Indian Statistical Institute, Kolkata

M. Tech. (Computer Science) Dissertation

Study of Neural Learning in Text Processing

A dissertation submitted in partial fulfilment of the requirements
for the award of Master of Technology
in
Computer Science

Author:
Debjyoti Paul
Roll No: MTCS-1418

Supervisor:
Dr. Utpal Garain
CVPR Unit, ISI

M.Tech(CS) DISSERTATION THESIS COMPLETION CERTIFICATE
Student: Debjyoti Paul (MTCS1418)
Topic: Study of Neural Learning in Text Processing
Supervisor: Dr. Utpal Garian

This is to certify that the thesis titled "Study of Neural Learning in Text Processing" submitted by Debjyoti Paul in partial fulfilment for the award of the degree of Master of Technology is a bona fide record of work carried out by him under my supervision. The thesis has fulfilled all the requirements as per the regulations of this Institute and, in my opinion, has reached the standard needed for submission. The results contained in this thesis have not been submitted to any other university for the award of any degree or diploma.

Date: Utpal Garain

Dedicated

To my supervisor, parents, the Almighty and all my well wishers, without your help and encouragement it would not have been possible.

Acknowledgements

I would like to thank my dissertation supervisor Dr. Utpal Garain for agreeing to guide me and for helping me to undertake work in the topic and supporting me through the research.

I would also like to thank Dr. Mandar Mitra, who also guided me during my dissertation. I would like to acknowledge the help of Mr. Akshay Chaturvedi (PhD student under Dr. Garain), Mr. Dwaipayan Roy (PhD student under Dr. Mitra) and Mr. Krishanu Nayak (presently Masters' student and doing dissertation under Dr. Garain) for their help and support, who were my co-authors in various papers or research work done as a part of this thesis.

Finally I would like to thank Abhishek Chakraborty, Anabik Pal of NLP Lab and Debojyoti Sinha of MIU Lab at ISI Kolkata, for their support, motivation and guidance.

Abstract

This study deals with the exploration of different Neural Learning frameworks in Natural Language Processing and Information Retrieval. Distributed neural language model *Word2Vec* has been reported to provide elegant word embedding as they capture semantic and syntactic information. Recent studies have also shown that such feature embedding coupled with various Neural Network models have been able to set new benchmarks in various problems of text processing. The aim of this research is to study different neural models and the word embedding framework and explore about their effectiveness and limitations in different challenges in text processing. Three problems have been explored in this study are (i)Learning document embedding from word embedding and analyzing it's effectiveness in document classification (ii) Automatic query expansion using neural word embedding (iii) Biomedical information extraction for Cancer Genetics. Effective use of neural framework for learning document representation for document classification is challenging as existing techniques performs remarkably well and also, the extension from word embedding model is not straightforward. Our study has found that learning such document embedding doesn't yield to any advantage in document classification when compared with naive Term Frequency-Inverse Document Frequency embedding. Semantically related term can be obtained by finding the most similar terms to the query terms using word embedding. In the second problem, Query expansion using such semantically K- nearest neighbor term in the vocabulary do help in improving the result over the baseline retrieval using language model. But it is found that, query expansion for ad-hoc retrieval requires terms to be occurring with high frequency in the relevant documents along with query terms, in addition to terms which are semantically related. But query expansion using word embedding fails to include terms which co-occurs with high frequency anywhere in the relevant document as *Word2Vec* model measures co-occurrence in a limited context window. Our third problem, Biomedical information extraction essentially requires identification of events and finding relation among events and entities. It is found that word embedding is extremely useful in biomedical relation extraction. Also neural architecture like Convolutions neural network provide superior result in event identification. We propose a parser architecture for biomedical document concerning cancer genetics, using neural architecture and word vector as feature. The parser outperforms the state of the art results.

Contents

Chapter 1

Introduction

1.1 Motivation

The fundamental challenge in any text processing is feature representation of terms and documents. The traditional representation of terms in form of embedding includes one-hot encoding, or frequency encoding techniques which were sparse representation. None of these techniques were able to capture the semantic and contextual information of the corpus in the embedding. The recent advancement in learning improved word embedding using distributed, unsupervised and neural framework has managed to overcome this difficulty. Such word embedding provides an low dimensional feature vector representation which capture lexical regularities, for terms, the fundamental unit of any NLP and IR problem. In short this neural based word embedding provide a generic feature set, rich in semantic and contextual information.

The Machine Learning community has witnessed a recent resurgence of Neural networks based frameworks, proposed during the 1980s and 1990s. The main difficulty of Neural networks was the excessive need of computing resource and time to train complex networks. But new advances in training methods, optimization techniques and use of GPUs has overcome the problem to a great extent. It is for this reason neural network has emerged as a good framework to utilize the simple feature encoded in the word embedding for various tasks. On top of that, there are different varieties of neural network models available, each offers a specific advantage in representing the features. For example, unlike conventional neural network with fully connected layers, Convolutional Neural Network provides the scope of using the property of local invariance in feature vector, which is extremely important in Computer Vision. Similarly Recurrent Neural Network allows time series modelling, Auto encoders provide a model for feature compression.

The ability of neural networks to learn complex hypothesis and model nonlinear hypothesis, without the need of modeling complex features as input, makes them interesting in pattern recognition. Hence it is an ideal framework which can be utilized to extract the information encoded in neural word embedding.

These advances in learning word embedding and the availability of improved training framework for neural network, has rekindled the need of investigation into the existing challenges in Natural Language Processing and Information Retrieval.

1.2 Scope and Background Studies

1.2.1 Neural Networks

Artificial Neural Networks has the ability to learn complex reasoning from simple feature representation. Motivated from biological neural networks, it has gained popularity in Machine Learning because of its simplicity in representation and flexibility in modelling. The basic unit of neural networks are neurons which process information at input to form a output following a nonlinear transformation by activation unit. The past few years have witnessed an extensive research in application of neural word embedding and various neural architecture in various problems of text processing. Many of this models have set new benchmarks in well researched problem in text processing.

1.2.2 Hyper Parameters

The general hyper parameters of Neural Network are number of hidden layer, activation function, learning rate, regularization parameter etc. There has been considerable research on different activation functions like sigmoid, tan hyperbolic, Rectified Linear Units (ReLu).

Activation Unit

Activation function acting on the input to the neuron plays a crucial role in the weight update process. Popular activation function which are considered over and over again in the literature are Sigmoid, Tan Hyperbolic, ReLu, Leaky ReLu, Maxout etc. Sigmoid function, $f(x) = \frac{1}{1+\exp(-x)}$ is simplest, though it saturates early and stops the learning. Tan Hyperbolic, $f(x) = 2 \times sigmoid(2x) - 1$ have been explored extensively and [1] it works better than Sigmoid because the gradients are large and also less prone to saturation. Figure 1.1 [1], taken from , shows the activation function: ReLu, Sigmoid and Tanh.

ReLu [2], defined as $f(x) = max(0, x)$ doesn't saturates has become popular in the recent years, since it has shown great success in accelerating Stochastic Gradient Descent and has avoids expensive sigmoid operation unlike Sigmoid and Tanh. The obvious drawback of ReLu is that, it might happen that a large gradient might be flowing through a ReLU neuron, which could cause the weights to update in such a way that the neuron will never activate on any datapoint again. If this happens, then the gradient flowing through the unit will forever be zero from that point on. Leaky ReLu manages to overcome this difficulty by having a small negative slope $f(x) = I(x < 0)(\alpha x) + I(x \geq 0)x$. Maxout activation proposed by [3] generalizes the concept of ReLu and Leaky-ReLu into a single function, $f(x) = max(w_1x + b_1, w_2x + b_2)$.

[1]`http://www.slideshare.net/oeuia/neural-network-as-a-function`

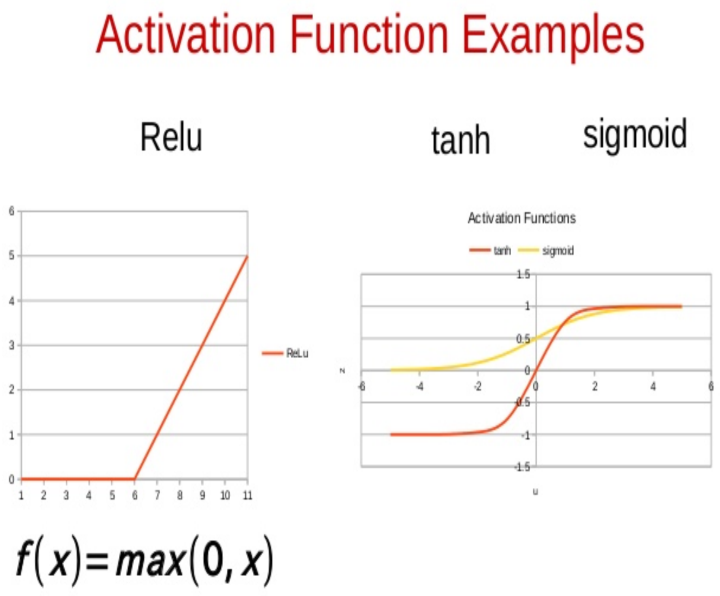

Figure 1.1: ReLU, Sigmoid and Tanh activation function

where w_1, b_1, w_2, b_2 are parameters depending on the desired saturation points.

Regularization

Neural Networks containing one or more hidden layer to learn complex features mapping input to output, from less number of training samples, tends to suffers from the problem of overfitting. Regularization is an important part in Neural Network Learning to counter overfitting. Popular regularization techniques includes L_2 norm and L_1 norm. Dropout [4] is a recent technique introduced in neural network to circumnavigate the overfitting problem. The dropout neural network model "drops" neurons in hidden layers with probability p during each iteration. This leads to generation of different network architecture at each iteration in a random manner. In a way this is similar to ensemble techniques where different classifier are choosen at random and the overall performance is average of all the different architecture generated at different time steps. Figure 1.2 shows the difference between neuron in a dropout neural network model and an ordinary neural network. For a neural network with M hidden layers works as follows. Let $m \in 1, \ldots, M$ be the index the hidden layers of the network. Let y^m denote the vector of inputs into layer m, y^{m+1} denote the vector of outputs from layer m and p be a random variable drawn from a Bernoulli distribution with probability p' then the drop out neural model is defined as:

$$p_j^m \sim Bernoulli(p') \tag{1.1}$$

$$\tilde{y}^m = p^m y^m \tag{1.2}$$

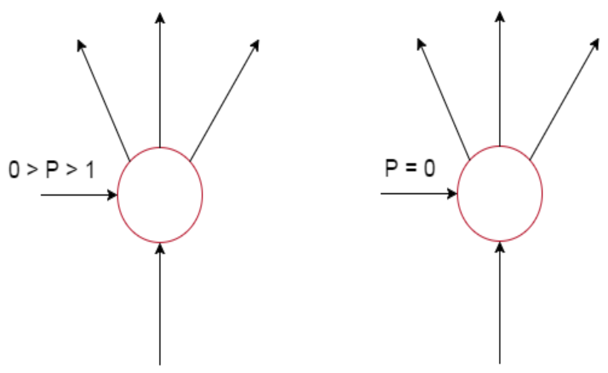

Figure 1.2: Left: Neuron with dropout Right: Neuron withut dropout

$$y_i^{m+1} = f(W_i^{m+1}\tilde{y}^m + b_i^{m+1}) \tag{1.3}$$

Where, $f(x)$ is the activation function and W and b are the weights and bias parameter.

