Attention Models in Text Classification

Yanting Miao[20841196] and Xun Lu[20820321]

University of Waterloo, Ontario, Canada
y43miao@uwaterloo.ca
x256lu@uwaterloo.ca

Abstract. Text classification is a fundamental task in many Natural Language Processing (NLP) applications. Traditional classification algorithms, such as SVM, Logistic Regression, and Decision Tree may lose contextual information and cause misclassification. Hence, we introduce three models, including one baseline model and two novel and robust attention models to learn latent semantic information in sentences. We also design experiments on four popular datasets and compare them with a novel model, RCNN [6], in academia. The results of experiments show that the performance of our attention models is close to RCNN in the text classification problem and outperform RCNN in sentiment analysis tasks.

Keywords: Deep learning · Attention models · RNN · LSTM · GRU · Text classification.

1 Introduction

With the rapid development of Internet and information technology, a wide variety of digital information fills every corner of human society, among which text information occupies a pivotal position. Text classification is a supervised learning process involving many key technologies in machine learning, data mining and other related fields. There are many factors that affect the performance of it, mainly including text preprocessing, feature extraction, dimensionality reduction, text representation, classifier design and evaluation criteria, etc. Due to the high dimensionality and high sparsity of traditional text representation models, designing efficient text representation models and reducing the dimensionality of text representation are main topics in the field of text classification.

In terms of text classification, it includes feature selection, feature weight estimation, and classification algorithms. Feature selection is the main method of text dimensionality reduction, therefore a large number of research documents have studied feature selection methods from two types of screening feature selection methods and multiple selection feature selection methods, from different feature search strategies and different feature set evaluation methods. Many research documents have analyzed and studied the factors affecting the classification effect of feature selection methods in unbalanced corpus, and improved the classification effect by improving the feature selection methods. Xing et al.

[8] discuss feature selection of high-dimensional data and the distribution characteristics of mutual information are used by [9] to improve the original mutual information feature selection method and improve the text classification effect.

Due to the weak feature expression capabilities of traditional text classification and the limited ability of classification models to handle complex problems, it is necessary to introduce advanced deep learning techniques to solve them. The text classification based on deep learning replaces the shallow classification model with a deep learning neural network model as a classifier. Meanwhile, the multi-layer structure of deep learning improves the ability to learn the features of the data set during the training process so that achieve the approximation of complex functions. Therefore, the utilization of deep learning models can solve the problems in traditional text classification origin from too complex generalization, and such limitation can lead to shortcoming of low accuracy.

In this work, we propose three deep learning models for text classification problem. One is CNN baseline model and the other two novel RNN-based models use attention mechanism. In these two attention models, first, we apply LSTM layer as role as encoder. Then, our attention models capture contextual information and compute attention weights via attention mechanism. Here, one attention model will forward the output of encoder and attention weights to decoder stack, and convert text classification problem to machine translation problem. second attention model is similar to the first one, but this attention model drop the decoder stack, to reduce overfitting. These two attention models are robust and they can achieve very high accuracy rate with a very small proportion of training samples. Our source code is available at `https://github.com/andrew-miao/ECE657A_Project-text-classification`. To summarize, our contributions are as follows:

- We propose two novel attention models for text classification tasks and attempt to convert text classification problem to machine translation problem. These two models can learn contextual information, which is important to classify text.
- Results on experiments demonstrate that the performance of two attention models are close to a novel model in academia, RCNN [6] in text classification task.

2 Related Work

To improve accuracy for text classification task, increasing researchers start to use deep learning model in recent years. For Convolutional Neural Network, in 1998, LeCun Y et al. used the CNN model to realize the task of handwritten font recognition [3]. The completion of this task made CNN a research hotspot in deep learning methods. Subsequently, more and more scholars applied it to NLP aspect. In 2015, Xiang et al. [2] used a CNN model in character-level and achieved an amazing improvement effect on the better technical level at the time.

