A Short Text Classification Method Based on Fusion of Word Vectors and LDA (Postprint)

Authors: Qun Zhang, Wang Hongjun, Wang Lunwen

Date: 2017-11-08T00:00:00+00:00

Abstract

【Purpose】To address the issues of poor thematic focus and severe feature sparsity in short texts, we propose a short text classification method based on the fusion of word embeddings and LDA topic model.

【Method】Fine-grained semantic modeling is performed on short texts simultaneously from both "word" granularity and "text" granularity perspectives. First, word embeddings are trained using Word2Vec, and short text vectors at the "word" granularity level are synthesized through an additive averaging method. The LDA topic model is trained via Gibbs sampling, and feature expansion is conducted on short texts according to the principle of maximum topic probability. Subsequently, weights of expanded features are calculated based on word embedding similarity to obtain short text vectors at the "text" granularity level. Finally, a fused word embedding and LDA short text representation model is constructed through vector concatenation, upon which short text classification is accomplished via a nearest neighbor classification algorithm.

【Results】Compared with three traditional single-model-based classification methods—namely Vector Space Model-based, word embedding-based, and LDA topic model-based approaches—the proposed fusion method demonstrates improvements in accuracy, recall, and F1-score, with minimum enhancements of 3.7%, 4.1%, and 3.9%, respectively.

【Limitations】The method has only been applied to the nearest neighbor classifier and has not yet been extended to various other classifiers such as Naive Bayes and Support Vector Machines.

【Conclusion】 Classification based on the fused word embedding and LDA short text representation model can effectively overcome the problems of poor thematic focus and feature sparsity in short texts, thereby improving short text classification performance.

Full Text

Preamble

Classifying Short Texts with Word Embedding and LDA Model

Zhang Qun, Wang Hongjun, Wang Lunwen
(Electronic Engineering Institute of PLA, Hefei 230037, China)

Abstract

[**Objective**] This paper proposes a short text classification method that integrates word embedding with the LDA topic model to address the problems of poor topic focus and severe feature sparsity in short texts. [**Methods**] We simultaneously model short texts at both the "word" and "text" granularities for fine-grained semantic representation. First, we train word vectors using Word2Vec and synthesize short text vectors at the "word" granularity through additive averaging. We then train an LDA topic model using Gibbs sampling and expand short text features according to the principle of maximum topic probability. Next, we calculate the weights of expanded features based on word vector similarity to obtain short text vectors at the "text" granularity. Finally, we construct an integrated short text representation model by concatenating these vectors, upon which classification is completed through a nearest neighbor classification algorithm. [**Results**] Compared with traditional classification methods based on single models—Vector Space Model, word embedding alone, and LDA topic model alone—the proposed fusion method achieves improvements of at least 3.7%, 4.1%, and 3.9% in precision, recall, and F1-score, respectively. [**Limitations**] The method has only been applied to nearest neighbor classifiers and has not yet been extended to other classifiers such as Naive Bayes and Support Vector Machines. [**Conclusions**] The proposed short text representation model, which fuses word embedding with LDA, effectively overcomes the problems of poor topic focus and feature sparsity in short texts, thereby improving short text classification performance.

Keywords: Short text classification; Word embedding; LDA topic model; Nearest neighbor classifier

1. Introduction

The intelligence of mobile terminals has catalyzed the rapid development of mobile Internet. To accommodate mobile users' reading habits, Internet content is increasingly presented in short text forms such as microblogs and instant news feeds. Automatically classifying massive amounts of short text content has become a hot research topic.

Over the past decades, scholars have proposed and refined a series of classic machine learning algorithms, including k-Nearest Neighbors (k-NN) [1], Naive Bayes (NB) [2], and Support Vector Machines (SVM) [3], successfully applying them to text classification with satisfactory results. However, compared with

conventional long texts, emerging mobile Internet short texts exhibit characteristics of brevity, weak descriptive capacity, and dispersed topics, causing these classic methods to face severe feature sparsity problems [4] and resulting in suboptimal classification performance.

