Proceedings of CONF-MLA 2025 Symposium: Applied Artificial Intelligence Research
DOI: 10.54254/2755-2721/2025.BJ38141

A Parameter-Independent Fewshot Text Categorization Incorporating Semantic Information

Yibin Wang

Zhejiang Wanli University, Ningbo, China
2362576901@qq.com

Abstract. Deep learning is currently widely used in text classification due to its high accuracy. However, deep learning tends to require a large amount of training resources, while deep learning also tends to suffer from overfitting with fewer labeled samples. Jiang et proposed an alternative scheme without training parameters that is simple, lightweight, and versatile in text catego-rization: a simple text compressor (e.g., gzip) combined with a k-nearest-neighbor classifier is used for text categorization. However, npc-gzip only considers the presence of identical words, ignoring semantic similarity. Therefore, we propose a method based on that paper to integrate semantic information, which outperforms the non-pretrained baseline method, npc-gzip, in the few-shot setting.

Keywords: npc-gzip, Huffman coding, k-Nearest Neighbors, Word embeddings

1. Introduction

With the development of deep learning already, text categorization has progressed by leaps and bounds. However, deep learning models often require a large amount of labeled data in order to have a better generalization effect, while also investing large training resources [1, 2]. Deep learning is able to capture latent knowledge with this addition, and is able to recognize harder cases such as negative transfer and false correlation, but for simple semantic categorization tasks such as topic categorization, deep learning is often too complex for such tasks, and lightweight methods like machine learning are generally more suitable. For example, Shen et al. [3] demonstrate that even word embedding-based classification methods can achieve results comparable to those of Convolutional Neural Networks (CNN) and Long Short-Term Memory Networks (LSTM) [4].

In addition to this npc-gzip proposes a text categorization method that combines a text compressor, distance between compressed texts and a k-nearest neighbor classifier (kNN). It utilizes a compressor to compress the text into sequences after converting it into utf-8 encoding, and then the distance measure between the compressed sequences is used as the similarity between the text and the text. Finally using the distance vectors of test samples and labeled samples, text classification is performed using kNN. However, the method directly uses the compressed sequence to compute the similarity, which often leads to the similarity computation focusing only on the same single words, thus ignoring semantic similarity. For example, the semantic similarity of the words 'supermarket' and 'mall' is very high, however, when looking at the spelling of the two words intuitively, they are completely unrelated sequences, and the similarity between the two words is

© 2026 The Authors. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).

DOI: 10.54254/2755-2721/2025.BJ38141

very small, which affects the effectiveness of text classification. In order to combine semantic similarity and tree, we propose an encoding algorithm based on semantic similarity: using the embedding similarity of words as a measure, the greater the similarity, the more identical prefixes will be in the encoded sequences of the words, and thus the sequences of the semantically similar words will have greater similarity after encoding.

Our contributions are as follows:(1) we propose a text encoding algorithm that complements the problem that directly compressed text cannot take semantic similarity into account;(2) experiments show that our method performs well in a few sample experiments with sparsely labeled data.

2. Methods

In general, we build upon the npc-gzip method, which consists of a lossless compressor (gzip), a distance metric based on the compressor, and a k-nearest neighbor classifier. The gzip compressor generalizes the principle of assigning shorter sequences to symbols occurring at high frequencies, aiming to use as few bits as possible to represent the information. The compression algorithm is employed to calculate the distance between samples, as samples from the same category tend to have similar vocabulary representations., e.g., s1 and s2 belong to the same category, and s1 and s3 belong to a different category. s1 and s2 will have a higher probability to have the same vocabulary, and the compression rate of gzip(s1+s2) will be higher than that of gzip(s1+s3), which will make it easy to obtain gzip(s1+s2) - gzip(s2) - gzip(s1+s2) - gzip(s1+s2) - gzip(s1+s2) - gzip(s1+s3) gzip(s2)< gzip(s1+s3) - gzip(s3). From there we obtain the distance metric, but the formula is affected by the absolute length of the text, we expect to obtain the relative distance, and the final distance metric formula used by npc-gzip is:

$$\mathrm{dis}\left(\mathrm{s1},\ \mathrm{s2}\right) = \frac{\mathrm{gzip(s1\ +\ s2)} - \min(\mathrm{gzip(s1)},\ \mathrm{gzip(s2)})}{\max(\mathrm{gzip(s1)},\ \mathrm{gzip(s2)})}$$

However, this method can only achieve high compression rate when the same word occurs, and cannot effectively compress words with the same semantic meaning. Example For example, 'supermarket' and 'mall' are two semantically similar words, we expect the distance between them to be small, and by the above formula, dis('supermarket', 'mall') = 0.3548; and the distance between 'supermarket' and 'superman', which does not want to die at all semantically, is dis('supermarket', 'superman') = 0.2580.

To address the above problem, we expect to construct an encoding method that will semantically similar words have the same prefix, so we use Huffman coding to recode the words, using each word as the number of individual nodes and the semantic similarity between the word pre-training word vectors as the distance metric, and we will select the two node trees with the largest semantic similarity at each level to merge, and use the two trees' mean value of glove embedding as the value of the new tree, and repeat the above steps until only one tree remains. And as with Huffman encoding, encode each leaf node of the tree, thus ensuring that the encoding of the two words with the largest semantic similarity is only the last bit inconsistent, while all prefixes are consistent; and for the words with more similar semantics, it can also be ensured that most of the prefixes are consistent, with only a small number of suffixes differing. Figure 1 shows an example of word semantic-driven Huffman coding method.

