how to apply them in Natural Language Processing tasks. Novel methods of learning lower or higher level features of natural languages are given in which word and phrase dense representations are derived from unlabelled corpora. Word representations are learned by training Deep Neural Networks to predict context from each sentence while phrase representations are learned by unsupervised learning with Convolutional Restricted Boltzmann Machine. It is shown that word representations learned from architectures which preserve text input as sequences have better word similarity and relatedness than bag-of-word approaches. Additionally phrase representations learned with Convolutional Restricted Boltzmann Machine when combined with bag-of-word features improve results of text classification tasks over only bag-of-word features. Beside learning word and phrase representations, to the best of my knowledge, the work in the thesis is first to explore Deep Neural Networks in Adverse Drug Reaction detection task where my architectures when used with pre-trained word representations significantly outperform the state-of-the-art models. In addition, outputs from my proposed attentional architecture can be used to highlight important word spans without explicit training labels. In the future I propose the learned representations to be used with the discussed Deep Neural Networks in different NLP tasks such as Dialog Systems, Machine Translation or Natural Language Inference.

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

The ability for computers to understand and extract information from languages is one of the central problems of Artificial Intelligence. The field of Natural Language Processing (NLP) was born to foster research that helps computers tackle challenges such as text classification, machine translation, natural language generation, reading comprehension, sentiment analysis and so on. We first talk about the text classification task and then discuss about how other NLP tasks can be derived into forms of text classification.

A text classification task is a task where we are given a set X of documents and a set of classes $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$. An example is $\mathbb{X} = \{$"La Vie En Rose", "Life in rosy colours"$\}$ and $\mathbb{C} = \{\text{French}, \text{English}\}$ for a language detection task. We wish to learn a classification function (or a model) $\gamma\colon \mathbb{X} \to \mathbb{C}$. Popular text classification tasks include topic classification, sentiment analysis, subjectivity and objectivity classification.

Solutions to many of NLP tasks however can be derived from more granular classifiers. For example in machine translation, given a finite dictionary of the target language, the translation can be generated by correctly classifying the next word from the dictionary and the translation stops at a special word that indicates the end. Similarly in reading comprehension where computers give answers to asked questions, the answers can be generated by classifying each word consecutively from a finite dictionary. While in some other tasks such as topic classification or some simple forms of sentiment analysis, the required technique is in the pure form of text classification. Given text classification is one of the most prominent tasks or sub-tasks in the many NLP areas, my research goal is to develop novel methods in this field of NLP.

Text classification models are usually trained in entirely supervised manner with labelled data. The limited availability of labelled data confines trained models in terms of both performance and capability. Labelling data can be expensive and require experts for annotation. In addition models trained in one domain or one task can not be transferred into a different domain or a different task. It is desirable that we distil knowledge from unlabelled data and apply the learned knowledge across different tasks and domains. Since the dawn of the internet, the amount of retrievable data has increased significantly. In addition to printed data, nowadays we have vast amount of data in the digital forms of blogs, micro-blogs, comments, reviews, articles, e-books, chats, files, emails, videos. In fact, according to a study by Gantz and Reinsel (2011), the digital universe is more than doubled every two years. Along with the impressive growth in data volume, new hardware has also elevated the available computational power. Devices like GPUs, TPUs and other ASICs. An NVIDIA Tesla P100 can deliver up to 50x performance boost over an Intel E5 CPU[1]. The current development in both data volume and hardware has intrigued me to search for methodologies of **how to leverage the increasing availability of large cor-**

[1] `https://www.nvidia.com/en-us/data-center/tesla-p100/`

pora and computational power to learn better language representations and utilise the learned representations in supervised text classification? Specifically the overall goal of the thesis is to look for answers of:

- How can we utilise unlabelled data to learn language representations?
- How can we utilise Deep Neural Networks with pre-trained representations from unlabelled data for supervised text classification?

There are usually three major language-specific components in a text classification task: data, feature representations and the model. Given the current growth of data size, we have found remaining limitations in the areas of feature representations and modelling:

1. **Feature representations - the word representations:** the traditional text classifiers usually break documents into small word fragments (n-grams) and represent them as separate dimensions in the fragment hyperspaces. Each document is a vector with weighted numbers of occurrences of these fragments as its dimensional magnitudes. Although this representation can preserve word orders to some extent by increasing the word fragment size, doing so increases the number of dimensions of the fragment hyperspace prohibitively. The growth of hyperspace requires proportional data sets and computing power for efficient training. These requirements are usually impractical with limited available data sets and contemporary computing power. A workaround is to project these words into a dense low-dimensional space so that words appearing in similar context are projected to be close to each other in the new space. Different approaches to learning these representations including constructing or factorising

Pointwise Mutual Information (PMI) matrices and Neural Networks to predict context using unlabelled corpora have achieved relative successes. However matrix factorisation and these shallow networks are still considered bag-of-word models as they do not preserve the sequential property of languages while learning word representations. I hypothesise that training word representations with sequential models like Recurrent Neural Networks (RNNs) helps preserve sequential attributes of words in their learned representations. These types of models have the advantages of sequential models but disadvantages of different training problems such as gradient exploding or vanishing. It is also hard to train these sequential models in parallel. In this study, we discuss how to overcome these problems to be able to use RNNs for training word representations.

2. **Feature representations - the higher level features:** common approaches to text classification with deep learning are to project separate words into low-dimensional word representations and feed these values into supervised Deep Neural Networks for final classifications. The problem with this approach is that while word representations can be trained in an unsupervised manner with abundance of unlabelled data, the discriminative Deep Neural Networks need to be trained with labelled data, which are usually scarce. To overcome this issue, we propose to use generative Deep Neural Networks, specifically Convolutional Restricted Boltzmann Machines (RBMs), to learn higher level features from unlabelled data. Weights of these RBM layers can be potentially used to initialise the weights of deep Convolutional Neural Networks (CNNs) which are trained with labelled data.

3. **The models:** we look for applying word representations and Deep Neural Networks into a text classification task, specifically Adverse Drug Reactions (ADRs) where shallow models with engineered features are dominant. Recently unstructured data such as medical reports or social network data have been used to detect content that contains ADRs. Case reports published in scientific bio-medical literature are abundant and generated rapidly. Social networks are other generous sources of data with unstructured format. While an individual tweet or Facebook status that contains ADRs may not be clinically useful, a large volume of these data can expose serious or unknown consequences. Common approaches to detect content with ADRs are to use maximum entropy classifiers with engineered features. These features normally include n-grams with different weighting schemes. These models suffer the same problems of that we discussed previously. In this research, we investigate different Deep Neural Network architectures for ADR detection. We show that even without engineered features, our Neural Networks with word embeddings outperform maximum-entropy classifiers with different weighting schemes for n-gram features.

1.2 Contribution and outline of this thesis

Common approaches to training word representations are bag-of-word methods. These methods are usually computationally efficient and can be trained with corpora that consist of billions of words. However like other bag-of-word models in other NLP tasks, these methods disregard word ordering that is one of the fundamental properties of languages. I introduce different sequential models that work with sequential data and still can be trained for corpora up to billions of tokens. Word representations produced by these models have superior performance than those produced by state-of-the-art methods with small data sets and on-par with these when trained with a large book corpus. Next in order to leverage the abundance of large unlabelled corpora, I explored the usages of generative models in learning higher levels features of sentences. Previously Restricted Boltzmann Machines were proven to be useful for pre-training Deep Neural Networks for image classifications but were not studied on NLP tasks. Using Convolutional Restricted Boltzmann Machines, I trained Neural Networks that produce higher level features from input sentences. These higher levels features, when combined with traditional features, improve text classification results. Finally we apply word representations and different Deep Neural Networks to improve the state-of-the-art in detecting Adverse Drug Reactions from text documents.

Chapter 2 gives some background on the history of text classification and recent state-of-the-art methods in text classification including methods to train word representations and different Deep Neural Network architectures. The next three chapters outline my work in learning word representations, higher level features of sentences and their applications in Adverse Drug Reaction detection. Chapter 6 distils these findings, discusses shortcomings and finally proposes future directions for potential improvements.

1.2.1 Chapter 3 summary: Learning word representations with sequential modelling

In this chapter, I investigate different Recurrent Neural Networks that learn word representations by summarising sentences into fixed size vectors and use them to predict appearances of words in surrounding sentences. Matrix factorisation methods

(Pennington et al. 2014) or window-based methods (Mikolov et al. 2013) are essentially bag-of-word based, which are efficient but do not utilise sequential property of languages. Meanwhile sequential models like RNNs have been widely used in language models, there is not much work in optimising these models for learning word representations. It naturally raises a research question:

RQ1: How can sequential models be trained to learn word representations from unlabelled data?

I hypothesise that word representations learned using these sequential model preserve sequential attributes of languages and, thus, are better word representations for text classification. Specifically I train encoder-decoder models where the encoders go through each word sequentially. Word representations are learned so that the model can predict word occurrences given the encoded representation. Compared to sequence-to-sequence models where, in prediction phase, each word is predicted conditioned on the previously predicted word, my models predict word occurrences independently in parallel. Therefore the prediction phase is more efficient and hundreds of words can be predicted at the same time instead of one at a time as in sequence-to-sequence models. At the end, I report different tricks to train the Neural Networks and quality evaluations of my word representations compared to state-of-the-art methods.

1.2.2 Chapter 4 summary: Learning higher level features with Convolutional Restricted Boltzmann Machines

The generative Neural Network, Convolutional Restricted Boltzmann Machine (CRBM), was previously applied successfully to learn higher-level features from images (Lee et al. 2009). These pre-trained networks and produced higher level features from raw images are proven to be useful for downstream image classification. A sentence where each word is embedded by a numerical vector becomes a matrix where CRBMs can be applied, similarly to a grey-scale image. In this chapter, I address a research question:

RQ2: How we can utilise CRBMs to reconstruct these embedded sentences and higher-level features from trained hidden layers for downstream text classification?

As convolutions are applied to the inputs, hidden layers are expected to capture interactions of consecutive words or phrases. In supervised tasks, outputs of these hidden layers then can be used as additional features to word-level features. I experimented with two data sets, MPQA Subjectivity classification and Movie Review data sets. For each of the data sets, in unsupervised phase, CRBMs are trained with pre-trained word embeddings. In supervised phase, features produced by CRBMs are combined with sums of word embeddings and bag-of-word features for classification. Overall models trained using word embeddings outperform traditional maximum entropy classifier with tf-idf features in both Subjectivity and MR data sets. The additional features trained from CRBMs help improve the performance of our classifiers in both data sets. In addition, nearest neighbours of example phrases in the projected space are shown to have some certain related semantic. The content of this chapter is based on the paper by Huynh et al. (2015).

1.2.3 Chapter 5 summary: Adverse Drug Reaction classification with Deep Neural Networks

In this chapter, I investigate the application of pre-trained word representations and Deep Neural Networks in Adverse Drug Reaction (ADR) content classification. Detecting ADRs is critical for healthcare providers in both pre-market and post-market safety monitoring. The process is expensive and samples are limited in numbers. The ability to automatically detect ADR content from medical records and social media content is thus very valuable to both healthcare providers and patients. Before my work, popular approaches including SVM, Maximum Entropy or Naive Bayes were used. Aligned with the original research question, in this chapter I address a research question:

RQ3: How can we utilise pre-trained word representations and Deep Neural Networks to improve Adverse Drug Reaction classification?

Specifically I experimented with training different Deep Neural Network architectures including CNN, Recurrrent Convolutional Neural Network (RCNN), Convolutional Recurrent Neural Network (CRNN) and Convolutional Neural Networks with Attention (CNNA) using Glove word representations. My experiments show that all the Deep Neural Networks outperform the baseline results including a maximum entropy classifier with tf-idf features that is also a baseline in Chapter 4. Additionally attention weights from CNNA can be used to extract important words in the decision making. This is valuable as the ability to intuitively interpret classification decisions helps detecting classifier's flaws and gaining trust from classifier end users. The content of this chapter is based on the paper by Huynh et al. (2016).

1.2.4 Summary

We will now introduce related background and analyses of traditional methods of text classification and new methods to learn language representations. The reviewed literature in next chapter inspires and provides motivations of experiments in the rest of the dissertation.

Chapter 2

Background

The problem of single-label text classification is defined as follows. $\mathcal{D} = X_1, \ldots, X_N$ is a set of training text chunks. Each chunk is labelled with a class label from 1 to k. For a given text record with an unknown label, a trained model is used to predict the class (from 1 to k) or the probability of this instance assigned to each class. Overall a text classification is a sub-task of general classification problems. Therefore approaches that are applicable to generic classification problems are also applicable to the text classification problem with naive input features (e.g. one-hot-encoding features). In the scope of this literature review, only data sets and techniques that are unique to text classification are discussed.

There are two common sets of approaches to text classification: rule-based approaches (Brill 1995; Han et al. 2003; Tayyar Madabushi and Lee 2016) and Machine Learning-based approaches (Kowsari et al. 2019). Rule-based approaches use sets of hand-crafted linguistic rules to group text into different classes. The rule-based approaches are easy to be interpreted by human and give the owners full control of adding new rules or removing existing ones. However, these approaches require deep domain knowledge and extensive effort for experts to curate rules. These systems are harder to maintain and improve once the size of rules grows as adding a new rule could lead to ineffectiveness of existing ones. The Machine Learning-based approaches can address these issues by automatically deriving rules from pre-labelled data with Machine Learning algorithms (classifiers). Labelling data is usually much easier and requires less expert knowledge than generating rules. Some labelled data are also abundant and generated publicly, for example Amazon customer reviews with ratings[1] or news data with classified topics[2]. However this is not always the case. Labelled data in many tasks such as machine translation, where a document in an original language needs to be translated to a target language, or reading comprehension, where computers need to give answers to questions related a document, are much less abundant and are expensive to be generated. In addition, the vast number of topics and complex semantic expressed in natural language make it hard to transfer what a machine learns from one task to another. One usually needs to train a new Machine Learning model for each different task. Therefore the effectiveness of Machine Learning approaches is often limited to tasks where the size of available labelled data is large enough and target domains are popular. This is highly inefficient as learned knowledge is discarded. There are a few remedies to this problem. First we know that the majority of words and phrases in the same language have the same meaning in different documents or domains. It is easy to see that if this information is transferred and reused in different tasks, one would need less labelled data than learning from scratch. Second many words have similar meanings (synonyms), opposite meanings (antonyms) or are simply related. This leads to the need of compressing the original space of unique words where each unique word is a separate dimension into a denser lower-dimensional space. Third one can notice that unlabelled data are actu-

[1] `https://s3.amazonaws.com/amazon-reviews-pds/readme.html`

[2] `https://www.kaggle.com/rmisra/news-category-dataset`

ally heavily labelled depending on how we frame a task. For example, an unlabelled document can be seen as labelled data in language models where one has to predict word by word in the document sequentially. With this observation, we bring the vast amount of information to available supervised models. Inspired by these motivations, recently there has been significant effort in extracting knowledge from abundant unlabelled data and utilise the new learned knowledge in supervised tasks to improve their effectiveness (Collobert et al. 2011; Mikolov et al. 2013; Pennington et al. 2014; Devlin et al. 2019). Typically word and sequence representations (their projections in a lower dimensional space) are learned from unlabelled data. Inferred representations of inputs are then fed into a classifier for final classification. In this chapter I survey methods that are common in text classification and in learning representations from unlabelled or labelled data. First I discuss different popular text classification models. Next I review different methods that generate word representations and how to evaluate them. Finally I review methods that learn sequence representations and their evaluation.

2.1 The classifiers

A classification model is technically a function that transforms an input into a vector of probabilities whose values are probabilities that the input is classified into different classes. This Section explains models used in popular baselines as well as discussing their strengths and weaknesses.

2.1.1 Naive Bayes

Naive Bayes is one of the simplest methods for classification and is particularly popular within text classification owing to its simplicity, scalability and effectiveness.

Therefore Naive Bayes remains a popular baseline in text classification (Russell and Norvig 2002).

In Naive Bayes, the probability of a document d belonging to class c can be computed according to Bayes' theorem:

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)}$$

as $P(d)$ is a common denominator for all classes, we can write

$$P(c|d) \propto P(d|c)P(c). \tag{2.1}$$

A document d can be tokenised into a sequence of tokens $t_1, t_2, \ldots, t_k$. Naive Bayes is "naive" due to its assumption of the independence of each token given the class c. Therefore, Equation 2.1 can be written as

$$P(c|d) = P(c) \prod_{1 \leq i \leq k} P(t_i|c) \tag{2.2}$$

The probability of class c can be estimated directly from training data as

$$\hat{P}(c) = \frac{N_c}{N}, \tag{2.3}$$

where N_c is the number of documents which belong to class c and N is the number of documents in the training set. The estimated probability $\hat{P}(t_i|c)$ is the ratio of number of times the token t_i appearing in class c documents and total number of tokens in all class c documents.

Although Naive Bayes is simple and scalable, there are limitations in this method. First training data usually does not contain all possible term-class associations. Second both $\hat{P}(c)$ and $\hat{P}(t_i|c)$ are approximations and can be far from their true values depending on the data coverage and quality. Third the independence assumption is unrealistic. One can overcome this problem by expanding tokens into bags of n-grams. However doing so introduces additional noise from even rougher approximations as n-grams are less popular than uni-grams that compose them.

2.1.2 Maximum Entropy Classifier

Another popular family of models in text classification is Maximum Entropy (MaxEnt) classifiers (Berger et al. 1996). The Maximum Entropy concept adopts Occam's razor of the least complex hypotheses and is somewhat similar to the concept of regularisation. Specifically a MaxEnt model seeks to maximise the conditional entropy

$$H(p) = -\sum_{d,c} \tilde{p}(d) p(c|d) \log p(c|d), \tag{2.4}$$

where $\tilde{p}(d)$ is the empirical distribution of x. With some transformation using Lagrange multiplier, the solution to Equation 2.4 is

$$\lambda^* = \arg\max_{\lambda} - \sum_{d} \tilde{p}(d) \log Z_\lambda(d) + \sum_{i} \lambda_i \tilde{p}(f_i),$$

where f_i is feature i and $Z_\lambda(d)$ is a normalising constant

$$Z_\lambda(d) = \sum_{c} \exp\left(\sum_{i} \lambda_i f_i(d, c) \right).$$

Training MaxEnt models can be done with iterative scaling algorithms such as Generalized Iterative Scaling and Improved Iterative Scaling or gradient-based methods such as L-BFGS and Coordinate Descent (Yu et al. 2011).

The MaxEnt model is closely related to Naive Bayes but offers more flexibility to pass different types of features. The main drawback is the features have to be manually engineered and, often, require expert or domain knowledge. In addition, it is not clear how we can leverage knowledge from unlabelled data to the classification task.

2.1.3 Support Vector Machine

Support Vector Machine (SVM) is also another model that is extensively used in text classification. Amongst models with maximum two layers of transformations, Support Vector Machine (SVM) constantly appears as one of the strong performers (Turney 2002; Pawar and Gawande 2012). In the simplest form, an SVM is a binary classifier that constructs a hyperplane that has the largest distance to the nearest training-data point of any class (Vapnik and Lerner 1963). Formally we assume a training data set of n points $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where y_i is either -1 or 1. We can select two parallel hyperplanes that separate the two classes of data so that the distance between them is as large as possible. These hyperplanes can be defined as

$$w \cdot x - b = 1$$

and

$$w \cdot x - b = -1.$$

As the distance between the hyperplanes is $\frac{2}{\|w\|}$, we want to minimise $\|w\|$ under the constraints

$$y_i(w \cdot x - b) \leq 1 \quad \forall i \ 1 \leq i \leq n.$$

For data that are not linearly separable, a soft-margin goal is minimised instead:

$$\left[\frac{1}{n}\sum_{i}^{n} \max\left(0, 1 - y_i(w \cdot x_i - b)\right)\right] + \lambda\|w\|.$$

To find solutions for the soft-margin problem, we solve the Lagrange dual

$$\text{maximise} f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} y_i c_i (x_i \cdot x_j) y_j c_j$$

subject to $\sum_{i=1}^{n} c_i y_i = 0$ and $0 \leq c_i \leq \frac{1}{2n\lambda}$ for all i.

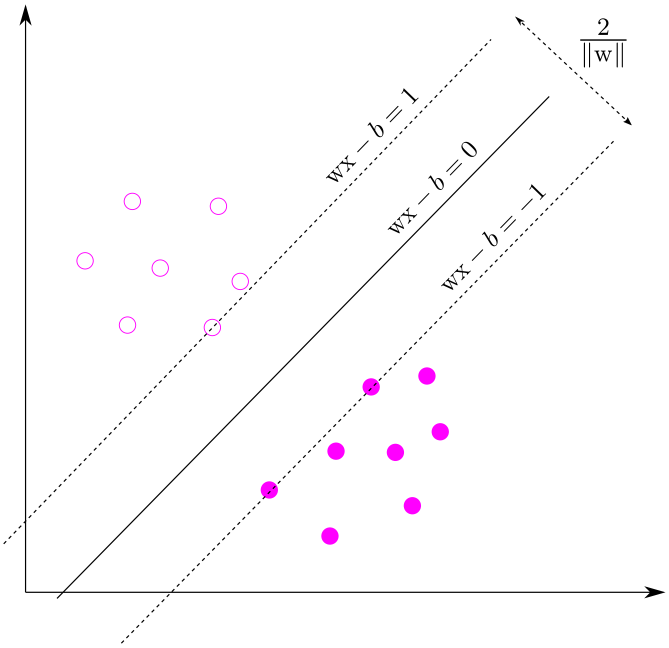


Figure 2.1: Selected hyperplanes and separator hyperplane that separates data points from two classes with maximum margin.

For non-linear classification, we solve the optimisation problem:

$$\text{maximise} f(c_1, \ldots, c_n) = \sum_{i=1}^{n} c_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i c_i k(x_i, x_j) y_j c_j$$

subject to $\sum_{i=1}^{n} c_i y_i = 0$ and $0 \leq c_i \leq \frac{1}{2n\lambda}$ for all i, where $k(x_i, x_j)$ is called the kernel function. In the case of linear SVM, $k(x_i, x_j)$ is the dot product of x_i and x_j.

Compared to Naive Bayes and MaxEnts, SVMs are robust when working with high dimensional data. As SVMs are trained by selecting support vectors (data points on marginal planes) that are most further apart, they are independent of the dimensionality of the feature space. Text classification is described as an ideal choice for SVMs by Thorsten Joachims (1998) as text data are highly dimensional and extremely sparse.

2.1.4 Neural Networks

Naive Bayes, Maximum Entropy Classifier or SVM are considered shallow architectures as they typically apply only one or two (non-linear SVM) transformations on top of their inputs. This limits their ability to capture complex hidden relationships amongst their inputs. Different to the above architectures, Neural Networks are typically stacked to form architectures with multiple layers (hence *deep*) where each layer is expected to capture relationships of inputs from the previous layer. Noticeably we can also stack a Maximum Entropy Classifier (equivalent to Single-layer perceptron or Softmax layer) or an SVM on top a Neural Network to form a deep architecture (Tang 2013).

In simple terms, a Neural Network is a collection of connected units. Each connection is analogous to a synapse and each unit is analogous to a neuron. A synapse carries a weight that multiplies its inputs and the connected neuron transforms the multiplied inputs to produce a correspondent output. We now discuss various popular types of Neural Networks from related literature.

2.1.4.1 Feedforward Neural Network

Feedforward Neural Networks are Neural Networks whose connections do not form a cycle. These are distinguished from Recurrent Neural Networks or Restricted Boltzmann Machines that are discussed later Sections. The simplest form of Feedforward Neural Network is a Single-layer perceptron (SLP) that has no hidden layer and the inputs are connected directly to the outputs. Incoming values to each output node are summed and optionally transformed. These SLPs can be stacked to form a Multi-layer Peceptron (MLP) (Figure 2.2) that has greater power of expression as the inputs go

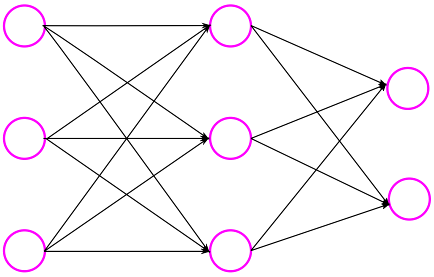


Figure 2.2: A MLP with 3 inputs, 2 outputs and a hidden layer with 3 units.

through many layers of transformation. MLPs can be trained by Stochastic Gradient Descent with back-propagation (Rumelhart et al. 1986).

2.1.4.2 Convolutional Neural Network

A Convolutional Neural Network (CNN) is a biologically inspired variant of Multi-Layer Perceptron (MLP) (Hubel and Wiesel 1968; Fukushima 1980). CNNs exploit spatial proximity by learning interaction between neurons of adjacent layers. Weights contain sub-regions, called *receptive-fields* or *filters*. The sub-regions are tiled to cover the entire visual field (Figure 2.3). Output from applying each filter to the input forms a *feature map*. If we denote x as the input, the k-th feature map at a given layer as h^k, whose filters are determined by the weights W^k and bias b_k, then the feature map h^k can be obtained as follow:

$$h^k_{ij} = \tanh\left((W^k * x)_{ij} + b_k\right)$$

where $*$ denote convolution. Similarly to Feed-forward Neural Networks, Convolutional Neural Networks can also be trained using standard back-propagation (LeCun 1989).

LeCun et al. (1998) introduce a *sub-sampling* layer stacked on each convolutional layer in LeNet-5 network (Figure 2.4). The *sub-sampling* layers perform local averaging (another convolution). The purpose of sub-sampling layers is to reduce overfitting.

