Deep Learning for Sentence Classification

Abdalraouf Hassan
Dep of Computer Science and Engineering
University of Bridgeport, CT, 06604, USA
abdalrah@my.bridgeport.edu

Ausif Mahmood
Dep of Computer Science and Engineering
University of Bridgeport, CT, 06604, USA
mahmood@bridgeport.edu

***Abstract*—Most of the machine learning algorithms requires the input to be denoted as a fixed-length feature vector. In text classifications (bag-of-words) is a popular fixed-length features. Despite their simplicity, they are limited in many tasks; they ignore semantics of words and loss ordering of words. In this paper, we propose a simple and efficient neural language model for sentence-level classification task. Our model employs Recurrent Neural Network Language Model (RNN-LM). Particularly, Long Short-Term Memory (LSTM) over pre-trained word vectors obtained from unsupervised neural language model to capture semantics and syntactic information in a short sentence. We achieved outstanding empirical results on multiple benchmark datasets, IMDB Sentiment analysis dataset, and Stanford Sentiment Treebank (SSTb) dataset. The empirical results show that our model is comparable with neural methods and outperforms traditional methods in sentiment analysis task.**

Keywords—Sentiment analysis, Natural Language Processing, Deep Learning, Recurrent neural network, LSTM, RNN

I. INTRODUCTION

The classic approach for text classification typically starts with the feature extraction stage, and then followed by classifier stage [1, 2]. Text categorization is significant for natural language processing (NLP) with many applications, such as document classification, web search, ranking, information retrieval, and spam filtering. Many neural networks based models attracted researchers recently, and these methods has achieved good results in practice. However, they seem to be very slow during the training and testing time, which make them only useful for large dataset.

In recent years with the progress of machine learning techniques, it becomes possible to train more complex models on much larger dataset, and they typically outperform the simple models. Perhaps the most efficient model is to use distribution representation of word. Neural Network Language Models outperform n-gram models [3, 4]. Currently for text classification problems, linear classifiers are considered to be the conventional approach and strong Baselines. Regardless of their simplicity, linear classifiers often obtains the state-of-the-art performances, especially when the right features are selected.

Deep neural network (DNN) has recently become very popular in computer vision. Where feature extractions and classification perform jointly [5]. DNN based method convention in most cases; an input document denoted as a sequence of words, and then each sequence represented as one-hot-vector, then each word in the sequence projected into a continuous vector space by multiplying it with weight matrix, forming a sequence of dense, real valued vector. This sequence then feed into a deep neural network, which processes the sequence in multiple layers, finally resulting in prediction probability [6]. This pipeline tuned jointly to maximize the classification accuracy on training set. A simple way to improve accuracy is to use unsupervised word representations as extra word features in present supervised NLP system [7].

In this work, we proposed a neural language model that is able to overcome the weakness of bag-of-words and n-grams models; they ignore semantics of words, and also they lose the ordering of words. Our method is able to capture long-term dependences in sequence of sentence. We utilized pre-trained word vectors, which were trained on 100 billion of words of Google News [14], and then we trained a Long Short-Term Memory (LSTM) to avoid the loss of details in local information, and overcome the problem of vanishing gradient [15].

II. RELATED WORK

A. Traditional Methods

In text classification, the standard method is to represent a sentence as a (bag-of-words) features, and then feed the fixed length sequence of vectors to train linear classifier (e.g. a logistic regression). However, bag-of-words loses all information about words, for instance, it ignores the order of words, and grammar, which might lead to conflict in word representations, where different sentences could have the same word representation.

Another popular method is n-gram, which is used for statistical language modeling, it takes in account the word order in short sentences. However, n-gram suffer from data sparsity. Simple techniques are limited in many tasks, there are situations where simple scaling up of basic techniques will not result in good performance or any significant progress.

n-gram models has lack of long-range dependency, which is due to the only explicit dependency range *(n-1)* tokens for n-gram model, and because of long range correlations drop exponentially with distance for any Markov model. n-gram model have not made much impact on linguistic theory, where the explicit goal is to model such dependencies.

Unlike n-grams model, neural network language modeling [3, 13] offer several advantages. Because of the projection of the entire vocabulary into a small hidden layer; semantically similar word clustered close to each other. Thus, n-gram counts defined by neural network based models could lead to better estimation for n-grams.

One-hot vector is a popular method to represent a word, where each dimension of the vector corresponds to a

distinct word. In addition, in the simple bag-of-words representation of a document, the document is represented as the sum of the one-hot vectors of the words. However, these representations lose much of the semantic meaning that is associated with text. Recent works have focused on creating vector representations for both words and documents, with the idea of retaining as much information as possible. In word2vec [8], vector representations are computed for each word, with the result being that words whose meanings are related are generally closer in terms of Euclidean distance than between unrelated words.