After completing the construction of the Huffman tree, we can recode each word, if we use the Huffman coding of '01' encoding, it will inevitably cause a large number of repetitions, for this reason, we will be given to each node a nonrepeating code, the overall process is shown in the

DOI: 10.54254/2755-2721/2025.BJ38141

following figure The leaf nodes are all the words, represented in black font, and the red font is the encoded character of the current node, for the single word 'the', the red font is the encoded character of the current node. For example, our encoding algorithm transforms the word 'the' into 'abd', while the word 'market' is encoded as 'aci...$', and 'mall' is encoded as 'aci...%'. The semantically similar words 'market' and 'mall' will differ only in the final character of their encoding, with the majority of their prefixes remaining consistent. After obtaining the new word encoding, we will splice the original text, and then use the above distance measure formula to obtain the distance between the samples to be detected and all labeled samples, and obtain the topn labeled samples with the shortest distance, and then use the plurality label as the final prediction label.

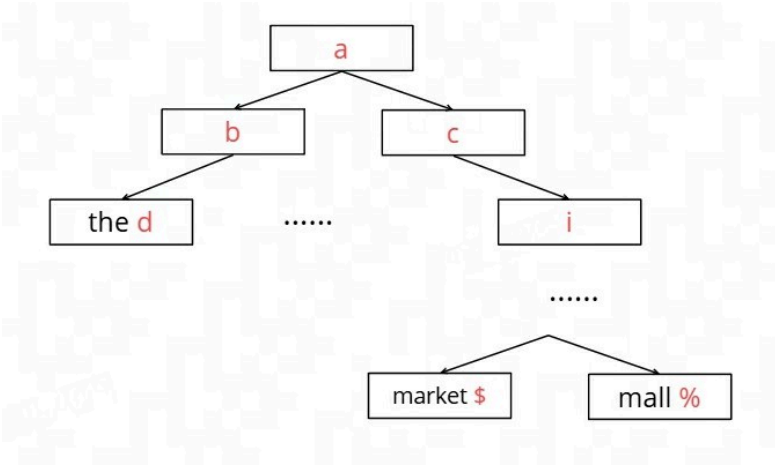


Figure 1. Word semantic-driven Huffman coding method

3. Results

In the low-resource labeling (fewshot) case, we compare our method with fasttext [5], Bi-LSTM+Attn [6], W2V [7], and npc-gzip, which are non-pre-trained baseline methods. In our experiments, we used two datasets AG news, DBpedia to validate the effectiveness of the methods using 5shot, 10shot, 50shot and 100shot respectively. Experimental settings: in the pre-training word vector in the coding algorithm, we chose the 25-dimensional glove word vector trained using twitter data, in the classification algorithm, the k hyperparameter used by the knn algorithm we set to 2; in the selection of the training data, we used each dataset, each category of 5, 10, 50, and 100 as the training samples, of which the following table of the npc-gizp experimental results are our reproduction results, not the paper results. Table 1 shows the results of AG new dataset, Table 2 shows the results of DBpedia Dataset.

Table 1. AG new dataset results

Model	5 Shot	10 Shot	50 Shot	100 Shot
fastText	0.273	0.273	0.329	0.550
Bi-LSTM+Attn	0.269	0.331	0.549	0.665
W2V	0.388	0.546	0.531	0.531
npc-gzip	0.528	0.600	0.698	0.734
sen-gzip(outs)	0.596	0.646	0.716	0.759

DOI: 10.54254/2755-2721/2025.BJ38141

Table 2. DBpedia Dataset results

Model	5 Shot	10 Shot	50 Shot	100 Shot
fastText	0.616	0.475	0.616	0.767
Bi-LSTM+Attn	0.506	0.648	0.818	0.862
W2V	0.325	0.402	0.675	0.787
npc-gzip	0.639	0.639	0.823	0.858
sen-gzip(outs)	0.675	0.735	0.852	0.883

Our method significantly outperforms previous nonpretrained approaches in the few-shot experiments., and our method also carries a significant improvement compared to the npc_gzip algorithm that does not incorporate semantics on the AG news dataset. On the AG News dataset, our method improves by 6.8%, 4.6%, 1.8%, and 1.5% over the npc-gzip algorithm for 5-shot, 10-shot, 50-shot, and 100-shot settings, respectively.

4. Conclusion and future work

In this paper, we used combined semantic encoding approach with compression algorithm and knn for text categorization. And in the case of low-resource sample labeling, our method shows significant improvement compared to other baseline methods. However, our method currently encodes based solely on fixed word semantic information, and in the future, we will further extend it by combining sentence semantic information.

References

[1] Jiang, Z., Yang, M., Tsirlin, M., Tang, R., Dai, Y., & Lin, J. (2023). "LowResource" text classification: A parameter-free classification method with in findings of the Association for Computational Linguistics: ACL 2023, pages 6810-6828, Toronto, Canada. Association for Computational Linguistics.

[2] Li, Q., Peng, H., Li, J., Xia, C., Yang, R., Sun, L., Yu, P. S., & He, L. (2022). A survey on text classification: From traditional to *ACM Transactions on Intelligent Systems and Technology (TIST), 13*(2), 1-41.

[3] Shen, D., Wang, G., Wang, W., Min, M. R., Su, Q., Zhang, Y., Li, C., Henao, R., & Carin, L. Baseline needs more love: On simple wordembedding-based models and associated pooling mechanisms. more love: On simple wordembedding-based models and associated pooling mechanisms. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume I)*, New York, USA. Computational Linguistics (Volume 1: Long Papers), pages 440-450.

[4] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation, 9*(8), 1735-1780.

[5] Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *EACL 2017*, 427.

[6] Wang, Y., Huang, M., Zhu, X., & Zhao, L. (2016). Attention-based LSTM for aspect-level sentiment classification. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 606-615.

[7] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv*, 1301.3781.