Rumelhart et al. [12] proposed recurrent neural networks (RNN) in 1986. In RNN, the nodes of the hidden layer are also connected to each other. Therefore,

the input of each hidden layer includes not only the output of the previous hidden layer but also the input of the input layer at the current moment, reminding that the output of the previous hidden layer is not included in the traditional neural network. Theoretically in the design of this hidden layer, RNN can handle sequence data of any length. However, in practice when the distance between the current predicted position and the text related to it is large, RNN cannot use the information that is far away because RNN uses back propagation algorithm to optimize the network, which may cause the gradient explosion and gradient disappearance phenomenon [13].

The attention mechanism is widely used in the field of natural language processing (NLP). It became popular in machine translation from 2014 to 2015 and at the same time, the neural network model combining attention mechanism and RNN plays an essential role in the field of natural language. In 2015 and 2016, the neural network model with attention mechanism and CNN became a hot topic in research. For instance, [5] applied the attention mechanism to machine translation tasks and used the attention mechanism to link the expression learned by each word in the source language with the word predicted to be translated, which is also the first application of the attention mechanism in the field of natural language processing. Afterward, Luong et al. [14] introduced global and local attention mechanisms such as adding attention mechanism before inputting CNN or between CNN feature extraction and pooling. This is also the first exploration of the attention mechanism on CNN.

3 Models

We propose three deep neural models, including a baseline CNN model and two attention models, to solve text classification tasks. To begin with, we describe the notations throughout this paper. We denote D_i as the i-th document in the corpus, w_j^i as the j-th word in D_i, and the number of words in a document will be defined as n. Hence, $D_i = [w_1^{(i)}, \ldots, w_n^{(i)}]$. Our models will compute the probability of D_i being class k, which is $P(k|D_i, w^{(i)}, \theta)$. θ is the parameters of this network. To simplify this task, we use Glove [1] to present words (word embedding) in our models.

3.1 Baseline Model CNN

The architecture of the our baseline model CNN in AGNews dataset [2] is shown in Fig.1. This model is inspired by LeNet [3]. In this CNN model, we use three convolution layers to capture context information. In full-connected layers, we use relu function as activation function. To reduce overfitting problem, we insert 3 dropout layers [4] in last 3 full-connected layers. For efficiency, this baseline model contains max-pool layer as downsampling layers. However, max-pool layers will change original meaning of $w_j^{(i)}$ or create a new word which not in the original sentence. But, compare with other two attention models, this baseline model is easier to train and implement. In our experiments, this baseline model

can achieve higher than 80% accuracy rate in non-sentiment analysis tasks. The parameters of this CNN in AGNews are illustrated in Table 1. This baseline model can keep local connectivity between words, but for a long sentence this model will lose information or cause some words lose original meaning, and thus we propose two attention based model.

Table 1. Parameters in baseline model that we used in AGNews dataset

Layer	(Depth, Height, Width)	Operation	Height, Width
1	(1, 300, 50)	Convolution	(5, 5)
2	(3, 296, 46)	Max-Pool	(2, 2)
3	(3, 148, 23)	Convolution	(3, 3)
4	(6, 146, 21)	Max-Pool	(2, 2)
5	(6, 73, 11)	Convolution	(3, 3)
6	(3, 71, 9)	Max-Pool	(2, 2)
7	(3, 35, 4)	Dense	None

