Final report : Research on Text classification

Yifei Xu (304880196), Yaxuan Zhu (704947072), Ruiqi Gao (104864885)
Department of Statistics
University of California, Los Angeles
fei960922@ucla.edu, yaxuanzhu@ucla.edu, ruiqigao@ucla.edu

1 Introduction and Related Work

Text classification is a common task in natural language processing (NLP). It predicts the most relevant label(s) to a given sentenses or paragraphs. It is widely used in tag recommendation, information retrieval, document categorization, etc.

A good representation of text is key to this problem. There are multiple works on this field including [8] [4] [1] [2]. ELMO [5] generalize the classical context-independent word embedding to context-dependent ones using the hidden layer features of a bi-directional LSTM. BERT [3] replaces the LSTM module with self-attention blocks and get superior results in 11 NLP tasks. The self-attention blocks, proposed in [6] replaces the original word embedding with the weighted combination of all the words in the content, where the weights is computed by the similarity between the pivot word and all other words. It is believed that this method can capture the long term dependency more efficiently than traditional recurrent based model such as RNN and LSTM because each word can directly interact with all other words in the content. In addition to computing the attention based on the information in the sentence, a recent work, LEAM [7] proposes to also embed the each classification category as vectors and use this information to compute attention weights on each input sentences. One can understand this as a mechanism that filters out information that is irrelevant to the classification task.

Although achieving good results, the original LEAM paper employs only the context-independent word embedding Glove as the input. Besides, it aggregates the information across different positions in the sentence by a simple sum, instead of more complex recurrent model, e.g. LSTM. Therefore, in our project, we want to bring all these methods together. We try to answer the following questions:

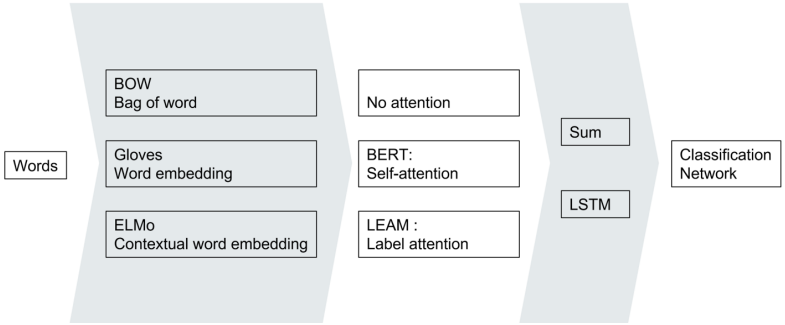
- Whether introducing context-dependent embedding (BERT and ELMO) will help.
- Whether using more complex information aggregate function (LSTM) helps.

2 Methodology

2.1 Work Flow

Our text classification problem can be roughly divided into four main parts — word representation, attention, text representation and classification.

- Word representation : Transform alphabet word into a vector. In this stage, we consider both context independent embedding like BOW and Glove, as well as the context dependent embedding, like ELMO and BERT.
- Attention mechanism : When a word sequence is converting into vectors, attention mechanism may be employed to apply different weight to different positions. The attention mechanism we compare here are self-attention used in BERT, which extract information based on the context itself as well as class embedding used in ELMO, which filters out classification-unrelated information.



- Text representation : Extract text representation from a list of word vectors, which typically have variant length for different text. We consider direct summing and LSTM model here.
- Classification : Classify the text by traditional Neural network. We employ a MLP model for this stage.

2.2 ELMO

Proposed in [5], ELMO generalize the context independent embedding into context-dependent ones. It employs a multi-layer bi-directional LSTM model as encoder and trains this encoder using the reconstruction loss (i.e. predict a word given its previous and later words). Then, given a word and its context, the output of each hidden layer can be seen as a context-dependent embedding of this words. Since there are multiple layers, the author use the weighted linear combination across all the hidden layers. Shown in 1, for each word k, the final ELMO embedding is the linear combination of across the hidden layer j. The parameter γ^{task} and s_j^{task} are trained with the downstream task. s_j^{task} is the output of a softmax layer and it controls the relative importance of features from each layer while γ^{task} serves as a scaling factor.

$$ELMO_k^{task} = \gamma^{task} \sum_{j=0}^{L} s_j^{task} h_{kj} \tag{1}$$

2.3 BERT

Unlike ELMO, BERT [3] replace the LSTM module with transformer module. It is trained on 2 tasks, Masked LM and Next Sentence Prediction. In Masked LM, a small percentage of a given sentence is randomly masked out and the model needs to learn how to predict this missing word given its surrounding context. This setting solve the indirectly "see yourself" problem in the original ELMO training process. On the other hand, in the Next Sentence Prediction task, the model is asked to predict whether the second sentence is the next sentence of the first one. This is helpful for the downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI). Given a word sequence, the BERT can encode each word into a context-dependent word embedding. It will also generate a classifier token [CLS] to represent the information of the whole sentence. Usually, people directly using the classifier token as the representation of the whole sentence and do classification on the top of it. However, in our project, to combine BERT with LEAM, we take the embedding of each word and pass them to the LEAM model instead of the classifier token.