Text data representation is crucial for text classification, as its quality directly affects classification effectiveness. Traditional text classification algorithms typically rely on the Vector Space Model (VSM), representing texts through vectors of feature terms and weights [5]. This method ignores semantic relationships between words, fails to capture deep-level topic information, and suffers from high-dimensional sparsity—problems that become more severe when representing short texts. Recent research has addressed this issue from three main directions. Some scholars introduce external knowledge bases (such as search engines, Wikipedia, and HowNet) to semantically expand text features and enrich inter-word semantic relationships [6-7]. While these methods can alleviate sparsity to some extent, they heavily depend on the quality of external knowledge bases, cannot handle topics not covered in these bases, and involve high computational costs and long processing times, making them less effective for topic-dispersed short texts. Other researchers map original high-dimensional feature spaces to lower-dimensional latent semantic or topic spaces to mine potential semantic structures. For instance, Latent Semantic Analysis (LSA) represents texts as low-dimensional latent semantic vectors [8], reducing dimensionality and noise while improving sparsity, but may impair classification performance and yields semantically ambiguous dimensions. In contrast, the Latent Dirichlet Allocation (LDA) topic model represents texts as probability distributions over latent topics [9], significantly improving high-dimensional sparsity, overcoming LSA' s classification impairment, and providing interpretable topic dimensions, thus gaining wide application. References [10-11] directly classify texts in the LDA topic space, but this approach offers limited improvement for short text sparsity due to poor topic focus. References [12-14] expand short text features based on LDA topic models, achieving better results than direct LDA application. The aforementioned VSM, LSA, and LDA models all directly derive short text vectors for representation, belonging to "text" -granularity models. The latest research considers modeling texts at the "word" granularity for finer semantic expression, first deriving word vector representations and then composing them into short text vectors [15]. This approach effectively addresses short texts' topic dispersion and poor focus, though simple yet effective composition methods remain to be studied—references [16-17] construct composition models through neural networks with high complexity.

Building upon this analysis, this paper organically fuses word embedding with LDA, proposing a novel short text classification method that simultaneously models short texts at both "word" and "text" granularities to address feature sparsity and poor topic focus. We synthesize "word" -granularity short text vectors through simple additive averaging, avoiding complex composition processes. For "text" -granularity modeling, rather than directly applying LDA to map short texts to topic dimensions, we expand short text features based on

maximum topic probability principles and calculate expanded feature weights using word vector similarity, thereby constructing a fused representation model. Additionally, training word vectors and LDA models does not rely on labeled data—only classifier training requires a small amount of labeled data, making this a semi-supervised learning approach [18].

2. Word Vector Training and LDA Modeling

2.1 Word Vector Training Based on Word2Vec

Word vectors provide a mathematical representation where each dimension corresponds to a semantic feature, and distances or similarities between vectors reflect semantic relationships between words. The Distributional Hypothesis states that word semantics are determined by their contexts. A neural network-based method for obtaining word vectors has been widely studied, modeling the relationship between target words and their contexts to obtain low-dimensional dense vectors rich in semantic information. Bengio et al. proposed the Neural Network Language Model (NNLM), where word vectors are obtained as a byproduct during language model training [19]. NNLM employs a three-layer feedforward neural network structure, as shown in [Figure 1: see original paper] [19].

In the NNLM structure, w_i is the target word with context being a word sequence $\{w_{i-n+1}, \dots, w_{i-1}\}$. The input layer maps context words to word vectors through a matrix C, then concatenates them sequentially as model input:

$$x = (C(w_{i-n+1}), \dots, C(w_{i-1}))$$

The hidden and output layers are defined as [19]:

$$h = \tanh(Hx + b)$$

$$y = Uh + Wx + d$$

where tanh is the hidden layer activation function, H is the weight matrix from input to hidden layer, U is the weight matrix from hidden to output layer, W is the direct connection weight matrix from input to output layer (often ignored), and b, d are bias terms. The model normalizes output y to a probability distribution over target words through the Softmax function [19]:

$$P(w_i|w_{i-n+1}, \dots, w_{i-1}) = \frac{\exp(y(w_i))}{\sum_{j=1}^{|V|} \exp(y(w_j))}$$

The model iteratively optimizes parameters to maximize this probability, including the word vector parameter matrix C.

NNLM' s computational cost concentrates on the matrix multiplication Uh in the hidden-to-output layer. Additionally, when vocabulary size $|V|$ is large, the Softmax calculation becomes very time-consuming.