Optimization and Learning rate annealing

The recent success of neural network to an extent can be attributed to the optimization techniques that has been proposed. *Stochastic Gradient Descent*(SGD) as an alternative to *Batch Gradient Descent* has shown it's potential to accelerate the leaning process. Though it is more prone to the local extrema problem, there is always a way out in such cases by assigning a different initialization point and re-training. . SGD, updates the network parameter W along the gradient with respect to the cost function $J(W; x^{(i)}; y^{(i)})$ for i^{th} training sample $(x^{(i)}, y^{(i)})$.

$$W = W - \eta \nabla_W J(W; x^{(i)}; y^{(i)}) \tag{1.4}$$

SGD inherently suffers from oscillation problem. So a *Momentum factor* μ is added to 1.4 to overcome this problem. The *momentum* term is added by calculating the weight update $v_t = \mu v_{t-1} - \eta \nabla_W J(W)$ and updating W as $W = W - v_t$. The learning rate η is crucial, and it has been found that a high value of learning rate should be gradually decreased at each iterations. This problem has been addressed by various annealing of the learning rate η. Two most successful learning rate annealing algorithm are *RMSprop* [5] and *Adam* [6]. *Adadelta* [7] is a generalized version of RMSprop. ***Adadelta*** uses a different learning rate for every parameter W_i at every time step t. The gradient of the objective function with respect to the parameter W_i at time step t: Let $g_{t,i} = \nabla_W J(W_i)$ The running average of $E[g^2]_t$ at time step t depends on the previous and current gradient: $E[g^2]_t = (1-\gamma)E[g^2]_{t-1} + \gamma E[g^2]_t$

$\Delta_w W_t = -\eta g_{t,i}$ and $W_{t+1} = W_t + \Delta_w W_t$ The parameter update of the Adadelta [7] is then defined as follows:

$\Delta W_t = -\frac{\eta}{\sqrt{E[g^2]_t + \epsilon}}$

or $\Delta W_t = -\frac{\eta}{RMS[g]_t} g_t$

But the units in this update in the numerator and the denominator do not match [7], i.e. the update don't have the same "hypothetical units" as the parameter. To realize this, they first define another exponentially decaying average, this time not of squared gradients but of squared parameter updates:

$E[\Delta W^2]_t = (1-\gamma)E[\Delta W^2]_{t-1} + \gamma \Delta W_t^2$

$RMS[\Delta W]_t = \sqrt{E[\Delta W^2]_t + \epsilon}$ ΔW_t is unknown quantity and it is approximated by $RMS[\Delta W]_t$ and the weight update equation follows:

$$\Delta W_t = -\frac{RMS[\Delta W]_t}{RMS[g]_t} g_t \tag{1.5}$$

RMSprop is a special case of **Adagrad** where, $\gamma = 0.1$.

Adaptive Moment Estimation (***Adam***) is also a learning rated annealing method proposed in [6]. In addition to storing an exponentially decaying average of past squared gradients like Adadelta and RMSprop, Adam also keeps an exponentially decaying average of past gradients , similar to momentum:

$m_t = \beta_1 m_{t-1} + (1-\beta_1)g_t$

$v_t = \beta_2 v_{t-1} + (1-\beta_2)g_t^2 v_t$

m_t and v_t are estimates of the mean and the variance of the gradients respectively. As m_t and v_t are initialized as vectors of $W's$. [6] observed that they are biased towards zero, especially during the initial time steps, and especially when the decay rates are small (i.e. β_1 and β_2 are close to 1).

They counteract these biases by computing bias-corrected first and second moment estimates:

$\tilde{m}_t = \frac{m_t}{1-\beta_1^t}$

$\tilde{v}_t = \frac{v_t}{1-\beta_2^t}$

Then the weights W_t is updated as follows:

$$W_{t+1} = W_t - \frac{\eta}{\sqrt{\tilde{v}_t + \epsilon}} \tilde{m}_t \tag{1.6}$$

[6] propose default values of 0.9 for β_1, 0.999 for β_2. Empirical results shown in [6] that Adam works improves the learning of neural networks and can be fairly compared to other adaptive learning-method algorithms, but in this work we observed that is it prone to overfitting even when proper measures are in place. The Adam algorith is extremely fast in convergance when compared to it's other counterpart

1.2.3 Single and Multi Layered Perceptron

The most basic form of neural network is where every neuron of a particular layer are connected to all the neurons of the previous and the next layer. In single layered perceptron there is no hidden layer. Multi layered Perceptron (MLP) contains one of more hidden layers. [8] [9] have proposed a joint learning based using MLP model for basic NLP tasks like Parts of Speech-Tagging, Named Entity Recognition, Chunking, and have performed very well compared to the existing benchmarks.

1.2.4 Back Propagation

For a classification problem, with the following training sequence $(X, Y) = (x^{(1)}, y^{(1)}), \dots (x^{(N)}, y^{(N)})$, the cost function is defined as,

$$J(W, b) = \frac{1}{N} \sum_{i=1}^{N} ||y^{(i)} = h_{W,b}^{(i)}||^2 + \lambda \sum_{l=1}^{L} \sum_{i=1}^{n_l} \sum_{j=1}^{n_{l+1}} ||W_{(i.j)}^{(l)}||_p \tag{1.7}$$

Where, W, b are the network weights and bias. $h_{W,b}^{(i)}$ is the hypothesis value in the output layer, predicted by forward pass of the input pattern through the network. The cost function depends on the deviation of the hypothesis from the actual output. The second part of the cost fucntion is the regularization term to control overfitting. The network weights W are updated as :

$$W_{ij}^{(l)} = W_{ij}^{(l)} - \alpha \frac{\partial}{\partial W_{ij}^{(l)}} J(W, b) \tag{1.8}$$

$$b_i^{(l)} = b_i^{(l)} - \alpha \frac{\partial}{\partial b_i^{(l)}} J(W, b) \tag{1.9}$$

Given a training example (x, y), the input is passed through each layer to compute all the activations throughout the network. The output value of the hypothesis $h_{W,b}(x)$ is calculated and an error term $\delta_i^{(l)}$ is calculated that measures how much that node i in the l^{th} layer deviates from the actual value of that node. For an output layer n_l, error is obtained by the difference between the network's activation and the true target value $\delta^{(n_l)}$. The error update for the hidden layer δ^L based on a weighted average of the error terms of the layer that uses $a_i^{(l)}$ as an input. The weighting factior is $w_{(l+1,l)}$ In detail, here is the backpropagation algorithm [1]:

1. Perform a feedforward pass, computing the activations for layers L_2, L_3, and so on up to the output layer L_{n_l}.

2. For each output unit i in layer n_l (the output layer), set $\delta_i^{(n_l)} = \frac{\partial}{\partial z_i^{(n_l)}} \frac{1}{2} \left\| y - h_{W,b}(x) \right\|^2 = -(y_i - a_i^{(n_l)}) \cdot f'(z_i^{(n_l)})$ For $l = n_l - 1, n_l - 2, n_l - 3, \dots, 2$

3. For each node i in layer l, set
   $\delta_i^{(l)} = \left(\sum_{j=1}^{s_{l+1}} W_{ji}^{(l)} \delta_j^{(l+1)} \right) f'(z_i^{(l)})$

4. Compute the desired partial derivatives, which are given as:

$$\frac{\partial}{\partial W_{ij}^{(l)}} J(W, b; x, y) = a_j^{(l)} \delta_i^{(l+1)} \tag{1.10}$$

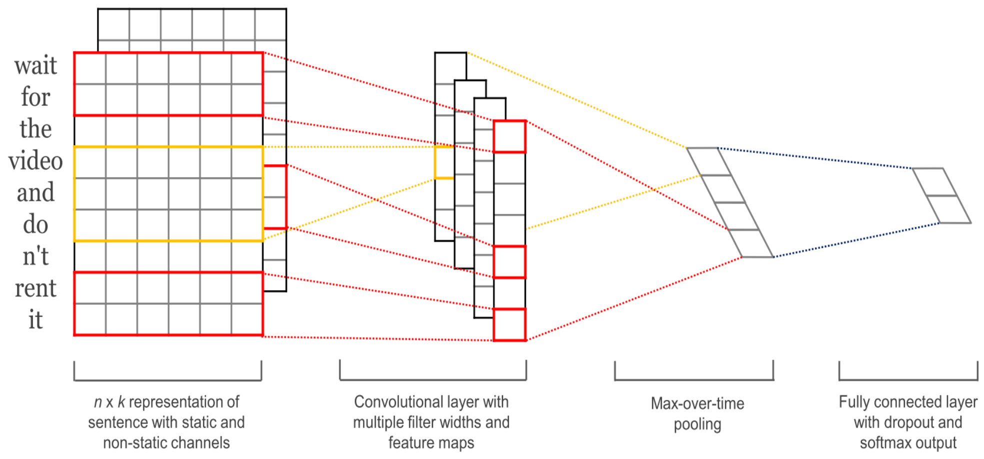
wait
for
the
video
and
do
n't
rent
it
n x *k* representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 1.3: Convolutional Neural Network for NLP

$$\frac{\partial}{\partial b_i^{(l)}} J(W, b; x, y) = \delta_i^{(l+1)}. \tag{1.11}$$

1.2.5 Convolutional Neural Network

Convolutional Neural network is a special kind of neural network, which applies different convolution operation on the input signal, followed by optional down sampling and a fully connected MLP. It takes care of local invariance that might exist in the input signal, especially in case of image. As shown in the Figure 1.3, CNN essentially consist of several Convolution-Pooling layer combination followed by fully connected layer. The weights of the convolution layer is shared and hence CNN leads to huge reduction in network parameters compared to MLP of same size. The pooling layer is a down sampling step, the most popular being Max-Pooling which preserve the maximum response in a particular window.

Though it's application in text processing is slightly unintuitive, introduction of word embedding has removed that limitation. There has been several attempts to implement CNN for task like sentiment analysis [10], relation classification [11], and sentence modeling [12] [13] and classification [14] which have shown great potential. There has been also some works on document summary generation [15] using CNN which has performed quite commendably. Figure 1.2 [2] shows the CNN model used for NLP problems. A matrix of dimension $N \times K$ is given as input, where N is the number of terms and K is the corresponding embedding dimension.

CNN consists of three layers [3], [4]:

- **Convolutional Layer**: Convolutional layers consist of neurons arranged in a rectangular grid. The convolutional layers works across spatial domain, where the input is

[2] `http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/`
[3] `http://cs231n.github.io/convolutional-networks/`
[4] `http://andrew.gibiansky.com/blog/machine-learning/convolutional-neural-networks/`

also a rectangular grid. There may be multiple such convolution filters which basically convolves with the input. The weights of the convolutional layers generates the output response by a weighted product with the input signal with window size equal to the convolutional layers

Let the input consists of $N \times N$ square neuronal layer is followed by the $m \times m$ convolutional layer W. Then size of the output response would be $(N-m+1) \times (N-m+1)$. The input to $(i,j)^{th}$ neuron of layer l, $y_l^{(i,j)}$, following the input response at layer $l-1$ follwed by the convolution layer and activation function f is given by:

$$y_l^{(i,j)} = f(x_l^{(i,j)}) = f(\sum_{a=0}^{m-1}\sum_{b=0}^{m-1} W_{(a,b)} y_{l-1}^{(i+a,j+b)}) \tag{1.12}$$

where x is the pre-nonlinearity response of the $l-1^{th}$ layer. The backpropagation is done in a similar way as described above. If J is the error cost function, then the error is propagated from the neuronal output of $(i,j)^{th}$ neuron of l^{th} layer, $y_l^{(i,j)}$ as $\frac{\partial J}{\partial y_l^{(i,j)}}$ and the weight update $W_{(a,b)}$ is computed as:

$$\frac{\partial J}{\partial W_{(a,b)}} = \sum_{i=0}^{N-m}\sum_{j=0}^{N-m} \frac{\partial J}{\partial x_l^{(i,j)}} \frac{\partial x_l^{(i,j)}}{\partial W_{(a,b)}} = \sum_{i=0}^{N-m}\sum_{j=0}^{N-m} \frac{\partial J}{\partial x_l^{(i,j)}} y_{l-1}^{(a+i,b+j)} \tag{1.13}$$

$$\frac{\partial J}{\partial x_l^{(i,j)}} = \frac{\partial J}{\partial y_l^{(i,j)}} \frac{\partial y_l^{(i,j)}}{\partial x_l^{(i,j)}} = \frac{\partial J}{\partial y_l^{(i,j)}} \frac{\partial}{\partial x_l^{(i,j)}} (f(x_l^{(i,j)})) = \frac{\partial J}{\partial y_l^{(i,j)}} f'(x_l^{(i,j)}) \tag{1.14}$$

$$\frac{\partial J}{\partial y_{l-1}^{(i,j)}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial J}{\partial x_l^{(i-a,j-b)}} \frac{\partial x_l^{(i-a,j-b)}}{\partial y_l^{(i,j)}} = \sum_{a=0}^{m-1}\sum_{b=0}^{m-1} \frac{\partial J}{\partial x_l^{(i-a,j-b)}} W_{(a,b)} \tag{1.15}$$

since, $\frac{\partial x_l^{(i-a,j-b)}}{\partial y_{l-1}^{(i,j)}} = W_{(a,b)}$

- **Max-Pooling**: After each convolutional layer, there may be a pooling layer. The pooling layer takes input the convolutional layer as small blocks out of the rectangular blocks and subsamples it to produce a single output from that block. There are several ways to do this pooling, such as taking the average or the maximum, or a learned linear combination of the neurons in the block. But in general Max-Pooling, which assign the output as the maximum response of each block is followed in this paper.

- **Fully-Connected**: The "high-level reasoning" in the neural network is done via fully connected layers. A fully connected layer is similar to MLP. It takes all neurons in the previous layer and is connected to every neuron of the next layer. It can be visualized as a 1×1 convolution operation or layer.