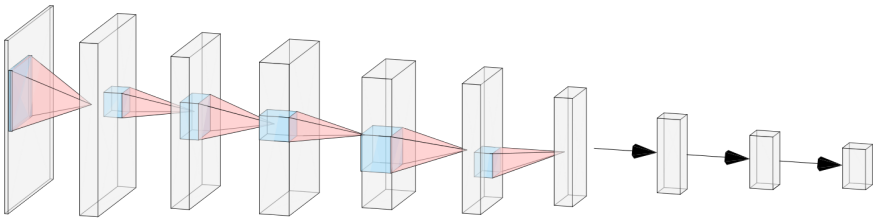

Fig. 1. The architecture of CNN baseline model in AGNews dataset

3.2 LSTM Encoder and GRU Decoder

Considering a long word sequence may lose information in RNN (forgetting issue), for example, and thus we introduce attention mechanisms in the text classification problem. Therefore, we also attempt to convert this text classification to the fixed length (the number of classes) *machine translation problem*. Fig. 2 displays the structure of our first attention model, which contains encoder and decoder stacks, and attention mechanism allow us to compute attention weights. The encoder stack contains one LSTM layer and the decoder composed by a one GRU layer.

In our model, the input document will be encoded via encoder, which can be rewrite as

$$e(D^{(i)}) = f(W^{(e)}D^{(i)} + b^{(e)}), \tag{1}$$

where $W^{(e)}$ and $b^{(e)}$ is the weight and bias of the LSTM layer respectively. In our implementation, we adopt zero as initial hidden state. Meanwhile, input document will be compressed by first full-connected layer and the result of the first full-connected layer be denoted as $c_1(D^{(i)})$. To match dimensionality to the encoder output, the second full-connected layer will be inserted, and its output is $c_2(D^{(i)})$. Softmax layer is responsible for computing attention weights. The word that has a great influence on the result will be given a higher attention weight. For example, the sentence *"Football: England Beats Poland in 2006 World Cup Qualifier"* in AGNews dataset, *"Football"* and *"World Cup"* will be assigned more attention weights. If we only consider context information like *"England Beats Poland"*, but forgetting global information, this sentence will be misclassified as *"world news"*. Hence attention mechanism allow us to remember global information and still keep the context information. Here, we write the attention weights (the output of softmax layer) for $D^{(i)}$ as

$$\alpha_j^{(i)} = \text{Softmax}(c_2(D^{(i)})_j), \tag{2}$$

where $c_2(D^{(i)})_j$ is the j-th "word" in the $c_2(D^{(i)})$. The attention weights are assigned to the encoder output $e(D^i)$, and concatenate with $c_1(D^i)$ to generate attention output:

$$A^{(i)} = [\alpha^{(i)} e(D^{(i)}); c_1(D^{(i)})] \tag{3}$$

$A^{(i)}$ contains latent semantic vector for $D^{(i)}$. The output layer adopts $A^{(i)}$ as input. In decoder part, we use "many to one" approach to produce output $d(A^{(i)})$. After decoder stack, our model is similar to the linear layer of the traditional neural network,

$$\tilde{y}^{(i)} = \text{relu}(Wd(A^{(i)}) + b). \tag{4}$$

In the last softmax layer, our model compute probability of each class for $D^{(i)}$, it is defined as

$$p_k = \text{Softmax}(\tilde{y}_k^{(i)}) \tag{5}$$

3.3 LSTM with Attention

In practice, first attention model faces overfitting problem and it is hard to train. Hence, to simplify this model, we describe the second attention model, which drop decoder stack and modify operations in the attention mechanism. In [5], Bahdanau et al. propose *annotation*, which composed by hidden state of encoder. In our model, we also use the concept of annotation, but we employ the concatenate of hidden state and cell state of LSTM layer,

$$a^{(i)} = [h^{(i)}; c^{(i)}], \tag{6}$$

where $h^{(i)}$ and $c^{(i)}$ is hidden state and cell state of $D^{(i)}$ in the LSTM layer respectively. The composition of attention layer are one linear layer, one softmax

