Deep Learning for text in limted data settings

Bhargav Lad
Department of Computer Science
Lakehead University
Thunder Bay, Canada
bamratbh@lakeheadu.ca

Jinan Fiaidhi
Department of Computer Science
Lakehead University
Thunder Bay, Canada
jfiaidhi@lakeheadu.ca

Pathikkumar Patel
Department of Computer Science
Lakehead University
Thunder Bay, Canada
ppatel73@lakeheadu.ca

***Abstract*— During the last few years, RNN models have been extensively used and they have proven to be better for sequence and text data. RNNs have achieved state-of-the-art performance levels in several applications such as text classification, sequence to sequence modelling and time series forecasting. In this article we will review different Machine Learning and Deep Learning based approaches for text data and look at the results obtained from these methods. This work also explores the use of transfer learning in NLP and how it affects the performance of models on a specific application of sentiment analysis.**

***Keywords*—Recurrent Neural Networks, Machine Learning, Deep Learning, Sentiment Analysis.**

I. INTRODUCTION

Sentiment analysis is contextual mining of text which identifies and extracts subjective information in source material, and helping a business to understand the social sentiment of their brand, product or service while monitoring online conversations. The dataset selected for this experiment is the IMDB movie review dataset. It is an opensource dataset and is easily accessible on the internet. The dataset contains a text.csv file which contains the reviews given by different users and the sentiment attached to that review. The dataset has three columns in total namely: text, is_valid and label. The *text* column has the review sentences and all the sentences are of variable lengths. The column *is_valid* defines if a specific review belongs to the validation set or not. The target feature here is *label* which states if a review is positive or negative. There are no missing values in the dataset.

This experiment will use a Naïve Bayes classifier and an RNN based model to predict the *label* of these reviews. Pytorch framework has been used to develop and test the model. The pipeline of the experiment is stated as follows:

- Data Pre-processing
- Splitting the data
- Numerical Representation of data
- Model Creation
- Evaluating the model

Data pre-processing, in this case means tokenization, stop-word removal, punctuation removal and stemming. The data loading and pre-processing steps have been carried out using the torchtext library. The data has already been split into the training and testing set which can be differentiated using the *is_valid* feature. Torchtext uses two parts namely TEXT and LABEL to read the data. Here, TEXT contains all the reviews and the LABEL contains the labels of these reviews.

Word embeddings, which are used for representing the text data into a numerical form are a type of word representation that allows words with similar meaning to have a similar representation. They are a distributed representation for text that is perhaps one of the key breakthroughs for the impressive performance of deep learning methods on challenging natural language processing problems. A preview of the dataset is as shown in figure-1.

	Text	Label	Is_valid
0	Un-bleeping-believable! Meg Ryan doesn't even ...	Negative	False
1	This is a extremely well-made film. The acting...	Positive	False
2	Every once in a long while a movie will come a…	Negative	True

Figure-1

The rest of the paper is organised as follows: Section II provides the Literature review which is followed by Section III shows the Exploratory Data analysis. Section IV explains Naïve Bayes Classifier model followed by the Word embedding techniques in Section V and Section VI talks about deep learning models. Section VII describes the system architecture and setup. Section VIII provides the results and Section IX gives a conclusion which is followed by the references.

II. LITERATURE REVIEW

Vu et al.[1] investigate CNN and basic RNN (i.e., no gating mechanisms) for relation classification. 1D CNN have been used for time series forecasting and text classification problems. The authors of [2] have shown that the 1D CNN model has achieved higher performance in terms of accuracy than the traditional methods. It has shown that 1D CNNs