2.4 LEAM

We denote the embedding of a word as e-dimensional vector , and construct a embedding matrix concatenating all the words in the whole document as: $\mathbf{Q} = \{_1, ..., _i, ..., _m\} \in^{(e \times m)}$. We also embed the labels into the same embedding space, denoted as $\mathbf{C} = \{\mathbf{c}_1, ..., \mathbf{c}_l, ..., \mathbf{c}_L\} \in^{(e \times L)}$. Then we align all the label-word pairs via the cosine similarity:

$$\mathbf{G} = \frac{\mathbf{CQ}}{|\mathbf{C}| \cdot |\mathbf{Q}|} \tag{2}$$

$\mathbf{G} \in^{(L \times m)}$ measures the similarity between each label and word token with respect to the document, in order to determine how relevant a word is to each label. Nonetheless, the local context can impact the word meaning, so an $1-d$ convolution with ReLU activation is added on to obtain an alignment between the label and each $2r+1$ gram window center at word $_i$:

$$a = \{a_1, ..., a_i, ..., a_m\} \qquad a_i = ReLU\left(\mathbf{G}_{i-r:i+r}W + b\right) \tag{3}$$

Then the attention α is obtained by max-pooling and softmax, resulting in the weights for each word in the document, and finally the composition of the document vector is computed as the weighted average:

$$\alpha_i = softmax\left(max\_pooling\left(a_i\right)\right) \tag{4}$$

$$X = \sum_{i}^{m} \alpha_{ii} \tag{5}$$

The label-attended document representation can be used as the input to a multi-label classification algorithm, the embedding weights and attention parameters can be learnt or fine-tuned simultaneously with the classifier. In our experiments, we compare on different datasets the effect of using a pre-trained word embedding and learn the embedding from random initialization in the target corpus.

An interesting byproduct of LEAM, is the attention weights on each word α, such that we may highlight the most 'important' (more weights in the representation composition) words in the document. It will also be exemplified in the experiments. In addition, there are several potential variants of LEAM architecture. The word-label alignment can be computed in alternative ways. Some of them will be compared in the experiments.

3 Experiments

Code is available at `https://github.com/fei960922/CS269_LEAM`.

3.1 Dataset

In our project, we are planning to use one of these public dataset: Reuters Newswire Topic Classification (Reuters-21578). This is a multi-label classification task. The data is a collection of news documents that appeared on Reuters in 1987 indexed by categories. It has 90 classes, 7769 training documents and 3019 testing documents. Each of these classes can independently has a label of 1 or 0. In other words, there are 90 independent binary classification task.

3.2 Evaluation methods

For our multi-label classification task, we use two metrics. We denote the true positive as Y , predicted positive as $\hat{Y}$, true negative as Y^c and predicted positive as $\hat{Y}^c$.

1. In order to account for partial correctness, Godbole et al. adopted **Hamming Score** to measure **Multilable Accuracy** as a symmetric measure of the distance between two binary vectors. We also introduce **False Positive Rate** (FPR) and **Recall** measure for multi-label classification. Using this definition, we can show the ROC curve.

$$Accuracy = \frac{1}{N}\sum_{d=1}^{N}\frac{|Y_d \cap \hat{Y}_d|}{|Y_d \cup \hat{Y}_d|} \qquad FPR = \frac{1}{N}\sum_{d=1}^{N}\frac{|Y_d^c \cap \hat{Y}_d^c|}{|Y_d^c|} \qquad Recall = \frac{1}{N}\sum_{d=1}^{N}\frac{|Y_d \cap \hat{Y}_d|}{|Y_d|}$$

2. The final metric is a strict one, **Exact Match Ratio** : $EMR = \frac{1}{N}\sum_{d=1}^{N} 1(Y_d = \hat{Y}_d)$. Only all multi label is right hit an exact match.

3.3 Results

First we only consider the LEAM as our attention model. We combine LEAM with context independent embeddings learning from random and pretrained by Gloves. We also compare using sum and using LSTM to extracting text vector.