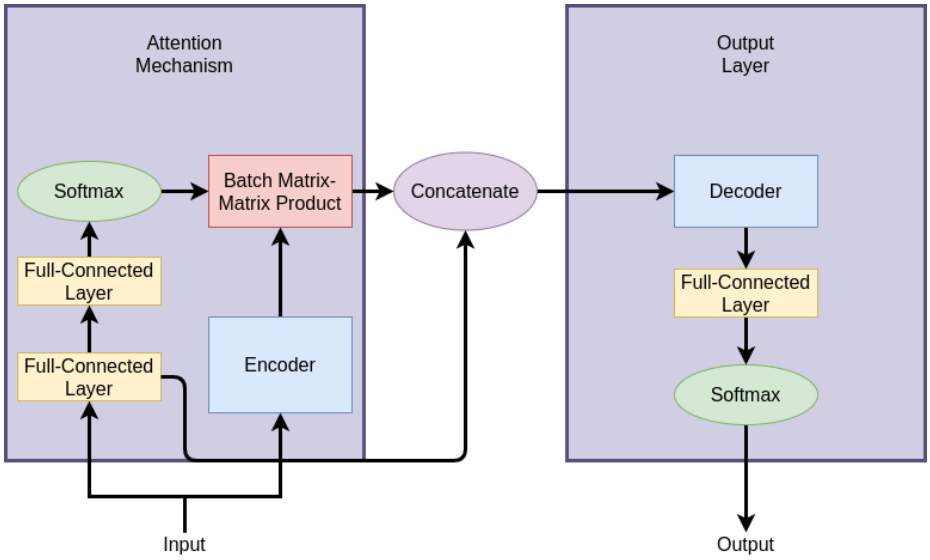

Fig. 2. The structure of the LSTM encoder with attention and GRU decoder. Input is the word embedding result of a document D_i. Output is the probability $P(k|D_i, w^{(i)}, \theta)$ of $D^{(i)}$ being each class k.

layer, and batch matrix-matrix product operation. Attention layer first compute attention weights via annotations:

$$\alpha_j^{(i)} = \text{Softmax}((\beta a^{(i)} + b_1)_j), \tag{7}$$

where β and b_1 is the weight and bias of linear layer respectively. $(\beta a^{(i)} + b_1)_j$ is seen as the j-th "word" of linear transformed annotations. Finally, the attention layer computes context vectors:

$$C^{(i)} = \alpha^{(i)} l^{(i)}, \tag{8}$$

where $l^{(i)}$ is the output of $D^{(i)}$ in the LSTM layer. In the context vectors, the words that have great effect on the classification will be assigned large values. Based on this context vectors, we can drop decoder to simplify the first attention model and reduce overfitting. The output p_k of this model can be denoted as

$$\tilde{y}^{(i)} = \text{relu}(WC^{(i)} + b_2), \ p_k = \text{Softmax}(\tilde{y}_k^{(i)}), \tag{9}$$

where W and b_2 is the weight and bias of the full connected layer.

4 Experiments

4.1 Datasets

To prove the effectiveness of our attention models, we perform experiments with the following our datasets: AGNews, Dbpedia, Amazon Review Polarity, and

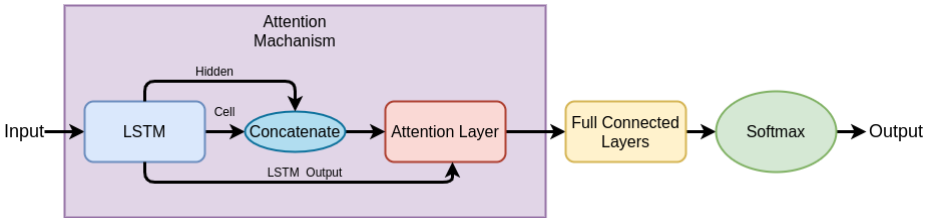

Fig. 3. The architecture of LSTM with attention. The input of this model is word embedding output of $D^{(i)}$. Attention layer use the LSTM output, cell state and hidden state of LSTM. The output of this model is the probability of $D^{(i)}$ being each class k

Yelp Review Polarity in [2]. Because our device doesn't have a enough memory space to store the dataset for Dbpedia, Amazon Review Polarity, and Yelp Review Polarity, we random choose 120000 training samples from training sets and 7600 test samples from test sets they provide. The detailed information about each dataset is shown in Table2.