perform better for some applications than their 2D counterparts. The reasons for this are stated as follows: (1) Rather than matrix operations, FP and BP in 1D CNNs require simple array operations. This means that the computational complexity of 1D CNNs is significantly lower than 2D CNNs. (2) Recent studies show that 1D CNNs with relatively shallow architectures (i.e. small number of hidden layers and neurons) are able to learn challenging tasks involving 1D signals. On the other hand, 2D CNNs usually require deeper architectures to handle such tasks. Obviously, networks with shallow architectures are much easier to train and implement. (3) Usually, training deep 2D CNNs requires special hardware setup (e.g. Cloud computing or GPU farms). On the other hand, any CPU implementation over a standard computer is feasible and relatively fast for training compact 1D CNNs with few hidden layers. (4) Due to their low computational requirements, compact 1D CNNs are well-suited for real-time and low-cost applications especially on mobile or hand-held devices. The authors of [3] designed and trained a 1D CNN to locate and quantify structural damage in a five-story structure. Qianzi Shen *et.al*[4] have proposed an approach where thye combine convolutional neural networks (CNNs) and BLSTM (bidirectional Long Short-Term Memory) as a complex model to analyze the sentiment orientation of text. They have shown that this structure gives better results than a single CNN or an LSTM model. The authors of [5] have used CNN for Sentiment analysis of twitter data and have shown that it performs better than the traditional methods used for classification purpose. Akhtar Shad *et.al.*[6] proposed a method to learn sentiment embedded vectors from the Convolutional Neural Network (CNN). These are augmented to a set of optimized features selected through a multi-objective optimization (MOO) framework. The sentiment augmented optimized vector obtained at the end is used for the training of SVM for sentiment classification.

Statistical methods like the naïve bayes classifier have been extensively used until the rise of Deep Learning models. The authors of [7] have studied different event based models of naïve bayes and shown some results obtained using these models. The work done by the authors of [8] discuss the usage of different deep learning models with respect to the text data. Their extensive research shows us comparative performances of different models on different NLP tasks. Lee JY *et.al*[9] have proposed a model that uses an RNN along with a CNN classifier for text classification problems. They have used RNN model to create fixed length vector representations of sentences which are then fed into a CNN model with a fixed input size. This approach has shown good results for short-text classification problems such as dialogue act prediction.

Word embeddings have also been vastly used as of the recent past as they provide an efficient way to represent words in a vector form based on their similarities and also help reduce the computational complexity as compared to its traditional counterparts such as one-hot representation. One of the popular word embedding technique *glove*[10] used for word vector representations has been used in our experiment. Pre-trained word embeddings have proven to be highly useful in neural network models for NLP tasks such as sequence tagging Lample *et al.*[11] and text classification Kim *et.al*[12].

III. Exploratory Data Analysis

The distribution of different classes in the dataset is as shown in figure-3. The sentiments are divided into two classes namely: positive and negative and figure-2 shows the frequency associated with each sentiment class.

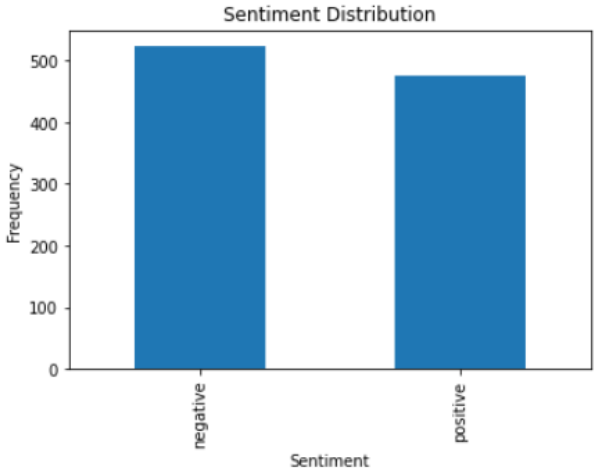

Figure-2

A frequency distribution plot for length of the sentences is as shown in figure-3.