Linear classifiers become the strong models for text classification problems. However, linear classifiers does not share parameters between the features and classes; this might edge the generalization in the context for large output, where as some classes have few cases. One general way to solve this issue is to factorize the linear classifier into low rank matrices [8]. Another way to solve this issue is using multilayer neural networks [9, 10].

B. Deep Learning Methods

Convolutional Neural Network (CNN) has reached an outstanding result in computer vision, where handcrafted features were used (e.g. Scale-Invariant Features Transform), followed by classifier. The objective of the CNN is to consider features extraction and classifier as one jointly trained task. The use of neural network inspired many researchers after the successful approach in [3, 10, 11], this area have been investigated by researchers in the recent years, particularly by using multi CNN and pooling layers, and consecutively extracting hierarchical representation of the input.

Recently, deep neural networks based models have shown very good results for several tasks in NLP. However, it has not outperformed the state-of-the-art by tremendous improvement, as reported in computer vision and speech recognition. Traditional methods using hand-crafted feature extractor and logistic linear regression classifier were compared to deep learning methods by [9]. However, there is no single machine-learning model that can work for all types of dataset.

Recursive Neural Network proved to be efficient in structure sentence representations. The model has a tree structure, which is able to capture the semantics of sentences. However, this is a time-consuming task due to the constructing of the textual tree complexity [12].

Recurrent Neural Network (RNN) has enhanced time complexity; in this model text is analyzed word by word then the semantics of all the previous text are preserved in a fixed-sized hidden layer, and the capability to capture superior appropriate statistics could be valuable to capture semantics of long text in recurrent network. However, RNN is biased model, because recent words are more significant than earlier words. Thus, the key components could appear anywhere across the document and not only at the end; this might reduce the efficiency when used to capture the semantics of the whole document. LSTM model was introduced to overcome the RNN drawbacks [15].

Deep neural networks, and representation learning approaches have led to new methods for solving the data sparsity problem. Several neural network based models for learning word representations followed this approach. Word embedding is the neural representation of a word and is real vector; word embedding allows to measure similarities between words by simply using distance between two embedded vectors [11, 13]. Recently, many researchers observed that it is not necessary for deep neural network to perform at word level, as long as the document is represented as one-hot-vector, the model could work without no change, regardless if each one-hot vector corresponds to a word [5].

III. Model Architectures

A. Word Embedding

To convert document d to input layer x we need to process two-steps to get a sequence of words into vector representation x to feed it as the input layer to the network.

- Firstly convert each word w in document d to a word vector; to achieve that we have to represent w by a one-hot vector using a vocabulary that consist of N high frequent words, then we map w to a vector.

- Secondly build x by concatenating these word vectors, and form the word vector of w by concatenating its one-hot vector representation, then each sentence is indicated as a sequence of one-hot vectors. A one-hot vector of the $i\text{-}th$ character in the vocabulary is a binary vector whose element is zero except for the $i\text{-}th$ element, which is set to one. Thus, each sentence is considered as a sequence of token or words; token are usually processed in sequential order, from left to right, a sequence is T one-hot vectors $(x_1, x_2, ..., x_T)$. An embedding layer projects each of the one-hot vector into a d-dimensional continuous vector space $\mathbb{R}^d$. This is done by multiplying the one-hot vector from left with weight matrix $W \in \mathbb{R}^{d|v|}$

  Where: $|V|$ is the number of unique character in vocabulary $e_t = Wx_t$

- After the embedding layer, the input sequence of one-hot vectors becomes a sequence of dense, real valued vectors $(e_1, e_2, ..., e_T)$.

B. Pre-trained word Vectors

We initialized the model using word vectors obtained from unsupervised neural language model, which is a popular method to improve performance in the absence of a large supervised training set. We exploits word2vec that were trained on 100 billion words from Google News. [8]. Words not existent are initialized randomly.

C. Recurrent Neural Network

RNN based model was introduced in neural network based language model to overcome certain limitations of the feedforward neural network language model, for instance the need to specify the context length. Moreover, because RNN can efficiently represent more complex pattern than the shallow neural networks [18]. RNN connects hidden layers to itself, using time-delayed connections, this allows the model to formulate types of short term memory, as information from the past can be represented by hidden layer state that gets update based on

the current input and the state of the hidden layer from the previous time step.

$h_t = f(x_t, h_{t-1})$,

Where: $x_t \in \mathbb{R}$ one time step from the input sequence is $(x_1, x_2, ..., x_T)$ $h_0 \in \mathbb{R}^{d`}$, is often initialized as an all-zero vector.

D. Long Short-Term Memory

LSTM models are a variety of RNN [15]. In RNN the prediction in sequence, where the hidden layer from one prediction is the hidden layer of the next prediction, this will assign a memory to the network, therefore, results from earlier estimation could lead to improve future predictions.