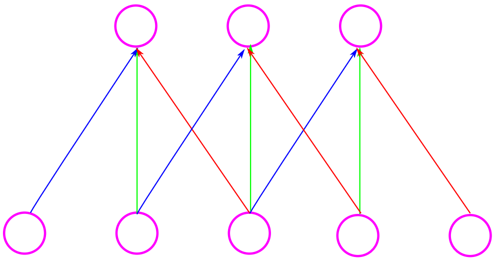


Figure 2.3: An example of the connection of 3 hidden units within the same receptive field. Weights of the same colours are shared - being identical.

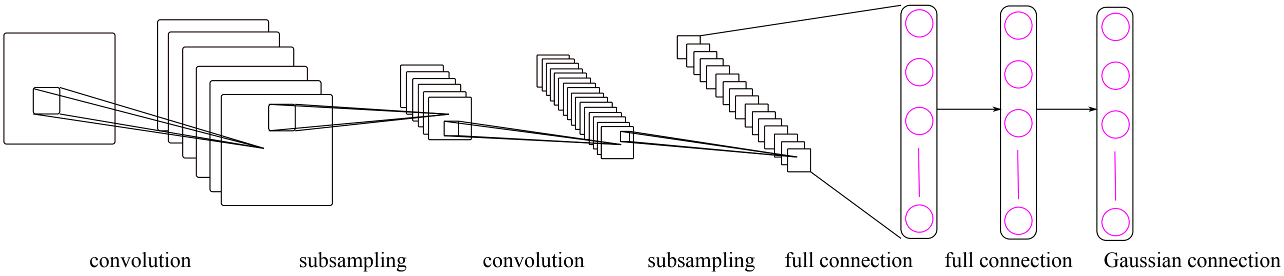


Figure 2.4: LeNet-5, a Convolutional Neural Network for digit recognition.

The outputs of the deep stacked CNNs are normally connected to an MLP (LeNet-5) either for regression or classification.

2.1.4.3 Recurrent Neural Network

Contrary to Feedforward Neural Network, Recurrent Neural Network (RNN) is a class of Neural Networks whose connections between neurons form directed cycles. This type of architectures enables the networks to model data with sequential or temporal nature. These can also have modules that behaves like memories which store signals from previous time steps and feed these signals back in the current time step.

There are several common types of RNNs which are fully-connected RNN, Elman networks (Elman 1990), Hopfield networks (Hopfield 1982) and Jordan networks (Jordan 1989). The simplest RNN is fully-connected RNN whose recurrent layer has nodes that fully connect to themselves. The standard method for training a RNN is Gradient Descend with Back-Propagation Through Time (BPTT) (Rumelhart et al. 1986). In BPTT, the network is unfolded in a number of time steps so it becomes a feedforward network. The gradient is computed in a similar manner to vanilla back-propagation for each time step, however, RNNs when trained with BPTT suffer from gradient vanishing and gradient exploding problems. These are caused by the error gradient while being back-propagated grows or shrinks exponentially depending on the size of their weights (Hochreiter 1991). Noticeably these problems also apply to deep MLPs where gradient back-propagated to the first layers reduces or increases exponentially. To resolve these issues, several different types of neurons with modified inner connections are introduced such as Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997), Gated Recurrent Unit (GRU) (Cho et al. 2014), Residual Networks (He et al. 2015) or Highway Networks (Srivastava et al. 2015).

We now discuss the popular Long Short Term Memory (LSTM) networks that address these issues of RNNs which can back-propagate error gradient efficiently in excess of 1000 steps of time intervals (Hochreiter and Schmidhuber 1997).

2.1.4.4 LSTM

Hochreiter and Schmidhuber (1997) introduce Long Short Term Memory (LSTM) networks whose hidden neurons are composed of gated units. The idea of gated units is for the network to be able to control the flow of information that contributes to the final output. With these gated units, it is possible that only a few inputs participate into the network output, thus, error gradient can be back-propagated through very few nodes.

There are many variants of LSTMs. We now discuss the vanilla architecture that incorporates changes from Gers et al. (2000) and Gers and Schmidhuber (2000) into Hochreiter and Schmidhuber (1997). In detail, each vanilla LSTM (Figure 2.5) module at time t contains a central value acting as memory c_t. There are also input gate i_t, output gate o_t and forget gate f_t. These gates are results of combinations of c_t, input x_t and output h_{t-1}. A new value c_t is the result of gated input and gated c_{t-1}. Output of the module is c_t gated by output gate o_t. The mathematical equations of these units are detailed below:

$$\begin{aligned}
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
o_t &= \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \\
c_t &= f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \\
h_t &= o_t \circ \sigma_h(c_t)
\end{aligned}$$

where:

$x_t \in \mathbb{R}^d$	is input vector to the LSTM unit.
$f_t \in \mathbb{R}^h$	is the forget gate's activation vector.
$i_t \in \mathbb{R}^h$	is the input gate's activation vector.
$o_t \in \mathbb{R}^h$	is the output gate's activation vector.
$h_t \in \mathbb{R}^h$	is the output vector of the LSTM unit.
$c_t \in \mathbb{R}^h$	is the cell state vector.
$W \in \mathbb{R}^{h \times d}, U \in \mathbb{R}^{h \times h}$ and $b \in \mathbb{R}^h$	weight matrices and bias vector parameters.
σ_g	sigmoid function.
σ_c	hyperbolic tangent function.
σ_h	can be sigmoid, hyperbolic tangent or identity function.

Weight matrices and bias vectors can be trained using SGD with BPTT.

LSTM nowadays are critical components in state-of-the-art NLP systems such as speech recognition (Chiu et al. 2018), machine translation (Wu et al. 2016) and language modelling (Shen et al. 2018).

2.1.4.5 Restricted Boltzmann Machine

Next we examine a family of generative Neural Network architectures that learn joint probabilities of inputs and their hidden units from unlabelled data. Specifically we are

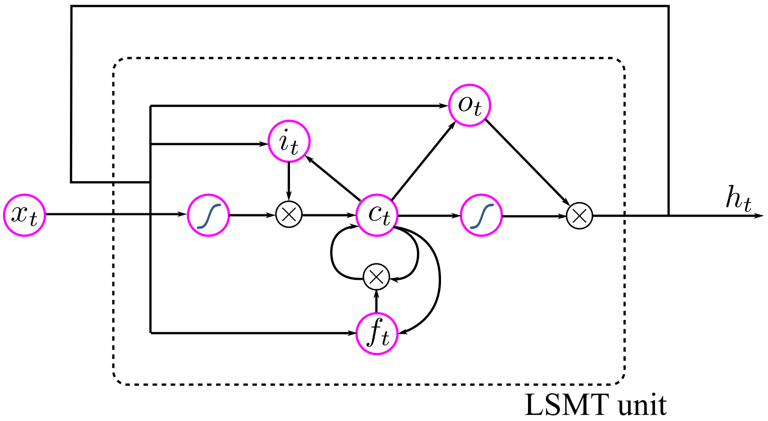


Figure 2.5: Long Short Term Memory network

interested in the property where these models can infer the conditional probability of hidden values given visible values. Once the conditional probability is known, we can sample hidden values given inputs, thus, project the original inputs into a smaller hyper-space. The projected values can be fed into a classifier with less number of parameters than if being fed with the original inputs. The reduction in number of parameters helps avoid overfitting where labelled data is scarce.

An RBM is an undirected bipartite network with a set of hidden units h, a set of visible units v, and symmetric connection weights between these two layers represented by a weight matrix W (Figure 2.6).

With an energy function $E(v, h)$, the generative probability of the network is given by

$$P(v, h) = \frac{1}{Z} \exp(-E(v, h)), \tag{2.5}$$

where Z is the partition function that ensures the sum of $P(v, h)$ over all possible v‘and h is one.

The hidden units are binary-valued, and the visible units are binary-valued or real-valued. For binary visible units, the energy function is given by

$$E(v,h) = -\sum_{i\in visible} b_i v_i - \sum_{j\in hidden} c_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{2.6}$$

where b_i are visible unit biases, c_j are hidden unit biases.

In many applications, inputs are normally real-valued and noisy. In this cases, the energy function is given by

$$E(v,h) = \sum_{i\in visible} \frac{(v_i - b_i)^2}{2\sigma_i^2} - \sum_{j\in hidden} c_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij}, \tag{2.7}$$

where σ_i is the standard deviation of the assumed noise level for unit i .

In real applications, it is easier to first normalise each component of the data to have zero mean and unit standard deviation and then use contrastive divergence (Hinton 2002) with zero noise to learn the RBM parameters.

With the proposal of Contrastive Divergence (Hinton 2002), a fast-learning algorithm, RBM has been successfully applied in a wide range of applications. To explain Contrastive Divergence, we assume that we have an input X, and model parameters Θ. The likelihood is given by

$$p(X;\Theta) = \prod_{k=1}^{K} \frac{1}{Z(\Theta)} f(x_k;\Theta), \tag{2.8}$$

where $Z(\Theta)$ is defined as the partition function

$$Z(\Theta) = \int f(x;\Theta)dx, \tag{2.9}$$

and K is the number of observations. Maximising this likelihood is equivalent to minimising the negative log of $p(X;\Theta)$, denoted $E(X;\Theta)$, where

$$E(X;\Theta) = \log Z(\Theta) - \frac{1}{K}\sum_{k=1}^{K} \log f(x_k;\Theta). \tag{2.10}$$

One of the ways to find a local minimum is to use gradient descent with line search. We write down the partial derivative of Equation 2.10:

$$\frac{\partial E(X;\Theta)}{\partial\Theta} = \frac{\partial \log Z(\Theta)}{\partial\Theta} - \frac{1}{K}\sum_{i=1}^{K}\frac{\partial \log f(x_i;\Theta)}{\partial\Theta} \tag{2.11}$$

$$= \frac{\partial \log Z(\Theta)}{\partial\Theta} - \left\langle \frac{\partial \log f(x;\Theta)}{\partial\Theta} \right\rangle_X \tag{2.12}$$

The right term can be easily computed from the given data, the left term can be expanded as:

$$\frac{\partial \log Z(\Theta)}{\partial\Theta} = \frac{1}{Z(\Theta)}\frac{\partial Z(\Theta)}{\partial\Theta} \tag{2.13}$$

$$= \frac{1}{Z(\Theta)}\frac{\partial}{\partial\Theta}\int f(x;\Theta)dx \tag{2.14}$$

$$= \frac{1}{Z(\Theta)}\int \frac{\partial f(x;\Theta)}{\partial\Theta}dx \tag{2.15}$$

$$= \frac{1}{Z(\Theta)}\int f(x;\Theta)\frac{\partial \log f(x;\Theta)}{\partial\Theta}dx \tag{2.16}$$

$$= \int p(x;\Theta)\frac{\partial \log f(x;\Theta)}{\partial\Theta}dx \tag{2.17}$$

$$= \left\langle \frac{\partial \log f(x;\Theta)}{\partial\Theta} \right\rangle_{p(x;\Theta)} \tag{2.18}$$

From here we can see that the left term can be approximated by drawing samples from the proposed distribution $p(x;\Theta)$. Hinton has found that empirically even one-step Gibbs sampling works well.

Larochelle and Bengio (2008) and Salakhutdinov and Hinton (2009) show that discriminative fine-tuned Deep Boltzmann Machine can perform well in handwritten digit and 3D object recognition tasks. However, these settings require input objects to be re-scaled to a fixed size that can cause information loss. It also requires a significant large number of parameters for full connections between visible layer and hidden layer.

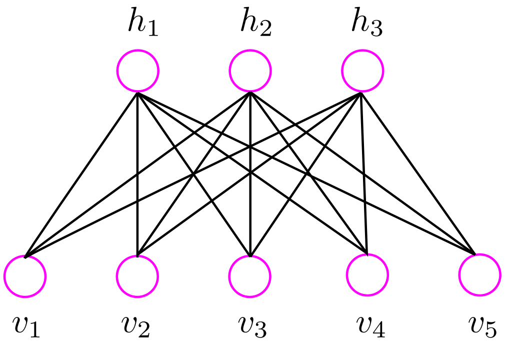


Figure 2.6: Restricted Boltzmann Machine with 5 visible units and 3 hidden units

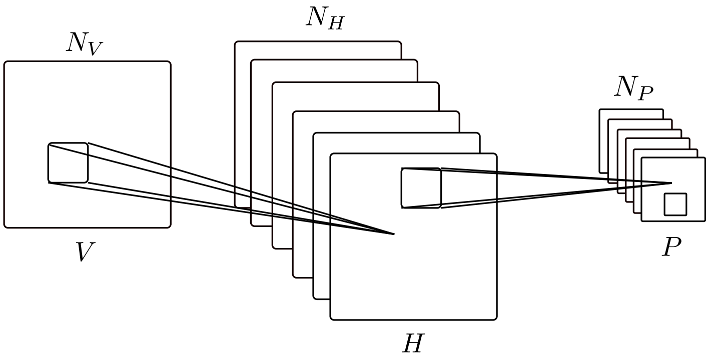


Figure 2.7: Convolutional Restricted Boltzmann Machine.

2.1.4.6 Convolutional Restricted Boltzmann Machine

A Convolutional Restricted Boltzmann Machine (CRBM) (Lee, Grosse, Ranganath, and Ng 2009) is similar to a normal RBM but weights between visible units and hidden units are shared among all locations in the hidden layer (Figure 2.7). This property makes CRBM a good fit for sequential inputs like sentences or documents. Only a fixed number of weights are needed regardless of its input length.

In a two-dimensional setting, the input layer is an array with dimension of $N_V \cdot N_V$[3]. The hidden layer consists of K groups with each group an $N_H \cdot N_H$ array of binary units. Each of the K groups is associated with N_W-dimensional filter weights, which are shared across all the hidden units within the group. This results in N_W being set

[3] For notation convenience, we assume a squared matrix. Note that there is no requirement that the inputs must be equal-sized or even two-dimensional in CRBM.

as $N_V - N_H + 1$. Denoting b_k as a shared bias for each group, c a shared bias for the visible units, the energy function of the network is defined by

$$E(v,h) = -\sum_{k=1}^{K}\sum_{j=1}^{N_H}\sum_{r=1}^{N_W} h_j^k W_r^k v_{j+r-1} - \sum_{k=1}^{K} b_k \sum_{j=1}^{N_H} h_j^k - c\sum_{i=1}^{N_V} v_i. \tag{2.19}$$

As there are K *groups* in the hidden layer, it is easy for the model to learn trivial features and be overfitted. Lee, Ekanadham, and Ng (2008) suggest adding a regularisation term that penalise a deviation of the expected activation of the hidden units from a fixed level p. Hence, the optimisation problem for a training set $(v^{(1)}, \ldots, v^{(m)})$ is defined as

$$\text{minimize}_{\{w_{ij},c_i,b_j\}} - \sum_{l=1}^{m} P(v^{(l)}, h^{(l)}) + \lambda \sum_{j=1}^{n} \left| p - \frac{1}{m}\sum_{l=1}^{m} E[h_j^{(l)}|v^{(l)}] \right|^2,$$

where p is the target sparsity, a constant controlling the sparseness of the hidden units, λ is a regularisation constant, n is the number of hidden units. The optimisation can be done with Gradient Descent where the gradient of the left term can computed by Contrastive Divergence (Lee et al. 2009) and computing gradient of the right term is straight forward.

2.1.5 Denoising Autoenconders

A more generic framework to learn representations from unlabelled data is autoencoders (Rumelhart, Hinton, and Williams 1986; Elman and Zipser 1988). An autoencoder has an *encoder* and a *decoder*. The *encoder* transforms an input vector v to a hidden representation h. The *decoder* transforms the hidden representation h into v' in the input space so that $P(V|Z = v')$ generate V with high probability. The training goal is to maximise the likelihood $P(V|Z = v')$.

As one can easily learn an autoencoder simply with identity mapping and retain complete information of its input, autoencoders are usually applied with constraints.

One of the popular constraints is to use hidden representations in lower dimensional space than the input space. In this case, if only linear mappings and squared error are used, the autoencoder is essentially equivalent PCA (Baldi and Hornik 1989). Another popular constraint is to use sparse coding where the autoencoders are trained to limit the number of non-zero units in the representations (Ranzato et al. 2007; Ranzato et al. 2008). Vincent et al. (2010) propose a denoising criterion for the decoder to reconstruct the original input from a corrupted input. Vincent et al. (2010) show that stacked denoising autoencoders have reconstruction and classification performance that is better than stacked ordinary autoencoders and equivalent to stacked RBMs.

Given that RBMs have better performance than ordinary autoencoders and similar performance to denoising autoencoders, we decide to explore usage of RBMs for learning language representations in Chapter 4.

2.2 Word representations

As the document universe is large, most solutions to text classification break text documents into smaller separate tokens (tokenisation). These tokens, usually words, are used as inputs into the classification models instead of the whole documents. We now review different methods to represent these input units.

2.2.1 n-grams

If each token is treated as a separate dimension in the token hyperspace, we call these models *uni*-grams. If each n consecutive tokens are treated as a separate dimension, we have n-gram models. Although n-gram models are simple and usually efficient, representing each unique n-gram as a separate dimension in the hyperspace has multiple problems. One obvious problem is that words with similar or related meanings are represented as totally unrelated tokens. The shared properties are completely not embedded in this representation. Besides this type of representations requires words in testing data to appear in training data to be used for prediction. Another problem with n-grams models is the vocabulary grows prohibitively in size when n increases. This leads to the need of huge corpora cover this size of the vocabulary in order to avoid overfitting. Even though this problem can be somewhat relived by smoothing methods like Laplace smoothing (Manning et al. 2008), Good-Turing smoothing (Gale and Sampson 1995) or Kneser Ney smoothing (Ney et al. 1994), these smoothed values are purely estimations. To overcome these problems, several alternative representations have been studied.

2.2.2 Co-occurence matrices

Since Harris (1954) introduces the distributional hypothesis, in which words sharing similar contexts imply similar meaning, different methods have been proposed to learn word representations from their contexts. One of the dominant approaches to represent contexts is to use co-coccurence matrices. An introduction to popular co-occurence matrices are presented in this Section and how they are used to learn word representations is discussed in the next Section.

One of the most popular co-occurence matrices in information retrieval is term-document matrices (Deerwester et al. 1990). A term-document matrix has unique terms as rows, documents as columns and number of times term i appearing in document j as the value of cell (i, j). These frequency values are often weighted by tf-idf where the cell values are multiplied by inverse of their numbers of appearances.

Church and Hanks (1989) suggest to use Pointwise Mutual Information (PMI) as an indication of word association. Given a text corpus, A PMI score of two words x

and y is defined as

$$I(x, y) = \log_2 \frac{P(x, y)}{P(x)P(y)}$$

where $P(x, y)$ is the probability when x and y occurs together in the corpus. The PMI metric positively correlates with this frequency while negatively correlating with how often x and y occur independently. When there is a genuine association between x and y, one might expect $P(x, y)$ to be larger than $P(x)P(y)$. This leads to $I(x, y) > 0$. While there is no association between x and y, we can expect $P(x, y) \approx P(x)P(y)$, consequently $I(x, y) \approx 0$. $P(x)$ and $P(y)$ are estimated by the ratio of number of times the word x and y appear in a corpus to the size of the corpus. $P(x, y)$ is the ratio of number of times x is followed by y in a window of w words to the size of the corpus.

Lund and Burgess (1996) construct term-term co-occurrence matrices based on rolling contexts. By moving a window of 10 words over a source corpus, co-occurrence counts of terms are collected. Each co-occurrence count is inversely scaled by the distance (in words) between two words in the window. This finally forms a matrix of co-occurrences of every single word pair.

Bullinaria and Levy (2007) examine different co-occurrence statistics with different distance metrics when evaluating word semantic and syntactic similarity. The examined statistics include PMI, Positive PMI, conditional probability $p(c|t)$ and probability ratio $\frac{P(c|t)}{P(c)}$ where t is a target word and c is a contextual word that appears within a fixed number of words around t. Examined distance metrics include:

$$\begin{aligned}
\textbf{Euclidean} \quad & d(t_1, t_2) = \Sigma_c |p(c|t_1) - p(c|t_2)|^2 \\
\textbf{City block} \quad & d(t_1, t_2) = \Sigma_c |p(c|t_1) - p(c|t_2)| \\
\textbf{Cosine} \quad & d(t_1, t_2) = 1 - \frac{\Sigma_c p(c|t_1) p(c|t_2)}{\left(\Sigma_c p(c|t_1)^2\right)^{\frac{1}{2}} \left(\Sigma_c p(c|t_2)^2\right)^{\frac{1}{2}}} \\
\textbf{Hellinger} \quad & d(t_1, t_2) = \Sigma_c \left(p(c|t_1)^{\frac{1}{2}} - p(c|t_2)^{\frac{1}{2}}\right)^2 \\
\textbf{Bhattacharya} \quad & d(t_1, t_2) = -\log \Sigma_c \left(p(c|t_1)^{\frac{1}{2}} p(c|t_2)^{\frac{1}{2}}\right) \\
\textbf{Kullback-Leibler} \quad & d(t_1, t_2) = \Sigma_c p(c|t_1) \log \left(\tfrac{p(c|t_1)}{p(c|t_2)}\right).
\end{aligned}$$

Bullinaria and Levy (2007) found that Positive PMI within short context window (less than 10 words) with cosine works best for word semantic similarity benchmarks while probability ratio with cosine has best performance in a syntactic benchmark. They also observe that closed class words, low frequency words, corpus size and quality add useful information for most tasks while dimensionality reduction may not be necessary for their performances.

2.2.3 Matrix factorisations

Deerwester et al. (1990) introduce Latent Semantic Analysis (LSA/LSI) that learns document and word latent representations by factorising a term-document matrix into lower rank matrices using Singular Value Decomposition (SVD). First a term-document co-occurrence matrix is built, often weighted by tf-idf due to its effectiveness. Let's call this matrix C, which is decomposed into $U\Sigma V^\top$. In a term-document matrix with m terms and n documents, U is a $m \times m$ unitary matrix, Σ is an $m \times n$ rectangular diagonal matrix and V is an $n \times n$ unitary matrix. In practice, as m and n are usually very large (m can be hundreds of thousands and n can be millions), C is usually approximated as a lower rank matrix $\tilde{C}$ based on minimising Frobenius distance between C and $\tilde{C}$. In this case, $\tilde{C} = U_k \Sigma_k V_k^\top$ where k is the chosen lower rank (typically hundreds), U_k is a $m \times k$ *term matrix*, Σ_k is a $k \times k$ rectangular diagonal matrix with the k largest eigen values, $V_k^\top$ is $k \times n$ *document matrix*. Once the co-occurence matrix is factorised, two documents i and j can be compared by comparing two vectors $\Sigma(V_i)^\top$ and $\Sigma(V_j)^\top$. Two terms i and j can be compared by comparing vectors $\Sigma(U_i)^\top$ and $\Sigma(U_j)^\top$.

Lund and Burgess (1996) analyse principle components of the term-term co-occurrence matrix (Hyperspace Analogue to Language - HAL) and find that word representations in the co-occurrence space share semantic and associative relationships. However HAL models are biased to words with high frequencies even though many of them have relative little information (i.e. stop words). From this observation, Rohde et al. (2006) introduce different normalisation strategies (COALS) to factor out this effect such as ignoring left/right distinction, reducing the window size from 10 to 4 words, discarding columns based on word frequency instead of word variance, replacing co-occurrence by correlation and replacing negative correlations by zeros. They find that the word representations achieved from COALS-SVD outperform LSA and HAL in word similarity tasks.

Blitzer et al. (2004) experiment with linear (Principal Componenent Analysis - PCA) and non-linear (Semidefinite Embedding - SDE) dimensionality reduction to learn bi-gram representations from bi-gram co-occurrence matrix. However this approach is not scalable as it is not practical to store a bi-gram co-occurrence matrix for all bi-gram pairs from the vocabulary.

Lebret and Collobert (2014) observe that when co-occurrence statistics are discrete probabilities, Frobenius distance is not suitable for co-occurrence matrix reconstruction. They propose to use Hellinger distance

$$H^2(P,Q) = \frac{1}{2}\sum_i(\sqrt{P_i} + \sqrt{Q_i})^2$$

where P and Q are discrete probability distributions while reconstructing co-occurrence matrices by PCA (hence HPCA). To use Hellinger distance, co-occurrence statistics have to be discrete probability measures. Specifically their co-occurrence statistic of a word w and a context T is computed as

$$p(w|T) = \frac{p(w,T)}{p(T)} = \frac{n(w,T)}{\sum_w n(w,T)}$$

where $n(w,T)$ is the number of time w and T appear together. The authors show that word representations built by Hellinger-distance PCA outperform ones built by Euclidean-distance SVD in Named Entity Recognition and Movie Review tasks.