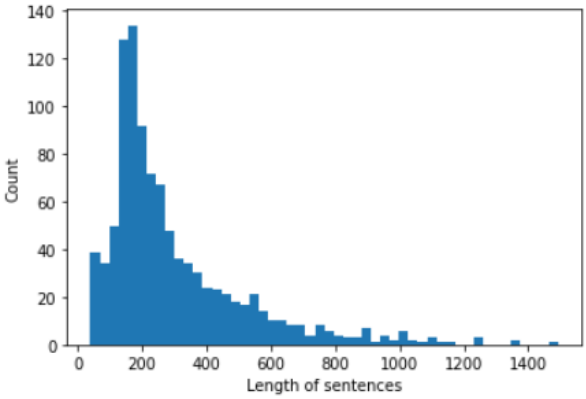

Figure-3

IV. Statistical Model For Text Classification

The Naïve Bayes (NB) model is one of the most simple and popular generative classifiers describing how to generate random instances X conditioned on the target value Y and is widely used as a learning algorithm for both discrete and continuous values [13]. Naive Bayes model has been used in various applications like real time prediction, recommendation system and numerous recognition tasks. Along with simplicity, Naïve Bayes is known to outperform even the most-sophisticated classification methods. It proves to be quite robust to irrelevant features, which it ignores. It learns and predicts very fast and it does not require lots of storage. However, Naïve Bayes algorithm works based on the following assumption: all features must be independent of each other. In reality, this is usually not the case; however, it still returns very good accuracy in practice even when the independent assumption does not hold. Equation (1) shows how posterior probability is calculated for a class given a set of features.

$$P(c \mid x) = \frac{P(x \mid c)\, P(c)}{P(x)} \qquad (1)$$

Statistical methods have three major drawbacks: (1) manual feature engineering, which requires domain expertise, (2) they are based on a complex set of hard if-then rules, which works good only for a specific task only and (3) Curse of dimensionality, which hinders joint probability function. An overview of the curse of dimensionality is illustrated by the following example: suppose a child wants to select a cookie and he/she has to choose based on its taste. This is quite easy as he/she can just choose the sweetest cookie out of all. Now, suppose that each cookie has three features: taste, shape and color. Now, making a decision becomes much harder than the previous case. This example perfectly illustrates the curse of dimensionality.

To address these limitations the following techniques were used:(1) Learning distributed representations of words existing in low-dimensional space to tackle the Curse of Dimensionality, (2) word embeddings learned from a large unlabeled corpus for the issue of Handcrafting Features and (3) Artificial Neural networks to learn the structure of the language instead of hard if-else rules.

V. Word Embeddings

Word-embedding vectors are one of the hot topics in text representations for NLP. They provide significant improvements over the formerly renowned one-hot encoding vectors in terms of sparsity reduction and better contextual information understanding. When each word is fed into the network, the Embedding layer retrieves its embedding vector, which the model learns to train using gradient descent.

Word2vec[17] is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. Word2vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are located close to one another in the space. The two models are as shown in figure-4.

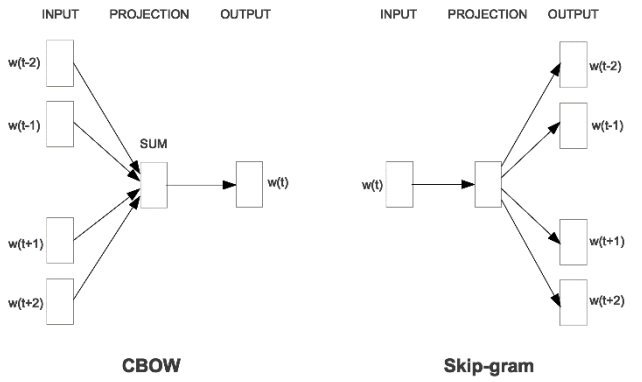

Figure-4

GloVe[18], coined from Global Vectors, is a model for distributed word representation. The model is an unsupervised learning algorithm for obtaining vector representations for words. This is achieved by mapping words into a meaningful space where the distance between words is related to semantic similarity. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

VI. Deep Learning Based Models

Deep learning models have achieved remarkable results in computer vision and speech recognition in recent years. Within natural language processing, much of the work with deep learning methods has involved learning word vector representations through neural language models [14] and performing composition over the learned word vectors for classification[15].Word vectors, wherein words are projected from a sparse, 1-of-V encoding (here V is the vocabulary size) onto a lower dimensional vector space via a hidden layer, are essentially feature extractors that encode semantic features of words in their dimensions. In such dense representations, semantically close words are likewise close in euclidean or cosine distance in the lower dimensional vector space[12].

Generally, RNN or CNN based architecture are used for NLP task such as sentiment analysis. The CNN network tries to extract information about the local structure of the data by applying multiple filters (each having different dimensions). The RNN based better suited to extract the temporal correlation of the data and dependencies in the text snippet.