1.2.6 Word Representation

In text processing, word is usually treated as the atomic unit. Representation of word which captures linguistic patterns and syntactic regularities have been always a potential research challenge. The representation must also allow calculation of similarity between words with general similarity functions. Different techniques for creating word embedding, mapping each vocabulary entry to a R^n, was introduced in Latent Semantic Analysis (LSA) [16] and probabilistic LSA [17]. Recently [18] proposed a neural framework to learn the word representation using a distributed and unsupervised setting. The novelty of the work is that the concept of similarity between the words is not only confined to syntactic regularities, but it can capture semantic as well as contextual information. For example words derived from different inflectional forms is expected to be similar. But [18] showed that representation learned by Word2Vec can encode linear relationship like **Man = Women - Queen + King**. By using hierarchical softmax, the weight update is made faster which helps in learning better word representation. By using negative sampling technique (as a simpler alternative to hierarchical softmax), iterative process over the entire vocabulary for weight update for each term in the vocabulary is avoided. The essential idea of the neural model proposed for learning word vector is to predict a particular word by seeing the surrounding words in a fixed window, which is essentially termed as continuous bag of word (CBOW) in [18]. The skip gram model is just the mirror image of CBOW. The paper also proposes composition technique to represent phrases (especially important for idiomatic phrases) by taking care of word order. Given a window of size C, the skip gram model for the word sequence $w_1, w_2 \dots w_T$ as described in [18]: $\frac{1}{T}\sum_{t=1}^{T}\sum_{-C \le i \le C, C>0} \log p(w_{t+i}|w_t)$ The basic architecture of skip-gram tries to learn $P(w_{t+i}|w_t)$ and expressed as the softmax probability:

$$P(w_j|w_i) = \frac{\exp({v'_{w_j}}^T v_{w_i})}{\sum_{p=1}^{W} \exp({v'_{w_p}}^T v_{w_i})} \tag{1.16}$$

The probability calculation for CBOW model will follow similarly Figure 1.4 taken from [18] shows the network architecture for learning word representation. Based on this word embedding framework, a series of research into neural based dependency parsing are quite promising.

1.2.7 Document Representation

Word representation using above framework can be extended to phrases using composition proposed in [19]. An intriguing question which arises from the above discussion on word representation. Can it be extended to higher units like paragraph and documents [19] proposed a neural model, *Doc2Vec*, which tries to achieve this goal. The main intuition is to model paragraph or sentences and then to generalize document representation from the paragraphs. A document or paragraphs contains a sequence of words and modeling a document essentially involves encoding this word information. Earlier implementation of document vector

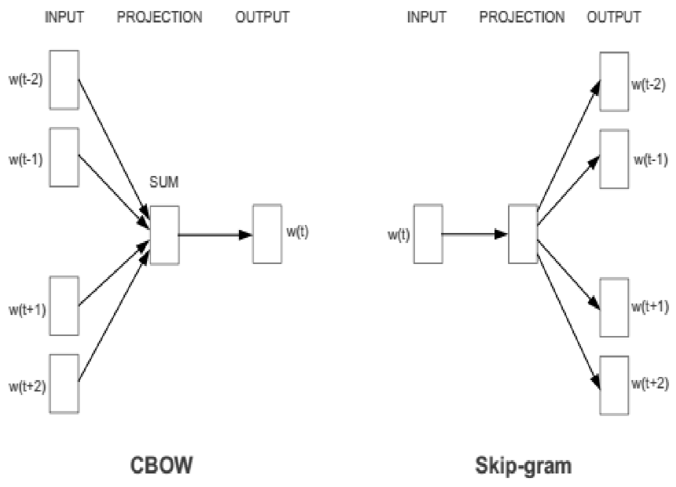

Figure 1.4: Conitnious Bag of Word and Skip-Gram Model

essentially corresponds to listing out presence or absence of words of the vocabulary either as binary vector or using some term weighting scheme. The sequential information of the words is lost in such model in addition to the vector being sparse and very high dimensional.

Doc2Vec overcomes this shortcoming by learning the document representation using a similar framework like in Word2Vec. It utilizes the word representation learnt by Word2Vec on the corpus and then learn the document representation by a similar context window approach. Two type of Doc2Vec model is proposed, Distributed Memory model (DM) and Distributed Bag of words model (DBOW). The former learn the document vector by iterating over each windows of k words $(w_1, w_2 \ldots w_k)$ in the document and to predict word w_{l+1} given $(w_{l-k}, w_{l-k+1} \ldots w_l)$ as input. The DM model memorizes the word sequence. An alternative is DBOW model which neglects the sequential information within a particular window. The paper also proposes a model to jointly learn both the model into a single representation. The performance of this document vectors or paragraph vectors has outperformed existing techniques in tasks like sentiment analysis and in paragraph similarity. Figure 1.5 taken from [19] shows the document vector learning, similar to word vector learning.

1.3 Our Work

From the above discussion, it is quite evident that word embedding and neural network models has shown potential in many challenging problems. Word embedding has provided a generic feature representation, different dimension of this feature possess represents different lexical meaning. This has motivated us in exploration of different text processing problems in light of this term embedding and to investigate it's effectiveness. In this study, we have explored the effectiveness of word embedding in three problem specific to NLP and IR: (a) A comparative study learning Neural Document Embedding from Word Embedding for Document Classification (b) Query Expansion using Word Embedding in Information Retrieval with improved feature (c) Information extraction for event detection and argument

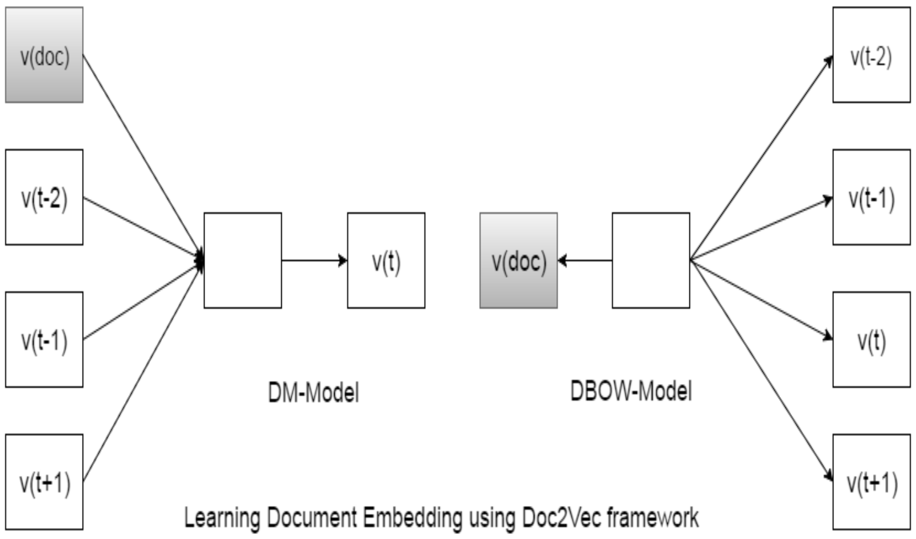

Figure 1.5: Learning Paragraph or Document Vector

extraction in Biomedical Documents This work focuses on exploration of the above three task using various neural framework. In this thesis, we have tried to answer the following questions: (a) Can the language model generating the word embedding extended to represent documents ? (b) How can the word representation be incorporated to help retrieve document in a IR setup (c) How can the semantic and contextual information utilized in information extraction system ?

Chapter 2

Learning Document Representation from Word Representation for Document Classification

2.1 Introduction

The problem of representation of documents in form of real vectors is one of the core component and a central challenge in any document processing and information retrieval task. Recently there has been extensive research on word representation using word embedding by encoding term co-occurrences and semantics information using neural network as learning framework [18], [20]. Such distributed language model, has the capability to learn word representation at low dimensions.

The research pursued in this study is mainly concerned with exploring different methods of getting the document embedding from word embedding and comparing them with respect to the problem of 20 Newsgroups document classification. There has been some recent reports which has extended word level embedding to document representation by learning the document semantic structure [19] (*Doc2Vec*). In addition to that we would like to compare the performance of such Word Embedding based document representation learning framework with traditional term frequency - inverse document frequency (TF-IDF) term weighting representation techniques.

A document is a collection of terms and for effective representation the important terms should be used to represent the document. There has been several researches on the ways to assign term importance which mainly involve some notion of term weighting. Weights are assigned to terms based on term frequency [21], inverse document frequency coupled with term frequency [22] or highlighting only the noun phrases [23].

2.1.1 Term weighting scheme

Documents may contain large number of terms. A method for learning document embedding from word embedding by *Doc2Vec* [19] involves learning the document vector using a Autoencoder framework where, the document vector and vectors of the word surrounding a particular word is used to predict the particular word. We followed a different path in forming the document vector. A document must be represented by the distinct terms occurring in it which is unique to the particular document. in this study, three types of term weighting are considered. The basic assumption is similar to TF-IDF, that is to give higher weights to important terms. Various term weighting methods are explored and the information obtained from *Word2Vec* model is added to it using vector sum composition or .

TF-IDF weighting

In this scheme, the terms are weighted based on term frequency and inverse document frequency measured over the entire corpus [22].

Noun Phrase

In this method, only the terms in noun phrases are used for composition. The basic hypothesis in this case is Noun Phrases are the sole representative of a document which is often true in many information retrieval problem [24], [23], [25]. Weights are then assigned to phrases are chosen based on

1. TF-IDF weighting on the nouns

2. Latent Dirichlet allocation (LDA) [26] is an efficient topic modelling techniques that can give meaningful insight to the major distinctive topics occurring in the corpora. LDA is used to assign model 10 topics in the corpus. For each topic top k nouns are chosen which represents the topic in the document under consideration. The final document vector is constructed by applying suitable composition function on each noun phrases corresponding to the nouns. The basic intuition behind using LDA for document classification [27] is that LDA might be able to detect the major topics which might be easily mapped to the different classes in this classification task. However since the main motivation is to test the document representation derived from Word2Vec, the LDA is constrained to model less topics than the number of classes. This essentially makes sure that the framework is restricted to take basic clues from LDA in order to filter important nouns and does not get a direct mapping to the class labels.

Using the above weighting techniques, terms with top N weights are taken and their word embedding are composed using some predefined function (e.g. sum or dimensionality reduction) to generate the document vector. This model depends mainly on term weighting scheme and doesn't entertain the semantic relationship in the document structure in form of sentences or phrases since the composition function is pre-defined. In this study, the composition function used are mainly sum and dimensionality reduction. Vector concatenation though provide an elegant way to represent phrases, but increases the dimension of the corresponding document embedding. This leads to inferior classifier performance due to increased number of parameters that are needed to be learned. Thus we give more preference to composition function which generates the document representation at lower dimension. The main idea in Doc2Vec method [19] is to learn the composition function for word embedding from the document in a sequential or distributed fashion. This method preserves the semantic relationship. The main goals of this paper are:

1. To explore how to form the document embedding using word embedding representation with a predefined, linear (vector sum) composition function and various term weighting schemes.

2. To find out the performance of such document embedding for document classification and compare with existing document embedding techniques and traditional TF-IDF weighting scheme.

2.1.2 Dataset description

All the experiments were done on the 20 Newsgroups[1] dataset. This dataset contains 11314 training examples and 7532 test examples. The task is to classify each document into one of 20 classes. Each document in the dataset mainly consists of three sub regions : Keywords, Subject and Context. The Keywords and Subjects are short snippets of text while the Context consists of a large volume of text. Only 1341 documents (training and test included) contain keywords however subject is present in every document. The main challenge of document representation in 20 Newsgroups data set is efficient representation of the context. The short snippets in keywords and subject can be easily represented as sum of word vectors of the words occurring in them.

2.2 Feature sets for document representation

In this section, we will introduce the various feature sets that were explored for the task of document classification in 20 Newsgroups dataset. Before forming different feature sets, we removed all the stop words present in the documents using NLTK toolkit [28].

2.2.1 Retraining Word Vectors

We obtained 300-dimensional word vectors[2] trained on GoogleNews corpus. Then these word vectors were retrained on 20 Newsgroups dataset. Retraining is done by initially

[1]`http://qwone.com/~jason/20Newsgroups/`

[2]GoogleNews-vectors-negative300.bin.gz

assigning random vectors to the words which didn't have vectors because they are specific to 20 Newsgroups corpus and either don't occur or occur with low frequency in Google News. Words which already have an entry in GoogleNews word vector model vocabulary are initialized with their corresponding vectors. With this initial parameter setting of the Autoencoder model described in [18], the network is trained with terms in the 20 Newsgroups corpus as described in [18]. The parameters for retraining the Word2Vec are essentially kept same as reported in [18] for GoogleNews dataset.