Slightly different to Lebret and Collobert (2014), Pennington et al. (2014) instead use $\log p(w|T)$ as their co-occurrence statistic. The motivation behind this idea is the log odd ratio $log\big(\frac{p(i|k)}{p(j|k)}\big)$, that represents relative relationship of word i and word j to a context k, is conveniently computed by $(v_i - v_j)^\top \tilde{v}_k$ where v_i, v_j are vector representations of words i, j and $\tilde{v}_k$ are vector representations of context k. In a word-word co-occurrence matrix, bias terms are introduced to keep the symmetry so that

$$v_i^\top \tilde{v}_k + b_i + \tilde{b}_k =_{ik}$$

where X_{ik} is the number of times word i appearing in context k. Variables v, $\tilde{v}$, b and $\tilde{b}$ can be found by solving weighted least square regression with the loss function shown in Equation 2.20. The weight function $f(X_{ij})$ gives more weight to popular co-occurring pairs and less weight to unpopular ones. In their experiments, the authors found $x_{\max} = 100$ and $\alpha = 3/4$ work best empirically.

$$J = \sum_{i,j=1}^{V} f(X_{ij})(w_i^\top \tilde{w}_j + b_i + \tilde{b}_j - \log X_{ij})^2 \tag{2.20}$$

$$f(x) = \begin{cases} (x/x_{\max})^\alpha & \text{if } x < x_{\max} \\ 1 & \text{otherwise} \end{cases} \tag{2.21}$$

It is found that their word representations (Glove) have better performance when compared to HPCA and SVD in both word similarity and word analogy tasks while trained with corpora with similar sizes.

2.2.4 Neural word representations

A family of more sophisticated methods to learn word representations are ones that utilise Neural Networks. Compared to matrix factorisation methods, non-linear Neural Networks can capture more complex relationship between their inputs as they are non-linearly transformed at different layers of the networks. Meanwhile recent advancements have shown that Deep Neural Networks can still be trained efficiently with variants of Stochastic Gradient Descent (Duchi et al. 2011; Zeiler 2012; Kingma and Ba 2014). Another advantage of Neural Networks is its architecture flexibility such as convolutional and recurrent connections models very well sequential lingual data such as sentences or documents.

Hinton (1986) study word and relation representations from structural data using feedforward neural networks with back-propagation. Their data set contains a set of agent-relation-patient triplets. To learn the representations, the authors build a network to predict the patient from the agent-relation pair for each triplet. Their networks contain an input layer that is a pair of one-hot-encoding vectors of input agent and relation. A one-hot-encoding vector is a vector whose only one dimension has a value of 1 and the rest have values of 0s. A categorical value can be represented by a one-hot-encoding vector whose number of dimension is number of categories and the dimension which represents the specified category has value of 1. Each one-hot-encoding vector is connected to a 6-unit hidden layer. These 12 hidden units are connected to a 12-unit second hidden layer. This layer is then fully connected to a 6-unit hidden layer. The penultimate layer is connected to an output layer for decoding the patient in the triplet. The entire network is trained with back-propagation and gradient descent. The final representations are achieved by the values of the hidden units activated directly from the one-hot-encoding inputs.

Even though successfully demonstrating the ability of neural networks to learn latent representations of different agents and relations, Hinton (1986)'s approach is restricted to the availability of this type of relational data sets. Bengio et al. (2003) push the idea to a large scale that focuses on language modelling and leverages the redundancy of text data. Given a set of training samples $w_1 \dots w_T$ where $w_t \in V$ (V is a vocabulary), they aim to learn $f(w_t, \dots, w_{t-n+1})$ that mimics the empirical probability $\hat{P}(w_t|w_1 \dots w_{t-1})$. They define a distributed representations matrix C of size $|V| \times m$ that maps any element i of V to a real vector $C(i) \in \mathbb{R}^m$. The function f then can be computed from

$$f(i, w_{t-1}, \dots, w_{t-n+1}) = g\big(i, C(w_{t-1}), \dots, C(w_{t-n+1})\big),$$

where g is *softmax* function

$$g\big(i, C(w_{t-1}), \dots, C(w_{t-n+1})\big) = \frac{e^{y_i}}{\sum_w e^{y_w}}.$$

The training is obtained by maximising the log-likelihood

$$\mathcal{L} = \frac{1}{T} \sum_{t=1}^{N} \log f(w_t, w_{t-1}, \dots, w_{t-n+1}; \theta) + \mathcal{R}(\theta),$$

where $\mathcal{R}(\theta)$ is the regularisation term and N is the number of words in the corpus.

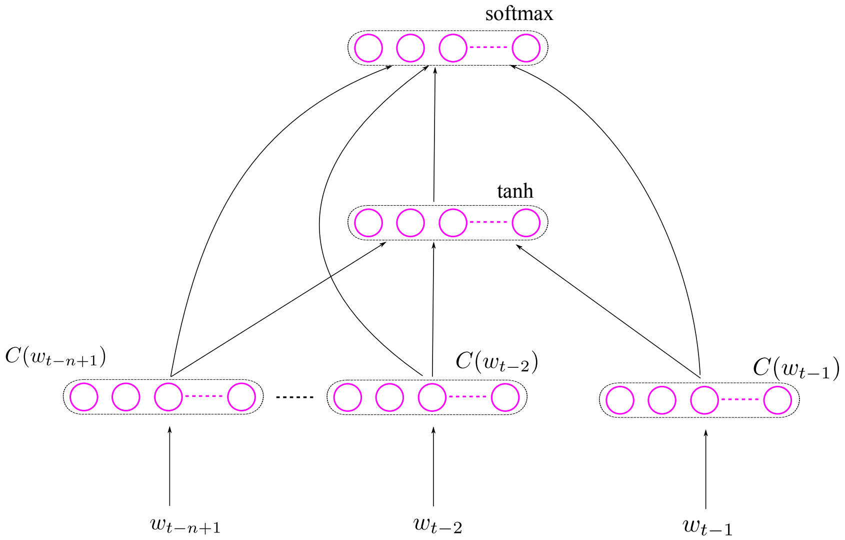


Figure 2.8: Neural probabilistic language model (NPLM)

The overall architecture of their network is described in Figure 2.8. The non-normalised log-probabilities y_i for each output word i is computed as

$$y = b + Wv + U\tanh(d + Hv),$$

where v is the concatenation of $C(w_{t-1}), \dots, aC(w_{t-n+1})$, H, U, W are the model's parameter matrices and b, d are biases.

The network is trained using parallelised asynchronous Stochastic Gradient Descent to train the parameter set $\theta = (b, d, W, U, H, C)$ for maximising regularised log-likelihood. Details of the parallel training can be found from the original paper. Bengio et al. (2003)'s experimental results show that significantly better perplexity[4] can be obtained using neural networks, in comparison with the best of the n-grams models: interpolated or smoothed tri-gram models (Jelinek and Mercer 1980), back-off n-gram models with the *Modified Kneser-Ney* algorithm (Kneser and Ney 1995;

[4] Geometric mean of inverse probability of each word given previous context. Perplexity is a measure for how well a distribution can be predicted.

Chen and Goodman 1996) and other class-based n-gram models (Brown 1990; Kneser and Ney 1993; Niesler et al. 1998). Morin and Bengio (2005) improve the NPLM by using hierarchical softmax instead of traditional softmax in the output layer. This technique reduces the computational cost for computing normalised probability of the predicted word from $O(|V|)$ to $O(\log |V|)$.

Mnih and Hinton (2007) propose two factored Restricted Boltzmann Machines (one temporal, one is non-temporal) and a log-bilinear language model that uses word representations to reduce one-hot-encoding inputs into a smaller space and boost training speed. RBMs are trained using Constrastive Divergence (Hinton and Salakhutdinov 2006) and the log-bilinear model is trained using stochastic gradient descent. The authors do not provide qualitative analysis on their word presentations but report an improvement on their language model over NPLM for a mixture of bi-linear and n-gram model. Also by using RBM, Dahl et al. (2012) use Metropolis-Hasting for training n-gram RBM with energy function

$$E(v,h) = -c^\top - \sum_{i=1}^{n} b^{*\top} v^{(i)} - h^\top U^{(i)} D v^{(i)}, \tag{2.22}$$

where D is the word representations matrix, v is the visible layer, h is the hidden layer, U is connection weights between visible and hidden layers. b, c are bias factors. The model utilises a word representation matrix to project the input to smaller dimension space. The trained word representations combined with bag-of-word features yields state-of-the-art sentiment classification when trained by linear SVM compared to previous models.

Collobert and Weston (2008) propose a Deep Neural Network architecture (Figure 2.9) for multi-task learning including part-of-speech tagging, semantic role labelling, language modelling, text chunking and named entity recognition. The authors use look-up tables for words and other features to project them into dense spaces. The lookup table for words is particularly similar to word representation matrix as a step to convert one-hot-encoding input into lower-dimension values. These projected values are inputs to a convolutional layer, then a Softmax layer for classification.

This look-up table is learned during the supervised and semi-supervised tasks. They do not provide an analysis on the quality of these word representations but show that both multi-task learning and semi-supervised learning improve the generalisation of the shared tasks.

Mikolov et al. (2011) suggest different strategies for training large-scale neural language models. The first strategy is to sort the data in order of their domain relevance. Their training data are split into 560 equally-sized chunks (each containing 40K sentences). A bi-gram model is trained for each chunk. Data chunks are sorted by their perplexity computed from their bi-gram models on the development set. The intuition behind this if less noisy data are processed at the end, they will have higher weights. The authors observe that sorting the data results in significant lower perplexity and quicker converge than SGD. Vocabulary size is reduced by considering only in-domain words (words that appear in only validation and test data) and grouping them into a smaller number of groups. Other strategies including hashing tricks and parallelisation of matrix calculations. Combining all the tricks, the authors could train a neural network model that achieves 10% relative reduction of word error rate on English Broadcast News speech recognition task against 4-gram model trained on 400M tokens.

Huang et al. (2012) use global word representations computed by weighted average of word representations appearing in the document to improve over the NPLM. Their

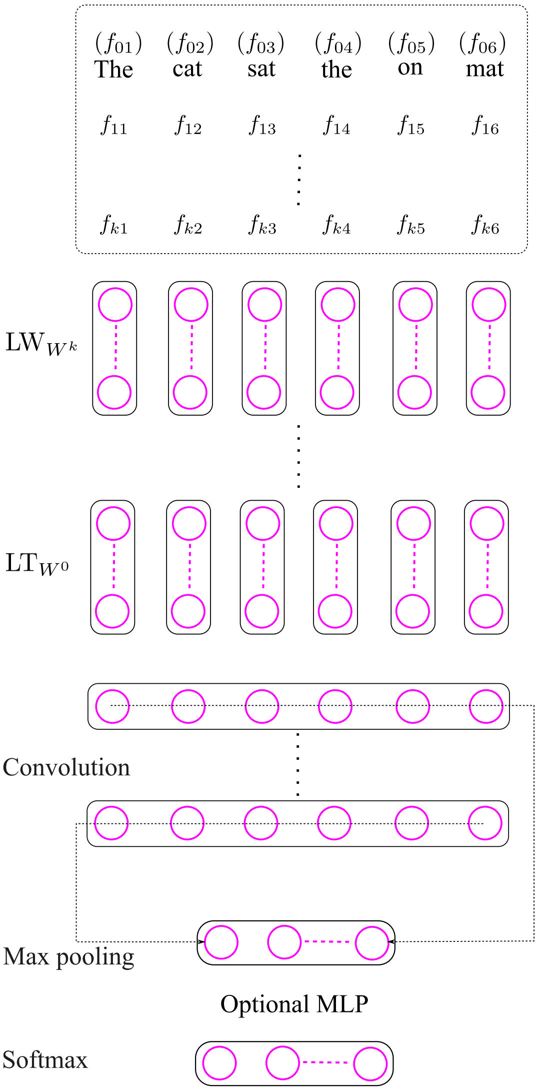


Figure 2.9: A Deep Neural Network with an embedding layer, a convolutional layer and a Softmax layer.

learning process adopts the sampling technique from Collobert and Weston (2008) to improve speed. Using trained single-prototype word presentation, the authors use K-mean clustering to further cluster weighted means of context word for each word into different sense clusters. Weighted average senses are used for word comparison and defined as

$$S = \frac{1}{K^2} \sum_{i=1}^{k} \sum_{j=1}^{k} p(w,i)p(w',j)d(\mu_i(w), \mu_j(w')), \tag{2.23}$$

where $\mu_i(w)$ is the cluster centre i of w, d is the distance function which is usually cosine similarity. More details of word representation evaluation is discussed in the next Section.

Mikolov et al. (2013) propose two new log-linear language models (Figure 2.10) that learn word representations directly without constructing the full language model. Their models which typically contain single hidden layer are much more efficient to train. The continuous bag-of-words (CBOW) model uses context words to predict the centre word while the continuous skip-gram model (Skipgram) use the centre word to predict the context words. These two architectures reduce the complexity for training to $ND + D\log(V)$ and $C(D + D\log(V))$ for each training word per epoch, where N is number of words in the context, D is the dimension of the word presentation, V is the vocabulary size and C is the maximum distance of the words (in number of words). In order to boost training speed, the authors use Hierarchical Softmax (Morin and Bengio 2005) and Asynchronous Gradient Descents (Le et al. 2012) for training. The quality analysis of the trained word representations from these two models suggests that semantic and syntactic analogical questions like "what is the word that is similar to *small* in the same sense as *biggest* is similar to *big*?" can be answered by performing simple algebraic operations with the vector representa-

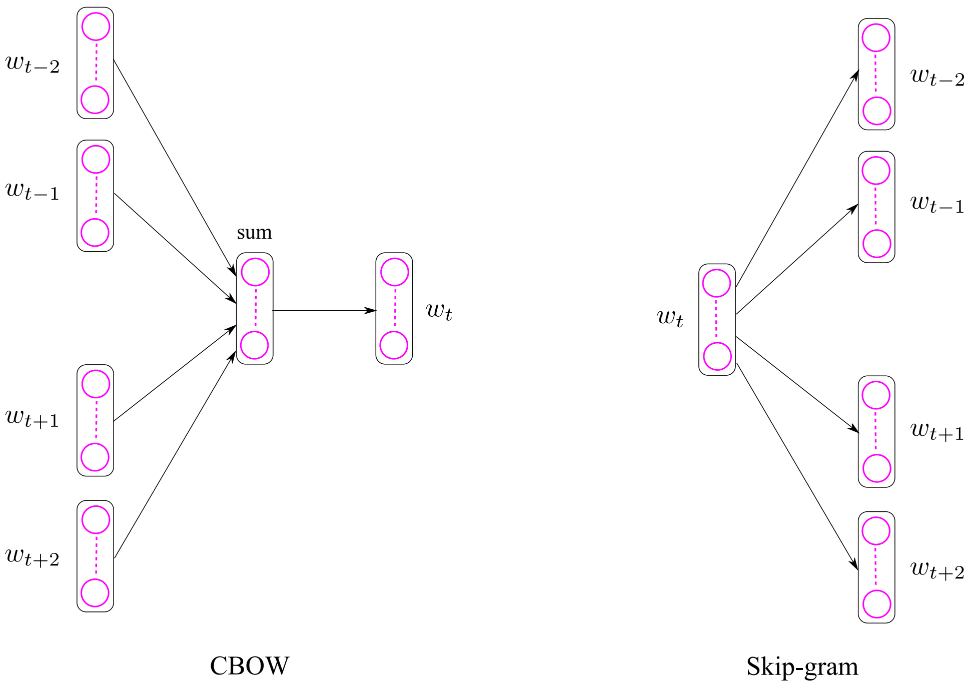


Figure 2.10: CBOW and Skip-gram model (Mikolov et al. 2013)

tions of the words. Skip-gram significantly outperforms representations generated from Neural Language Models and CBOW in semantic analogy task while Skip-gram and CBOW perform similarly in syntactic analogy task. Given both Skip-gram and CBOW can be trained much more efficiently than Neural Language Models, they give superior word representations when trained with bigger corpora and larger vector dimensions. The authors also show that their word representations when used with existing models improve the performances across different NLP tasks like measuring relational similarity, sentiment analysis, paraphrase detection and machine translation.

Mnih and Kavukcuoglu (2013) suggest to use noise-contrastive estimation (Gutmann and Hyvärinen 2010) for speeding up training the log-bilinear model (NCE). Mikolov et al. (2013) further propose a simplified version of noise-constrastive estimation called negative sampling that is even more speed efficient for training their

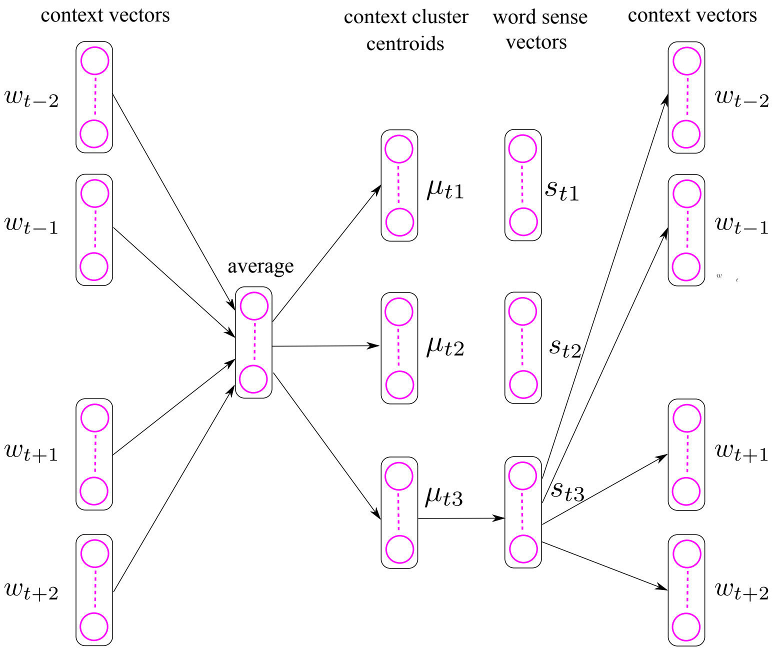


Figure 2.11: Multi-Sense Skip-gram with window size of 2, number of clustered senses of 3.

Skip-gram model. Additionally Mikolov et al. (2013) use sub-sampling to discard frequent words with a probability that is proportional to their frequency. In order to learn representations for phrases and named entities, e.g. "New York Times", "Toronto Maple Leafs", the authors use the PMI scores with a discount factor to detect these phrases and convert them into single tokens.

Readdressing the problem of single-prototype word presentation as in Huang et al. (2012), Neelakantan et al. (2014) further develop the Skip-gram into Multi-Sense Skip-gram (MSSG) model as in Figure 2.11. The overall idea is that, to each cluster of K context clusters, a sense embedding is assigned and used to predict the context words. With MSSG, all words have K number of context clusters. In order to overcome this issue, the authors also use an online K-mean clustering for non-parametric clustering (NP-MSSG). In the evaluation, their models outperform the Skip-gram model in two datasets WordSim-353 (Finkelstein et al. 2002) and SCWS (Huang et al. 2012).

Botha and Blunsom (2014) tackle compositional morphology by breaking words into smaller factors. The word representations can then be computed by adding these factor representations, e.g.:

$$\overrightarrow{\text{imperfection}} = \overrightarrow{\text{im}} + \overrightarrow{\text{perfect}} + \overrightarrow{\text{ion}}$$

These factor representations are then trained using log-bilinear model (Mnih and Hinton 2007) with class-based decomposition (Goodman 2001) for speeding training. They demonstrate that their morphology-guided CSLMs (continuous space language models) improve intrinsic language model performance when compared to baseline CSLMs and n-gram MKN models.

Word representations learned by training Neural Networks show superior quality compared to those constructed with Matrix Factorisation methods Mikolov et al. (2013). However so far, state-of-the-art methods do not to utilise deeper than 2-layer Neural Networks. In fact the next Section shows that some Neural Network-based methods actually implicitly optimise goals that are similar to Matrix Factorisation methods. The advantage of shallow Neural Networks is they are fast to be trained and scalable to corpora with billions of words. It is also not clear whether a more complicated architecture is needed to train better word representations for the quality metrics. To answer these questions, we shall revisit learning word representations with more complicated architectures in Chapter 3.

2.2.5 Implicit Matrix Factorisation

Levy and Goldberg (2014) reveal that the state-of-the-art NCE, Skip-gram Negative Sampling (SGNS) in fact implicitly factorise weighted shifted PMI matrices. The authors then propose alternative representations with shifted Positive PMI

$$\mathrm{SPPMI}_k(w,c) = \max\big(\mathrm{PMI}(w,c) - \log k, 0\big)$$

and their SVD. In SVD, an $m \times n$ matrix M is factorised as $U\Sigma V^\top$ where U is a $m \times m$ unitary matrix with complex numbers, Σ is an $m \times n$ rectangular diagonal matrix with real non-negative numbers and V is also a unitary matrix with complex numbers. Different to previous approaches where word representations are computed as $U\Sigma$ and context representations are computed as V, to achieve symmetry of word and context representations, the authors have word representations as $U\sqrt{\Sigma}$ and context representations as $V\sqrt{\Sigma}$. Their results show that SPPMI improves on two word similarity tasks and one analogy task. Their SVD representations achieve at least as good as SGNS for word similarity but worse in word analogy task.

Levy et al. (2015) further improve the above models by borrowing tricks from Mikolov et al. (2013). The authors found that smoothing context distribution while computing PMI matrix boosts performance across tasks

$$\mathrm{PMI}_\alpha(w,c) = \log\frac{P(w,c)}{P(w)P_\alpha(c)} \text{ where } P_\alpha(c) = \frac{\#(c)^\alpha}{\sum_c \#(c)^\alpha}.$$

For SGNS, using more negative samples results in better performances. Additionally, sometimes combinations of word and context representations can help with substantial gain compared to only word or context representations alone.

2.2.6 Evaluation

Word representations are typically evaluated by comparing word similarities to human annotated scores and finding target words in word analogy task (Mikolov et al. 2013).

In word similarity tasks, similarities are computed by *cosine* similarity of trained representations. Popular word similarity data sets include *WordSim353* (Finkelstein et al. 2002) that contains two sets of English word pairs and human-annotated scores of relatedness judgement. The first set contains 153 word pairs scored by 13 annotators. The second set contains 200 word pairs scored by 16 annotators. The final score is the *Pearson* correlation of computed *cosine* similarities and averaged human annotated scores.

However the *WordSim353* data set does not distinguish similarity and relatedness. In fact, the given instruction to annotators is to evaluate how related two words are. To understand the the difference between similarity and relatedness, we can look at the triple *cloth*, *closet* and *wardrobe*. While *cloth-closet* or *cloth-wardrobe* are highly related and frequently appear in same context, they are completely different things. Meanwhile *closet-wardrobe* might not appear together as often as with *cloth* but they indicate very similar types of containers. From this observation, Agirre et al. (2009) split *WordSim353* into two separate data sets *WordSim353 Similarity* and *WordSim353 Relatedness* to measure similarity and relatedness separately. Another effort to help with evaluating word representations based on similarity comes from the work of Hill et al. (2015). Hill et al. (2015) build *SimLex-999* that contains 999 word pairs sampled from a broad range of concrete and abstract nouns, adjectives and verbs. Each word pair is scored with a similarity rating by approximately 50 subjects.

Other popular data sets for word embedding evaluation include the *MEN* data set (Bruni et al. 2012) that contains 3,000 word pairs that were selected from a balanced range of relatedness levels according to a text-based semantic score and *A. Turk* data set (Radinsky et al. 2011) consists of 280 word pairs with relatedness annotated by Amazon's Mechanical Turk workers. Another data set is made by Luong and Manning

(2013) to focus on evaluating similarity for rare word pairs which are usually ignored in word embedding training method and other data sets.

Word analogy task is first proposed by Mikolov et al. (2013) for word representation evaluation. The idea is as there are many different types of relationships between words, e.g. *big* to *large*, *big* to *biggest*, *big* to *bigger* or *big* to *small*, Mikolov et al. (2013) build a data set (Google) consisting of quadruples of (a, a^*, b, b^*). The trained representations are evaluated by finding the word which is closest (using *cosine* similarity) to $\vec{a^*} - \vec{a} + \vec{b}$. For example, given the quadruple (*big*, *biggest*, *small*, *smallest*), first we find a word x which is closest to $\overrightarrow{biggest} - \overrightarrow{big} + \overrightarrow{small}$ (3CosAdd). The classification is correct if x is exactly *smallest*. The target word can also be found using 3CosMul

$$\arg\max_{b^* \in V} \frac{\cos(\vec{b}^*, \vec{b}) \cos(\vec{b}^*, \vec{a}^*)}{\cos(\vec{b}^*, \vec{a}) + \epsilon}$$

which has an effect of reducing the impact of large quantities and amplifying the impact of small quantities. The data set contains 8,869 semantic and 10,675 syntactic questions. Another similar data set (*MSR*) from Mikolov et al. (2013) focuses entirely on syntactic relationships and contains 8,000 test samples.

The benchmark results are shown in Table 2.1. For similarity and relatedness, there is no single method that outperforms the others in all data sets. While SGNS does better in WordSim Similarity, PPMI does better in WordSim Relatedness. For analogy tasks, SGNS are consistently amongst the best performers. However, when the context window is small, the margin between PPMI and SGNS reduces to be significantly small with 3CosMul metric.

win	Method	WordSim Similarity	WordSim Relatedness	MEN	A. Turk	Rare Words	SimLex	Google Add/Mul	MSR Add/Mul
2	PPMI	.732	**.699**	.744	.654	.457	.382	.552/.677	.306/.535
	SVD	.772	.671	.777	.647	**.508**	.425	.554/.591	.408/.468
	SGNS	**.789**	.675	.773	**.661**	.449	**.433**	.676/**.689**	.617/**.644**
	GloVe	.720	.605	.728	.606	.389	.388	.649/.666	.540/.590
5	PPMI	.732	**.706**	.738	**.668**	.442	.360	.518/.649	.277/.467
	SVD	.764	.679	.776	.639	**.499**	**.416**	.532/.569	.369/.424
	SGNS	**.772**	.690	.772	.663	.454	.403	.692/**.714**	.605/**.645**
	GloVe	.745	.617	.746	.631	.416	.389	.700/.712	.541/.599
10	PPMI	.735	**.701**	.741	.663	.235	.336	.532/.605	.249/.353
	SVD	.766	.681	.770	.628	**.312**	.419	.526/.562	.356/.406
	SGNS	**.794**	.700	**.775**	**.678**	.281	**.422**	.694/.710	.520/**.557**
	GloVe	.746	.643	.754	.616	.266	.375	.702/**.712**	.463/.519

Table 2.1: Performance of each method across different tasks using 2-fold cross-validation for hyper-parameter tuning (Levy et al. 2015).