Building on NNLM, this paper uses Word2Vec for word vector training. Word2Vec is an open-source tool based on Mikolov et al.' s CBOW (Continuous Bag-of-Words) and Skip-gram models [20]. These models resemble NNLM but differ in that NNLM obtains word vectors indirectly while training a language model, whereas CBOW and Skip-gram directly target word vector acquisition. Word2Vec thus simplifies and improves NNLM by: (1) Removing the hidden layer to avoid complex matrix multiplication Uh; (2) Replacing NNLM' s vector concatenation with additive averaging in CBOW' s input layer to reduce computational complexity [20]:

$$x = \frac{1}{n-1} \sum_{j=i-(n-1)/2, j\neq i}^{i+(n-1)/2} C(w_j)$$

where $w_{i-(n-1)/2}, \ldots, w_{i+(n-1)/2}$ (excluding w_i) represent CBOW' s context of $(n-1)/2$ words before and after target word w_i, providing more complete context than NNLM' s use of only preceding words.

CBOW predicts target words from context, while Skip-gram predicts context from target words. Their structures are shown in [Figure 2: see original paper] and [Figure 3: see original paper] [20]. To address NNLM' s expensive Softmax computation, Word2Vec employs two optimization algorithms: Hierarchical Softmax with Huffman coding [21] and Negative Sampling [15].

For short text classification tasks, we found that corpus size and model choice affect word vector quality and classification performance. We summarize the following empirical guidelines:

1. When corpus size exceeds 200MB, CBOW outperforms Skip-gram; below 100MB, the opposite holds; between 100MB-200MB, performance differences are negligible.
2. CBOW' s additive averaging reduces complexity but ignores word order information. We experimented with concatenation instead of averaging in CBOW' s input layer to incorporate order information, but results showed no significant performance difference.

2.2 LDA Modeling Based on Gibbs Sampling

LDA is a three-layer Bayesian generative model of "document–topic–word" that simulates text generation, modeling documents as probability distributions over mixed topics and topics as distributions over words, as shown in [Figure 4: see original paper] [9].

In [Figure 4: see original paper], symbols denote: M as total document count, N as word count per document, K as number of latent topics; θ as document-topic distribution matrix, Φ as topic-word distribution matrix, both following Dirichlet distributions with hyperparameters α and β; w as words, z as topic assignments; $D = \{d_1, d_2, \ldots, d_M\}$ represents the entire corpus, d_m the m-th document, w_m the word vector of d_m, and z_m the corresponding topic vector. The LDA generation process is:

1. For each document d_m, draw a topic distribution $\theta_m \sim \mathrm{Dir}(\alpha)$;
2. For each word w_{mn}, draw a topic $z_{mn} \sim \mathrm{Mult}(\theta_m)$;
3. Draw a topic-word distribution matrix $\Phi_m \sim \mathrm{Dir}(\beta)$, then determine word distribution $\Phi_{z_{mn}}$ for w_{mn} based on z_{mn};
4. Generate word $w_{mn} \sim \mathrm{Mult}(\Phi_{z_{mn}})$;
5. Repeat steps 2-4 for all N words to generate d_m;
6. Repeat steps 1-5 for all M documents to generate corpus D.

LDA aims to assign latent topics to each word in D and estimate θ and Φ by computing the posterior probability:

$$p(z, \theta, \Phi | w, \alpha, \beta) = \frac{p(w, z, \theta, \Phi | \alpha, \beta)}{p(w | \alpha, \beta)}$$

The denominator is intractable, so Gibbs Sampling is used as a simple and effective alternative.

Gibbs Sampling [22] is a special Markov Chain Monte Carlo (MCMC) method that generates a Markov chain through topic sampling for words. Let z_i be the current topic for vocabulary word t, depending on other sampled topics $z_{\neg i}$. The sampling formula is [22]:

$$p(z_i = k | z_{\neg i}, w) \propto \frac{n_{k,\neg i}^{(t)} + \beta}{n_{k,\neg i}^{(\cdot)} + |V|\beta} \cdot \frac{n_{m,\neg i}^{(k)} + \alpha}{n_{m,\neg i}^{(\cdot)} + K\alpha}$$

where $|V|$ is vocabulary size; $n_k^{(t)}$ counts word t assigned to topic k, $n_m^{(k)}$ counts topic k words in document d_m, and $\neg i$ denotes counts excluding current assignment.

After sampling, θ and Φ are estimated as [22]:

$$\theta_{mk} = \frac{n_m^{(k)} + \alpha}{n_m^{(\cdot)} + K\alpha}$$

$$\Phi_{kt} = \frac{n_k^{(t)} + \beta}{n_k^{(\cdot)} + |V|\beta}$$

3. Semi-supervised Classification Method Combining Word Vectors and LDA

3.1 Method Framework Description

LDA models texts at the "text" granularity and performs well on long texts but poorly on short texts. Word vectors excel at "word" granularity semantic similarity but require further research for text-level representation. Since short texts lie between these granularities, we propose a fusion method that simultaneously models them at both levels. Training word vectors and LDA requires no labeled data—only classifier training needs a small labeled set, making this semi-supervised [18]. The framework is shown in [Figure 5: see original paper].