CNNs have been very successful for several computer vision and NLP tasks in the recent years. They are specially powerful in exploiting the local correlation and pattern of the data through learned by their feature maps. One of the early works which used CNN for text classification is by Kim [12], which showed great performance on several text classification tasks.

To perform text classification with CNN, usually the embedding from different words of a sentence (or paragraph) are stacked together to form a two-dimensional array, and then convolution filters (of different length) are applied to a window of h words to produce a new feature representation. Then some pooling (usually maxpooling) is applied on new features, and the pooled features from different filters are concatenated with each other to form the hidden representation. These representations are then followed by one (or multiple) fully connected layer(s) to make the final prediction. Figure-5 shows general architecture of CNN.

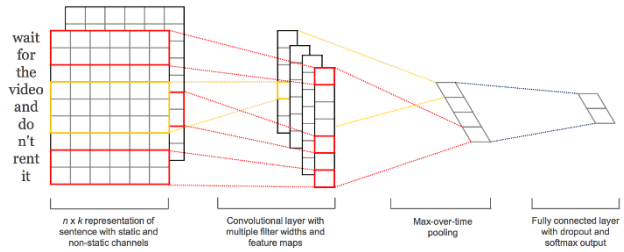

Figure-5

A recurrent neural network (RNN) [Elman, 1990] is able to process a sequence of arbitrary length by recursively applying a transition function to its internal hidden state vector h_t of the input sequence. The activation of the hidden state h_t at time-step t is computed as a function f of the current input symbol xt and the previous hidden state h_{t-1}.

It is common to use the state-to-state transition function f as the composition of an element-wise nonlinearity with an

affine transformation of both x_t and h_{t1}. Traditionally, a simple strategy for modeling sequence is to map the input sequence to a fixed-sized vector using one RNN, and then to feed the vector to a softmax layer for classification or other tasks Cho et al.[16]. A standard RNN architecture is as shown in figure-6.

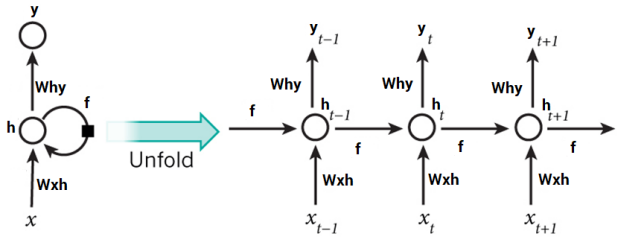

Figure-6

VII. System Architecture and Experimental Setup

The configuration of a RNN is formed by the following hyper-parameters: (1) output dimension on embedding layer (2) Number of hidden RNN and MLP layers/neurons. The model used in this experiment has one embedding layer with output dimension of 100 along with a RNN layer. The hidden dimension for the RNN layer is set as 256. There is one dense layer in the network. The output from the RNN layer is then fed as an input to the fully connected layers. The output of these dense layers is then passed through a softmax layer which calculates the probabilities for each of the class. The Sentiment Feature is the target variable in this experiment. The batch size used in this experiment is 20. The learning rate set for the experiment is 3e-3. The loss measure used here is the CrossEntropyLoss and the performance metric used is Accuracy. We also use pretrained GLOVE embeddings in the experiment to perform transfer learning.

VIII. Results

The results obtained are shown for all the three methods: Naïve Bayes classifier, RNN and RNN with pre-trained embeddings. The system architecture and setup is described in section VII.

The plot for training and testing accuracy with respect to number of epochs is as shown in figure-7

Figure-7

The plot for testing loss with respect to the number of epochs is as shown in figure-8.

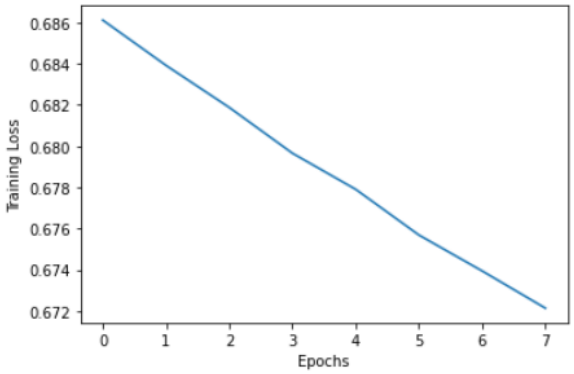

Figure-8

Table-1 gives us details of accuracies for different models on the dataset.