2.3 The sequence representations

It is very often in Natural Language Processing that we have to work with structures that are composed of sequences of individual words, specifically sentences, paragraphs and documents. Studying representations of these structures, either from learned word representations or from scratch, is one of the trending topics in Computational Linguistics. There are two popular approaches to this problem, one is the bottom-up approach (Semantic Composition) that learns semantic compositions from smallest compositional units (e.g. *adj-noun* word pairs) and another is similar to distributed word representations, which learns the compositional representations using their context.

2.3.1 Semantic composition

Smolensky (1990) uses tensor products to connect vectors of individual units to produce structured representations. Tensor product is a generalisation of outer product that produces tensors whose components are products of all possible pairs (u_i, v_j) of the components of vector u and v. However tensor products result in exponential growth in dimensionality when computing representations of structures with many constituents.

To overcome this problem, Plate (1995) proposes to use circular convolution to compress a tensor product into a vector whose dimensionality is the same as the input vectors. A circular convolution is defined as $\circledast : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}^n$ such that $t = t \circledast x$ where $t_i = \sum_{j=0}^{n-1} c_j x_{(i-j)\ mod\ n}$. Circular correlation can be used for lossy decompression. Kanerva (2009) proposes binary spatter codes that are typically random binary N-vectors. Compositional representations are combined by component-wise exclusive OR (XOR) of two vectors. Both binary spatter codes and circular convolutions can be computed efficiently meanwhile we can still keep the dimensions unchanged. To retrieve the component vectors, noisy versions can be reconstructed and compared to all components vectors.

While tensors can represent relations and role-filler bindings in a distributed fashion (e.g., in $loves(John,\ Mary)$), any binding scheme based on tensors violates the role-filler independence. It means that two tensors are only similar when both their roles and fillers are similar. Similar roles when bound to completely different fillers result in completely different tensor products (Holyoak and Hummel 2000). Landauer and Dumais (1997) propose to use simple addition $a+b$ to represent bi-gram ab. This naive model lacks order sensitivity, i.e. bi-gram ab has exactly the same representations of ba. Nevertheless, Mitchell and Lapata (2010) show the simple addition works in practice. They additionally suggest to use element-wise multiplication as a composition operation where $[ab]_i = a_i \cdot b_i$. The element-wise multiplication operation has the best performance in an experimental evaluation of seven compositional models and two non-compositional models.

In the holistic approach, frequent bi-grams are treated as individual words. Representations of bi-grams are constructed in the same manner as uni-grams. This approach works well for a small set of high-frequency n-grams (Turney 2012). Guevara (2010) use Partial Least Square Regression (PLSR) to predict new bi-gram compositional representations from holistic context vectors of a small set of bi-grams. Their results were promising with the representations generated by PLSR outperforming additive and multiplicative models in one of the two evaluations.

In other approaches, Clark and Pulman (2007), Widdows (2008), Clark et al. (2008), Grefenstette and Sadrzadeh (2011) combine symbolic models and distributional models with linear algebra such as tensor products. Socher et al. (2011), Turney (2012), Turney (2013) work directly with similarity rather than composition.

2.3.2 Distributed representations

In an entirely different approach, similar to distributed word representations, sentence and structured representations are learned based on the distributional hypothesis (Harris 1954), e.g. these representations can be learned from the context where they appear. Le and Mikolov (2014) introduce two models to learn representations of paragraphs. In PV-DBOW model, each word and paragraph is mapped to a unique vector. Similar to *word2vec*, *doc2vec* learns word and paragraph representations by predicting words using their context and the paragraphs the predicted words belong to. Specifically to predict each word, the context word and paragraph representations (including the predicted word) are either averaged or concatenated as a vector h. The vector h is then used as an input into a maximum entropy classifier to predict the word in the centre. At prediction time, the paragraph representations are obtained similarly as at training time with fixed word representations and other weights of the trained model. Le and Mikolov (2014) go further with another model PV-DM which is similar to PV-DBOW but without using the word vectors at all. The authors find that combining the paragraph representations from two methods usually helps across many tasks.

Also inspired by Skip-gram model, Kiros et al. (2015) introduce skip-thought vectors where sentence representations are learned by predicting the previous and the next sentences with an encoder-decoder model. For each triple S_{i-1}, S_i, S_{i+1}, first, a recurrent neural network encodes S_i into a fixed-length vector. This fixed-length vector, or sentence representation, is then used as an input to another recurrent neural network, combined with the current word, to separately predict the next word in S_{i-1} and S_{i+1} consecutively.

Hill et al. (2016) introduce another encoder-decoder model called Sequential Autoencoder (SAE) and Sequential Denoising Autoencoder (SDAE) which have the same architecture as sequence-to-sequence model (Vinyals and Le 2014). SAE encodes sentences and reconstructs them from the encoded version while SDAE works similarly but with corrupted sentences as inputs. Hill et al. (2016) also propose FastSent that is similar to Skip-gram to predict the presence of words in context sentences using the sum of word representations in the middle sentence.

To capture longer historical context, Gan et al. (2017) use a hierarchical CNN-LSTM encoder-decoder model which is similar to the above models but the last sentence representations are fed into an additional LSTM layer to combine with the current sentence's representations for the final encoded representations (hierarchical). Another proposed model is a CNN-LSTM that combines the auto-encoder and future predictor that predicts both the encoded sentence and the next sentence in a similar manner as SkipThought's (composite). Finally the authors combine the representations produced by the hierarchical model and the composite model to produce a combine representations (combine).

Pagliardini et al. (2018) extend C-BOW (Mikolov et al. 2013) to learn sentence representations by using additions of n-gram representations from a sentence to predict n-grams present in the same sentence (Sent2Vec). The sentence representation is computed by averaging the sentence's n-gram representations. The simple approach results in superior sentence representations and training time compared to the existing methods when trained with un-ordered sentences.

Arora et al. (2017) propose to use a simple weighted average of the sentence's word representations for computing the sentence representations

$$c_s = \sum_{w \in s} \frac{a}{p(w) + a},$$

where $p(w)$ is the probability of word w appearing in the corpus. The weighting scheme penalises frequent words and emphasises infrequent words. The weighting scheme is derived from an observation that given the latent embedding c_s of sentence s, a word $w \in s$ is sampled with a probability that is not only proportional to $\langle v_w, c_s \rangle$ but also $p(w)$. One explanation for this idea is that language is not composed of just semantic but, possibly, syntactic rules. The authors go further to remove the projection to the first principle component from the learned sentence representations. The idea comes from their observation that projections of word representations built by existing methods into the first principle components correlate to common words.

Chen (2017) revamps Huang et al. (2012) and Le and Mikolov (2014)'s idea to learn word representations and document representations by predicting the target word from context words and document's representations (Doc2VecC). The main difference to Le and Mikolov (2014)'s work is that the document representations is the average of the representations of words in the document. Additionally the document is randomly corrupted to increase the model's generalisation.

2.3.3 Evaluations

The learned representations are evaluated in supervised evaluations and unsupervised evaluations.

2.3.3.1 Unsupervised evaluations

Sentence representations are used to evaluate sentence similarities. Two popular sentence similarity data sets are SICK (Marelli et al. 2014) and STS (Agirre et al. 2014). The SICK data set consists of 10,000 pairs of sentences. For each sentence pair, 10 different human annotators score values from 1-5 to indicate whether two sentences are not at all related to highly related. The data set is split into 4,500 training pairs, 500 development pairs and 4,927 testing pairs. All sentence pairs are used for evaluation except the development pairs are held-out for tuning hyper-parameters in some methods. The STS data set consists of 3,750 sentence pairs and ratings. The STS sentence pairs come from six different linguistic domains. For evaluation, sentence-pair similarity is computed by *cosine* distance of their representations. The *cosine* distances are correlated with gold-standard human judgements.

Unfortunately, Arora et al. (2017) and Chen (2017) do not report the performances of their generated representations for this task. Overall even trained with unordered sequences Sent2Vec models are the best performers for reported tasks in all models trained with either unordered or ordered sequences. Interestingly Sent2Vec with uni-grams consistently outperforms, or at least on par with, Sent2Vec-bigrams except SICK and STS Forum data sets (Table 2.2). All complex approaches with

	STS 2014							SICK
Model	News	Forum	WordNet	Twitter	Images	Headlines	All	Test + Train
SAE	17/.16	.12/.12	.30/.23	.28/.22	.49/.46	.13/.11	.12/.13	.32/.31
SAE+embs.	.52/.54	.22/.23	.60/.55	.60/.60	.64/.64	.41/.41	.42/.43	.47/.49
SDAE	.07/.04	.11/.13	.33/.24	.44/.42	.44/.38	.36/.36	.17/.15	.46/.46
SDAE+embs.	.51/.54	.29/.29	.56/.50	.57/.58	.59/.59	.43/.44	.37/.38	.46/.46
DBOW	.31/.34	.32/.32	.53/.5	.43/.46	.46/.44	.39/.41	.42/.43	.42/.46
DM	.42/.46	.33/.34	.51/.48	.54/.57	.32/.30	.46/.47	.44/.44	.44/.46
Skip-gram	.56/.59	.42/.42	.73/.70	.71/.74	.65/.67	.55/.58	.62/.63	.60/.69
CBOW	.57/.61	.43/.44	.72/.69	.71/.75	.71/.73	.55/.59	**.64/.65**	.60/.69
Uni-gram TFIDF	.48/.48	.40/.38	.60/.59	.63/.65	72/.74	.49/.49	.58/.57	.52/.58
Sent2Vec uni.	**.62/.67**	.49/.49	**.75/.72**	**.70/.75**	**.78/.82**	**.61/.63**		.61/.70
Sent2Vec uni. + bi.	**.62/.67**	**.51/.51**	.71/.68	**.70/.75**	.75/.79	.59/.62		**.62/.70**
SkipThought	.44/.45	.14/.15	.39/.34	.42/.43	.55/.60	.43/.44	.27/.29	.57/.60
FastSent	.58/.59	.41/.36	.74/.70	.63/.66	.74/.78	.57/.59	.63/.64	.61/.72
FastSent+AE	.56/.59	.41/.40	.69/.64	.70/.74	.63/.65	.58/.60	.62/.62	.60/.65

Table 2.2: Performance of sentence representations models (Spearman/Pearson correlations) on unsupervised (relatedness) evaluations. Models with **+embs** are trained with fixed pre-trained word representations.

recurrent neural networks such as SAE, SDAE and Skip-Thought perform poorly compared to more light-weight approaches with simple average over word representations (Sent2Vec, FastSent, CBOW, Skip-gram). Hill et al. (2016) hypothesise that this may be owing to the sequential models computing non-linear encoding of internal sequence representations whose geometry may not be reflected in a simple cosine distance.

2.3.3.2 Supervised evaluations

Several text classification benchmarks are used for evaluating sentence representations generation methods. Typically 5 data sets are used: Movie Review sentiment (MR) (Pang and Lee 2005), Customer Reviews (CR) (Hu and Liu 2004), subjectivity/objectivity classification (SUBJ) (Pang and Lee 2004), opinion polarity (MPQA) (Wiebe et al. 2005) and question-type classification (TREC) (Li and Dan 2002). Each task contains a set of sentences or documents and human annotated labels. All documents are first converted into their representations by the compared methods. These representations are then applied with a logistic regression classifier and L2 regularisation. For data sets whose train-test split is not pre-defined, a 10-fold cross-validation is used. The L2 penalty is tuned by cross-validation on the train group. More details of this process can be found in (Kiros et al. 2015)[5]. The bag-of-word baseline for this task is NB-SVM (Wang and Manning 2012) which uses SVM with Naive Bayes log-count ratio on bi-grams. NB-SVM is reported to be a robust and state-of-the-art performer on some popular text classification tasks.

Another supervised evaluation method is paraphrase detection on the Microsoft Research Paraphrase corpus (Dolan et al. 2004). The data set consists of a training set of 4,076 sentence pairs and a test set of 1,724 sentence pairs. Each pair is labelled whether they are paraphrases. To classify whether two sentences are paraphrases, a concatenation of component-wise product $u \cdot v$ and their absolute difference $|u - v|$ is used as an input into a logistic regression model trained similarly to the above text classification tasks. Similar setting is also used to predict sentence similarity with SICK data set where similarity score is predicted in a supervised manner using the same set o features $u \cdot v$ and $|u - v|$ (Kiros et al. 2015).

Performances of the models in text classification task are shown in Table 2.3. The models are grouped by whether they require ordered sentences to be trained with. The results of Bi-LSTM and Tree-LSTM Tai et al. (2015) models that are trained directly with the data set are also reported to compare with the learned representations. The results of supervised sentence similarity task with SICK data set are shown in Table 2.4. Unfortunately, Arora et al. (2017) and Chen (2017) do not report performances of their models in the same textual classification task and Arora et al. (2017) report only *Pearson* correlation for sentence similarity task.

[5] code located at `https://github.com/ryankiros/skip-thoughts/blob/master/eval_classification.py`

Data	Model	MSRP (Acc/F1)	MR	CR	SUBJ	MPQA	TREC
	SAE	74.3 / 81.7	62.6	68.0	86.1	76.8	80.2
	SAE+embs.	70.6 / 77.9	73.2	75.3	89.8	86.2	80.4
	SDAE	76.4 / 83.4	67.6	74.0	89.3	81.3	77.6
	SDAE+embs.	73.7 / 80.7	74.6	78.0	90.8	86.9	78.4
	DBOW	72.9 / 81.1	60.2	66.9	76.3	70.7	59.4
	DM	73.6 / 81.9	61.5	68.6	76.4	78.1	55.8
	Skip-gram	69.3 / 77.2	73.6	77.3	89.2	85.0	82.2
	CBOW	67.6 / 76.1	73.6	77.3	89.1	85.0	82.2
	Uni-gramTFIDF	73.6 / 81.7	73.7	79.2	90.3	82.4	85.0
	Sent2Vec uni.	72.2 / 80.3	75.1	80.2	90.6	86.3	83.8
	Sent2Vec uni. + bi.	72.5 / 80.3	75.8	80.3	91.2	85.9	86.4
	SkipThought	73.0 / 82.0	76.5	80.1	93.6	87.1	92.2
	FastSent	72.2 / 80.3	70.8	78.4	88.7	80.6	76.8
	FastSent+AE	71.2 / 79.1	71.8	76.7	88.8	81.5	80.4
	Gan et al. (2017) hierarchical	74.0 / 82.5	75.2	78.0	91.7	88.2	90.0
	Gan et al. (2017) composite	74.7 / 82.2	76.3	79.93	92.5	88.8	91.4
	Gan et al. (2017) combine	75.5 / 82.6	77.2	80.9	93.1	89.1	91.8
	Gan et al. (2017) combine+embs.	**76.45 / 83.76**	**77.77**	**82.05**	**93.63**	**89.36**	**92.6**

Table 2.3: Performances of sentence representations in supervised textual classification.
(Hill et al. 2016; Gan et al. 2017)

Overall, Gan et al. (2017)'s combine model perform best across all different data sets. Individually both Gan et al. (2017)'s hierarchical and composite models perform comparably to Skip-Thought. On the contrary to the unsupervised sentence similarity task, sequential models outperform light-weight averaging models in all tasks. The representations generated by sequential models when used as inputs to a supervised logistic regression model perform at least comparably to averaging models (Doc2VecC, Arora et al. (2017)). Corrupting inputs also helps with generalisation when SDAE outperforms SAE in 4 out of 5 textual classification data sets. This mechanism also proves to be useful in Chen (2017). Fixing word representations (+embs models) with pre-trained word representations improves performances across different models (SAE, SDAE, Gan et al. (2017)'s models).

2.4 Conclusion

This chapter provides a broad overview of different methods in text classification and textual representation learning. The most effective methods that learn word representations are CBOW and Skip-gram. Even though these methods use Neural

Method	r	ρ	**MSE**
Skip-thought	0.8584	0.7016	0.2687
Gan et al. (2017) hierarchical	0.8333	0.7646	0.3135
Gan et al. (2017) composite	0.8434	0.7767	0.2972
Gan et al. (2017) combine	0.8533	0.7891	0.2791
Gan et al. (2017) combine+embs.	**0.8618**	**0.7983**	**0.2668**
Doc2VecC	0.8381	0.7621	0.3053
Arora et al. (2017)	0.8603		
Task-dependent methods			
Bi-LSTM	0.8567	0.7966	0.2736
Tree-LSTM	0.8676	0.8083	0.2532

Table 2.4: Performance of supervised sentence similarity with SICK data set. (Hill et al. 2016; Gan et al. 2017)

Networks for training, they optimise very similar goals to matrix factorisation methods. These networks are thus considered shallow and capture only simple interaction of linguistic discourses. The "shallowness" comes with an important advantage which is their short training time. However the amount of available computational power nowadays has made it much faster to train Deep Neural Networks. In some systems, thousands of Tensor Processing Units (TPUs) are used to synthesise data and train Neural Networks (Silver et al. 2018). The abundance of computational power has made me confident to study usage of Deep Neural Networks in learning language generation which I discuss in detail in Chapter 3. For sequence representation learning, sequential auto-encoders have been used widely. As reviewed earlier RBMs are at least as effective as auto-encoders in reported experiments meanwhile they were not used for this task before. Therefore in Chapter 4, I explore the usage of RBMs for learning sequence representation and evaluate the learned representations in different text classification tasks. Finally in Chapter 5, I investigate applications of different Deep Neural Networks and pre-trained language representations in a real-world task of Adverse Drug Reaction classification and show their effectiveness.

Chapter 3

Word and sentence representation with sequential modelling

Word representations have been learned by matrix factorisation methods or methods that optimise for similar goals. However, the studied methods have been limited to methods based on co-occurrence statistics or windowed bag-of-word shallow Neural Networks. These methods are usually so computationally efficient that they can be trained with huge corpora that may contain billions of words. These bag-of-word models do not utilise the structural nature of languages for their inferences: we hypothesise that by using structural architectures, specifically recurrent Neural Networks, derived word representations contain properties learned from the preserved sequential nature of the input text. Meanwhile the latest sequential models focus less on learning word representations but sentence representations. Additionally, sequential models like Skip-thought or Seq2seq (Sutskever et al. 2014; Cho et al. 2014) are trained to predict only the next immediate word conditioned on the current word and previous sentence representation. We hypothesise that this limits the capability of the models to learn word and sentence semantic as matrix factorisation methods have proven that predicting multiple words within a limited distance improves the learned word representations. In this chapter, I look at different methods to train sequential models to learn word representations and address the research question **RQ1**.

3.1 Methodologies

Similar to other sequential models, our model first summarises the content of the each sentence into a fix-sized vector h. This vector is then used to predict words appearing in its next sentence. For example, given a dictionary $\mathcal{V}$, $a_1, a_2, \ldots, a_n$ is a sequence of words from the current sentence, and $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m$ is a sequence of words in the next sentence, we would like to find a model P so that:

$$P(\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m | a_1, a_2, \ldots, a_n) = \max_{b_i \in \mathcal{V}} P(b_1, b_2, \ldots, b_m | a_1, a_2, \ldots, a_n) \tag{3.1}$$

The problem with this type of models is that with the vocabulary size of typical hundreds of thousands, the prediction space would be too large for contemporary computational power and require a huge number of training data. In Skip-thought or Seq2seq, the goal is simplified as:

$$\hat{b}_i = \arg\max_{b \in \mathcal{V}} P(b | \hat{b}_1, \ldots, \hat{b}_{i-1}, a_1, \ldots, a_n) \; \forall i \in [1 \ldots m]$$

This type of simplification is good for sentence generation as it takes as much information as possible to predict the next word but not necessarily good for learning word representations or sentence representations. Meanwhile Word2vec, Glove and other matrix-factorisation based methods show that predicting words in further distance helps with learning word representations with better quality (Levy et al. 2015).

We make a simple observation that to learn word and sentence representations, predicting the exact order of the words in the next sentence is not necessary. It is often that inserting, deleting or swapping one or two words from the sentence doesn't change its meaning drastically (Landauer et al. 1997). In many cases, given the context one can understand the sentence well without the correct order of the words at all. Based on this observation, we relax the target so that $\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_m$ are independently drawn. With this simplification we get

$$P(b_1, b_2, \ldots, b_m | a_1, a_2, \ldots, a_n) = \prod_i^m P(b_i | a_1, a_2, \ldots, a_n).$$

Computing $P(b_i | a_1, a_2, \ldots, a_n)$ is significantly less computationally expensive compared to the joint probability. However, a typical vocabulary with a size of hundred thousands of words is still not scalable when training a classification model with a vanilla $softmax$ function. To tackle this problem, we use negative sampling (Mikolov et al. 2013) to sample a small set of negative samples for each positive word. With negative sampling, our final goal is to find the maximum likelihood of a binary classification instead

$$\prod_i^m \left(P\big(y(b_i) = 1 | a_1, a_2, \ldots, a_n\big) \prod_{\tilde{b} \in N(b_i)} P\big(y(\tilde{b}) = 0 | a_1, a_2, \ldots, a_n\big) \right),$$

where $N(b_i)$ is a set of negative samples for word b_i, and $y(w)$ is the indication of whether the word w appears in the current sentence.

We use a one-layer LSTM (Figure 3.1), followed by a hidden linear layer h to summarise the current sentence $a_1, a_2, \ldots, a_n$ with an initially randomised embedding W. Let h be the output vector of the recurrent network, we now have:

$$P\big(y(w) | a_1, a_2, \ldots, a_n\big) = P\big(y(w) | h\big)$$

The probability of a word appearing in the current sentence can be computed from the dot product of its embedding and the previous sentence's summary h:

$$P\big(y(w) = 1 | h\big) = \frac{1}{1 + \exp(-W_x \cdot h)}$$

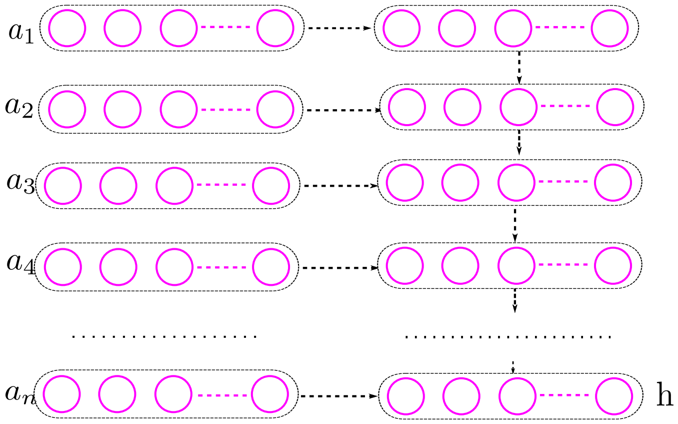


Figure 3.1: LSTM Recurrent Neural Network. The arrows show the directions of connections

So far, our model is only different to skip-thought or seq2seq models at predicting consecutive words or independent words. Now we discuss several tweaks that make our models distinctive from the existing models.

3.1.1 Predicting context

We hypothesise that a good sentence presentation should be able to capture not only future but also past and presence. In fact, Gan et al. (2017) show that a *composite* model that decodes both current and next sentence beats the *future* model that decodes only the next sentence. Similarly in learning word embeddings (word2vec, glove, svd), predicting contexts including words on both sides improves the learned word embedding quality. From this observation, we extend the model to predict words appearing in a fixed-size window of sentences centred at the current sentence rather than predict just words in the immediate next or previous sentence. In the experiment, we report how this affects the quality of learned word representations. To highlight the difference, let us have a look at the example of the following excerpt:

s_{-2}: I flipped open the pad and wrote: walking home.

s_{-1}: I shoved it back in my pocket and continued walking.

s_0: The sun was almost gone and shadows were starting to appear behind everything.

s_1: I looked at my own shadow, that danced behind me.

s_2: The phone buzzed again as Seth sent me a reply.

Skip-gram iterates through each word and tries to predict the words surrounding it independently. CBOW iterates through each word and tries to predict that word using its surrounding words. Skip-thought iterates through each sentence and predicts the words in the next and previous sentences consecutively. Our methods iterate through each sentence and predict words in the surrounding sentences and itself independently. Specifically in the given example, we iterate through each sentence s_{-2} to s_2. At s_0, we compute the representation of s_0 and use it to predict words in all sentences s_{-2} to s_2.

3.1.2 Stateful LSTM

We make a small modification to the traditional recurrent layer where the initial states of the recurrent units are either randomly initialised or set as zeros. Our LSTM initial states are initialised as the last values from the previous sentence. The purpose of this modification is to make the LSTM layer to take the representation from the last sentence into the computation of the current sentence's representation. However in order to achieve this, we have to shuffle the training sentences so that the data can be independently and identically distributed. This makes our optimisation be no longer standard Stochastic Gradient Descent. Nevertheless the empirical results show a good convergence of the loss function on the validation set and trained word representations' quality. It is worth noting that during training, the LSTM is not unfolded beyond the first word of each sentence. Thus the error gradient is not back propagated any further than the first word of the centre sentence.