Table 2. A summary of the datasets, including the number of classes, the number of train/validation/test samples, and average length of a document.

Dataset	Classes	Train/Val/Test	Length
AGNews	4	96000/24000/7600	177
Dbpedia	14	96000/24000/7600	130
Amazon	2	96000/24000/7600	478
Yelp	2	96000/24000/7600	492

- **AGNews**: 496,835 categorized news articles from larger than 2000 news sources from the 4 largest classes, including World News, Sports News, Business News, and Science/Technology News from AG's corpus of news articles, using only the title and description fields.
- **Dbpedia**: 560,000 training samples and 70,000 testing samples from 14 nonoverlapping categories from DBpedia 2014. The language we used is English.
- **Amazon Review Polarity**: This dataset from the Stanford Network Analysis Project (SNAP) and be categorized 2 classes, positive and negative. This full dataset contains 3,600,000 training samples and 400,000 test samples.
- **Yelp Review Polarity**: This dataset from Yelp Dataset Challenge 2015 and this dataset composed by 2 classes, positive and negative. The full dataset contains 560,000 training samples and 38,000 test samples.

4.2 Experiment Settings

For each dataset, we use nltk tokenizer to get tokens and use Glove [1] to perform word embeddings. We split 20% of the training dataset for validation and remain

the rest 80% training samples as our training set. To indicate our models are effective, our models will compare with another novel and robust model [6], RCNN. Considering the setting of hyper-parameters may affect our experiments, we use the same hyper-parameters for all models. The word embedding size will be set as 300 and the sequence length is 60. The learning rate is 0.001 and the dropout rate as 0.5. For evaluation metrics, we choose test accuracy and Macro-F1 score measure in all datasets. Further, to demonstrate the robustness of our models, we also set an experiment that only uses a few samples from the training set.

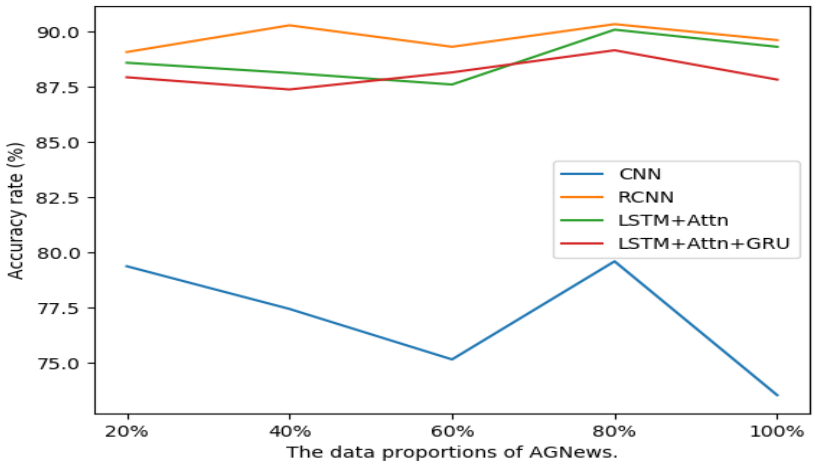

Fig. 4. Test accuracy in varying training data proportions of AGNews

4.3 Results and Discussion

Table 3 and Table 4 displays test accuracy rate and average Macro-F1 score respectively. The performance of our two attention models is close to the novel model RCNN. In the Yelp dataset, our model, LSTM encoder with attention and GRU Decoder, outperforms the RCNN model. We note that our attention models and RCNN perform well on AGNews and Dbpedia. The reason for this phenomenon is that each document has words indicating the class of this document in these datasets. Amazon and Yelp datasets are more suitable for sentiment analysis. Our attention models in sentiment analysis perform even better than RCNN. Our attention models may capture more context information than the baseline model and RCNN, which is helpful for sentiment analysis. In addition, the average length of sentence in Amazon and Yelp is much longer

than our setting sequence length value. Thus, our attention models may perform better if we set a higher sequence length for these long sentence datasets. Fig.4 shows the robustness of our attention models. The training sets of AGNews may have noisy or outliers. Therefore, the performance is decreasing when models use all training samples from the training set. Our attention models can only take 20% training samples and outperform the baseline model CNN using 80% training samples of the training set. The LSTM with attention model, has less parameters than another attention model, and hence, this model is easier to train and more robust than Encoder-Decoder attention model.