LSTM gives RNN more features to a fine-grained control over memory; this aspects control how much the present input matters for forming the new memory, also how much the prior memories matters in designing the new memory, and what parts of the memory are essential is producing the output. In our experiments, we followed the basic recipes of implementing for LSTM by extracting the cell outputs and gradients. Mostly applied method on IMDB and SSTb datasets such as (bag-of-words or n-grams), classically ignore long-range ordering information. LSTM consist of four sub-unites –inputs, output, forget and candidate memory cell.

First we compute the value for i_t, the input gate, and $\tilde{c}_t$ the candidate value for the states of the memory cells at time *t*:

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \quad (1)$$

$$\tilde{c}_t = \tanh(W_i x_t + U_i h_{t-1} + b_c) \quad (2)$$

Second, compute the value for f_t, the activation function of the memory cells forget gate at time *t*:

$$f_t = \sigma(W_f x_t + U_f h_{t-1}) + b_f \quad (3)$$

Given the value of the input gate activation i_t, the forget gate activation f_t, and the candidate state value $\tilde{c}_t$, now we can compute c_t the memory cells new state at time *t:*

$$c_t = i_t * \tilde{c}_t + f_t * c_{t-1} \quad (4)$$

With the new state of the memory cells, we can compute the value of the outputs gates and subsequently, their outputs:

$$o_t = \sigma(W_o x_t + U_i h_{t-1}) \quad (5)$$

$$h_t = o_t * \tanh(c_t) \quad (6)$$

LSTM unit first computes memory cell, and computes the output, or activation, then resulting sequence from the recurrent layer is $(h_1, h_2, ..., h_T)$,

Where T is the length of the input sequence to the layer. Furthermore, we establish that averaging the hidden states over sum of sentence; get better results. This increase the level of feedback to earlier words, dropout is very dominant regularize, even if the network is one layer deep. Increasing the number of the layers, where the output of one layer fed as input to the next layer, occasionally found to be operational.

IV. EXPERIMENTAL RESULTS AND DATA SETS

A. Datasets

We use two datasets for sentiment analysis to evaluate the performance of our model; Stanford Large Movie Review Dataset IMDB [16], and Stanford Sentiment Treebank (SSTb) dataset [19].

- IMDB dataset was proposed by [16] as a benchmark for sentiment analysis. It consist of 100,000 movie reviews taken from IMDB. The key aspect of this dataset is each movie review has several sentences. 100,000 reviews are divided into three datasets: 250,000 labeled training instances, 25,000 labeled test in sentences, and 50,000 unlabeled training instances. There are positive and negative labels balanced in both the training and the test set.

- This dataset was first proposed by [17] then extended by [19] as a benchmark for sentiment analysis, it consist of three sets; 8544 sentences from training. 2210 sentence for test and 1101 sentence for validation. In total there are 239,232 labeled phases in the dataset.

B. Model Setup

To improve performance in the lack of a large supervised training set it is a common method to initializing word vectors with those obtained from an unsupervised neural language model. We use word2vec vectors that were trained on 100 billion words from Google news [14]. The dimensionality of the vector is 300, word2vec model were trained using the continuous bag-of-words algorithm, and all the words are not present in the set of pre-trained are initialized randomly.

Training is done through stochastic gradient descent over shuffled mini-batches, rectifier linear units used for both datasets, we set the size of hidden state to be 128, and mini-batch size of 64, we select random 10% of the training data as dev set for early stopping, dropout showed to be a good regularize, we set dropout to be set to 0.5, which added ~3% relative performance.

For each data sets, we randomly split the full training examples into training and validation. The training size is the same as the corresponding test size and balanced in each class, to make use of available label data, we follow the experimental protocols as described in [19].

C. Results and Discussion

We perform several experiments to offer fair comparison to competitive model, by implementing several deep-learning approaches and deep-learning inspired approaches and evaluate these algorithms on sentiment-labeled datasets. We choose the best to our knowledge models that provide competitive results on the same dataset in [16,19,20].

Table 1. Table performance of our method in error rates compared to other approaches on the SSTb dataset.

Methods	SSTb
Naïve Bayes [19]	18.2 %
SVMs [19]	20.6 %
Bigram Naïve Bayes [19]	16.9 %
Word vector averaging [19]	19.9 %
RNN [19]	17.6 %
Matrix Vector-RNN [19]	17.1 %
RNTN [19]	14.6 %
Our *lstm-w2v*	14.3 %

Table 2. Table shown the error rates of our method on the IMDB dataset compared to the other methods.