3.1.3 Data interleaving

In order to set the last states of the previous sentence as the initial states of the next sentence, we first read a large batch of K consecutive sentences ($K = 10,000$ for small data sets and $K = 100,000$ for larger data sets). Each epoch of training is done by iterating through all mini-batches created from large batches. Each mini-batch has a size of 25 sentences. Mini-batch i from a large batch is formed by sentences with indices $i, \frac{K}{25} + i, 2\frac{K}{25} + i, \ldots, 24\frac{K}{25} + i$ from the large batch. Each large batch has $\frac{K}{25}$ mini-batches. By iterating through interleaved sentences from mini-batches, the last states of the LSTM from the previous sentence is remained as initial states of the LSTM for the next sentence.

3.1.4 Neural Networks

There are many different ways to transform a sentence into a fixed-size vector using Neural Networks. In this study, we experiment with 3 different simple architectures:

- A simple sum of word embeddings from the sentence that is the same as FastSent model (Hill et al. 2016).
- A simple forward LSTM, stacked by a linear layer (LSTM).
- A convolutional layer, followed by a forward LSTM layer and a linear layer (Conv-LSTM).

In FastSent model, the sum of the word embeddings is the sentence representation. In both LSTM and Conv-LSTM models, the output of LSTM at the last word of the

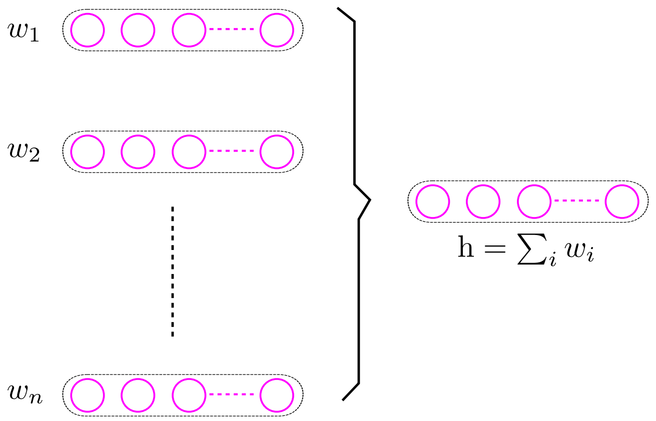


Figure 3.2: FastSent

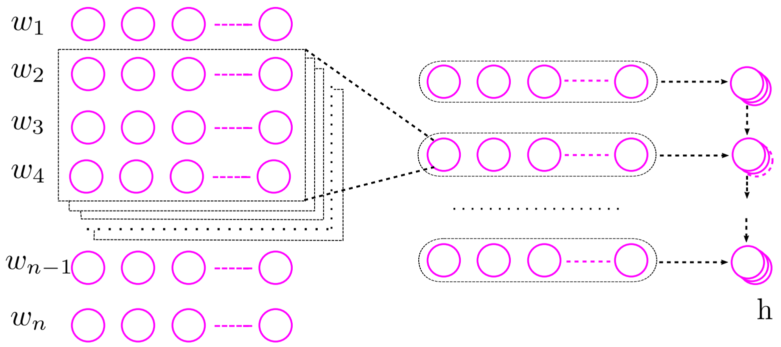


Figure 3.3: Conv-LSTM

sentence is our sentence representation.

3.2 Experiments

In this experiment, we evaluate the capability of our approach to learn word representations[1]. We train the model with two corpora: the Brown corpus[2] and the Book corpus (Zhu et al. 2015). The Brown corpus consists of one million words of American English text printed in 1961. The text is sampled for 15 different categories including press reports, press reviews, science fiction, etc. The Book corpus is a larger data set that consists of almost one billion words from 11,038 free books written by yet

[1] Source code of the experiments can be found at `https://github.com/trunghlt/DocumentModel`

[2] `https://www1.essex.ac.uk/linguistics/external/clmt/w3c/corpus_ling/content/corpora/list/private/brown/brown.html`

unpublished authors. With the Brown corpus, we experiment with different architectures and parameters as it is quicker to train with a smaller data set. With the Book corpus, we could only do one experiment with one configuration of parameters as it took us two months to train for only one epoch with a Geforce GTX 1080 Ti GPU. We now discuss different settings that are used for training the model in both corpora.

For both corpora, we choose the number of negative samples as 2 (samples are drawn from uniform distribution), maximum length of centre sentence as 50 (longer sentences are clipped and shorter sentences are padded with a special token at the beginning), maximum length of predicted sentences as 25 (clipped and padded similarly to centre sentence), word embeddings with 300 dimensions, the vocabulary size of $20,000$. The vocabulary is built from the top most frequent words for each corpus. All models are trained by Adagrad (Duchi et al. 2011). With Brown corpus, we iterate through 200 training epochs while we could only train with 1 epoch with the Book corpus as it took us 2 months to train for only 1 epoch.

We compare our trained word embeddings with ones learned by CBOW and Skip-gram. Both CBOW and Skip-gram models are trained by Gensim[3] with all words lower-cased, 1 epoch (we found these methods are quickly overfitted and work best when trained with 1 epoch), window size of 10 and dictionaries with similar size to the other models.

Following Section 2.3.3, we evaluate the trained word embeddings within word similarity task (data sets include WordSim353, WordSim353 Similarity, WordSim353 Relatedness), word analogy task (Google and MSR data sets) and text classification

[3] `https://radimrehurek.com/gensim/models/word2vec.html`

Method	WordSim353	WordSim353 Similarity	WordSim353 Relatedness	SimLex999	Google Add/Mul	MSR Add/Mul
CBOW	0.157	0.251	0.080	0.116	0.0177/0.0136	0.0325/0.0238
Skip-gram	0.265	0.325	0.186	**0.130**	**0.0287/0.0255**	**0.0512/0.0517**
FastSent	0.280†	0.317	0.200	0.067	0.0066/0.0057	0.0028/0.0022
LSTM	0.303†	**0.376**†	0.215†	0.060	0.0023/0.0021	0.0022/0.0017
Conv-LSTM	**0.318**†	0.369†	**0.240**†	0.104	0.0029/0.0013	0.0020/0.0017

Table 3.1: Word similarity and word analogy evaluations of word embeddings trained with Brown corpus.

Results with † are statistically significantly different from both CBOW and Skip-gram baselines. All tests are done using bootstrapping with p-value = 0.05 and Bonferroni correction. Bold values are the best for each data set. Bold values are the best for each data set.

Method	WordSim353	WordSim353 Similarity	WordSim353 Relatedness	SimLex999
CBOW	0.536	0.658	0.431	0.403
Skip-gram	**0.614**	**0.689**	**0.521**	**0.436**
Conv-LSTM	0.585	0.655	0.501	0.313

Table 3.2: Word similarity and word analogy evaluations of word embeddings trained with Brown corpus.

Bold values are the best for each data set.

(Movie Review and Subjectivity[4]) for Brown corpus. For Book corpus, we evaluate the trained word embeddings with word similarity tasks. The settings for word similarity and word analogy evaluations are the same as in Levy et al. (2015). For text classification, we simply compute the means of word embeddings in input documents and use them as inputs to a simple logistic regression model for binary classification. Results are the mean of accuracies from 20-fold cross validations.

With Brown corpus where we could train our models with 200 epochs, both results on word similarity and word relatedness trained by LSTM and Conv-LSTM when evaluated with WordSim353 data set are better than ones trained by bag-of-word

[4] `http://www.cs.cornell.edu/people/pabo/movie-review-data`

Method	Movie Review	Subjectivity
CBOW	0.602	0.651
Skip-gram	0.641	0.658
LSTM	**0.655**†	**0.683**†

Table 3.3: Text classification results using mean of embeddings trained from different methods with Brown.

Results with † are statistically significantly different from CBOW and Skip-gram baselines. All tests are done using student-t tests with p-value = 0.05 and Bonferroni correction. Bold values are the best in each data set.

Sentence window	WordSim353	WordSim353 Similarity	WordSim353 Relatedness	SimLex999
3	0.165	0.305	0.022	**0.118**
5	$0.236^{\dagger}$	$0.362^{\dagger}$	0.092	0.110
11	$\mathbf{0.273}^{\dagger}$	**0.367**	$\mathbf{0.153}^{\dagger}$	0.084

Table 3.4: Word similarity and word analogy evaluations of word embeddings trained by LSTM with different sentence windows.

Results with † are statistically significantly different from the results on the immediate row above. All tests are done using bootstrapping with p-value = 0.05 and Bonferroni correction. Bold values are the best in each data set.

Sentence window	WordSim353	WordSim353 Similarity	WordSim353 Relatedness	SimLex999
Shuffled data and non-stateful LSTM	0.232	0.294	0.143	**0.101**
Interleaved data and stateful LSTM	$\mathbf{0.273}^{\dagger}$	$\mathbf{0.367}^{\dagger}$	**0.153**	0.084

Table 3.5: Word similarity and word analogy evaluations of word embeddings trained by LSTM.

Results with † are statistically significantly different from the results on the immediate row above. All tests are done using bootstrapping with p-value = 0.05. Bold values are the best in each data set.

models Skip-gram and CBOW. When evaluated with SimLex999, even though Skip-gram is still the best model, the gap between LSTM and Skip-gram is small (0.006). In word analogy evaluations, LSTM and Conv-LSTM's results are significantly worse than than CBOW and Skip-gram. We hypothesise that analogy inferences from word embeddings trained by more complex sequential models require more sophisticated modelling than simple 3CosMul or 3CosAdd (Levy et al. 2015). In text classification, word embeddings trained from our LSTM model significantly outperform ones trained with CBOW and Skip-gram in Movie Review and Subjectivty data sets (Table 3.3).

To evaluate the impact of sentence window size in which we predict word appearances, we experiment with different window sizes of 3, 5 and 11. The results in Table 3.4 show that bigger window size is better for learning word representations. However the gain from bigger window size reduces when window size is larger than 5. Choosing larger window size however increases training time linearly. Therefore one would want to choose an appropriate window size so that the additional training time is worth of performance gain.

To evaluate the impact of stateful LSTM compared to non-stateful LSTM, we experiment with training the word representations with two models where data are interleaved with stateful LSTM and shuffled with non-stateful LSTM. The results from two experiments are reported in Table 3.5 which show stateful LSTM significantly outperforms non-stateful LSTM in 3 of out 4 data sets, especially by a large margin in WordSim353 Similarity data set.

With Book corpus where Conv-LSTM is trained with only 1 epoch, its word similarity evaluation is not as good as Skip-gram across different data sets but comparable to CBOW. Unfortunately as it takes us two months to train the model, we could not experiment with training Conv-LSTM with more epochs. In a hindsight, one should go with a simpler model, perhaps LSTM, it is quicker to train.

3.3 Conclusion

In this chapter, we propose and experiment with a novel approach of learning word representations where these word representations are trained by predicting word appearances in centre sentence and its surrounding sentences using summary of the centre sentence. Our ideas including predicting word appearances in distant sentences, stateful LSTM and interleaved training data prove to efficiently improve the learned word representations on state-of-the-art Skip-gram models on Brown corpus in similarity, relatedness metrics. Our embeddings also perform better than CBOW and Skip-gram in text classification when means of word embeddings are used as input. Unfortunately our model takes significantly more time to train than Skip-gram. Therefore in a larger Book corpus, we could only train our model with 1 epoch and could not achieve better word representations than Skip-gram's. Another drawback from our approach is our learned word representations do not perform as well as

Skip-gram and CBOW in analogy tasks. We hypothesise that as our representations are trained with more complex sequential models, it requires non-linear transformation for inferences rather than simple linear 3CosAdd or 3CosMul. In Chapter 4, I show that sums of word representations when combined with phrase-level representations further improves subjectivity classification on MPQA data set and sentiment classification on Movie Review data set. In Chapter 5, I show that using Deep Neural Networks with embedded word representations improves Adverse Drug Reaction classification over simple sums of word representations.

In the future, we would like to continue the experiments with sub-sampling as used in Mikolov et al. (2013) and more sophisticated sampling methods than currently used uniform sampling of negative words. Another improvement could be to make stateful LSTM propagate further than the first word of the centre sentence. This will help the error to propagate to words in previous sentences and, potentially, help the model to learn more effective representations as they capture longer relationship and extract more information from the same amount of training data. In term of speed, one could experiment with deep convolutional networks or Transformer network (Vaswani et al. 2017) as these networks can be parallelised more efficiently.

Chapter 4

Higher level features with generative models

As demonstrated in Chapter 3, semi-supervised learning for word representations and sentence representations helps improve performance for text classification as a whole. In this chapter, we further study learning the representations of higher-level discourse structures with generative models. Given that textual semantic changes depending on word compositions and discourse structures, we experiment with learning representations of phrases using existing word representations and stacked Convolutional Restricted Boltzmann Machine. With stacked unsupervised Neural Networks, new representations can be learned for n-grams, phrases or the whole sentences. We observe that the learned representations when combined with existing features help improve overall performances on two tasks: subjectivity vs objectivity classification and sentiment classification.

4.1 Related work

In text classification tasks where labelled data is scarce, it is useful to use features learned from unsupervised methods as additional features for training. At word level, work by Collobert and Weston (2008), Turian et al. (2010), Dahl et al. (2012) shows that word representations learned by generative models improve overall performance on different NLP tasks including POS tagging, chunking, Named Entity Recognition, Semantic Role Labelling and Sentiment Analysis. At sentence level, Ko and Seo (2000) use manually create keywords to classify sentences into predefined categories and feed these information into a Naive Bayes classifier for document category classification while Hill et al. (2016) use *autoencoder* and *denoising autoencoder* to learn sentence representations.

Subjectivity/Objectivity classification is a task where a sentence is classified as whether nor not it carries the author's own opinion. Pang and Lee (2004) show that removing subjective sentences and their adjacent sentences slightly improves accuracy over their baseline. In information retrieval, the ability of telling whether a sentence is subjective or objective also allows users to retrieve only opinions or facts. Riloff and Wiebe (2003), Wiebe and Riloff (2005) focus on a bootstrapping process that learns subjective and objective patterns from sentences. Wiebe and Riloff (2005) develop the idea further with training Naive Bayes classifiers from bootstrapped data. Other related works extending their work with various n-gram features and different lexical instantiation include papers by Wilson and Raaijmakers (2008), Raaijmakers et al. (2008), Murray and Carenini (2009).

Early work on sentiment analysis (Pang and Lee 2004; Turney 2002; Dave, Lawrence, and Pennock 2003; Beineke, Hastie, Manning, and Vaithyanathan 2004; Pang and Lee 2005) mainly focuses on combining bag-of-words with traditional features (POS tags, negation words, bi-grams, lexical information) based on Naive Bayes, MaxEnt or SVM models. Nakagawa et al. (2010) apply CRFs with hidden variables to sentence-level sentiment classification. Socher et al. (2011) introduced *recursive* *autoencoders* that learned vector space representation for multi-words phrases and achieved the state-of-the-art performance on sentence-level sentiment classification using only sentence-level training data set.

Recent work on subjectivity and sentiment classification relies extensively on word/sentence representations and supervised Deep Neural Networks (Kalchbrenner et al. 2014; Kim 2014; Kiros et al. 2015; Hill et al. 2016).

4.2 Methodologies

For learning higher level features in sentences, we chunk each sentence in our data sets into separate words using NLTK's Treebank Work Tokenizer[1]. Words that were not present in the pre-learned word embeddings were replaced by "UNKNOWN". Additionally we tagged words using the NLTK POS tagger and replaced proper nouns (NNP, NNPS) with "ENTITY". We did not perform stemming and kept punctuations since some punctuations such as "!" might be indicative of sentiment. Each word is then represented by its corresponding word embedding, which has a form of a vector of length N_V.

For a sentence containing L words, when we stack its word embeddings, we construct a matrix of size $L \times N_V$ shown as visible layer in Figure 4.1 (left). Therefore, each sentence is represented by a matrix with the same column size (N_V) although its row size (sentence length L) differs from each other. We use the first CRBM to learn hidden features from n-grams. These first level hidden features are then fed into the second CRBM to learn another higher-level sets of features. Combining all these features improves the sentiment analysis results as will be shown in Section

[1] `http://nltk.org/_modules/nltk/tokenize.html\#word_tokenize`

4.3. Figure 4.1 illustrates how we learn higher-level features from sentences using two hidden CRBM layers. All layers are trained using Contrastive Divergence with 1-step Gibbs sampling. Figure 4.2 shows examples of the input, hidden values and reconstructed inputs into the first CRBM layers.

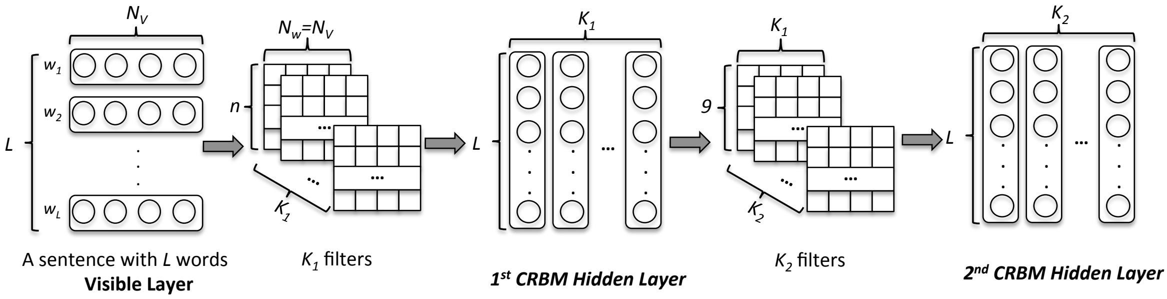
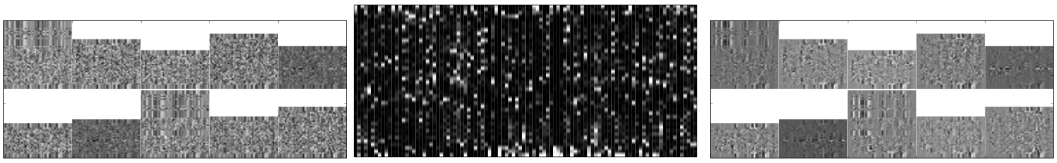


Figure 4.1: A CRBM network with two hidden layers. In all layers the width of filters is always equal to the width of the input layer. This helps the network learn interactions among all input features rather than with individual words.

For the first CRBM layer, we use K_1 Gaussian-binary unit filters[2]. Each filter has a size of $n \times N_w$. With the row size n, the CRBM can learn higher-level features from n-gram word sequences. In our work here, we set $n = 5$ as this is the typical word sequence length used for inducing word embeddings (Collobert and Weston 2008). We add two padding words at the beginning and the end of each sentence so that convolution can be done with sentences with less than 5 words. Each convolutional filter produces a $L \times 1$ vector where L is the length of an input sentence. The sum of these vectors forms a set of features that we call CRBM-layer1 features.

Stacking sequentially K_1 vectors produces a matrix of $L \times K_1$ binomial probabilistic units. We then applied another CRBM with K_2 binary-binary unit filters each of which has a size of $m \times K_1$. The parameter m is empirically set to 9. Sums of vectors

[2] We have real-valued visible units and binary-valued hidden units. As such, filters connecting the visible layer and the hidden layer have Gaussian-binary units.



(a) Visualisation of 10 randomly sampled sentences without padding.

(b) Hidden values (before pooling) from the first 100-filter CRBM layer of a sample sentence.

(c) Reconstructed representations of the sampled sentences in Figure (a) from the first CRBM layer.

Figure 4.2: Examples of sentence representations, their hidden values and reconstructed representations.

films provide some great insight	**film well worth seeing**	**of the greatest family-oriented**
abilities offers a solid build-up	is well worth seeing .	of the greatest natural sportsmen
ship makes a fine backdrop	is something worth seeing	of this italian freakshow .
still offers a great deal	is certainly worth hearing .	about the best straight-up
It makes a wonderful subject	this movie worth seeing .	to the greatest generation .
howard demonstrates a great eye	still quite worth seeing .	's very best pictures .
film delivers a solid mixture	it 's worth seeing .	of the finest kind ,

Table 4.1: Nearest neighbours of some example phrases in the CRBM first-layer hidden space with Continuous Bag-Of-Words embedding on the Movie Review data set. The distance is computed from vectors built by applying first layer filters to the phrases (so each vector has K_1 dimensions).

produced from this CRBM form another set of features that we call CRBM-layer2 features.

Table 4.1 demonstrates that the hidden features learned from the CRBM network co-locate syntactically related phrases. In addition, the learned hidden higher-level features can capture semantic similarities between phrases, for example, “films provide some great insight” and “film delivers a solid mixture” conveys a similar meaning.

4.3 Experiments

We evaluate our proposed framework on two tasks, sentence-level subjectivity classification (classify a sentence as subjective or objective) on the MPQA corpus and sentence-level sentiment classification (classify a sentence as positive or negative) on the MR dataset. Both datasets contain over 10,000 sentences and have roughly equal class distributions. For each dataset, we have experimented with two different embeddings, the C&W embeddings (Collobert and Weston 2008) and the Continuous Bag-Of-Words (CBOW) embeddings (Mikolov and Zweig 2012)[3].

C&W Embeddings Collobert and Weston (2008) construct a data set containing all possible windows of text from the entire of English Wikipedia. Positive examples are windows from Wikipedia while negative examples are the same windows but where the middle word has been replaced by a random word. They trained a two-class classification Neural Network that optimises

$$\sum_{s \in S} \sum_{w \in V} \max(0, 1 - f(s) + f(s^w)), \tag{4.1}$$

where S is the set of training windows, V is the word dictionary and s^w is the sentence where the middle word has been replaced by the word w. In practice, only a small sample of w used in training.

In this work, we use the modified version of this representation provided by Turian et al. (2010) trained with 5-gram windows as it is publicly available and widely used in other research.

Continuous Bag-Of-Words (CBOW) Embeddings In order to scale the training better with larger number of dimensions, Mikolov et al. (2013) introduce a simple one layer-neural network that predicts a word from sum of surrounding word representation. By using hierarchical softmax, the network computational complexity is $O(N \times D + D \times \log(V))$ where N is the number of surrounding words, D is number of representation dimensions and V is dictionary size. With that linear complexity,

[3] Source code of the experiments can be found at `https://github.com/trunghlt/StackedRBMs`.

the network can be easily scaled to thousands of dimensions. This embeddings has been shown by Mikolov et al. (2013) to outperform C&W embeddings in several semantic-related tasks.

4.3.1 Experimental Setup

We evaluate our proposed framework on two tasks, sentence-level subjectivity classification (classify a sentence as subjective or objective) on the MPQA corpus and sentence-level sentiment classification (classify a sentence as positive or negative) on the movie review data set (MR)[4]. Each corpus is experimented with two different architectures for two different embeddings.

C&W embeddings We experimented with two hidden layers. All layers are trained by Contrastive Divergence with 1-step Gibbs sampling.

With the first layer, we use the 50-dimension version of the word embeddings that is made publicly available by Turian et al. (2010)[5]. After transformation, each sentence then has a width of 50 columns. We used 100 Gaussian-binary unit filters and each filter has a size of 5×50 (5 rows and 50 columns). With this height, the network can learn higher level features from 5-gram word sequences (we chose 5 because it is consistent to embedding training processes). Because sentence lengths vary and can have a minimum length of 1, we added two padding words at the beginning and the end of each sentence so that convolution can be done with sentences which have less than 5 words. In order to prevent overfitting, we also set a target sparsity of the network to be $p = 0.01$. Each convolutional filter produces a $L \times 1$ ($M_H = L, N_H = 1$)

[4] `http://www.cs.cornell.edu/People/pabo/movie-review-data/`

[5] `http://metaoptimize.com/projects/wordreprs/`

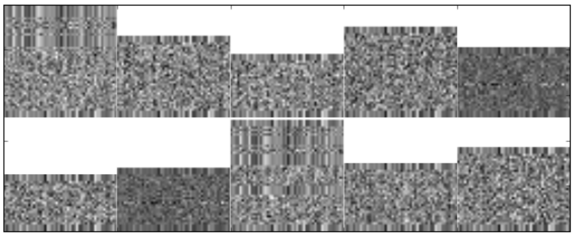


Figure 4.3: Stacked hidden layers of a sampled sentence

hidden layer where L is the length of an input sentence. Sum of these hidden layers form a set of features that we call CRBM-layer1.

Stacking horizontally 100 of these hidden layers produces images of $L \times 100$ binomial probabilistic units (e.g., see Figure 4.3). After padding these images with rows of zeros, we then apply another CRBM with 30 binary-binary unit filters and each filter has a size of 9×100. Sums of hidden layers produced from this CRBM form another set of features that we call CRBM-layer2.

We also experiment with different number of filters for each layer with the MPQA data set. Results are reported in Figure 4.4. Overall, the performance is better for more hidden layers in the first stack while it peaks at 50 layers in the second stacks. For the 2-layer stacked architecture, we choose 100 filters for layer 1 because of its relative performance and low computational intensity and memory usage compared to the model trained with 200 filters.

CBOW embedding Using the code from Google[6], we train two embeddings: first with 200 dimensions with CBOW models and data from first one billion characters from Wikipedia[7] used for subjectivity/objectivity classification task, second with 100 dimensions with data from Large Moview Review Dataset[8] used for sentiment classi-

[6] `https://code.google.com/p/word2vec/`

[7] `http://mattmahoney.net/dc/enwik9.zip`

[8] `http://ai.stanford.edu/~amaas/data/sentiment/`

fication task. Both embeddings were trained with windows of 5-grams and all other parameters of *word2vec* are set as default values. We use a similar CRBM network structure and training process as described in the previous section with only changes in number of filters and target sparsity at each layer. The first layer network has 200 filters with size of 5×200 and a target sparsity of $p = 0.01$. The second layer network has 50 filters with size of 9×200 and a target sparsity of $p = 0.5$.