2.2.2 Feature Representation from Word Embeddings

Subject and Keywords play a crucial role for document classification in 20 Newsgroups dataset as was mentioned in [29] where the authors assigned ten times more weight to words present in subject and keywords than to words present in context. Therefore all the feature sets explored in this study, unless mentioned otherwise, contain word vectors of subjects and keywords as features (word vectors for subject and keyword were obtained by summing up the word vectors of words present in subject and keyword respectively). Thus, we get 600 dimensional feature for subject and keywords. For documents not containing keywords, the feature is padded with zeros.

The various feature sets explored in this study are as follows:

1. **LDA + Subject + Keyword:** LDA was done on the set of noun phrases of all documents and the number of topics was set to 10 (Increasing or Decreasing this parameter degrades the classifier performance. However strict grid search on this parameter was not performed). Then for each topic, we select the top five noun phrases based on probability. In this way, we get a set containing a maximum of 50 noun phrases. Then we sum the vectors of noun phrases present in the aforementioned set to get a 300 dimensional vector. This results in 900 dimensional feature set (300 dimension for LDA and 600 dimension for subject and keywords). This feature set is referred as LDA SUB KEY in result section.

2. **Principal components + Subject + Keyword:**. Here two cases were considered:

   (a) For each document, Principal Component Analysis (PCA) is done on the vectors of noun phrases present in the document and the first principal component is considered as feature. This results in 900 dimensional feature set (300 dimension for principal component and 600 dimension for subject and keywords). This feature set is referred as PCA1 SUB KEY in result section.

   (b) Here the first two principal components are considered as feature resulting in a 1200 dimensional feature set. This feature set is referred as PCA2 SUB KEY in result section. No further improvement was observed by taking three or more principal components.

3. **Sum of top 10 TF-IDF + Subject + Keyword:** Here also, two cases were considered:

   (a) For each document, the sum of the word vectors of the words having top 10 TF-IDF scores, is taken. This results in 900 dimensional feature set (300 dimension for TF-IDF and 600 dimension for subject and keywords). This feature set is referred as TF-IDF SUB KEY in result section.

   (b) It is to be noted here that by simply taking the sum, we are assigning equal weight to all the top 10 words (based on TF-IDF scores). One way to assign weights to these words is to use their respective tfidf scores and take weighted sum. However, this didn't give satisfactory performance. Hence, in this feature set, the network is given the word vector of the top 10 words in the form of 300x10 matrix as an input followed by 1x10 convolution kernel. Thus the network is able to learn the weights during training. This feature set is referred as TF-IDF CONV SUB KEY in result section.

4. **Doc2Vec:** Doc2Vec [19] method returns a vector for every document. This vector is able to capture the semantic relation within a document. Here we used these vectors as feature set for our task of document classification. Three models are considered for Doc2Vec Distributed Memory(DM) Model, Distributed Bag of Words (DBOW) Model and Concatenated DBOW and DM model. It is to be noted that these feature sets were are not appended with separate feature for subject and keywords, since the whole document is used to generate the document vectors.

   (a) Distributed Memory Model (DM Model): The distributed memory model tries to capture the sequential and semantic structure if the document. The word vectors are initialized from pre-trained word embedding model on the same or different corpus. The document vectors are initialized randomly. The network is trained to predict a word given its context as input and the document vector is updated. It encodes the sequential information of each word, hence memory model, by using a window of words preceding the current word as context. The final feature vector obtained is of 300 dimension. This feature set is referred as Doc2Vec(DM Model) in result section.

   (b) Distributed Bag of Words (DBOW): It is a variant to the DM model, where the sequential training is not followed. Instead the context can be any word withing a window surrounding the current word. The final feature vector obtained is 300 dimension. This feature set is referred as Doc2Vec(DBOW Model) in result section.

   (c) Concatenated DBOW and DM Model: A concatenated model is used to preserve both the feature of DBoW and DM Model. We concatenate the vectors to form a feature vector of 600 dimension. This feature set is referred as Doc2Vec(Concatenated Model) in result section.

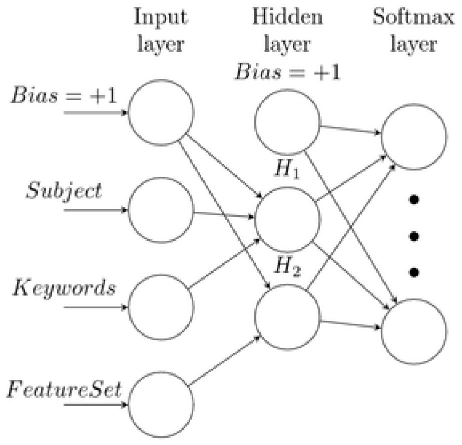

Figure 2.1: Classifier Architecture used to evaluate various feature sets. Here H_1 and H_2 consists of 100 neurons each. Note that in case of Doc2Vec feature sets, the nodes corresponding to Subject, Keywords and the hidden node H1 were absent.

2.3 Classifier Architecture

In order to compare between these feature sets, we maintained the same network architecture for all the feature sets. The network architecture is shown in Figure 2.1. The hidden layer of the network consists of 200 neurons (excluding bias), where the first 100 neurons are only connected to 600 neurons corresponding to subject and keywords and the other 100 neurons are connected to the rest of the feature set. In case of Doc2vec, this layer had only 100 neurons (excluding bias). ReLU [30] activation function is used as activation function for the hidden layer. This layer is then connected to softmax layer consisting of 20 neurons (number of classes). If Sigmoid and tanh activation is used in place of ReLU, the network takes longer to converge.
The classifier parameter is tuned by observing the training accuracy separately for the context feature (i.e. *FeatureSet* in Figure 2.1) and subject-keyword. A network is first designed to classify based on first 600 dimension feature corresponding to subject and keyword and training accuracy is observed by varying the hidden layer size in multiples of 50. Best training accuracy is obtained at hidden layer of size 100. Same procedure is followed for the context feature vector (i.e. *FeatureSet* in Figure 2.1) and peak training accuracy is obtained at 100. So by this procedure, the hidden layer parameter is fixed.

The classifiers are designed using Keras Neural Network library in Python [31]. Training was done using stochastic gradient descent (SGD) with batch size of 128 for all the feature sets. In order to reduce over-fitting, we use the dropout technique [32].

2.4 Results

The results for various feature sets are given in Table 2.1 As can be seen, Doc2Vec(DBOW model) has the best performance. This is an indication that semantics do play a role in document classification. TF-IDF SUB KEY performs better than the feature sets based on

Table 2.1: Classification Performance on 20 Newsgroups dataset

Feature Set	Avg. Precision	Avg. Recall	Avg. F1-Score
LDA SUB KEY	0.76	0.76	0.76
PCA1 SUB KEY	0.76	0.76	0.76
PCA2 SUB KEY	0.76	0.76	0.76
TF-IDF SUB KEY	0.78	0.78	0.78
TF-IDF CONV SUB KEY	0.77	0.77	0.77
Doc2Vec(DM model)	0.75	0.75	0.75
Doc2Vec(DBOW model)	0.82	0.82	0.82
Doc2Vec(Concatenated Model)	0.81	0.82	0.81
TF-IDF embedding	0.85	0.84	0.84

only noun phrases which suggests that although noun phrases are important, they are not sufficient for document classification. Amongst all the feature sets explored in this study, Doc2Vec(DM model) gave the least scores.

The results obtained clearly shows that DBOW learning method proposed in [19] is better than the model we explored. But the more interesting result is that TF-IDF embedding outperforms all the model. This clearly shows the problem in representing document for classification by extending word embedding model. It is to be noted that the result of the experiment clearly shows that the problem is in composition of word vector. The model we proposed by choosing top N tems and feature representation ultimately composes the word vectors using simple vector summing. DM-model focuses on the sequential word order, but the performance clearly shows such word order is not important for document classification.

2.5 Summary

This chapter looks at different types of feature sets for document classification. The various conclusion drawn from this paper are listed below:

1. The best performance was obtained by the Doc2Vec DBOW model, which suggests that it is indeed important to capture semantic relationship and contextual information within a document for document classification. Also since the DBOW model learns the composition function, this suggests that composition function is extremely important in forming document vector.

2. Having said that, the methods of constructing the document embedding using Noun Phrase, LDA or term weights are not lagging far behind. It must be noted that, only top 10 noun phrase or top 10 terms in case of TF-IDF are used to represent the entire document. This surely indicates the strength of distributed language model Word2Vec.

3. Doc2Vec DM model captures the sequential memory based semantics. Hence an interesting finding of this paper is that capturing sequential memory based semantics

degrades the performance of the classifier. Further investigations are needed to find out the possible reasons.

4. Amongst the features which didn't capture semantic relationship, the best performance was obtained by TF-IDF method. This suggests that only Noun phrases are not sufficient for representing documents as the performance gets stuck after a certain limit.

5. Another possible reason behind the poor performance of LDA feature set could be because the topics chosen by LDA were not consistent with the actual output labels of the 20 Newsgroups dataset.

6. The fact that traditional feature representation still works better leads to an obvious future endeavour in document representation. The main problem in the methods proposed in this study or the framework like Doc2Vec is inability to model larger or different sized documents. The main challenge in this problem is word vector composition. It is clear that the most general approach to model different size documents using top word representation doesn't work. The reason can be attributed to the inability of simple vector sum composition to represent a document.

7. We have done some local parameter tuning for the models proposed in this study, like number of noun phrases to consider, number of topics in LDA, number of terms to be chosen to represent a document based on TF-IDF weights. Though exhaustive search to find the best parameters has not been done, which opens up scope for further investigation in the future.

8. As part of future work, we plan to extend the comparison of various document embedding frameworks discussed here for other document classification and information retrieval tasks.

Chapter 3

Automatic Query Expansion using Word2Vec

3.1 Introduction

The recent advances in Deep Neural Network framework for text processing has motivated a few explorations in Information retrieval. A few research have focused in particular on the use of *word embeddings* generated using deep Neural Networks. The interest in the use of word embeddings has been recently been motivated because [18] showed how distributed neural embedding captures the semantic relatedness between words. Such embedding is reported to capture the semantic as well syntactic regularities accurately and easily comparable by linear vector similarity between the corresponding embeddings produced by this method. Thus, this method provides a convenient way of finding words that similar to any given word in a generic sense.

Since the objective of Query Expansion (QE) is to find words that are semantically as well as to some extent lexically related to a given user query, it should be possible to leverage word embeddings in order to improve QE effectiveness.

Let Q be a given user query consisting of the words $q_1, q_2, \ldots, q_m$. Let $w_e^{(i)}{}_1, w_e^{(i)}{}_2, \ldots, w_e^{(i)}{}_k$ be the k nearest neighbours (kNN) of q_i in the embedding space. Then, these $w_j^{(i)}$s constitute a set of obvious candidates from which terms may be selected and used to expand Q. It is obviously desirable to consider expansion term which are closer to the query as a whole than a particular term in the query, so that the concept expressed by the query about the user information need can be taken care of. [1]

Word embeddings has show some promise in some specialised applications (e.g., clinical decision support [33] and sponsored search [34]) and for cross-lingual retrieval [35]. But use of word embeddings for QE seems not to have been explored within the standard ad hoc retrieval task setting. Our goal in this work is to study how word embeddings may be applied to QE for ad hoc retrieval. Specifically, we are looking for answers to the following questions.

[1]This idea has been used in a number of traditional, effective QE techniques, e.g., RM3.

1. Does QE using the nearest neighbours of query terms improve retrieval effectiveness?
2. If yes, is it possible to characterise the queries for which this QE method does / does not work?
3. How does embedding based QE perform compared to an established QE technique like RM3 [36]?

In this study, we explore simple KNN bases and an incremental KNN embedding based QE methods. These methods are described in more detail in the next section. Experiments on a number of TREC collections (Section 3.3) shows that these QE methods shows significant improvements in retrieval effectiveness when compared to using the original, unexpanded queries. However, it cannot outperform the performance of RM3. We discuss these results in greater detail in Section 3.4. Section **??** concludes the paper.

3.2 Word Embedding based Query Expansion

In this section, we describe more precisely the basic, word embedding based QE method with the nearest neighbor method inspired by [37]. The nearest neighbors are computed in an incremental fashion as elaborated below. It is to be noted that the basic principle behind word embedding based QE is to use terms that are close to the query terms in the embedding space. In this study, we have used generic vector sum composition, where each query is represented by the centroid of the word embedding computed from the word vectors of its each component terms.

3.2.1 Composition of Terms and K-Nearest Neighbor

Consider the TREC query 301: *International Organized Crime*. If we search for similar terms of the individual query terms (similar words of *International*, *Organized* and *Crime*) we may end up with some new terms which are less associated to the actual information need. To achieve something more purposeful, it would be nice to have some expansion words which are related to the query as a whole (similar words of *International Organized Crime*). As all the terms have a vector corresponding to it, we can easily get the effect of composition of query terms by composing the vectors corresponding to it.