Table 3. The test accuracy rate for 4 models in 4 dataset

Dataset	baseline CNN	RCNN	LSTM Attention	Encoder-Decoder
AGNews	78.48%	**90.85**%	90.23%	88.67%
Dbpedia	84.73%	**96.57**%	95.88%	96.26%
Amazon	74.85%	**88.91**%	88.12%	87.38%
Yelp	75.29%	85.12%	85.20%	**85.45%**

Table 4. The average Macro-F1 score for 4 models in 4 dataset

Dataset	baseline CNN	RCNN	LSTM Attention	Encoder-Decoder
AGNews	0.78	**0.90**	0.90	0.88
Dbpedia	0.83	**0.97**	0.95	0.96
Amazon	0.73	**0.88**	0.88	0.86
Yelp	0.73	0.85	0.85	**0.86**

5 Conclusion and Future Work

We attempt to convert text classification to a output fixed length machine translation problem and propose two attention models. Our attention models learn context information via using attention mechanism. The experiments illustrate our attention models are effective and robust. In sentiment analysis tasks, our attention models even perform better than another novel model RCNN [6].

For future work, we will increase the sequence length for long sentence datasets and introduce TF-IDF in data preprocessing phase to improve the performance of our models. Further, we consider to use BERT [7], another novel word embedding pre-trained model, to replace Glove. At last, we think that researchers need to focus on the sentiment analysis tasks which may not have a word imply the class of this sentence.

References

1. Pennington, J., Socher, R. and Manning, C. D. "Glove: Global Vectors for Word Representation". EMNLP pp. 1532–1543 (2014)
2. Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification". In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1 (NIPS'15). MIT Press, Cambridge, MA, USA, 649–657.(2015)
3. Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, Nov. 1998.
4. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. (January 2014)
5. Bahdanau, Dzmitry ; Cho, Kyunghyun ; Bengio, Yoshua: Neural Machine Translation by Jointly Learning to Align and Translate, 2014
6. Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. Recurrent convolutional neural networks for text classification. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI'15). AAAI Press, 2267–2273. (2015)
7. Devlin, Jacob, Chang, Ming-Wei and Lee, Kenton and Toutanova, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2018)
8. Xing, Eric, Jordan, Michael and Karp, Richard. Feature Selection for High-Dimensional Genomic Microarray Data. Proc 18th International Conf on Machine Learning. (2001)
9. Zaffalon, M., Hutter, M. Robust feature selection by mutual information distributions. In Darwiche, A., Friedman, N. (Eds), UAI-2002: Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence. Morgan Kaufmann, San Francisco, 577–584. (2002)
10. LeCun, Y., Bengio, Y. and Hinton, G. Deep Learning. Nature, 521, 436–444. doi: 10.1038/nature14539 (2015)
11. Blei, D. M., Ng, A. Y., and Jordan, M. I. Latent dirichlet allocation. Journal of machine Learning research, 3(Jan), 993-1022. (2003)
12. Rumelhart, D., Hinton, G., and Williams, R. Learning representations by back-propagating errors. Nature 323, 533–536 (1986)
13. Y. Bengio, P. Simard and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," in IEEE Transactions on Neural Networks, vol. 5, no. 2, pp. 157-166, doi: 10.1109/72.279181. (1994)
14. Luong, Minh-Thang, Hieu Pham, and Christopher D. Manning. "Effective approaches to attention-based neural machine translation." arXiv preprint arXiv:1508.04025 (2015)