Model	Training EMR	Testing EMR	Training Accuracy	Testing Accuracy
Random+sum	0.8066	0.6565	0.8589	0.7055
Random+LEAM+sum	0.9197	0.7350	0.9510	0.7860
GloVes+sum	0.7009	0.6380	0.7614	0.6879
GloVes+LEAM+sum	**0.9065**	**0.7906**	**0.9468**	**0.8580**
Random+LEAM+LSTM	0.6866	0.5808	0.7330	0.6083
Glove+LEAM+LSTM	0.8419	0.6459	0.8915	0.7410

Table 1: LEAM Results Comparison

As we can see, in all setting, adding LEAM performs better. Also, a good pretrained word embedding do helps. Although about 50% of the words are not appeared in Gloves and we set them as <Unknown>, Gloves still performs better than training word embedding fram stratch. We believe it is due to the connections between words will definitely helps the classification while the training data itself is not capable to found such connection.

However, LSTM has a worse result. We can see a strong overfitting in LSTM setting. We try to decrease the hidden layers and hidden dimension in dense layer to make sure the parameter size are the same. However, it still led to overfitting. We think it is due to the parameter initialization and learning rate in both LSTM layer and embedding training are the same. We will try to seperate the later.

Since we found the result says that replacing sum with LSTM actually given us worse testing accuracy. so we just use sum in the later on experiments.

Secondly, we take into consideration those context-dependent embeddings (ELMO and BERT). We report the ROC curve in Figure 1 and the testing results in Table 1. From these results we can see that introducing text dependent embedding like ELMO and BERT can further improve the performance of LEAM. This means class embedding can not take the position of context embedding and self attention. On the other hand, if we compare models with LEAM as a class embedding method and those without LEAM, we can clear see that those without LEAM have a big accuracy degradation. This means context embedding also can not take the place of class embedding. The class attention mechanism can filter out the class irrelevant information and help the classification result.

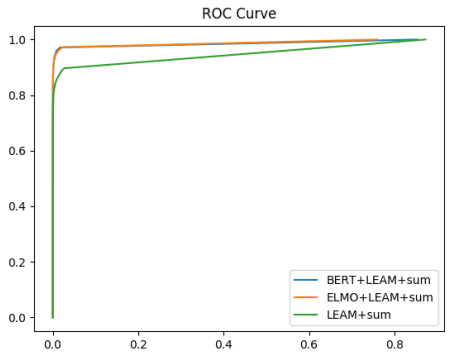


Figure 1: ROC curves for different methods

Model	Training EMR	Testing EMR	Training Accuracy	Testing Accuracy
ELMO+sum	0.8568	0.7208	0.8915	0.7389
ELMO+LEAM+sum	0.9334	**0.8614**	0.9574	0.8686
BERT+sum	0.9183	0.7320	0.9504	0.7945
BERT+LEAM+sum	**0.9632**	0.8191	**0.9764**	**0.8695**

Table 2: Text representation Comparison

4 Conclusion

In this project, we bring different embedding and attention methods together. Our results show that self-attention from context and attention from class label can extract useful information from different aspects. They will help the classification result and they can not replace each other. On the other hand, those complex aggregate methods like LSTM may not be needed when those attention methods are introduced. They may even degrade the performance due to hard to try.

Acknowledgments

Template is NIPS2017 template downloaded from NIPS2017 website. Proposal is for 2019 Spring CS269 Project. NO DISTRIBUTION.

References

[1] Hareesh Bahuleyan, Lili Mou, Olga Vechtomova, and Pascal Poupart. Variational attention for sequence-to-sequence models. *arXiv preprint arXiv:1712.08207*, 2017.

[2] Yuntian Deng, Yoon Kim, Justin Chiu, Demi Guo, and Alexander Rush. Latent alignment and variational attention. In *Advances in Neural Information Processing Systems*, pages 9712–9724, 2018.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[4] Yue Jiao, Jonathon Hare, and Adam Prügel-Bennett. Probabilistic semantic embedding, 2019.

[5] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[7] Guoyin Wang, Chunyuan Li, Wenlin Wang, Yizhe Zhang, Dinghan Shen, Xinyuan Zhang, Ricardo Henao, and Lawrence Carin. Joint embedding of words and labels for text classification. *arXiv preprint arXiv:1805.04174*, 2018.

[8] Ronghui You, Suyang Dai, Zihan Zhang, Hiroshi Mamitsuka, and Shanfeng Zhu. Attentionxml: Extreme multi-label text classification with multi-label attention based recurrent neural networks. *arXiv preprint arXiv:1811.01727*, 2018.