Table 4.1 shows phrases that are nearest to sampled phrases in the first-layer hidden space for movie review data set. As we can see, syntactically and semantically related phrases are mapped into close neighbourhoods. This proves our learned hidden higher-level features carry useful syntactic and semantic content. However even though indicating different sentiments, some phrases are projected closely to each other (e.g. "films provide some great insight" and "it lacks a strong narrative", "you want to hate it" and "you want to love it"). This is due to the properties of CBOW embeddings whose training process does not distinguish word-level sentiments existing in training data.

4.3.2 Results

Subjectivity Classification For sentence-level subjectivity classification, we combine word embeddings in sentences with higher level features learned from stacked CRBMs and train a linear SVM model[9] with default parameters. All models are cross-validated with 10 folds.

We compare our proposed approach with four baselines. Lexicon labelling uses the MPQA subjectivity lexicon[10] to label a sentence as subjective or objective depending

[9] `http://www.csie.ntu.edu.tw/~cjlin/liblinear/`

[10] `http://mpqa.cs.pitt.edu/lexicons/`

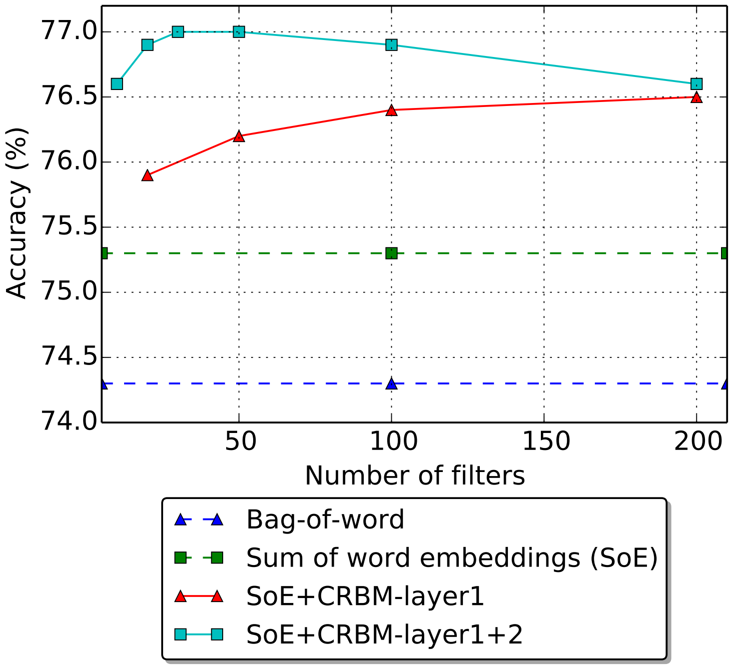


Figure 4.4: Subjectivity classification (MPQA) accuracies with different feature sets using C&W embeddings.

on the occurrence of polarity words in the sentence. SubjLDA is a weakly-supervised Bayesian modelling approach (Lin, He, and Everson 2011) that incorporates word polarity priors from the MPQA subjectivity lexicon into a variant of the Latent Dirichlet Allocation (LDA) model for subjectivity classification. The result of Naive Bayes (NB) was previously reported in (Wiebe and Riloff 2005) where a supervised NB classifier is trained from the MPQA corpus. We also train a linear SVM with bag-of-words features as another baseline.

It can be observed from Table 4.2(a) that simply training SVM from CBOW word embeddings already outperforms all the baselines. Additionally incorporating higher level features learned by CRBM further improves the accuracy. In particular, adding the CRBM-layer1 features seems to be quite effective. Further adding the CRBM-layer2 features only results in marginal improvements. Overall, with our proposed approach we observed 4% improvement in accuracy upon the best baseline model.

Model	Accuracy (%)
Lexicon labelling	63.1
subjLDA (Lin, He, and Everson 2011)	71.2
Naive Bayes (Wiebe and Riloff 2005)	73.8
SVM	74.3
With C&W 2008 embeddings	
Sum of embeddings (SoE)	75.3* **
SoE + CRBM-layer1	76.4* **
SoE + CRBM-layer1 & 2	**77***
With CBOW embeddings	
Sum of embeddings (SoE)	77.3*
SoE+CRBM-layer1	78.1* †
SoE+CRBM-layer1 & 2	**78.3***

(a) Subjectivity classification on MPQA.

Model	Accuracy(%)
ME-TFIDF	73.1
BoF+Reversal	76.4
Tree-CRF (Nakagawa, Inui, and Kurohashi 2010)	77.3
Greedy RAE (Socher, Pennington, Huang, Ng, and Manning 2011)	77.7
With C&W 2008 embeddings	
BoF+Reversal + SoE	76.9 *
BoF+Reversal + SoE + CRBM-layer1	77.3*
BoF+Reversal + SoE + CRBM-layer1 & 2	77.6*
With CBOW embeddings	
BoF+Reversal+SoE	78.1*
BoF+Reversal+SoE+CRBM-layer1	78.5* †
BoF+Reversal+SoE+CRBM-layer1 & 2	**78.7*** †

(b) Sentiment classification on MR.

* statistical significance ($p < 0.05$) with respect to the baselines
† statistical significance with respect to its next best model.
All hypothesis tests were done with student-t tests.

Table 4.2: Experimental results

Sentiment Classification For sentence-level sentiment classification, we compare our results with four baselines, Maximum Entropy with tf-idf features (ME-TFIDF), combining Bag-of-Features with Polarity Reversal (BoF+Reversal), a dependency tree based classification method employing Conditional Random Fields (Tree-CRF) (Nakagawa, Inui, and Kurohashi 2010), and the greedy Recursive Autoencoder (RAE) network (Socher, Pennington, Huang, Ng, and Manning 2011). Here, Bag-of-Features refer to the surface forms, base forms, and POS tags of word uni-grams. Polarity reversal indicates polarity reversing caused by content-word negators. These features are trained using linear SVMs with default parameters and and validated by 10-fold cross validation.

It can be observed from Table 4.2(b) that using CBOW word embeddings gives similar performance as Greedy RAE. With additional features from a two-layer CRBMs, the model outperforms all the baselines. Although the improvement may appear modest, they are very notable in comparison to the scale of improvements reported in similar literature (Nakagawa, Inui, and Kurohashi 2010; Socher, Pennington, Huang, Ng, and Manning 2011).

4.4 Conclusions

In this work, we have shown the ability to use Stacked Convolutional Deep Belief Network to learn useful higher level features of sentences with pre-trained word embeddings. These features when combined with existing common features boost the performance of simple shallow linear models (linear SVM) to outperform the baseline model using Maximum Entropy classifier with tf-idf features and state-of-the-art models in both tasks subjectivity/objectivity classification and sentiment classification. In two experimented embeddings, CBOW embeddings with higher number of dimension outperform C&W2008 in both experimented tasks.

Experiments in this chapter fulfil the research question **RQ2**. In future development, we are tempted to explore supervised deep convolutional networks with this pre-trainning setting. It is also interesting to investigate the effect of training with Fast Persistant Contrastive Divergence (Tieleman and Hinton 2009), Rectified Linear Hidden Units (Nair and Hinton 2010), Dropout (Hinton 2014) and Maxout (Goodfellow et al. 2013) on the current architecture. We are also interested in training an embedding that can distinguish between words with different sentiments and using this embedding for the experimented pipeline.

Chapter 5

Adverse Drug Reaction Classification with Deep Neural Networks

Adverse Drug Reactions (ADRs) are potentially very dangerous to patients and are amongst the top causes of morbidity and mortality (Pirmohamed et al. 2004). Many ADRs are hard to discover as they happen to certain groups of people in certain conditions and they may take a long time to expose. Healthcare providers conduct clinical trials to discover ADRs before selling the products but they are normally limited in numbers. Thus, post-market drug safety monitoring is required to help discover ADRs after the drugs are sold on the market. In the United States, Spontaneous Reporting Systems (SRSs) is the official channel supported by the Food and Drug Administration (FDA). However these systems are typically under-reported and many ADRs are not recorded in the systems. Recently unstructured data such as medical reports (Gurulingappa et al. 2012; Gurulingappa et al. 2012) or social network data (Ginn et al. 2014; Nikfarjam et al. 2015; Weng et al. 2017) have been used to detect content that contains ADRs. Case reports published in scientific biomedical literature are abundant and generated rapidly. Social networks are another source of redundant data with unstructured formats. While an individual tweet or Facebook status that contains ADRs may not be clinically useful, a large volume of these data can expose serious or unknown consequences.

In this chapter, to address the research question **RQ3**, we introduce different Neural Network architectures that classify unstructured documents to whether they contain ADR content. We show that even without engineered features our Neural Networks with word embeddings outperform Maximum-Entropy Classifiers with different weighting schemes for *n*-gram features.

5.1 Related Work

Natural Language Processing (NLP) approaches have been used to detect ADRs and their relations from Electronic Health Records (EHR) (Wang et al. 2009; Friedman 2009) and clinical reports (Aramaki et al. 2010; Gurulingappa and Fluck 2011). Both EHRs and clinical reports have several advantages over plain text or social network data such as they contain more complete records of patients' medical history, treatments, conditions. Leaman, Wojtulewicz, Sullivan, Skariah, Yang, and Gonzalez (2010) are ones of the first to attempt to extract ADRs from text and social networks. They generate a golden data set for DailyStrength[1], a social network where its users share health-related struggles and successes with each other, and lexicons created from UMLS Methathesaurus[2], SIDER (Kuhn et al. 2010) and The Canada Drug Adverse Reaction Database[3]. Their data set contains a total of $6,890$ comment records. Their approach is rather straight forward, which uses direct matches of terms in their built lexicons against terms tokenised from the comments. They report a precision of 78.3%, a recall of 69.9% and an F-score of 73.9%. Further work that focuses on

[1] http://www.dailystrength.org/

[2] National Library of Medicine. 2008. UMLS Knowledge Sources.

[3] http://www.hc-sc.gc.ca/dhp-mps/medeff/index-eng.php

exploring existing or expanded lexicons to find ADRs can be found at (Benton et al. 2011; Harpaz et al. 2012; Gurulingappa et al. 2012; Yates and Goharian 2013; Liu and Chen). Lexicon-based approaches are limited in the number of drugs studied or the number of target ADRs. Nikfarjam and Gonzalez (2011) introduce a rule-based approach on the same DailyStrength data set. Though it does not perform as well as the lexicon-based approach, it can detect expressions not included in the lexicons.

With the emergence of annotated data, there have been more machine-learning based approaches to ADRs detection. Gurulingappa and Fluck (2011) use Decision Trees, Maximum Entropy and SVMs with many engineered features. They obtain an F-score of 77% for ADR class with ADE data set. Sarker and Gonzalez (2015) use SVMs with different feature sets from combined data sets (ADE, Twitter and DailyStrength). They observe that combining Twitter with ADE data sets or DailyStrength with Twitter data sets helps improving their performances. Nikfarjam et al. (2015) use Conditional Random Fields to simultaneously detect ADRs and the condition for which the patient is taking the drug. In addition to traditional features, they introduce embedding clusters features trained with word2vec and k-means clustering. Rastegar-Mojarad et al. (2016) and Zhang et al. (2016) use ensemble models that combine decision trees (Random Forest) or different classifiers with various features.

Overall, approaches to ADR detection have been limited with shallow models and heavily engineered features. There has been a lack of an end-to-end approach that relies on redundancy of unannotated and annotated data.

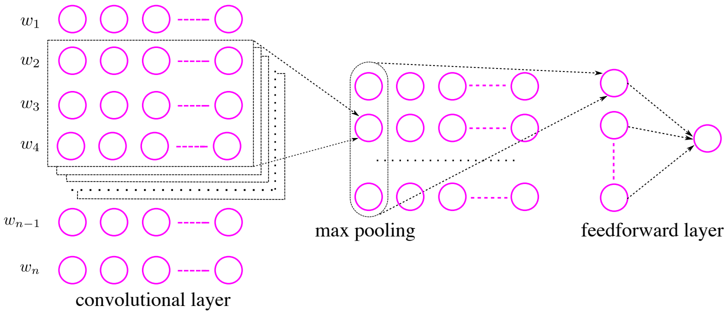


Figure 5.1: Convolutional Neural Network (CNN)

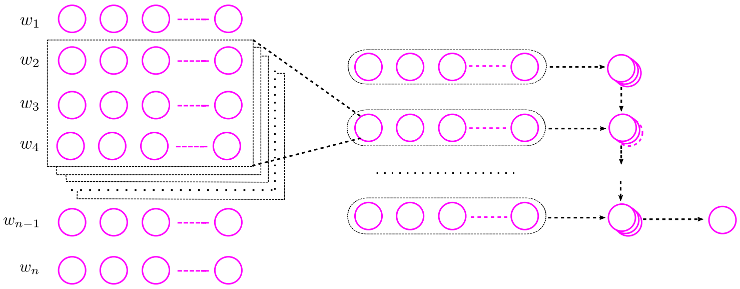


Figure 5.2: Recurrent Convolutional Neural Network (RCNN)

5.2 Deep Neural Network architectures

In this section, we introduce a number of Neural Network architectures and propose two new models, Convolutional Recurrent Neural Networks (CRNN) and Convolutional Neural Network with Attentions (CNNA)[4].

[4] Source code is available at `https://github.com/trunghlt/AdverseDrugReaction`

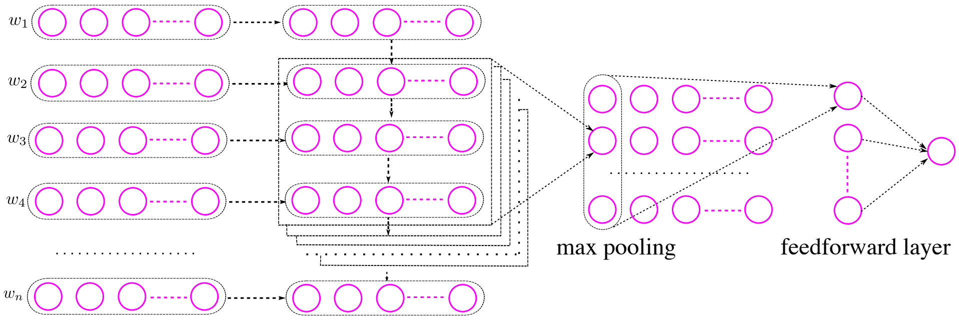


Figure 5.3: Convolutional Recurrent Neural Network (CRNN)

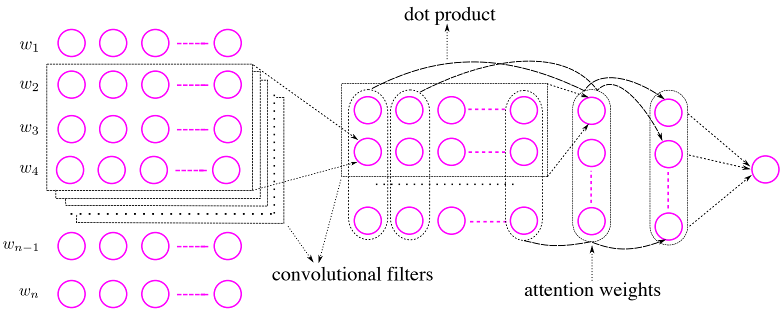


Figure 5.4: Convolutional Neural Network with Attention (CNNA)

5.2.1 Convolutional Neural Network (CNN)

Deep Convolutional Neural Networks (CNN)s are recently extensively used in many computer vision (Alex Krizhevsky et al. 2012; Szegedy et al. 2014; Simonyan and Zisserman 2014; He et al. 2015). In NLP, CNNs (Figure 5.1) were previously used successfully in sentence classification and sentiment analysis (Collobert et al. 2011; Kim 2014; Zhou et al. 2015). The network starts with a convolutional layer with Rectified Linear Units (RLUs) (Glorot et al. 2011). A RLU takes an input and returns the original input if it is larger than 0, otherwise, it returns 0. The convolutional filters normally have the same width as the word vectors, thus, produce feature maps with only 1 column. The network is then stacked by a max pooling layer that picks the maximum element from each column. The last layer is a feedforward layer to an output layer with either sigmoid (Equation 5.3) or softmax (Equation 5.4) activation depending on whether the classification is binary or multinomial. The mathematical formulations for different layers of the CNN are:

$$l^k_{1_{i1}} = \max\{(W^k_1 * X)_{i1}, 0\}, \tag{5.1}$$

$$l_{2_k} = \max_i\{l^k_{1_{i1}}\}. \tag{5.2}$$

If it is binary classification, we set

$$l_3 = \frac{1}{1 + \exp(-W_3^\top l_2 - b_3)}, \tag{5.3}$$

or, otherwise, if it is multinomial classification

$$l_{3_i} = \frac{\exp(W_3^\top l_2 + b_3)_i}{\sum_j \exp(W_3^\top l_2 + b_3)_j}. \tag{5.4}$$

Here, $X \in \mathbb{R}^{d \times s}$ is the input matrix after the projection, $d \in \mathbb{N}$ is the document length, $s \in \mathbb{N}$ is the word vector length, $*$ denotes convolution, $W_1^i \in \mathbb{R}^{h \times e}$, $W_3 \in \mathbb{R}^{k \times 1}$ are the Neural Network weights, $b_3 \in \mathbb{R}$ is the bias term, $h \in \mathbb{N}$ is the convolutional filter height and $k \in \mathbb{M}$ is the number of convolutional filters.

5.2.2 Recurrent Convolutional Neural Network (RCNN)

Another architecture that has achieved comparable results in sentence classification task is Recurrent Convolutional Neural Network (RCNN) (Zhou et al. 2015). The RCNN (Figure 5.2) also starts with a convolutional layer like the CNN but followed by a recurrent layer rather than a max pooling layer. The convolutional filters have the same width as the embedding and are applied in the manner that the outputs have the same number of rows as the input. We also use the Rectified Linear function as the activation function for the convolutional layer. For the recurrent layer, at time step t, the recurrent node takes the input from the outputs produced by all the convolutional filters at row t and previous values at time step $t - 1$. For activation, we use Gated Recurrent Units (Cho et al. 2014). Finally the nodes at the last time step are fully connected to a single node with a *sigmoid* activation to produce binary classification:

$$l_{1_{t1}}^{k} = \max\{(W_1^k * X)_{t1}, 0\} \tag{5.5}$$

$$l_{2_{t_j}} = \left(l_{1_{t1}}^{*}\right) \tag{5.6}$$

$$l_3 = \frac{1}{1 + \exp(-W_3^{\top} l_{2_d} - b_3),} \tag{5.7}$$

where $(X)\colon \mathbb{R}^{d\times k} \to \mathbb{R}^{d\times r}$ denotes Gated Recurrent Unit (GRU) with input X, $r \in \mathbb{R}$ is the size of the output of the RNN and $t \in \mathbb{R}$ denotes a time step that is equivalent to the order of the window that produces the values from convolutional filters.

GRUs are recurrent units which have additional gating units. The gating units modulate the flow of information inside the unit. The activation h_j^i of a GRU at time t is a linear interpolation between previous activations:

$$h_t^j = (1 - z_t^j)h_{t-1}^j + z_t^j \tilde{h}_t^j$$

Here z_t^j acts as a gate which decides how much the unit updates its content and it is computed by $z_t^j = \sigma(W_z x_t + U_z h_{t-1})^j$, while $\tilde{h}_t^j$ is a candidate activation, computed similarly to traditional recurrent unit, $\tilde{h}_t^j = \tanh\big(Wx_t + U(r_t \odot h_{t-1})\big)^j$, where r_t is a reset gate and $\odot$ is an element-wise multiplication. These reset gates can be computed similarly to the update gate $r_t^j = \sigma(W_r x_t + U_r h_{t-1})^j$.

The idea behind gated flows is to enable information further in the past to be propagated to the current unit with fewer time steps. With fewer time steps, the error gradient is passed by back-propagation more efficiently due to the propagated gradient is less prone to vanishing or exploding. Cho et al. (2014) show that GRUs have better performance than traditional tanh and comparable performance to LSTM units.

5.2.3 Convolutional Recurrent Neural Network (CRNN)

Inspired by RCNN, we introduce a new architecture called Convolutional Recurrent Neural Network (Figure 5.3) that stacks a convolutional layer on top of a recurrent layer, which is opposite to a RCNN. The intuition behind this is that the recurrent layer can capture the global contexts before information passed to the convolutional layer. The convolution and max-pooling layers replace the traditional average over hidden features or only hidden features at the last word in the sentence. We use GRUs for the recurrent layers and RLUs for the convolutional layer:

$$l_{1_i} = (X_{i*}), \tag{5.8}$$

$$l_{2^k_{i1}} = \max\{(W_2^k * l_1)_{i1}, 0\}, \tag{5.9}$$

$$l_3 = \frac{1}{1 + \exp(-W_3^\top l_2 - b_3)}. \tag{5.10}$$

5.2.4 Convolutional Neural Network with Attention (CNNA)

Inspired by the works from (Bahdanau et al. 2014; Hermann et al. 2015; Rush et al. 2015; Rocktäschel et al. 2016; Yang et al. 2016) which use the attention mechanism where the generation of outputs at each consecutive time step is conditioned on different subsets of the input, we introduce a new architecture built on top of the CNN with additional attention mechanism (Figure 5.4). The addition is one-filter convolutional layer on top of the direct outputs from the first convolutional layer. The outputs of this convolutional layer are normalised with *softmax* function so that they can have a sum of 1, which we call *attention weights*. These *attention weights* are then multiplied with the outputs from the first convolution (*dot product*). The outputs of this dot product are forward connected to a perceptron for binary classification.

The advantage of introducing the attention mechanism is that we can use these attention weights to extract words that the model mainly uses for the prediction. In practice, we found it very interesting and helpful to see which words are more weighted in the model's decisions (see Figure 5.5 in Section 5.4).

Even though getting more popular, attention mechanism has been mostly applied with Recurrent Neural Networks (Bahdanau et al. 2014; Hermann et al. 2015; Rush et al. 2015; Rocktäschel et al. 2016; Yang et al. 2016). There are recently some works that incorporate attention mechanism with CNNs (Yin et al. 2016; Yin et al. 2016). In Yin et al. (2016), attention weights are computed differently by taking the dot product between the representation of the input query and the sentences in question-answer tasks. In Yin et al. (2016), even though called attention, the attention layers behave more like feature maps than traditional attention weights (multiplied with features) and are computed by matching two feature maps.

5.3 Experiments

We examine our Neural Networks with ADR content classification from two separate data sets.

5.3.1 Data sets

We use two data sets for the evaluation of various Neural Network architectures. The first one is a Twitter data set (Sarker et al. 2016) published for a shared task in Pacific Symposium on Biocomputing, Hawaii, 2016. The tweets associated with the data were collected using generic and brand names of the drugs, and also their possible phonetic misspellings. The tweets were annotated for presence of ADRs. In the shared task, 70% $(7,575)$ of the original data set is shared for training and the rest of the data is used for evaluation. Owing to Twitter's data terms and conditions, only the tweet ids are contained in the original file. At the time of this experiment, as many tweets are no longer accessible we could download only 5, 108 tweets (out of 10, 822 original tweets) with 557 tweets with ADR descriptions. Due to the difference in the size of the experimental data set, we can not compare our results directly with the previously reported baselines. Thus we reuse the codes published by (Zhang et al. 2016) that perform classification with the various algorithms (see Section 5.3.2 for further details).

The second dataset, the ADE (adverse drug effect) corpus, was created by (Gurulingappa et al. 2012) by sampling from MEDLINE case reports[5]. Each case report provides important information about symptoms, signs, diagnosis, treatment and follow-up of individual patients. The ADE corpus contains 2, 972 documents with 20, 967 sentences. Out of these, 4, 272 sentences are annotated with names and relationships between drugs, adverse effects and dosages.

For both data sets, we use 10-stratified-fold cross-validation and report precision, recall and F-scores of various methods.

5.3.2 Baselines

For the Twitter data set, it was reported from the shared task that both the best (Rastegar-Mojarad et al. 2016) and the second best (Zhang et al. 2016) approaches are classifiers with engineered features. In order to directly compare our results with the existing approaches, we have reimplemented these classifiers based on the published code by (Zhang et al. 2016) including term-matching classifier based on

[5] `https://www.nlm.nih.gov/bsd/indexing/training/PUB_050.htm`

an ADR lexicon, maximum entropy with n-grams and tf-idf weightings or NB log-count ratio, and maximum entropy with word embeddings. We describe each of these methods below:

- *Term-matching based on an ADR lexicon* (**TM**). An existing ADR lexicon[6] is directly used for ADR detection. The lexicon contains 13,699 terms describing side effects from COSTART, SIDER, CHV and DIEGO_Lab. A document is classified as positive if it contains a term from the lexicon.

- *Maximum-Entropy classifier with n-grams and tf-idf weightings* (**ME-TFIDF**). For a document $d \in \mathcal{D}$, an n-gram i has a weight of

  $$F_i(d) = \begin{cases} \big(1 + \log(n_i(d))\big) \times \log\big(1 + \frac{|\mathcal{D}|+1}{|\{d' \in \mathcal{D} | n_i(d') > 0\}|+1}\big) & \text{if } n_i(d) > 0 \\ 0 & \text{otherwise,} \end{cases}$$

  where $n_i(d)$ is the number of times a term i appears in document d.