Deriving Composed Words Given a query of k terms, $Q = \{q_1, \ldots, q_k\}$, we compose the vectors corresponding to the query words in a linear chain from left to right, i.e., for a k term query, we obtain the centroid vector for the query by:

$Q_c = \sum_{i=1}^{k} v(q_i)$

Finding K-Nearest expansion terms Following the above approach, for a given query, the $Q = \{q_1, \ldots, q_k\}$, the expanded form will be the following:

$$Q_{exp} = NN(Q_c) \tag{3.1}$$

where *NN()* is a method that returns top N terms which are similar to the vector Q_c using cosine similarity.

The Q_{exp} terms contains all the candidate expansion terms obtained by sorting the terms according to cosine similarity and choosing the top N similar terms. This is the normal KNN approach that is followed in this study. We propose a incremental KNN approach next to mitigate query drift.

3.2.2 Pre-retrieval incremental Nearest Neighbor based approach for QE

The incremental nearest neighbor method, is a simple extension of the previous method. Instead of computing the nearest neighbor expansion term, for a query term, in a single step, we follow an incremental procedure. The first assumption in this method is that, the most similar expansion terms have comparatively lower drift than the terms occurring later in the list in terms of similarity. Since the top similar terms in the vocabulary are contender in becoming the expansion query terms, it can be assumed that these terms are also similar to each other, in addition to being similar to the query term. Following the above assumption a iterative process pruning of terms from the vocabulary is done at each step for every the terms in . At the first step, we compute the nearest neighbors of the query under consideration using the query centroid vector in Equation . We prune this set by a fixed amount K. Next we consider the most similar term in the pruned nearest neighbor set and make it the query term and repeat the above procedure for a fixed l number of steps. At each step the nearest neighbors set is computed based on the nearest neighbor of the previous set and the set is pruned. Essentially, by following the above procedure, we are constraining the nearest neighbor to be similar to each other in addition to being similar to the query term. A high value of $l \geq 10$ may lead to risk of query drift. A low value of $l \leq 2$ essentially performs similar to normal pre-retrieval model. By careful tuning we choose $l = 5$ as the number of iterations for this method. After l iteration, the terms are sorted according to the similarity with the query centroid vector and top N terms are returned

3.2.3 Retrieval

The expanded query is formed depending only on the original query terms (Q) and how they appear in the collection as a whole. Thus the actual retrieval can be done using any standard retrieval model. For our experiment, we used Language Model with Jelinek Mercer smoothing [38]. Linear smoothing parameter λ is tuned. The expansion term weights are assigned by normalizing the expansion term score (similarity with respect to the query centroid vector) by the total score obtained by summing over all top N expansion terms. This weights are multiplied with $1 - \lambda$ and added with the the original query scaled by λ to give the expanded query.

q

Table 3.1: Dataset Overview

Document Collection	Document Type	#Docs	Query Fields	Query Set	Query Ids	Avg qry length	Avg # rel docs
3*TREC	4*News	4*528,155	4*Title	TREC 6 ad-hoc	301-350	2.48	92.2
				TREC 7 ad-hoc	351-400	2.42	93.4
1*Disks 4, 5				TREC 8 ad-hoc	401-450	2.38	94.5
				TREC Robust	601-700	2.88	37.2
2*WT10G	2*Web pages	2*1,692,096	2*Title	TREC 9 Web	451-500	3.46	52.3
				TREC 10 Web	501-550	4.62	67.2

3.3 Evaluation

We explored the effectiveness of our proposed method on the standard ad-hoc task using TREC collection as well as on the TREC web collection. Preciously, we use the documents from TREC disk 4 and 5 with the query sets TREC 6, 7, 8 and Robust. For the web collection, we use WT10G collection. The overview of the dataset used is presented in Table 3.1. We implemented our method [2] using the Apache licensed Lucene search engine[3]. We used lucene distributed standard language model with linear smoothing [38].

3.3.1 Experimental Setup

Indexing and Word Vector Embedding At the time of indexing of the test collection, we removed the stopwords following the SMART[4] stopword-list. Porter stemmer is used for stemming of words. The stopword removed and stemmed index is then dumped as raw text for the purpose of training the neural network of *Word2Vec* framework. The vectors are embedded in an abstract 200 dimensional space with negative sampling using 5 word window on continuous bag of words model. These are as par the parameter setting prescribed in [39]. We removed any words that appear less than three times in the whole corpus.

Parameter setting In all our experiments, we only use the "title" field of the TREC topics as queries.The linear smoothing parameter λ was empirically set to 0.6 after varying it in the range $[0.1, 0.9]$.

3.3.2 Results

Table 3.2 shows the performance of the proposed method, compared with the baseline LM model and feedback model RM3 [36]. It can be seen that the QE methods based on word embeddings almost always outperforms the LM baseline model (often significantly). There does not seem to be a major difference in performance between the three variants, but the

[2]Available from `anonymysed-url` upon acceptance
[3]https://lucene.apache.org/core/
[4]ftp://ftp.cs.cornell.edu/pub/smart/

Dataset	Method	Parameters			Metrics		
		K	$feedbackdocs$	α	MAP	GMAP	P@5
4*TREC-6	LM	-	-	-	0.2303	0.0875	0.3920
	Pre-ret	120	-	0.55	0.2311	0.087	0.4280
	Increm.	120	-	0.65	0.2376	0.0942	0.4200
	RM3	30	70	-	$0.2634^{k,p,i}$	0.0957	0.4360
5*TREC 7	LM	-	-	-	0.1750	0.0828	0.4080
	Pre-ret	120	-	0.60	0.1800*	0.0896	0.4160
	Increm.	60	-	0.90	0.1888*	0.1041	0.4400
	RM3	20	70	-	$0.2151^{k,p,i}$	0.1038	0.4160
5*TREC 8	LM	-	-	-	0.2373	0.1318	0.4320
	Pre-ret	120	-	0.65	0.2441	0.1406	0.4440
	Increm.	70	-	0.90	0.2613*	0.1565	0.4960
	RM3	20	70	-	$0.2701^{k,p,i}$	0.1543	0.4760
5*Robust	LM	-	-	-	0.2651	0.1710	0.4424
	Pre-ret	90	-	0.65	0.2759	0.1769	0.4646
	Increm.	120	-	0.60	0.2935*	0.1972	0.5051
	RM3	20	70	-	$0.3304^{k,p,i}$	0.2177	0.4949
5*WT10G	LM	-	-	-	0.1454	0.0566	0.2525
	Pre-ret	80	-	0.6	0.1718	0.0660	0.3027
	Increm.	80	-	0.60	0.1753	0.0770	0.3030
	RM3	20	70	-	$0.1915^{k,p,i}$	0.0782	0.3273

Table 3.2: MAP for baseline retrieval and various QE strategies. A * in the kNN and Increm. columns denotes a significant improvement over the baseline. A k, i, and p in the RM3 column denotes a significant improvement over the kNN and Incremental QE techniques. The parameter K for RM3 is the number of terms used for QE. Significance testing has been performed using paired t-test with 95% confidence.

incremental method seems to be the most consistent in producing improvements. However, RM3 appears to be significantly superior for all the query sets.

3.4 Summary

Distributed neural language model word2vec, possesses the semantic and contextual information. This contributes to the performance improvement over text similarity based baseline for each of the two methods Query expansion intuitively calls for finding terms which are similar to the query, and terms which occurs frequently in the relevant documents (captured from relevance feedback). In the proposed embedding based QE techniques, the terms which are similar to the query terms in the collection-level abstract space are considered as the expansion terms. Precisely, in the K-NN based QE method, expansion terms are chosen from the entire vocabulary, based on the similarity with sum-composed query terms. However this techniques fails to capture the other features of potential expansion terms, such as terms, frequently co-occurring with query terms. Experiments on the TREC ad-hoc and wec datasets shows that the performance of RM3 is significantly better than the proposed methods which indicates that the co-occurrence statistics is more powerful than the similarity in the abstract space.

The incremental approach for K-NN based QE perform better as compared to pre-retrieval QE with K-NN. The reason is both the superior methods try to minimize the effect of query drift that occurs due to generalization imposed by word2vec. For the incremental approach, terms which are similar both semantically and contextually are given precedence by considering incremental computation of nearest neighbor. Such terms should rank higher in terms of similarity with respect to the query and hence the drift is reduced because the methods needs overall similarity to be high. Thus similar terms are only being searched in the document with higher vocabulary overlap with the query. Thus the generalization effect can be mitigated to an extent. However it can't be completely removed in the current setting, because the original Word2Vec model is trained on the entire corpus. Another pitfall in word2vec framework is the composition of phrases. Linear average of the word vectors of individual term often may lead to results which are same or marginally better when neighbors of individual terms are included in the query expansion list. So the approach to capture the entire query sense into the word vector for finding the nearest neighbor needs to be studied more rigoruously in future work.

A drawback of the incremental KNN computation compared with pre-retrieval KNN QE is that the former takes more time, due to iterative pruning step involved.

An obvious future work, in this direction, is to apply the embeddings in combination with co-occurrence based techniques (e.g. RM3). In this work, we restrict the use of embeddings only to select similar words in the embedded space. Thus a possible future scope is to use the embeddings exhaustively for utilizing other aspects of the embedded forms. In our experiments, we trained the neural network over the entire vocabulary. A possible future work is thus the investigation of local training of word2vec from pseudo-relevance documents which might get rid of the generalization effect when trained over the whole vocabulary.

Chapter 4

Event detection and Argument Extraction for Biomedical Documents

4.1 Introduction

In the recent years, there is an exponential growth of Bio-medical literature [40],[41],[42]. As a result, extensive focus has been devoted by the research community to develop improved **information extraction** techniques for biomedical event extraction. Efforts are being made to identify hierarchical relation between specialized keywords and statements bearing biological meaning from scientific literature[40],[43].

4.1.1 Problem Definition

An **event** (*Plan_proc*, *regulation*, *growth* in Fig 1.) is defined as a recursive interaction between **entities** (*L-NAME*, *adenocarcinoma*) and/or **events**. It will have an **event trigger**(treatment, inhibited and growth) and one or several **arguments**(*Theme*(Th), *Cause*, *Instrument*(Informationnist)). Arguments represents the relationship among the event under consideration and the entities and/or events it interacts with. An **Entity** is defined as a representative belonging to a particular biological organization of various hierarchical level. Each entity possesses a type or a class (e.g., in Figure 4.1, *treatment* is an entity of class **Plan_proc**) that depicts the level of hierarchy it belongs to.

The problem of event extraction in biomedical documents involves two basic steps: 1. **Event Identification and classification** 2.**Event enrichment through Argument detection**.

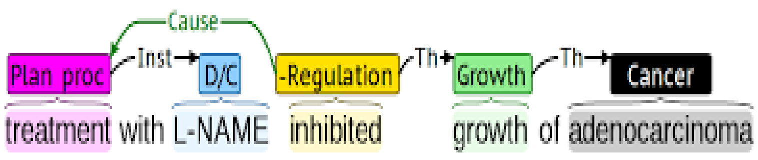

Figure 4.1: An example of event extraction in Cancer Genetics

4.1.2 Our Contribution

In addressing the event detection and classification problem, one of our major goals is to design the classifier with a low dimensional feature set enriched in semantic and contextual information and reduce the need of extensive feature engineering. For this purpose, the recent advances in word embeddings are explored, where generation of word vectors uses a human-interference free distributed neural framework and is capable of capturing distributional characteristics of words and phrases [18],[44],[20]. Such a language model has made it possible to learn contextual and semantic information in text and generate a low dimensional embedding(compared to vocabulary size) in space for each vocabulary entry.

The salient contributions of this paper are as follows: (a) Feature representation using word embedding (along with other features or as a standalone) for trigger detection and event classification in Biomedical corpora. This has not been well explored before in understanding Biomedical text. (b) Use of CNN-based architecture for extracting interaction between event and the context in a better way through use of convolution layers, (c) Exploring different classification architectures and parameter settings, which can successfully exploit the feature representation given by word embeddings and finally, (d) presenting a generic event classification framework for different BioNlp tasks. The proposed framework for event classification is evaluated on two tasks of Bio-NLP-Shared Task 2013: Cancer Genetics (CG) [42] and Genia Event Extraction (GE)[45]. Both the datasets suffer from high class imbalance problem which poses a serious challenge to design an efficient classifier. (e) Designing a Biological dependency parser for biomedical knowledge extraction as arguments between events an entities by leveraging information stored in form of word embedding, parts of speech and phrase level syntactic dependency information. The argument extraction framework is evaluated on Cancer Genetics (CG) [42] dataset of BioNLP-ST2013. As in case of event classification. the argument detection problem also sufferes from serious class imbalance problem.