Methods	IMDB
BoW (bnc) [16]	12.20 %
Bow [16]	11.77 %
LDA [16]	32.58 %
Full+BoW [16]	11.67 %
Full+Unlabeled+BoW[16]	11.11 %
MNB-uni [20]	16.45 %
MNB-bi [20]	13.41 %
SVM-uni [20]	13.05 %
SVM-bi [20]	10.84 %
NBSVM-uni [20]	11.71 %
Our lstm-w2v	11.32 %

Results of our method compared to other baselines are reported in Table 1, and 2. As we can see from the table significant improvement comes from [20], beside to other variations, NBSVM on bigram features works the best and yield in improvement of 2 % in term of error rate. Since the sentences are long, one might expect that it is difficult for recurrent networks to learn. We found that with tuning, it is possible to train LSTM recurrent networks to fit the training set.

Our experiments on the sentiment analysis task show that the method is competitive with state-of-the-art to previous reported results. The good performance validates the qualities of the model in term of capturing the semantics and similarity relationships in short texts. In fact, our proposed model has the potential to overcome many weaknesses of traditional method e.g., bag-of-words and n-gram models where order and information about word are vanished. We observed that there is no single machine-learning method that perform the state-of-the-art for all types of the datasets, we also found out there are many factors could effects the performance of the model, such as the size of the dataset, choice of the feature extractor, and type of machine learning classifier will be used. These factors could play an important role in deciding which method is the best fit for specific natural language processing system.

V. CONCLUSION

In this work, we present a new neural network techniques to capture sentiment analysis in short text based on recurrent neural network LSTM and pre-trained word vectors word2vec, we validate our model on two benchmark datasets. Our results show that a simple LSTM with one single layer perform well with unsupervised word vectors, which become an important feature in natural language processing and deep learning. Our experiments on sentiment analysis datasets showed that our method is competitive with state-of-the-art of deep learning methods, and likely overcome some flaws of bag-of-words and n-grams models; using pre-trained vector representations can overcome some obstacles of traditional methods. It would be interesting to see if the architecture introduced in this paper is variable for other text classification tasks such as machine translation or information retrieval. Combination of convolutional neural network with recurrent layer is also direction worth to explore.

REFERENCES

[1] Joachims, T. Text categorization with support vector machines: Learning with many relevant features. in European conference on machine learning. 1998. Springer.

[2] Mu, Y., et al., Event-related theta and alpha oscillations mediate empathy for pain. Brain research, 2008. 1234: p. 128-136.

[3] Bengio, Y., et al., Neural probabilistic language models, in Innovations in Machine Learning. 2006, Springer. p. 137-186.

[4] Mikolov, T., et al. Strategies for training large scale neural network language models. in Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on. 2011. IEEE.

[5] Kim, Y., Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882, 2014.

[6] Shen, Y., et al. learning semantic representations using convolutional neural networks for web search. in Proceedings of the 23rd International Conference on World Wide Web. 2014. ACM.

[7] Turian, J., L. Ratinov, and Y. Bengio. Word representations: a simple and general method for semi-supervised learning. in Proceedings of the 48th annual meeting of the association for computational linguistics. 2010. Association for Computational Linguistics.

[8] Mikolov, T., et al., Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781, 2013.

[9] Zhang, X., J. Zhao, and Y. LeCun. Character-level convolutional networks for text classification. in Advances in Neural Information Processing Systems. 2015.

[10] Collobert, R. 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. 2008. ACM.

[11] Collobert, R., et al., Natural language processing (almost) from scratch. Journal of Machine Learning Research, 2011. 12(Aug): p. 2493-2537.

[12] Socher, R., et al. Parsing natural scenes and natural language with recursive neural networks. in Proceedings of the 28th international conference on machine learning (ICML-11). 2011.

[13] Bengio, Y., et al., A neural probabilistic language model. journal of machine learning research, 2003. 3(Feb): p. 1137-1155.

[14] Mikolov, T., et al. Distributed representations of words and phrases and their compositionality. in Advances in neural information processing systems. 2013.

[15] Hochreiter, S. and J. Schmidhuber, Long short-term memory. Neural computation, 1997. 9(8): p. 1735-1780.

[16] Maas, A.L., et al. learning word vectors for sentiment analysis. in Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1. 2011. Association for Computational Linguistics.

[17] Pang, B. and L. Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. in Proceedings of the 43rd annual meeting on association for computational linguistics. 2005. Association for Computational Linguistics.

[18] Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. "Recurrent neural network based language model." In Interspeech, vol. 2, p. 3. 2010.

[19] Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank." In Proceedings of the conference on empirical methods in natural language processing (EMNLP), vol. 1631, p. 1642. 2013.

[20] Wang, Sida, and Christopher D. Manning. "Baselines and bigrams: Simple, good sentiment and topic classification." In Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2, pp. 90-94. Association for Computational Linguistics, 2012.