- *Maximum-Entropy classifier with n-grams and NB log-count ratio* (**ME-NBLCR**). Each n-gram i has a weight of

  $$f_i = \begin{cases} \log\big(\frac{1+\sum_{d:y(d)=1} n_i(d)}{\sum_{i' \in \mathcal{V}}(1+\sum_{d:y(d)=1} n_{i' \in \mathcal{V}}(d))} \times \frac{\sum_{i' \in \mathcal{V}}(1+\sum_{d:y(d)=-1} n_{i'}(d))}{1+\sum_{d:y(d)=-1} n_i(d)}\big) & \text{if } n_i(d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

  where $\mathcal{V}$ is a set of all n-grams and $y(d) \in \{1, -1\}$ is the true label of each document.

- *Maximum-Entropy classifier with mean word embeddings* (**ME-WE**). This method simply uses the average of embeddings of words in each document as their input into a maximum-entropy classifier.

[6] `http://diego.asu.edu/downloads/publications/ADRMine/ADR_lexicon.tsv`

For the ADE data set, the best performance published is 0.81 in F-score using SVMs trained from a rich set of features including n-grams, UMLS semantic types and concept IDs, synset expansions, polarity indicator features, ADR lexicon matches, and topics, etc. (Sarker and Gonzalez 2015). However, since our ME-NBLCR outperforms SVMs on ADE, we don't report the results using SVMs here.

5.3.3 Training of Neural Networks

In all the described Neural Network architectures in Section 5.2, the training algorithm is Adadelta (Zeiler 2012) with learning rate of 1.0, decay rate (ρ) of 0.95 using the library Keras[7]. The word embeddings initialized with Glove vectors[8] and are trained together with other parameters. For each fold, we split the training data set into training and validating sets. The training stops when there is no performance improvement on the validation set after 5 consecutive epochs. The batch size is set as 50. All convolutional window has a size of 5.

5.4 Results

We compare the precision, recall and F-scores of the positive class (instances labeled as containing the description of adverse drug reactions) of Neural Network architectures with the baselines in Table 5.1. Since both the Twitter and ADE data sets contain imbalanced class distribution, we also report the Area Under the ROC Curve (AUC) results. It can be observed that in general, results on the ADE data set are better than those on the Twitter data set. This is perhaps not surprising since tweets contain a lot of ill-grammatical sentences and short forms. Simply relying on an ADR lexicon

[7] `http://keras.io/`

[8] `http://nlp.stanford.edu/data/glove.840B.300d.zip`

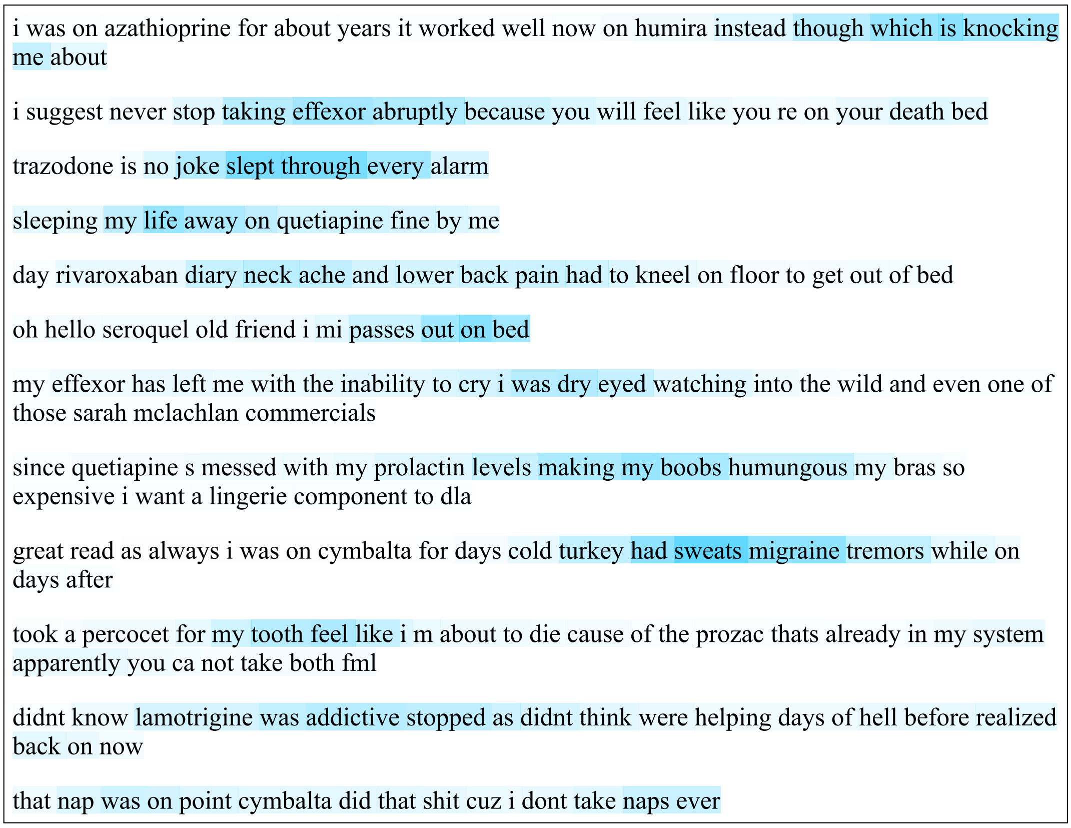
Method	Twitter Dataset				ADE Dataset			
	Precision	Recall	F1	AUC	Precision	Recall	F1	AUC
TM	0.13	0.89	0.23	0.59	0.30	0.99	0.46	0.53
ME-TFIDF	0.33	0.70	0.45	0.85	0.74	0.86	0.80	0.94
ME-NBLCR	0.79	0.14	0.23	0.83	0.91	0.79	0.84	0.95
ME-WE	0.27	0.73	0.40	0.82	0.48	0.70	0.57	0.76
CNN	0.47	0.57	**0.51**†	**0.88**†	0.85	0.89	**0.87**†	**0.97**†
CRNN	0.49	0.55	**0.51**†	0.87†	0.82	0.86	0.84	0.96
RCNN	0.43	0.59	0.49†	0.87†	0.81	0.89	0.83	0.92
CNNA	0.40	0.66	0.49†	0.87†	0.82	0.84	0.83	0.95

Table 5.1: Adverse drug reaction classification results on the Twitter and ADE datasets. Results with † are statistically significantly different from best results from the baselines (top rows). All tests are done using student-t tests with p-values = 0.05.

for the detection of ADRs from text gives the worst results. Among the baselines, the best performing method is ME-TFIDF on the Twitter data set where an F-score of 0.45 and an AUC value of 0.85 are obtained. But on the ADE data set with more formal language, ME-NBLCR gives superior results compared to ME-TFIDF with an F-score of 0.84 and an AUC value of 0.95. Training MaxEnt from aggregated word embeddings (ME-WE) outperforms the term matching method (TM), but performs worse than both ME-TFIDF and ME-NBLCR.

All the Neural Network architectures perform similarly on the Twitter data set and they improve upon the best baseline method ME-TFIDF by 4-6% in F-score and 2-3% in AUC. On the ADE data set, CNN outperforms other Neural Network architectures and its performance gain over ME-NBLCR is 7% in F-score and 3% in AUC. Overall, CNN gives the best results although CRNN and CNNA are quite close to CNN in terms of AUC values. It is not very straightforward to explain why CNNs are better than the recurrent architectures in our experiments. Our hypothesis is that as ADR descriptions are composed of short fragments of texts, convolutions with small windows are enough to capture necessary information for ADR classification.

Since CNNA assigns a weight to each word when making classification decision, we show in Figure 5.5 a visualisation of attention weights of sampled tweets from the



i was on azathioprine for about years it worked well now on humira instead though which is knocking me about

i suggest never stop taking effexor abruptly because you will feel like you re on your death bed

trazodone is no joke slept through every alarm

sleeping my life away on quetiapine fine by me

day rivaroxaban diary neck ache and lower back pain had to kneel on floor to get out of bed

oh hello seroquel old friend i mi passes out on bed

my effexor has left me with the inability to cry i was dry eyed watching into the wild and even one of those sarah mclachlan commercials

since quetiapine s messed with my prolactin levels making my boobs humungous my bras so expensive i want a lingerie component to dla

great read as always i was on cymbalta for days cold turkey had sweats migraine tremors while on days after

took a percocet for my tooth feel like i m about to die cause of the prozac thats already in my system apparently you ca not take both fml

didnt know lamotrigine was addictive stopped as didnt think were helping days of hell before realized back on now

that nap was on point cymbalta did that shit cuz i dont take naps ever

Figure 5.5: Sampled tweets with weighted highlights from attention weights.

Twitter data set. Words with higher attention weights are highlighted with darker blue colour. We can observe that most of the highlighted words are indeed related to descriptions of adverse drug effects. For example, “neck ache” and “lower back pain” in the fifth tweet and “dry eyed” in the seventh tweet. The above results suggest that although CNNA gives slightly worse results compared to CNN for ADR classification, it presents results in a more interpretable form and could be potentially used for the extraction of word sub-sequences actually describing ADRs. As such, CNNA would be a better candidate than CNN for more fine-grained ADR extraction.

5.5 Conclusion

This chapter has explored different Neural Network (NN) architectures for ADR classification. In particular, it has proposed two new Neural Network models, Convolutional Recurrent Neural Network (CRNN) and Convolutional Neural Network with Attention (CNNA). Experimental results show that all the NN architectures outperform the baseline Maximum Entropy classifiers trained from tf-idf features of n-grams with different weighting strategies considerably on both the Twitter and the ADE data sets. Among NN architectures, no significant differences were observed on the Twitter data set. But CNN appears to perform better compared to other more complex CNN variants on the ADE dataset. Nevertheless, CNNA allows the visualisation of attention weights of words when making classification decisions and hence is more appropriate for the extraction of word sub-sequences describing ADRs.

Chapter 6

Conclusion

Since the work of Collobert et al. (2011), Deep Neural Networks and pre-trained representations have gained tremendous attention from the community. They have indeed successfully out-performed other methods in many of NLP tasks without the need of heavily feature engineering from domain experts. In many of recent works, Decision Trees, SVMs, Maximum Entropy Classifiers, HMMs, CRFs are now replaced by Convolutional Neural Networks, Recurrent Neural Networks or other variants. Features such as Part-Of-Speech tags, designated lexicons, thesauruses even though being useful in some cases are gradually replaced partially or completely by pre-trained word, phrase and sentence representations. Given their effectiveness, I have investigated and introduced new methods using Deep Neural Networks in learning language representations using available unlabelled data and in text classification. The new methods are compared to state-of-the-art methods and a common baseline using Maximum Entropy classifier with tf-idf features. Specifically three main new ideas have been presented in this thesis. First I explore how to learn better word representations with sequential models to address the research question of **how sequential models can be trained to learn word representations from unlabelled data (RQ1).** Second I conduct experiment with Convolutional Restricted

Boltzmann Machines (CRBMs) to investigate **how we can train CRBMs to reconstruct these embedded sentences and utilise higher-level features from trained hidden layers for downstream text classification (RQ2)**. Finally I apply pre-trained word presentations and Deep Neural Networks to extract content of Adverse Drug Reactions from medical records and social media data, addressing **how we can utilise pre-trained word representations and Deep Neural Networks to improve Adverse Drug Reaction classification (RQ3)**.

In utilising sequential models to learn word representations, when trained with a corpus with an appropriate size, I have found that Recurrent Neural Networks with LSTM yield word representations with better similarity and relatedness when compared to ones learned from state-of-the-art bag-of-word approaches like CBOW or Skip-gram. These word representations when used in simple text classification tasks prove to yield better performances than ones trained by CBOW and Skip-gram. However in order to achieve these results, one should be careful to select an architecture with statefulness where the states of the previously trained sentences should be carried forward to compute the state of the next sentence. Given this setting helps our models to train better word representations, ones might want to try the same setting in other tasks with other models. For example, current language models are often trained conditioned on only one immediate previous sentence and training data sets come in form of shuffled pairs of sentences (or triplets of sentences in case of predicting next and previous sentences). Other models that might similarly benefit from the same setting are machine translation models, text-to-speech or speech-to-text models. Word representations trained with these methods, however, do not perform as well as bag-of-word methods in one of the popular evaluations of this kind which is the analogy task. We hypothesise that as our methods learn word representations using more complex models, it requires more sophisticated transformations to infer different word relationships. Another caveat of this method is it is more computationally expensive than bag-of-word approaches which makes them hard in practice to train with corpora with billions of words. In the future, I would like to explore how to train the models more efficiently and better infer word analogies from these word representations and what kind of transformations are required. For training speed, there are many strategies including trying other architectures that are easier to be parallelised, e.g. deep convolutional models or transformer networks (Vaswani et al. 2017), training on faster hardware like TPU[1] or make it distributed on multiple GPUs or TPUs. It is also interesting to evaluate the trained representations in different tasks such as text classification, sentence similarity or paraphrase detection.

In utilising CRBMs to learn higher-level features from unlabelled data, from the best of my knowledge, I was first to apply CRBMs to reconstruct sentences using pre-trained word representations and learn phrase-level features from textual data as described in Chapter 4. These features when combined with other word-level features such as part-of-speech tags or tf-idf improve over word-level features alone in subjectivity and sentiment classification when trained with linear SVMs. Recently discriminative models have achieved state-of-the-art results in different text classification thank to vast amount of training data and pre-trained word representations (that I have shown in Chapter 5). In cases where labelled data are sparse, there is little improvement over bag-of-word approaches and these improvements usually come from selecting quality word representations. By acquiring good representations

[1] `https://en.wikipedia.org/wiki/Tensor_processing_unit`

of phrases or n-grams from unlabelled data, one might be able to achieve similar gain as with word representations. These representations can also be used to compute similarity between n-grams or phrases that would be useful in Information Retrieval, Natural Language Generation or Natural Language Understanding. Similar models can also be applied at higher levels than discourses such as sentences, paragraphs or documents. It is also interesting to investigate the effect of training with Fast Persistent Contrastive Divergence, Rectified Linear Hidden Units, Dropout, MaxOut on the current architectures or to compare their performances to related unsupervising architectures such as Auto-encoders.

In utilising pre-trained word representations and Deep Neural Networks in the application of Adverse Drug Reaction classification, I was first to investigate different DNN architectures with pre-trained word representations in Adverse Drug Reaction classification and achieved state-of-the-art results for Twitter and ADE data sets. Additionally we introduced a simple Convolutional Neural Network with Attention that can assign importance of tokens from the input text in classification results. This mechanism provides an intuitive explanation of the model and provides insights on its strengths and weaknesses. An advantage of this architecture is it is not limited within ADR classification and can be used in other domains. In the future, I would like to apply models in Chapter 3 and 4 into learning word and phrase representations from medical data sets. These new word representations can then replace or be combined with the existing word representations into our DNNs. Since our work, DNNs with pre-trained word representations in related tasks such as Named Entity Recognition for Clinical Notes or Multi-Task Pharmacovigilance Mining for Social Media Posts.

Overall I have introduced new methodologies and fulfilled the research goals to learn better language representations from unlabelled data with stateful LSTMs, Convolutional Restricted Boltzmann Machines and to utilise language representations with Deep Neural Networks to achieve better text classification than the baseline method that does not use representation pre-training (ME-TFIDF) as shown in sentiment and ADR classification. Additionally outputs from my newly introduced Neural Network with Attentions can highlight salient words and, hence, be used to explain the model's classification decisions. Neural Networks with pre-trained representations have achieved relative successes in NLP but computers are still far from humans in learning from languages or understanding them. For example while humans can learn to execute different tasks with little instruction, computers need large data sets of examples. Humans can learn from different sources of data and apply the learned knowledge efficiently across tasks while computers are often trained separately for different tasks with different models. Perhaps one can train a unified model of language modelling for Question & Answer, dialog systems, text classification or information extraction (generating answers for appropriate questions) by utilising vast amount of unlabelled data.

The author would like to express his gratitude to Professor Stefan Rüger, Dr Yulan He and Dr Alistair Willis for their continued advice and encouragement. To my beloved wife, Phuong Ho, I can't say enough how much I owe you and thank so much for her long endurance of my frequent absence and lack of my attention while working on my thesis. I would also like to thank my parents for their continuous encouragement to push my limit. Without all your of support, I would not have been able to complete this thesis.

Appendix A

Notations

$\mathbb{R}$	Set of real numbers
$\mathbb{R}^n$	Set of n-dimensional real-valued vectors
$\mathbb{R}^{n \times m}$	Set of $n \times m$ real-valued matrices
$[a, b]$	Closed interval between a and b
(a, b)	Open interval between a and b
$\{a, b, c\}$	Set containing elements a, b and c
$\mathbb{N}$	Set of natural numberes
$\log$	Logarithm with base e
$\log_a$	Logarithm with base a
$\mathcal{S}$	An arbitrary set
$\lvert\mathcal{S}\rvert$	Number of elements in set $\mathcal{S}$
$s \in \mathcal{S}$	An element in set $\mathcal{S}$
$\mathcal{X}$	Input space
$\mathcal{Y}$	Output space
$\mathbb{H}$	Feature space
$\langle \cdot, \cdot \rangle$	Inner product in feature space
v	An arbitrary vector (can be column or row vector depending on its context)
$\mathbf{1}$	Vector of all ones
v_i	ith component of v
$\lVert v \rVert$	L_2 norm of v
$\lVert v \rVert_p$	L_p norm of v
$u \circ v$	Hadamard or entry-wise product of vectors u and v
$f \circ g$	Composition of functions f and g
$f * g$	Convolution of f and g
M	An arbitrary matrix
$\lVert M \rVert_2$	Spectral norm of M
$\lVert M \rVert_F$	Frobenius norm of M
$M^\top$	Transpose of M
M_i	Row vector i of M
$M_{i,j}$	Element at row i, column j of matrix M
$\mathbf{I}$	Identity matrix
$K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$	Kernel function over $\mathcal{X}$

Table A.1: Summary of notations

Bibliography

Agirre, E., E. Alfonseca, K. Hall, J. Kravalova, M. Paşca, and A. Soroa (2009). A Study on Similarity and Relatedness using Distributional and WordNet-based Approaches. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, pp. 19.

Agirre, E., C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, R. Mihalcea, G. Rigau, and J. Wiebe (2014). SemEval-2014 Task 10: Multilingual Semantic Textual Similarity. *International Workshop on Semantic Evaluation* (SemEval), 81–91.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012). Imagenet Classification with Deep Convolutional Neural Networks. *Neural Information Processing Systems (NIPS)*, 1097–1105.

Aramaki, E., Y. Miura, M. Tonoike, T. Ohkuma, H. Masuichi, K. Waki, and K. Ohe (2010). Extraction of Adverse Drug Effects from Clinical Records. *Studies in Health Technology and Informatics 160*(PART 1), 739–743.

Arora, S., Y. Liang, and T. Ma (2017). A Simple but Tough to Beat Baseline for Sentence Embeddings. *International Conference on Learning Representations (ICLR)*, 1–14.

Bahdanau, D., K. Cho, and Y. Bengio (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *International Conference on Learning Representations (ICLR)*, 1–15.

Baldi, P. and K. Hornik (1989, January). Neural Networks and Principal Component Analysis: Learning from Examples Without Local Minima. *Neural Networks. 2*(1), 53–58.

Beineke, P., T. Hastie, C. Manning, and S. Vaithyanathan (2004). Exploring Sentiment Summarization. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text Theories and Applications*, Volume 07, pp. 12–15. AAAI Press.

Bengio, Y., R. Ducharme, P. Vincent, and C. Jauvin (2003). A Neural Probabilistic Language Model. *Journal of Machine Learning Research 3*, 1137–1155.

Benton, A., L. Ungar, S. Hill, S. Hennessy, J. Mao, A. Chung, C. E. Leonard, and J. H. Holmes (2011). Identifying Potential Adverse Effects using the Web: A New Approach to Medical Hypothesis Generation. *Journal of Biomedical Informatics 44*(6), 989–996.

Berger, A. L., S. A. D. Pietra, and V. J. D. Pietra (1996). A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics 22*, 39–71.

Blitzer, J., K. Q. Weinberger, L. K. Saul, and F. C. N. Pereira (2004). Hierarchical Distributed Representations for Statistical Language Modeling. In *Neural Information Processing Systems (NIPS)*, Volume volume17, pp. 185–192.

Botha, J. a. and P. Blunsom (2014). Compositional Morphology for Word Representations and Language Modelling. In *International Conference on Machine Learning (ICML)*, Beijing, China.

Brill, E. (1995). Transformation-Based Error-Driven Learning and Natural Language Processing : A Case Study in Part-of-Speech Tagging. In *Computational Linguistics*.

Brown, P. (1990). Class-based N-gram Models of Natural Language. In *The IBM NLP Workshop*. Paris, France.

Bruni, E., G. Boleda, M. Baroni, and N.-K. Tran (2012). Distributional Semantics in Technicolor. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics 1*(July), 136–145.

Bullinaria, J. A. and J. P. Levy (2007). Extracting Semantic Representations from Word Co-occurrence Statistics: A Computational Study. *Behavior Research Methods 39*(3), 510–526.

Chen, M. (2017). Efficient Vector Representation for Documents through Corruption. In *International Conference of Learning Representations (ICLR)*, pp. 1–13.

Chen, S. F. and J. Goodman (1996). An Empirical Study of Smoothing Techniques for Language Modeling. In *ACL-96*, Santa Cruz, CA, pp. 310–318. Association for Computational Linguistics (ACL).

Chiu, C., T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani (2018, April). State-of-the-Art Speech Recognition with Sequence-to-Sequence

Models. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4774–4778.

Cho, K., B. van Merrienboer, C. Gulcehre, F. Bougares, D. Bahdanau, H. Schwenk, Y. Bengio, Kyunghyun Cho, Bart van Merrie nboer, Caglar Gulcehre, Fethi Bougares Holger Schwenk, and Yoshua Bengio (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *Empiricial Methods in Natural Language Processing (EMNLP).*

Church, K. W. and P. Hanks (1989). Word association norms, mutual information, and lexicography. *Proceedings of the 27th annual meeting on Association for Computational Linguistics - 16*(1), 76–83.

Clark, S., B. Coecke, and M. Sadrzadeh (2008). A Compositional Distributional Model of Meaning. *Proceedings of the Second Quantum Interaction Symposium (QI-2008)* (Schuetze 1998), 133–140.

Clark, S. and S. Pulman (2007). Combining Symbolic and Distributional Models of Meaning. *Proceedings of the AAAI Spring Symposium on Quantum Interaction*, 52–55.

Collobert, R. and J. Weston (2008). A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. In *International Conference in Machine Learning (ICML)*, Volume 20.

Collobert, R., J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa (2011). Natural Language Processing (Almost) from Scratch. *Journal of Machine Learning Research 12*, 2493–2537.

Dahl, G. E., R. P. Adams, H. Larochelle, R. P. Adams, and H. Larochelle (2012). Training Restricted Boltzmann Machines on Word Observations. In *International Conference in Machine Learning (ICML).*

Dave, K., S. Lawrence, and D. M. Pennock (2003). Mining the Peanut Gallery: Opinion Extraction and Semantic Classification of Product Reviews. In *Proceedings of the 12th International Conference on World Wide Web*, New York, NY, USA, pp. 519–528. ACM.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science 41*(6), 391–407.

Devlin, J., M.-W. Chang, K. Lee, and K. Toutanova (2019, jun). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *North American Chapter of the Association for Computational Linguistics (NAACL)*, Minneapolis, Minnesota, pp. 4171–4186. Association for Computational Linguistics.

Dolan, B., C. Quirk, and C. Brockett (2004). Unsupervised Construction of Large Paraphrase Corpora. In *Proceedings of the 20th international conference on Computational Linguistics - COLING '04*, pp. 350–es.

Duchi, J., E. Hazan, and Y. Singer (2011). Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *Journal of Machine Learning Research 12*, 2121–2159.

Elman, J. L. (1990). Finding Structure in Time. *Cognitive Science 14*(2), 179–211.

Elman, J. L. and D. Zipser (1988). Learning the Hidden Structure of Speech. *The Journal of the Acoustical Society of America*.

Finkelstein, L., E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin (2002). Placing Search in Context: the Concept Revisited. *ACM Transactions on Information Systems 20*(1), 116–131.

Friedman, C. (2009). Discovering Novel Adverse Drug Events Using Natural Language Processing and Mining of the Electronic Health Record. In *Proceedings of the 12th Conference on Artificial Intelligence in Medicine, AIME '09*, pp. 1–5.

Fukushima, K. (1980). Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*.

Gale, W. A. and G. Sampson (1995). Good-Turing Frequency Estimation Without Tears. *Journal of Quantitative Linguistics*.

Gan, Z., Y. Pu, R. Henao, C. Li, X. He, and L. Carin (2017, sep). Learning Generic Sentence Representations Using Convolutional Neural Networks. In *Empirical Methods in Natural Language Processing (EMNLP)*, Copenhagen, Denmark, pp. 2380–2390. Association for Computational Linguistics.

Gantz, J. and D. Reinsel (2011). Extracting Value from Chaos. Technical report.

Gers, F. and J. Schmidhuber (2000). Recurrent Nets That Time and Count. In *International Joint Conference on Neural Networks*, pp. 189–194 vol.3.

Gers, F. A., J. Schmidhuber, and F. Cummins (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation 12*(10), 2451–2471.