4.1.3 Related Works

Most of the previous works have centered around use of rule bases, SVM or Conditional Random Field based classification approach with hand encoded features for trigger detection and event classification. **Turku Event Extraction System (TEES)** 2.1 [46] tees (graphical approach) and **NaCTeM EventMine** [47] uses SVM based model for event extraction. In GE, **EVEX** [48], which performs a re-ranking step over TEES prediction, achieved the best performance in Bio-NLP ST13. [43] proposed a system for extraction indirect biomedical concepts from MEDLINE abstracts and their visualization called **FACTA+**, which has given some benchmark on accuracy of trigger detection and classification for GE event. [49] addresses the problem of word sense disambiguation in Biomedical event using an entropy based approach. [40] showed that, additional feature like n-gram or bag of words may be required along with word embeddings.

4.2 Method

As noted before, our feature representation module makes use of Word2Vec, proposed by [18], as word embedding technique . For generation of word vectors, a Word2Vec model pre-trained on PubMed corpus[1] is used. The Word2Vec model is retrained on CG and GE corpus with the following parameters : Negative Sampling : 15, Window Size: 5, feature dimension : 200 and continuous bag of word model (cbow).

4.2.1 Feature Representation for Event Detection

The task of trigger detection and event classification essentially requires a classifier which, given a term, classifies it to one of the event types and to a *no event* class if it is not a trigger. So the class labels consist of an extra class for *no event*. To detect multi-term triggers, the dataset is marked with Inside-Outside-Between (IOB) tags. A sliding Window model for feature representation is used. The window moves over the text and at each step it considers a context size of K. For a sentence of N terms, it considers $W_1, W_2, \ldots, W_K, \ldots, W_{2K}, W_{2K+1}$ for classification of the event type of the Kth word and the set of neighboring $2K$ words is as it's context. Special words $START$ and $STOP$ are used for boundary condition Two different types of feature representation are explored in this study:

• *Term Embedding* (TE): The basic feature representation is to replace each token by it's word embedding obtained from the Word2Vec model. So the input size is $(2K + 1) \times 200$ which is reshaped to a row-vector such that the first 200 entries corresponding to the word embedding of the first word and so on. This reshaping step is not required in case of CNN.

• *Term + Parts of Speech(POS) Embedding* (TE+POS-E): In this case, the word embedding based feature set is appended with a POS embedding of the center word whose event type is to be determined. The dimension of this POS embedding L, is same as the total number of POS types that occurs in the dataset. The dimension of this feature representation is $(200 \times (2K + 1)) + L$. The basic intuition behind using POS tags, is to make the classifier learn the dependency of the class label on POS information.

4.2.2 Classification Architectures for Event Detection

Multi Layered Perceptron

The *Multi-Layered Perceptron* based Named Entity Recognition model proposed in [8] is extended here for the event tagging and classification purpose. For TE, the concatenated embedding input for the $2K+1$ words is given as input: $X = [X_1, \ldots, X_k, \ldots, X_{2K+1}]$. This classifier is denoted as MLP (TE) in the result section. The hidden layer response, h is given by

$$h = f(W_{ih}X + b), \tag{4.1}$$

[1]http://bio.nlplab.org/

where W_{ih} is the weight matrix mapping input X to h, b is the bias term. These symbols are kept consistent through out the paper. Different superscripts are added to denote parallel layer.

For TE + POS-E feature representation two different architectures for the input layer of MLP are explored:

• POS feature is concatenated with the term embedding. In this case
$X = [X_1, \ldots, X_k, \ldots, X_{2K+1}, POS_k]$
Mapping to hidden layer is same as in 1. This classifier is denoted as MLP (TE + POS-E 1) in the result section, i.e. Section-3.

• Term embedding and POS embedding are mapped separately to the hidden layer. The hidden layer outputs are merged to form the hidden layer response as follows.

$$h^1 = f(W^1_{ih}X + b^1), h^2 = f(W^2_{ih}X^2 + b^2) \tag{4.2}$$

$$h = [h^1, h^2] \tag{4.3}$$

where the inputs are $X^1 = [X_1, \ldots, X_k, \ldots X_{2K+1}]$ and $X^2 = POS_k$. This classifier is denoted as MLP (TE + POS-E 2) in Section-3.

The softmax probability for i^{th} class is computed as follows:

$$p(y_i|X) = \frac{\exp W_{hy_i}h}{\sum_{c=1}^{i=C} \exp W_{hy_c}h}, \tag{4.4}$$

where W_{hy} is the weight matrix mapping hidden to output layer and C is the total number of class labels. The cross entropy cost function is then obtained as:

$$J = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{p(y_i|X)}{\sum_{j=1}^{C} p(y_j|X)} \tag{4.5}$$

For MLP classifier, RMSprop optimizer [50] is used.

Convolutional Neural Network for Event Detection

CNN for event detection and classification is applied to the Term Embedding feature set, which can be thought as a $(2K + 1) \times 200$ 2D image. An architecture consisting of two 2D convolution-Pooling layers followed by a fully connected layer is used. First layer consists of 4 filters of dimension 2×41 (empirically chosen) forming the convolution layer, followed by a 2×2 max-pooling layer. Second convolution layer consists of 8 filters of dimension 1×21 (network half-ed by max-pooling), followed by a 1×2 max-pooling layer. For CNN, Adam [6] optimizer is used.

Two different CNN architectures are explored in this work, one with two sets of convolution-pooling, denoted as **CNN-2-Layer** in Section-3. In the second one, we remove the second layer of convolution-pooling and it is denoted as **CNN-1-Layer**. The main motivation behind using CNN is that it may be able to capture the interaction between the trigger words and the context in a better way because of convolution. Also it has lesser parameters than MLP.

All the classifiers are implemented with ReLU [51] activation and with Dropout value 0.25. The models are implemented in Keras [52] and the code will be made available on acceptance.

4.2.3 Designing a Biological Parser for argument extraction

The next goal is to design a neural parser which will mine for argument given an trigger of a particular event. An argument extraction problem is essentially mining for the dependency between events and entities. The relationship between events and entities essentially depends on the language properties.

The assumption of our approach in argument detection is that relationship between events in a particular sentence and entities or other events are confined withing that sentence. There are instances where there are relationship between entities and events belonging to different sentence. Mining arguments in such cases required anaphora resolution, which is skipped in this study. The neural parser designed in this study is a classifier which predicts the type of argument for every pair of event- entity or event-event pair. Our approach is essentially to design a classifier which will be able predict the type of argument like Theme, cause, etc or if it is an argument at all for every event-entity or event-event pair in a sentence.

The main intuition in designing the argument parser, is to use feature which can model the syntactic dependencies between the events and entities in the sentence. (i) **Word Embedding**, which essentially incorporate the contextual information is a good candidate for the feature set. The first 400 dimension of the feature vector is the word embedding of the trigger word of the event under consideration (X_{ev}), followed by word embedding of the entity (X_{en}). Since trigger or entity can have multiple tokens, normalized vector sum over the tokens is used as composition function. (ii) In argument detection, the type of **event and entity class** is extremely important. Some event classes possesses only particular types of arguments, and hence the classifier should leverage that. Also certain entity class like of the type organ usually represent argument of type location. An embedding for the event type (X_{ev}^{type}) and entity type (X_{en}^{type}) is incorporated next in the feature set. The embedding size for event type(entity type) is equal to the number of event classes(entity classes) $C_{ev}(C_{en})$. Since we are looking for dependency between events and entities, words like preposition, conjunction, adverbs, adjective, verb etc., either following the entity (if entity occurs before the event in the sentence) or preceding it (otherwise), can provide hints to the argument characterizing the interaction. For example in Figure 4.2, including the adverb like **dramatically** or preposition like **in** in the word vector feature as *connecting words* (X_{cw}) between events and entities or event might be helpful.

(iii) The **parts of speech** (POS) information of both the event and the entity is also crucial as parts of speech may provide hint regarding the type of entity and event. Two binary vectors X_{ev}^{pos} and X_{en}^{pos} with dimension equal to the number of Parts of speech (C_{pos})

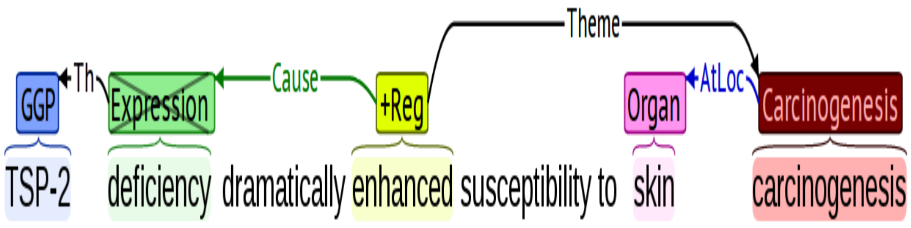

Figure 4.2: Event-Entity relation with arguments

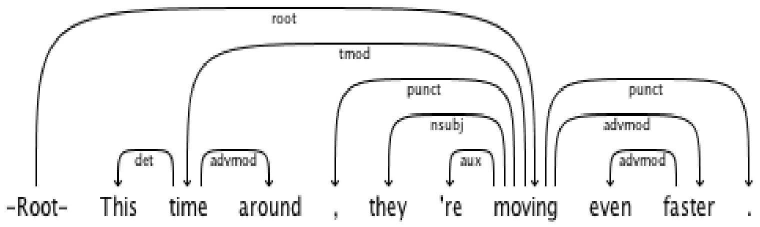

Figure 4.3: Dependency Parsing of a Sentence

is used as an embedding. The POS information is represented identically like bag of words representation. The POS embedding for an entity or event is thus a binary vector, where each bit represent a POS-type. If a term of a particular POS is present in the event or the entity, the corresponding bit is set to 1.

(iv) The goal of solving this information extraction problem is finding out if there exist any syntactic dependency between a particular event and an entity or between two events. **Language level dependency** can be obtained by paring the language dependency tree structure. There are several well known dependency parser for english language for example ***MaltParser*** [53] and ***Stanford Dependency Parser*** [54]. Several neural network based dependency parser has also been proposed in the recent years [55] [56]. Figure 4.3 shows the dependency tree output of *Stanford Neural Dependency Parser* [55].

Thus the transitional dependency parser, given a sentence outputs the syntactic dependency between the words. Such language level dependency should be present between the event terms and the entity terms as events and entities capture the concept of biological interaction expressed through the sentence. An event or an entity may be multi-word token. So we are looking for dependency between phrase and we leverage the word level dependency information into phrase level by using a bipartite graph model. So the type of dependency between each word in event and entity are encoded in a single vector. For each event -event pair or event-entity pair the dependency vector essentially contains the type of dependency that exist between various token of the two pair. We follow a simple approach of Bipartite graph checking to finding the dependencies, where the nodes of the graph are the tokens of the event or the entity. The edges is the dependency type(or no dependency) that exist between the nodes as obtained by parsing the dependency tree of the sentence to which the event or the entity belongs. The dimension of the the dependency embedding is equal to $C_{dep} + 1$, C_{dep} the total number of types of dependency relation X^{dep}.

Finally we design two classification architecture similar to the event detection step:

(i) A fully connected MLP (MLP-FC) is used to classify the argument given the concatenated feature vector input $X_{in} = [X_{ev}, X_{en}, X_{cw}, X_{ev}^{type}, X_{en}^{type}, X_{ev}^{pos}, X_{en}^{pos}, X^{dep}]$. The input is mapped to the hidden layer h using $h = f(W_h in X_{in} + b_h in)$ and mapped to the argument class using softmax function

$$p(y_i|X_{in}) = \frac{\exp W_{hy_i} h}{\sum_{c=1}^{i=C} \exp W_{hy_c} h}, \tag{4.6}$$

(ii) The embedding of different type are mapped with different weights and the responses are merged in the hidden layer. Finally again a softmax function is used to predict class probabilities. The features are group together as: $X_1 = [X_{ev}, X_{en}, X_{cw}]$, $X_2 = [X_{ev}^{type}, X_{en}^{type}]$, $X_3 = [X^{dep}, X_{ev}^{pos}, X_{en}^{pos}]$

$$h_{merged} = [f(W_{h_1 in_1} X_1 + b_{h_1 in}), f(W_{h_2 in_2} X_2 + b_{h_2 in}), f(W_{h_3 in_3} X_3 + b_{h_3 in}) \tag{4.7}$$

$$p(y_i|X_{in}) = \frac{\exp(W_{hy_i h_{merged}})}{\sum_{c=1}^{i=C} \exp(W_{hy_c} h_{merged})}, \tag{4.8}$$

The basic intuition behind this MLP architecture, is that, it tries to model the relation between similar features while mapping to the hidden layer. The relation between features of different types are captured in the softmax layer. Finally the categorical cost function defined in Equation 4.5 is optimized by SGD or its suitable variants.