Classification Model	Accuracy
Naïve Bayes	52.56%
RNN	57.04%
RNN with pre-trained embeddings	73.68%

Table-1

IX. Conclusion

It can be said for this experiment that the use of pre-trained embeddings can prove to be quite useful to boost up the accuracy and reduce the computational complexity. However, the use of pre-trained embeddings also depends on the domain of an application. Thus, word vector representation needs to be computed as a universal vector that can understand the context of words in multiple domains. Although, pre-trained embeddings may not be able to represent words across multiple domain but they prove to be of great use when working on a dataset that has a limited amount of text. These vectors can quickly learn from the given set of data and can be used as an effective learning technique.

X. References

[1] Vu NT, Adel H, Gupta P, Schütze H. Combining recurrent and convolutional neural networks for relation classification. arXiv preprint arXiv:1605.07333. 2016 May 24.

[2] Kiranyaz S, Ince T, Hamila R, Gabbouj M. Convolutional neural networks for patient-specific ECG classification. In2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC) 2015 Aug 25 (pp. 2608-2611). IEEE.

[3] Avci O, Abdeljaber O, Kiranyaz S, Inman D. Structural damage detection in real time: implementation of 1D convolutional neural networks for SHM applications. InStructural Health Monitoring & Damage Detection, Volume 7 2017 (pp. 49-54). Springer, Cham.Aslam N, Xia K, Ali A, Ullah S. Adaptive TCP-ICCW congestion control mechanism for QoS in renewable wireless sensor networks. IEEE sensors letters. 2017 Oct 2;1(6):1-4.

[4] Shen Q, Wang Z, Sun Y. Sentiment analysis of movie reviews based on cnn-blstm. InInternational Conference on Intelligence Science 2017 Oct 25 (pp. 164-171). Springer, Cham.

[5] Liao S, Wang J, Yu R, Sato K, Cheng Z. CNN for situations understanding based on sentiment analysis of twitter data. Procedia computer science. 2017 Jan 1;111:376-81.

[6] Akhtar MS, Kumar A, Ekbal A, Bhattacharyya P. A hybrid deep learning architecture for sentiment analysis. InProceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers 2016 Dec (pp. 482-493).

[7] McCallum A, Nigam K. A comparison of event models for naive bayes text classification. InAAAI-98 workshop on learning for text categorization 1998 Jul 26 (Vol. 752, No. 1, pp. 41-48).

[8] Yin W, Kann K, Yu M, Schütze H. Comparative study of cnn and rnn for natural language processing. arXiv preprint arXiv:1702.01923. 2017 Feb 7.

[9] Lee JY, Dernoncourt F. Sequential short-text classification with recurrent and convolutional neural networks. arXiv preprint arXiv:1603.03827. 2016 Mar 12.

[10] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014 Oct (pp. 1532-1543).

[11] Lample G, Ballesteros M, Subramanian S, Kawakami K, Dyer C. Neural architectures for named entity recognition. arXiv preprint arXiv:1603.01360. 2016 Mar 4.

[12] Kim Y. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882. 2014 Aug 25.

[13] Tom M. Mitchell, "Generative and Discriminative Classifiers: Naïve Bayes and Logistic Regression", https://www.cs.cmu.edu/~tom/ mlbook/NBayesLogReg.pdf

[14] Bengio Y, Ducharme R, Vincent P, Jauvin C. A neural probabilistic language model. Journal of machine learning research. 2003;3(Feb):1137-55.

[15] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P. Natural language processing (almost) from scratch. Journal of machine learning research. 2011;12(Aug):2493-537.

[16] Cho K, Van Merriënboer B, Gulcehre C, Bahdanau D, Bougares F, Schwenk H, Bengio Y. Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078. 2014 Jun 3.

[17] Mikolov T, Chen K, Corrado G, Dean J. Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. 2013 Jan 16.

[18] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. InProceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) 2014 Oct (pp. 1532-1543).