Ginn, R., P. Pimpalkhute, A. Nikfarjam, A. Patki, K. Oconnor, A. Sarker, K. Smith, and G. Gonzalez (2014). Mining Twitter for Adverse Drug Reaction Mentions : A Corpus and Classification Benchmark. *Workshop on Building and Evaluating Resources for Health and Biomedical Text Processing (BioTxtM)* (1).

Glorot, X., A. Bordes, and Y. Bengio (2011). Deep Sparse Rectifier Neural Networks. In *COLING 2010*, Volume 15, pp. 315–323.

Goodfellow, I. J., D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio (2013). Maxout Networks. In *International Conference on International Conference on Machine Learning (ICML)*. JMLR.org.

Goodman, J. (2001). Classes for Fast Maximum Entropy Training. In *International Conference: Acoustics, Speech, and Signal Processing (ICASSP)*, pp. 561–564. IEEE.

Grefenstette, E. and M. Sadrzadeh (2011). Experimenting with Transitive Verbs in a DisCoCat. *GEMS 2011 Workshop on Geometrical Models of Natural Language Semantics*, 5.

Guevara, E. (2010). A Regression Model of Adjective-Noun Compositionality in Distributional Semantics. *In Proceedings of the 2010 Workshop on Geometrical Models of Natural Language Semantics, ACL 2010, Uppsala, Sweden, 16 July 2010*, 33–37.

Gurulingappa, H. and J. Fluck (2011). Identification of Adverse Drug Event Assertive Sentences in Medical Case Reports. *The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases: Workshop on Knowledge Discovery in Health Care and Medicine*.

Gurulingappa, H., A. Mateen-Rajput, and L. Toldo (2012). Extraction of Potential Adverse Drug Events from Medical Case Reports. *Journal of Biomedical Semantics 3*, 15.

Gurulingappa, H., A. M. Rajput, A. Roberts, J. Fluck, M. Hofmann-Apitius, and L. Toldo (2012). Development of a Benchmark Corpus to Support the Automatic Extraction of Drug-related Adverse Effects from Medical Case Reports. *Journal of Biomedical Informatics 45*(5), 885–892.

Gutmann, M. and A. Hyvärinen (2010). Noise-contrastive Estimation: A New Estimation Principle for Unnormalized Statistical Models. In *Aritificial Intelligence and Statistics*.

Han, H., E. Manavoglu, C. L. Giles, and H. Zha (2003). Rule-based Word Clustering for Text Classification. In *26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, Toronto, Canada, pp. 445–446.

Harpaz, R., W. DuMouchel, N. Shah, D. Madigan, P. Ryan, and C. Friedman (2012, 05). Novel Data-Mining Methodologies for Adverse Drug Event Discovery and Analysis. *Clinical Pharmacology and Therapeutics 91*, 1010–21.

Harris, Z. S. (1954). Distributional Structure. *Word 10*(23), 146–162.

He, K., X. Zhang, S. Ren, and R. Sun (2015). Deep Residual Learning for Image Recognition. *arXiv preprint arXiv:1512.03385*.

Hermann, K. M., T. Kočiský, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom (2015). Teaching Machines to Read and Comprehend. In *Neural Information Processing Systems (NIPS)*.

Hill, F., K. Cho, and A. Korhonen (2016, June). Learning Distributed Representations of Sentences from Unlabelled Data. In *North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, San Diego, California, pp. 1367–1377. Association for Computational Linguistics.

Hill, F., R. Reichart, and A. Korhonen (2015). SimLex-999: Evaluating Semantic Models with (Genuine) Similarity Estimation. In *Computational Linguistics.*

Hinton, G. (1986). Learning Distributed Representations of Concepts. In *the Eighth Annual Conference of the Cognitive Science Society*, Amherst, pp. 1–12. Lawrence Erlbaum Associates.

Hinton, G. (2014). Dropout : A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research (JMLR) 15*, 1929–1958.

Hinton, G. E. (2002). Training Products of Expers by Minimizing Constrastive Divergence. *Neural Computation* (14), 1771–1880.

Hinton, G. E. and R. R. Salakhutdinov (2006, jul). Reducing the Dimensionality of Data with Neural Networks. *Science (New York, N.Y.) 313*(5786), 504–507.

Hochreiter, J. (1991). Untersuchungen zu Dynamischen Neuronalen Netzen. *Master's thesis, Institut fur Informatik, Technische Universitat, Munchen*, 1–71.

Hochreiter, S. and J. Schmidhuber (1997). Long Short-Term Memory. *Neural Computation 9*(8), 1735–1780.

Holyoak, K. J. and J. E. Hummel (2000). The Proper Treatment of Symbols in a Connectionist Architecture. *Cognitive dynamics: Conceptual Change in Humans and Machines*, 229–265.

Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational Abilities. *Proceedings of the National Academy of Sciences 79*(8), 2554–2558.

Hu, M. and B. Liu (2004). Mining and Summarizing Customer Reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177. ACM New York, NY, USA.

Huang, E. H., R. Socher, C. D. Manning, and A. Y. Ng (2012). Improving Word Representations via Global Context and Multiple Word Prototypes. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL): Long Papers-Volume 1*, 873–882.

Hubel, D. H. and T. N. Wiesel (1968). Receptive Fields and Functional Architecture of Monkey Striate Cortex. *The Journal of Physiology*.

Huynh, T., Y. He, and S. Rüger (2015). Learning Higher-Level Features with Convolutional Restricted Boltzmann Machines for Sentiment Analysis. In A. Hanbury, G. Kazai, A. Rauber, and N. Fuhr (Eds.), *Advances in Information Retrieval*, Cham, pp. 447–452. Springer International Publishing.

Huynh, T., Y. He, A. Willis, and S. Rüger (2016, dec). Adverse Drug Reaction Classification With Deep Neural Networks. In *Proceedings of COLING 2016: Technical Papers*, pp. 877–887. COLING.

Jelinek, F. and R. L. Mercer (1980, may). Interpolated Estimation of Markov Source Parameters from Sparse Data. In *In Proceedings of the Workshop on Pattern Recognition in Practice*, Amsterdam, The Netherlands: North-Holland, pp. 381–397.

Jordan, M. I. (1989). Serial Order: A Parallel, Distributed Processing Approach. In J. L. Elman and D. E. Rumelhart (Eds.), *Advances in Connectionist Theory: Speech*. Hillsdale, NJ: Erlbaum.

Kalchbrenner, N., E. Grefenstette, and P. Blunsom (2014). A Convolutional Neural Network for Modelling Sentences. *American Journal of Computational Linguistics*.

Kanerva, P. (2009). Hyperdimensional Computing: An Introduction to Computing in Distributed Representation with High-dimensional Random Vectors. *Cognitive Computation 1*(2), 139–159.

Kim, Y. (2014, aug). Convolutional Neural Networks for Sentence Classification. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 1746–1751.

Kingma, D. P. and J. Ba (2014). Adam: A Method for Stochastic Optimization. In *International Conference of Learning Representations (ICLR)*, San Diego.

Kiros, R., Y. Zhu, R. Salakhutdinov, R. S. Zemel, A. Torralba, R. Urtasun, and S. Fidler (2015). Skip-Thought Vectors. In *Neural Information Processing Systems (NIPS)*, Montreal, Canada.

Kneser, R. and H. Ney (1993). Improved Clustering Techniques for Class-Based Statistical Language Modelling. In *EUROSPEECH-93*, pp. 973–976.

Kneser, R. and H. Ney (1995, may). Improved Backing-off for M-gram Language Modeling. In *In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Volume I, Detroit, Michigan, pp. 181–184.

Ko, Y. and J. Seo (2000). Automatic Text Categorization by Unsupervised Learning. *Proceedings of the 18th conference on Computational linguistics - 1*, 453–459.

Kowsari, K., K. J. Meimandi, M. Heidarysafa, S. Mendu, L. E. Barnes, and D. E. Brown (2019). Text Classification Algorithms: A Survey. *Computing Research Repository abs/1904.08067*.

Kuhn, M., M. Campillos, I. Letunic, L. J. Jensen, and P. Bork (2010). A Side Effect Resource to Capture Phenotypic Effects of Drugs. *Molecular systems biology 6*(343), 343.

Landauer, T. K. and S. T. Dumais (1997). A Solution to Plato's Problem: The Latent Semantic Analysis Theory of Acquisition, Induction, and Representation of Knowledge. *Psychological Review 104*(2), 211–240.

Landauer, T. K., D. Laham, B. Rehder, and M. E. Schreiner (1997). How Well Can Passage Meaning be Derived without Using Word Order ? A Comparison of Latent Semantic Analysis and Humans. *Proceedings of the 19th annual meeting of the Cognitive Science Society*.

Larochelle, H. and Y. Bengio (2008). Classification Using Discriminative Restricted Boltzmann Machines. *Proceedings of the 25th International Conference on Machine Learning (ICML)*, 536–543.

Le, Q. and T. Mikolov (2014). Distributed Representations of Sentences and Documents. *International Conference on Machine Learning (ICML) 32*, 1188–1196.

Le, Q., M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng (2012). Building High-level Features Using Large Scale Unsupervised Learning.

In *International Conference in Machine Learning (ICML).*

Leaman, R., L. Wojtulewicz, R. Sullivan, A. Skariah, J. Yang, and G. Gonzalez (2010, July). Towards Internet-Age Pharmacovigilance: Extracting Adverse Drug Reactions from User Posts in Health-Related Social Networks. In *Proceedings of the 2010 Workshop on Biomedical Natural Language Processing*, Uppsala, Sweden, pp. 117–125. Association for Computational Linguistics.

Lebret, R. and R. Collobert (2014). Word Embeddings through Hellinger PCA. *European Chapter of the Association for Computational Linguistics - EACL*, 482–490.

LeCun, Y. (1989). Generalization and network design strategies. In R. Pfeifer, Z. Schreter, F. Fogelman, and L. Steels (Eds.), *Connectionism in Perspective.* Elsevier.

LeCun, Y., L. Bottou, Y. Bengio, and P. Haffner (1998). Gradient-based Learning Applied to Document Recognition. *Proceedings of the IEEE 86*(11), 2278–2323.

Lee, H., C. Ekanadham, and A. Y. Ng (2008). Sparse Deep Belief Net Model for Visual Area V2. In *Advances in Neural Information Processing Systems (NIPS).*

Lee, H., R. Grosse, R. Ranganath, and A. Y. Ng (2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, 1–8.

Levy, O. and Y. Goldberg (2014). Neural Word Embedding as Implicit Matrix Factorization. *Advances in Neural Information Processing Systems (NIPS)*, 2177–2185.

Levy, O., Y. Goldberg, and I. Dagan (2015). Improving Distributional Similarity with Lessons Learned from Word Embeddings. *Transactions of the Association for Computational Linguistics 3*, 211–225.

Li, X. and R. Dan (2002). Learning Question Classifiers. *International Conference on Computational Linguistics*, 1–7.

Lin, C., Y. He, and R. Everson (2011). Sentence Subjectivity Detection with Weakly-Supervised Learning. In *5th International Joint Conference on Natural Language Processing*, Chiang Mai, Thailand, pp. 1153–1161.

Liu, X. and H. Chen. AZDrugMiner: An Information Extraction System for Mining Patient-reported Adverse Drug Events in Online Patient Forums. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, pp. 134–150.

Lund, K. and C. Burgess (1996). Producing High-dimensional Semantic Spaces from Lexical Co-occurrence. *Behavior Research Methods, Instruments and Computers 28*(2), 203–208.

Luong, M.-t. and C. D. Manning (2013). Better Word Representations with Recursive Neural Networks for Morphology. *Conference on Computational Natural Language Learning (CoNNL 2013).*

Manning, C. D., P. Raghavan, and H. Schütze (2008). *Introduction to Information Retrieval.* New York, NY, USA: Cambridge University Press.

Marelli, M., S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, and R. Zamparelli (2014, May). A SICK Cure for the Evaluation of Compositional Distributional

Semantic Models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, Reykjavik, Iceland, pp. 216–223. European Languages Resources Association (ELRA).

Mikolov, T., K. Chen, G. Corrado, and J. Dean (2013). Distributed Representations of Words and Phrases and their Compositionality. *Neural Information Processing Systems (NIPS)*, 1–9.

Mikolov, T., G. Corrado, K. Chen, and J. Dean (2013). Efficient Estimation of Word Representations in Vector Space. In *Workshop at International Conference of Learning Representations (ICLR).*

Mikolov, T., A. Deoras, D. Povey, L. L. Burget, J. Cernocky, and J. Cernocký (2011, dec). Strategies for Training Large Scale Neural Network Language Models. *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, 196–201.

Mikolov, T., W.-t. Yih, and G. Zweig (2013). Linguistic Regularities in Continuous Space Word Representations. In *The North American Chapter of the Association for Computational Linguistics (NAACL).*

Mikolov, T. and G. Zweig (2012). Context Dependent Recurrent Neural Networkl Language Model. Technical report, Microsoft Research Technical Report.

Mitchell, J. and M. Lapata (2010). Composition in Distributional Models of Semantics. *Cognitive Science 34*(8), 1388–1429.

Mnih, A. and G. Hinton (2007). Three New Graphical Models for Statistical Language Modelling. In *International Conference on Machine Learning (ICML).*

Mnih, A. and K. Kavukcuoglu (2013). Learning Word Embeddings Efficiently with Noise-contrastive Estimation. *Neural Information Processing Systems (NIPS)*, 1–9.

Morin, F. and Y. Bengio (2005). Hierarchical Probabilistic Neural Network Language Model. *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 246–252.

Murray, G. and G. Carenini (2009). Predicting Subjectivity in Multimodal Conversations. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 3 - Volume 3*, EMNLP '09, Stroudsburg, PA, USA, pp. 1348–1357. Association for Computational Linguistics.

Nair, V. and G. E. Hinton (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, Number 3, Haifa, Israel, pp. 807–814.

Nakagawa, T., K. Inui, and S. Kurohashi (2010). Dependency Tree-based Sentiment Classification using CRFs with Hidden Variables. *Computational Linguistics* (June), 786–794.

Neelakantan, A., J. Shankar, A. Passos, and A. Mccallum (2014). Efficient Nonparametric Estimation of Multiple Embeddings per Word in Vector Space. *Empirical Methods in Natural Language Processing (EMNLP).*

Ney, H., U. Essen, and R. Kneser (1994). On Structuring Probabilistic Dependences in Stochastic Language Modelling. *Computer Speech and Language*.

Niesler, T., E. W. D. Whittaker, and P. C. Woodland (1998). Comparison of Part-Of-Speech and Automatically Derived Category-based Language Models for

Speech Recognition. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 177–180. IEEE.

Nikfarjam, A. and G. H. Gonzalez (2011). Pattern Mining for Extraction of mentions of Adverse Drug Reactions from User Comments. *AMIA Annual Symposium Proceedings 2011*, 1019–1026.

Nikfarjam, A., A. Sarker, K. O'Connor, R. Ginn, and G. Gonzalez (2015). Pharmacovigilance from Social Media: Mining Adverse Drug Reaction Mentions Using Sequence Labeling with Word Embedding Cluster Features. *Journal of the American Medical Informatics Association 22*(3), 671–681.

Pagliardini, M., P. Gupta, and M. Jaggi (2018, June). Unsupervised Learning of Sentence Embeddings Using Compositional N-gram Features. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers) (NAACL)*, New Orleans, Louisiana, pp. 528–540. Association for Computational Linguistics.

Pang, B. and L. Lee (2004). A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of the 42Nd Annual Meeting on Association for Computational Linguistics*, ACL '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Pang, B. and L. Lee (2005). Seeing Stars : Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Association for Computational Linguistics (ACL)*, Number 1.

Pawar, P. Y. and S. H. Gawande (2012). A Comparative Study on Different Types of

Approaches to Text Categorization. *International Journal of Machine Learning and Computing*.

Pennington, J., R. Socher, and C. Manning (2014). Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1532–1543.

Pirmohamed, M., S. James, S. Meakin, C. Green, A. K. Scott, T. J. Walley, K. Farrar, B. K. Park, and A. M. Breckenridge (2004). Adverse Drug Reactions as Cause of Admission to Hospital: Prospective Analysis of 18 820 Patients. *British Medical Journal 329*(7456), 15–19.

Plate, T. A. (1995). Holographic Reduced Representations. *IEEE Transactions on Neural Networks 6*(3), 623–641.

Raaijmakers, S., K. Truong, and T. Wilson (2008, October). Multimodal Subjectivity Analysis of Multiparty Conversation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Honolulu, Hawaii, pp. 466–474.

Radinsky, K., E. Agichtein, E. Gabrilovich, and S. Markovitch (2011). A Word at a Time: Computing Word Relatedness using Temporal Semantic Analysis. In *International Conference on World Wide Web (WWW)*, pp. 337–346.

Ranzato, M., Y.-L. Boureau, and Y. LeCun (2008). Sparse Feature Learning for Deep Belief Networks. *Advances in neural information processing systems (NIPS)*.

Ranzato, M., C. Poultney, S. Chopra, and Y. LeCun (2007). Efficient Learning of Sparse Representations with an Energy-Based Model. In *Advances in neural* *information processing systems (NIPS).*

Rastegar-Mojarad, M., R. K. Elayavilli, Y. Yu, and H. Liu (2016). Detecting Signals in Noisy Data - Can Ensemble Classifiers Help Identify Adverse Drug Reaction in Tweets? In *Proceedings of the Social Media Mining Shared Task Workshop at the Pacific Symposium on Biocomputing.*

Riloff, E. and J. Wiebe (2003). Learning Extraction Patterns for Subjective Expressions. *Empirical Methods in Natural Language Processing (EMNLP)*, 105–112.

Rocktäschel, T., E. Grefenstette, K. M. Hermann, T. Kočiský, and P. Blunsom (2016). Reasoning about Entailment with Neural Attention. In *International Conference on Learning Representations (ICLR)*, Number 2015, pp. 1–9.

Rohde, D., L. Gonnerman, and D. Plaut (2006). An Improved Model of Semantic Similarity based on Lexical Co-occurrence. *Communications of the Association for Computing Machinery (ACM).*

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning Representations by Back-propagating Errors. *Nature 323:533.53.*

Rush, A. M., S. Chopra, and J. Weston (2015). A Neural Attention Model for Abstractive Sentence Summarization. *In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)* (September), 379–389.

Russell, S. J. and P. Norvig (2002, December). *Artificial Intelligence: A Modern Approach (2nd Edition).* Prentice Hall.

Salakhutdinov, R. and G. Hinton (2009). Deep Boltzmann Machines. *Artificial Intelligence* (2).

Sarker, A. and G. Gonzalez (2015). Portable Automatic Text Classification for Adverse Drug Reaction Detection via Multi-corpus Training. *Journal of Biomedical Informatics 53*, 196–207.

Sarker, A., A. Nikfarjam, and G. Gonzalez (2016). Social Media Mining Shared Task Workshop. pp. 581–592.

Shen, Y., Z. Lin, C.-W. Huang, and A. Courville (2018). Neural Language Modeling By Jointly Learning Syntax And Lexicon. In *International Conference on Learning Representations (ICLR)*, pp. 1–9.

Silver, D., T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis (2018). A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-play. *Science*.

Simonyan, K. and A. Zisserman (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ImageNet Challenge*, pp. 1–10.

Smolensky, P. (1990). Tensor Product Variable Binding and the Representation of Symbolic Structures in Connectionist Systems. *Artificial Intelligence 46*(1-2), 159–216.

Socher, R., E. Huang, and J. Pennington (2011). Dynamic Pooling and Unfolding Recursive Autoencoders for Paraphrase Detection. *Advances in Neural Information Processing Systems (NIPS)*, 801–809.

Socher, R., J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning (2011). Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions. In

Empirical Methods in Natural Language Processing (EMNLP), Number i, pp. 151–161.

Srivastava, R. K., K. Greff, and J. Schmidhuber (2015). Highway NetWorks: Training Very Deep Networks. *Neural Information Processing Systems (NIPS).*

Sutskever, I., O. Vinyals, and Q. V. Le (2014). Sequence to Sequence Learning with Neural Networks. *Advances in Neural Information Processing Systems (NIPS)*, 3104–3112.

Szegedy, C., W. Liu, Y. Jia, and P. Sermanet (2014). Going Deeper with Convolutions. *arXiv preprint arXiv: 1409.4842.*

Tai, K., R. Socher, and C. Manning (2015). Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Association for Computational Linguistics (ACL).*

Tang, Y. (2013). Deep Learning using Linear Support Vector Machines. In *International Conference on Machine Learning (ICML).*

Tayyar Madabushi, H. and M. Lee (2016). High Accuracy Rule-based Question Classification using Question Syntax and Semantics. *Proceedings of the 26th International Conference on Computational Linguistics (COLING-16).*

Thorsten Joachims (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In *European Conference on Machine Learning (ECML).*

Tieleman, T. and G. Hinton (2009). Using Fast Weights to Improve Persistent Contrastive Divergence. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, New York, NY, USA, pp. 1033–1040. ACM.

Turian, J., L. Ratinov, and Y. Bengio (2010). Word Representations : A Simple and General Method for Semi-supervised Learning. *Computational Linguistics* (July), 384–394.

Turney, P. D. (2002). Thumbs Up or Thumbs Down? Semantic Orientation Applied to Unsupervised Classification of Reviews. *Computational Linguistics* (July), 417–424.

Turney, P. D. (2012). Domain and Function: A Dual-space Model of Semantic Relations and Compositions. *Journal of Artificial Intelligence Research 44*, 533–585.

Turney, P. D. (2013). Distributional Semantics Beyond Words: Supervised Learning of Analogy and Paraphrase. In *Association for Computational Linguistics (ACL).*

Vapnik, V. and A. Lerner (1963). Pattern Recognition using Generalized Portrait Method. *Automation and Remote Control 24*(6), 774–780.

Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin (2017). Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), *Advances in Neural Information Processing Systems (NIPS)*, pp. 5998–6008. Curran Associates, Inc.

Vincent, P., H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol (2010). Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research 11*(3), 3371–3408.

Vinyals, I. S. O. and Q. V. Le (2014). Sequence to Sequence Learning with Neural Networks. *Neural Information Processing Systems (NIPS)*, 1–9.

Wang, S. and C. Manning (2012). Baselines and Bigrams: Simple, Good Sentiment and Topic Classification. *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics* (July), 90–94.

Wang, X., G. Hripcsak, M. Markatou, and C. Friedman (2009). Active Computerized Pharmacovigilance Using Natural Language Processing, Statistics, and Electronic Health Records: A Feasibility Study. *Journal of the American Medical Informatics Association 16*(3), 328–337.

Weng, W.-H., K. B. Wagholikar, A. T. McCray, P. Szolovits, and H. C. Chueh (2017). Medical Subdomain Classification of Clinical Notes using a Machine Learning-based Natural Language Processing Approach. *BMC Medical Informatics and Decision Making 17*(1), 155.

Widdows, D. (2008). Semantic Vector Products: Some Initial Investigations. *Second AAAI Symposium on Quantum Interaction 26*(March), 28th.

Wiebe, J. and E. Riloff (2005). Creating Subjective and Objective Sentence Classifiers from Unannotated Texts. In *the Conference on Computational Natural Language Learning (CoNLL).*

Wiebe, J., T. Wilson, and C. Cardie (2005, feb). Annotating Expressions of Opinions and Emotions in Language. *Language Resources and Evaluation 39*(2), 165–210.

Wilson, T. and S. Raaijmakers (2008). Comparing Word, Character, and Phoneme

N-grams for Subjective Utterance Recognition. In *INTERSPEECH*, pp. 1614–1617.

Wu, Y., M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, Ł. Kaiser, S. Gouws, Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean (2016). Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *ArXiv e-prints*, 1–23.

Yang, Z., D. Yang, C. Dyer, X. He, A. Smola, and E. Hovy (2016). Hierarchical Attention Networks for Document Classification. *North American Chapter of the Association for Computational Linguistics (NAACL).*

Yates, A. and N. Goharian (2013). ADRTrace: Detecting Expected and Unexpected Adverse Drug Reactions from User Reviews on Social Media Sites. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7814 LNCS*, 816–819.

Yin, W., S. Ebert, and H. Schütze (2016). Attention-Based Convolutional Neural Network for Machine Comprehension. In *Proceedings of The North American Chapter of the Association for Computational Linguistics (NAACL): Human-Computer QA Workshop.*

Yin, W., H. Schütze, B. Xiang, and B. Zhou (2016). ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. In *Transactions of the Association for Computational Linguistics (TACL).*

Yu, H. F., F. L. Huang, and C. J. Lin (2011). Dual Coordinate Descent Methods for Logistic Regression and Maximum Entropy Models. *Machine Learning*.

Zeiler, M. D. (2012). ADADELTA: An Adaptive Learning Rate Method. *Computing Research Repository (CoRR) abs/1212.5701.*

Zhang, Z., J.-y. Nie, and Xuyao Zhang (2016). An Ensemble Method for Binary Classification of Adverse Drug Reactions From Social Media. *Proceedings of the Social Media Mining Shared Task Workshop at the Pacific Symposium on Biocomputing*.

Zhou, C., C. Sun, Z. Liu, and F. C. M. Lau (2015). A C-LSTM Neural Network for Text Classification. *Computing Research Repository (CoRR) abs/1511.08630.*

Zhu, Y., R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler (2015). Aligning Books and Movies: Towards Story-like Visual Explanations by Watching Movies and Reading Books. In *Proceedings of the IEEE International Conference on Computer Vision*, Volume 2015 Inter, pp. 19–27.