4.2.4 Dataset and Preprocessing

The CG dataset[2] consists of 38 event types and a total of 46,495 training samples and 15,115 test samples. The GE dataset [3] consists of 24 event types and a total of 50,011 training samples and 51,045 test samples. Both the dataset show high class imbalance. For some classes there are less than 5 training instances which make the problem extremely challenging. For both the dataset, the development set is used as test set in this study. After tokenization and POS tagging, the datasets are marked with *IOB* tags for detection of multi-word triggers. The argument extraction framework using the biomedical document relation parser proposed in this study is evaluated on the Cancer Genetics Data.

4.2.5 Class Imbalance

To mitigate the effect of class imbalance problem, the following steps are taken at preprocessing and classification steps: (a) The terms which are exclusively tagged as *entity* in the dataset are not considered for classification. (b) In the training set, some terms with particular POS (like Determiner(DT), Connecting Conjunction(cc) etc.) appear as triggers with very low frequency (≤ 5) and hence, they are not considered for classification. (c) Class weights are assigned during classification. For Genia Event task, the classes which have at

[2]http://2013.bionlp-st.org/tasks/cancer-genetics
[3]http://bionlp-st.dbcls.jp/GE/

Classifier	Avg. Precision	Avg. Recall	Avg. F1-Score
MLP(TE)	0.93260	0.8008	0.8533
MLP(TE-POS-E 1)	0.9308	0.9311*	0.9298*
MLP(TE-POS-E 2)	0.9445*	0.8795	0.9057
CNN-1-Layer	0.9446	0.8308	0.8749
CNN-2-Layer	0.9435	0.7029	0.7875

Table 4.1: Performance of different classifier on Genia Event dataset

least 10 training samples are given higher weights than to the other classes, especially the *Not Event* class. For Cancer Genetics, the classes with at least 40 training examples are given higher weights. The terms which are pruned away are however considered as part of the context for the other words.

4.3 Experimental Results

4.3.1 Trigger Detection and Event Classification

Table 4.1 shows the performance of the classifiers evaluated on Genia Event (GE) dataset whereas, Table 4.2 shows the performance on Cancer Genetics (CG) dataset.

The MLP(TE-POS-E 1) classifiers for GE (Table 4.1) outperforms the other classifiers in recall and F1-score measures. Note that GE has very low number of events as compared to *Not Events*. The performance of CNN-1-Layer is better than MLP(TE) because it has less number of parameters. Both MLP(TE-POS-E 1) and MLP(TE-POS-E 2) show overall better performance than MLP(TE) and two CNN-based architectures which indicates that POS information be important in GE event. Note that the classifiers implemented in this work have outperformed the best trigger detection and event classification accuracy of 78.9% reported in [49] and 0.7790 (precision) reported in [43].

A contrasting performance of the classifiers is observed for CG (Table 4.2). For CG dataset, the overall performances of CNN-1-Layer and CNN-2-Layer are better than that of MLP(TE), MLP(TE-POS-E 1) and MLP(TE-POS-E 2). The POS information is still helpful as MLP(TE-POS-E 1) and MLP(TE-POS-E 2) perform better than MLP(TE). The CNN frameworks work better for CG dataset because it has less number of parameters. Also, due to more number of event classes, there is much more interaction between the trigger and the context window, which supports the intuition behind using CNN. Figure 4.4 shows some prediction instances of CNN 2-Layer

Classifier	Avg. Precision	Avg. Recall	Avg. F1-Score
MLP(TE)	0.8837	0.8060	0.8334
MLP(TE-POS-E 1)	0.8932	0.8567	0.8698
MLP(TE-POS-E 2)	0.9028	0.8582	0.8724
CNN-1-Layer	0.9150	0.9183	0.9154
CNN-2-Layer	0.9185*	0.9203*	0.9168*

Table 4.2: Performance of different classifier on Cancer Genetics dataset

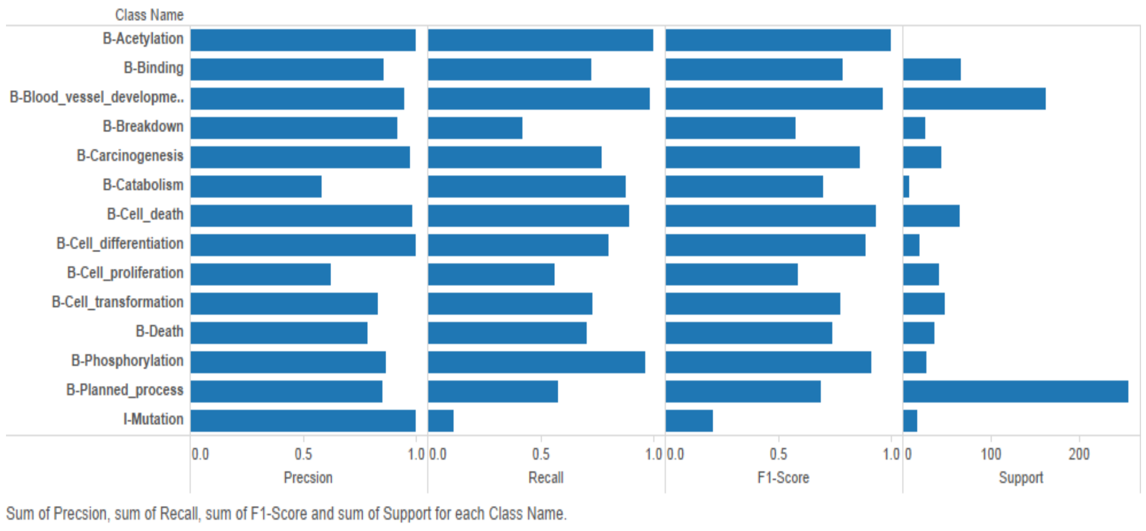

Figure 4.4: Some instances of event detection by CNN

4.3.2 Argument Extraction

The different classifier architecture proposed for argument extraction are explored next. The word embedding of entity-event pair or event-event pair (X_{ev}, X_{ev} or X_{en}) are obviously indispensable for argument extraction. An attempt to verify the above statement by removing either of the two features (event-entity word vector and dependency) results in poor performance on argument classification. The event and argument type is also useful as some of the events don't possess certain arguments. The necessity of POS information and connecting words (cw) are under question and it is rigorously verified below. First we explore the classifier which doesn't uses the connecting words in the word vector feature and the results are shown in Table 4.3. The precision for both MLP(FC) and MLP(Merging) is higher that as reported in [46] and [47], but both recall and F1-score are inferior when compared to [46]. The F1-score of both the classifier is better than [47].

Next we explore the feature set from which POS has been excluded and the results are shown in Table 4.4. MLP(FC) outperforms both [46] and [47] in precision and F1-score, but recall is still inferior. As before MLP(Merging) perfoms inferior to MLP(FC).

Including both POS and connecting words leads to a improved performance in recall and

Classifier	Avg. Precision	Avg. Recall	Avg. F1-Score
MLP(FC)	0.8203*	0.4261	0.5580
MLP(Merging)	0.8175	0.4069	0.5406

Table 4.3: Performance of different classifier for argument extraction in CG. Connecting Word feature excluded

Classifier	Avg. Precision	Avg. Recall	Avg. F1-Score
MLP(FC)	0.8128	0.4299	0.5589
MLP(Merging)	0.7769	0.4176	0.5400

Table 4.4: Performance of different classifier for argument extraction in CG. POS feature excluded

as a result both MLP(FC) and MLP(Merging) outperforms [46] and [47] by huge margin with respect to precision and F1-score as shown in Table 4.5. The recall of both the classifier is nearly same as that in [46] and better than [47].

Classifier	Avg. Precision	Avg. Recall	Avg. F1-Score
MLP(FC)	0.7402	0.5025*	0.5930*
MLP(Merging)	0.7465	0.4933	0.5858

Table 4.5: Performance of different classifier for argument extraction in CG with all the features

Figure 4.5 shows the class wise performance of the MLP(FC) with all the features. From the above analysis there are few interesting observation to be made. Using Word vectors and Dependency information for designing the biomedical relation parser out performs the existing benchmark in cancer genetics set by [46] and [47]. The following feature analysis shows that POS and connecting words are also important feature and they capture patterns useful in mining event arguments. What is more interesting is that MLP(FC) performs better that MLP(merging) consistently. The reason may be attributed to the non-linear interrelation between the lexical features, event or entity types and word embedding features captured by MLP(FC) which is useful in argument extraction. For some classes with very small training sample the classifier proposed here fails to detect the class. Application of high class weighting to such classes inherently leads to over fitting and hence lowers classification performance for other classes significantly. Figure 4.6 shows the overall framework propesed in this work for information extraction in Cancer Genetics.

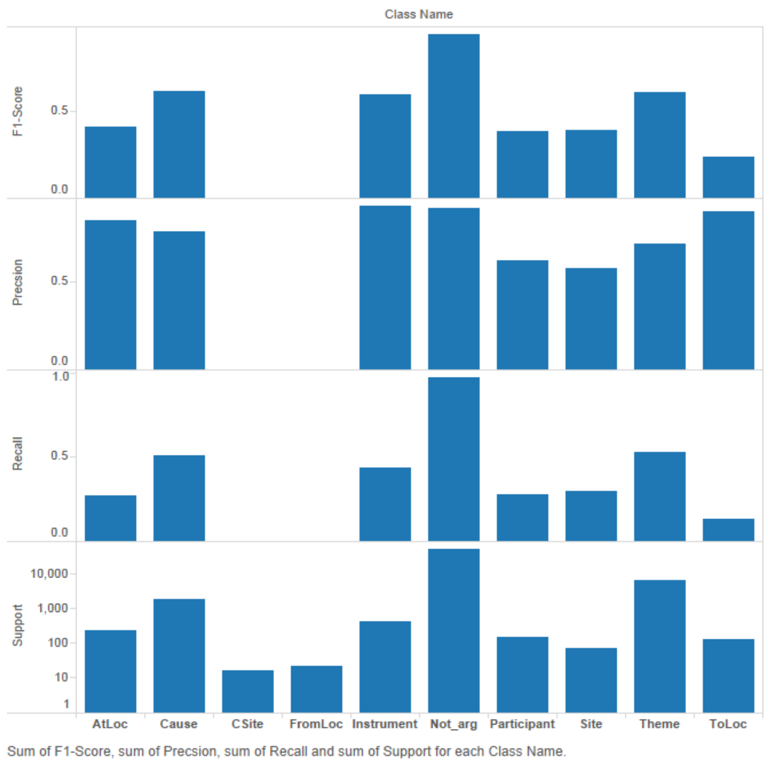

Figure 4.5: Classwise Performance of Argument extraction

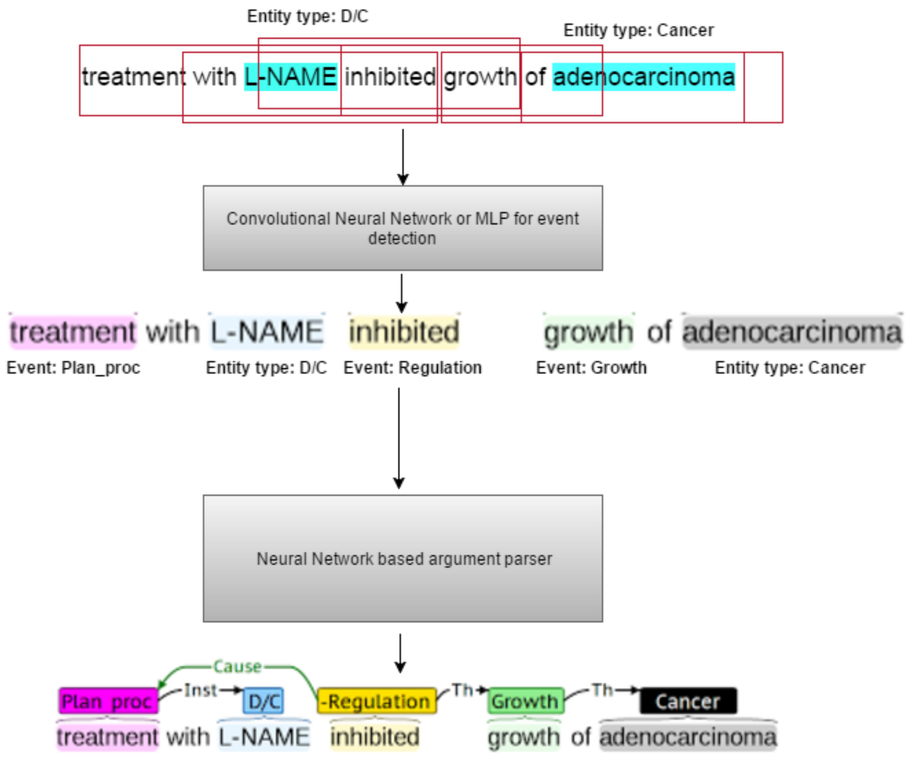
Entity type: D/C
Entity type: Cancer
treatment with L-NAME inhibited growth of adenocarcinoma
Convolutional Neural Network or MLP for event detection
treatment with L-NAME inhibited growth of adenocarcinoma
Event: Plan_proc
Entity type: D/C
Event: Regulation
Event: Growth
Entity type: Cancer
Neural Network based argument parser
Cause
Inst
Th
Th
Plan_proc
D/C
-Regulation
Growth
Cancer
treatment with L-NAME inhibited growth of adenocarcinoma

Figure 4.6: The overall Cancer Genetics Information extraction framework proposed

4.4 Summary

This chapter presents a novel neural-based framework for event detection, classification and argument enrichment of events for biomedical information extraction. Use of word embedding in representing features could be considered as a distinct contribution for the given task. The MLP and CNN based architectures show their potential in extracting the interaction between events and entities for event classification. This has motivated further investigation into the second sub-task, i.e., event enrichment with arguments with word embedding. The parser architecture for Biomedical document relation parsing with Word embedding and dependency parsing has not been explored before and the above experiments shows how word embedding captures syntactic regularities in the sentence. It is to be noted that the lexical features like POS and language dependency along with the syntactic features in word embedding, captures consistent patterns and leads to the superior result of the parser. The class imbalance problem makes the classification task and argument extraction task difficult. Techniques like sub-sampling of *Not Event* class or over-sampling of event instances may be further investigated. However, such sampling might affect the sequence information in IOB tagging. Except the classes with too few training samples, the proposed classifiers and the biomedical document parser perform remarkably well and it clearly shows the effectiveness of word embedding as feature and efficacy of neural framework.

Chapter 5

Conclusion

Neural learning in text processing has shown great promise in recent developments and this study also confirms that. Neural language model, generating word embedding indeed captures both semantic and syntactic relationships as it is observed in various problems explored in this study. But an important drawback of word embedding framework is the absence of composition function which can be use for representation of large text documents. *Doc2Vec* has alleviated it to an extent, but when it comes to document classification, it is still lagging behind the traditional TF-IDF representation. The future endeavor for document embedding is to redefine the composition function or learning the composition function so that paragraph or document embeddings can be learnt for document classification.

Also as observed in automatic query expansion, our model, isn't able to perform better for two reasons, composition and lack of global sense which can't be captured by *Word2Vec*. The document or collection level co-occurances are not captured on *Word2Vec*, hence only terms which are semantically related are captured in query expansion. But RM3 clearly takes into account the co-occurrence statistics over a document level or collection level and hence terms which are not semantically similar but co-occurs often in pseudo-relevant documents are also included. Thus the question that should be address in our future works is how to augment this document level co-occurrence information along with the semantic information already captured in *Word2Vec*. The consistant better performance of incremental KNN over simple KNN, motivates such investigation further.

The problem of information extraction is dependent more on semantic and lexical relationship between terms than on mere co-occurrence statistics. The CNN and MLP for event detection and argument extraction have set new benchmarks for the argument extraction problem.

The problems explored in this study have answered the questions we have started with: How effective is the neural network framework along with neural embedding *Word2Vec*? From the above study it is clear that *Word2Vec* is an effective feature representation for terms. But there is still obstacles when extending it to model variable sized document or paragraphs. Neural Network models have emerged as very effective framework, the superior performance of CNN and MLP stands as a proof to that. But there are still challenges in designing problem specific composition function for *Word2Vec* when it comes to problem of document embedding and information retrieval. The reason for this is very simple, unlike information extraction task which is proposed as supervised classification problem, document embedding and query expansion are not supervised task. Though document classification is a supervised task, learning general document representation is unsupervised and hence simple composition function will not be able to achieve the desired objective.

Having said that, the Neural Network framework and word embedding have shown positive hope in many areas of text processing. Research efforts should be given to explore improved techniques for composition and ways to incorporate task specific information in word embedding.

Bibliography

[1] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines*, vol. 3. Pearson Upper Saddle River, NJ, USA:, 2009.

[2] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, 2010.

[3] I. J. Goodfellow, D. Warde-Farley, M. Mirza, A. C. Courville, and Y. Bengio, “Maxout networks.,” *ICML (3)*, vol. 28, pp. 1319–1327, 2013.

[4] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting.,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[5] T. Tieleman and G. Hinton, “Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude,” *COURSERA: Neural Networks for Machine Learning*, vol. 4, no. 2, 2012.

[6] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[7] M. D. Zeiler, “Adadelta: an adaptive learning rate method,” *arXiv preprint arXiv:1212.5701*, 2012.

[8] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa, “Natural language processing (almost) from scratch,” *The Journal of Machine Learning Research*, vol. 12, pp. 2493–2537, 2011.

[9] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, 2008.

[10] C. N. dos Santos and M. Gatti, “Deep convolutional neural networks for sentiment analysis of short texts.,” in *COLING*, pp. 69–78, 2014.

[11] D. Zeng, K. Liu, S. Lai, G. Zhou, J. Zhao, *et al.*, “Relation classification via convolutional deep neural network.,” in *COLING*, pp. 2335–2344, 2014.

[12] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” *arXiv preprint arXiv:1404.2188*, 2014.

[13] S. Poria, E. Cambria, and A. Gelbukh, “Deep convolutional neural network textual features and multiple kernel learning for utterance-level multimodal sentiment analysis,” in *Proceedings of EMNLP*, pp. 2539–2544, 2015.

[14] Y. Kim, “Convolutional neural networks for sentence classification,” *arXiv preprint arXiv:1408.5882*, 2014.

[15] M. Kågebäck, O. Mogren, N. Tahmasebi, and D. Dubhashi, “Extractive summarization using continuous vector space models,” in *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pp. 31–39, Citeseer, 2014.

[16] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *JASIS*, vol. 41, no. 6, p. 391, 1990.

[17] T. Hofmann, “Probabilistic latent semantic indexing,” in *Proc. SIGIR*, pp. 50–57, ACM, 1999.

[18] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, pp. 3111–3119, 2013.

[19] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” *arXiv preprint arXiv:1405.4053*, 2014.

[20] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation.,” in *EMNLP*, vol. 14, pp. 1532–1543, 2014.

[21] H. Larochelle and Y. Bengio, “Classification using discriminative restricted boltzmann machines,” in *Proceedings of the 25th international conference on Machine learning*, pp. 536–543, ACM, 2008.

[22] P. Pantel and D. Lin, “Document clustering with committees,” in *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 199–206, ACM, 2002.

[23] D. A. Evans and C. Zhai, “Noun-phrase analysis in unrestricted text for information retrieval,” in *Proceedings of the 34th annual meeting on Association for Computational Linguistics*, pp. 17–24, Association for Computational Linguistics, 1996.

[24] H.-T. Zheng, B.-Y. Kang, and H.-G. Kim, “Exploiting noun phrases and semantic relationships for text document clustering,” *Information Sciences*, vol. 179, no. 13, pp. 2249–2262, 2009.

[25] T. Tomokiyo and M. Hurst, "A language model approach to keyphrase extraction," in *Proceedings of the ACL 2003 workshop on Multiword expressions: analysis, acquisition and treatment-Volume 18*, pp. 33–40, Association for Computational Linguistics, 2003.

[26] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *the Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[27] C. Xing, D. Wang, X. Zhang, and C. Liu, "Document classification with distributions of word vectors," in *Asia-Pacific Signal and Information Processing Association, 2014 Annual Summit and Conference (APSIPA)*, pp. 1–5, IEEE, 2014.

[28] E. Loper and S. Bird, "Nltk: The natural language toolkit," in *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics - Volume 1*, ETMTNLP '02, (Stroudsburg, PA, USA), pp. 63–70, Association for Computational Linguistics, 2002.

[29] H. Guan, J. Zhou, and M. Guo, "A class-feature-centroid classifier for text categorization," in *Proceedings of the 18th international conference on World wide web*, pp. 201–210, ACM, 2009.

[30] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[31] F. Chollet, "keras." `https://github.com/fchollet/keras`, 2015.

[32] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[33] T. Goodwin and S. M. Harabagiu, "UTD at TREC 2014: Query expansion for clinical decision support," in *Proc. TREC 2014*, 2014.

[34] M. Grbovic, N. Djuric, V. Radosavljevic, F. Silvestri, and N. Bhamidipati, "Context- and content-aware embeddings for query rewriting in sponsored search," in *Proc. SIGIR 2015*, pp. 383–392, 2015.

[35] I. Vulic and M. Moens, "Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings," in *Proc. SIGIR '15*, pp. 363–372, 2015.

[36] N. Abdul-Jaleel, J. Allan, W. B. Croft, O. Diaz, L. Larkey, X. Li, M. Smucker, and C. Wade, "Umass at trec 2004: Novelty and hard," in *Proc. TREC*, 2004.

[37] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[38] C. Zhai and J. Lafferty, "A study of smoothing methods for language models applied to information retrieval," *ACM Trans. Inf. Syst.*, 2004.

[39] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. NIPS '13*, pp. 3111–3119, 2013.

[40] C. Li, R. Song, M. Liakata, A. Vlachos, S. Seneff, and X. Zhang, "Using word embedding for bio-event extraction," *ACL-IJCNLP 2015*, p. 121, 2015.

[41] D. Sinha, U. Garain, and S. Bandyopadhyay, "Event extraction from cancer genetics literature," in *Advances in Pattern Recognition (ICAPR), 2015 Eighth International Conference on*, pp. 1–6, IEEE, 2015.

[42] S. Pyysalo, T. Ohta, and S. Ananiadou, "Overview of the cancer genetics (cg) task of bionlp shared task 2013," in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 58–66, Citeseer, 2013.

[43] Y. Tsuruoka, M. Miwa, K. Hamamoto, J. Tsujii, and S. Ananiadou, "Discovering and visualizing indirect associations between biomedical concepts," *Bioinformatics*, vol. 27, no. 13, pp. i111–i119, 2011.

[44] Y. Goldberg and O. Levy, "word2vec explained: Deriving mikolov et al.'s negative-sampling word-embedding method," *arXiv preprint arXiv:1402.3722*, 2014.

[45] J.-D. Kim, Y. Wang, and Y. Yasunori, "The genia event extraction shared task, 2013 edition-overview," in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 8–15, Association for Computational Linguistics, 2013.

[46] J. Björne and T. Salakoski, "Tees 2.1: Automated annotation scheme learning in the bionlp 2013 shared task," in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 16–25, 2013.

[47] M. Miwa and S. Ananiadou, "Nactem eventmine for bionlp 2013 cg and pc tasks," in *Proceedings of BioNLP Shared Task 2013 Workshop*, pp. 94–98, 2013.

[48] K. Hakala, S. Van Landeghem, T. Salakoski, Y. Van de Peer, and F. Ginter, "Evex in st'13: Application of a large-scale text mining resource to event extraction and network construction," in *Proceedings of the BioNLP Shared Task 2013 Workshop*, pp. 26–34, Association for Computational Linguistics, 2013.

[49] D. Martinez and T. Baldwin, "Word sense disambiguation for event trigger word detection in biomedicine," *BMC bioinformatics*, vol. 12, no. 2, p. 1, 2011.

[50] Y. N. Dauphin, H. de Vries, J. Chung, and Y. Bengio, "Rmsprop and equilibrated adaptive learning rates for non-convex optimization," *arXiv preprint arXiv:1502.04390*, 2015.

[51] R. Livni, S. Shalev-Shwartz, and O. Shamir, "On the computational efficiency of training neural networks," in *Advances in Neural Information Processing Systems*, pp. 855–863, 2014.

[52] F. Chollet, "Keras." `https://github.com/fchollet/keras`, 2015.

[53] J. Nivre, J. Hall, and J. Nilsson, "Maltparser: A data-driven parser-generator for dependency parsing," in *Proceedings of LREC*, vol. 6, pp. 2216–2219, 2006.

[54] M.-C. De Marneffe and C. D. Manning, "The stanford typed dependencies representation," in *Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pp. 1–8, Association for Computational Linguistics, 2008.

[55] D. Chen and C. D. Manning, "A fast and accurate dependency parser using neural networks.," in *EMNLP*, pp. 740–750, 2014.

[56] C. Dyer, M. Ballesteros, W. Ling, A. Matthews, and N. A. Smith, "Transition-based dependency parsing with stack long short-term memory," *arXiv preprint arXiv:1505.08075*, 2015.