The method comprises four steps: 1. Construct a large unlabeled dataset and a small labeled dataset, with preprocessing; 2. Train word vectors and LDA on the large unlabeled dataset; 3. Fuse word vectors and LDA to model short texts on the small labeled dataset; 4. Build a k-NN classifier for new short texts and evaluate performance.

3.2 Dataset Construction and Preprocessing

Dataset construction is critical for classification. While supervised learning requires large labeled sets, our semi-supervised approach needs only small labeled sets, reducing annotation workload. We construct large unlabeled set D and small labeled set D' with requirements: 1. Natural language expression patterns; 2. Domain consistency with classification tasks; 3. Comprehensive and balanced coverage of potential topics; 4. Sufficient domain/topic-related words in D.

Preprocessing includes Chinese word segmentation and stopword filtering. For the small labeled set, we additionally perform feature selection using χ^2 statistics, which measures term t' s correlation with category c [2]:

$$\chi^2(t,c) = \frac{(AD - BC)^2}{(A+B)(C+D)}$$

Parameter meanings are shown in .

3.3 Fused Short Text Representation Model

We train word vectors and LDA on large unlabeled data, then fuse them for short text modeling.

Input: 1. Small labeled short text training set D'; 2. Word vectors trained on D; 3. LDA model trained on D.

Output: Fused representation model for D'.

Step 1: Word Vector Composition
Use additive averaging to obtain word-vector-based short text representation:

$$d'_m = \frac{1}{N_m} \sum_{j=1}^{N_m} v(w_j)$$

where d'_m is the m-th short text representation, w_j are its words, N_m is word count, and $v(w_j)$ are word vectors.

Step 2: LDA-based Feature Expansion
Match each word in D' with LDA' s topic-word distribution matrix Φ, selecting its maximum-probability topic $z_{\max}$. Then match $z_{\max}$ with the topic-word file to select top r words as expansion features, yielding:

$$d''_m = \{w_{m1}, (c_{11}, \dots, c_{1r}), \dots, w_{mN_m}, (c_{N_{m1}}, \dots, c_{N_{mr}})\}$$

where d''_m is the LDA-based feature expansion model.

Step 3: Expanded Feature Weighting via Word Vectors
Gibbs sampling outputs θ, Φ, and a topic-word file showing top n words per topic (example in). For original feature w_{mn}, we compute TFIDF weight:

$$\text{weight}(w_{mn}) = \text{TFIDF}(w_{mn}) = \frac{\text{TF}(w_{mn}) \times \text{IDF}(w_{mn})}{\sum_{w \in d_m} [\text{TF}(w) \times \text{IDF}(w)]}$$

For expanded feature c_{nr}, its weight depends on: (1) its topic' s importance in the text, represented by w_{mn}' s TFIDF weight; (2) its relevance to the topic, measured by semantic similarity between c_{nr} and w_{mn} via cosine similarity of their word vectors:

$$\text{sim}(c_{nr}, w_{mn}) = \frac{v(c_{nr}) \cdot v(w_{mn})}{|v(c_{nr})| \times |v(w_{mn})|}$$

Thus, expanded feature weight is:

$$\text{weight}(c_{nr}) = \text{TFIDF}(w_{mn}) \times \text{sim}(c_{nr}, w_{mn})$$

Step 4: Vector Concatenation
Since words may have multiple senses, the same expanded feature might appear multiple times. Merge duplicates by summing weights. Finally, concatenate the feature expansion model d''_m with the word-vector model d'_m:

$$d_m = d'_m; d''_m$$

where ";" denotes sequential concatenation, yielding the fused representation.

3.4 Building the k-NN Classifier

k-NN is a classic, stable algorithm for text classification. It compares new data with training samples, selects the top k most similar samples' class labels as candidates, and assigns the majority label. We use cosine similarity for comparison.

4. Experiments

4.1 Experimental Setup

We use Fudan University' s Chinese text classification corpus as large-scale unlabeled data for training word vectors and LDA. A small labeled set of 1,000 short texts (<150 characters) is constructed, balanced across five domains: computer science, economics, environment, arts, and sports (200 per domain). A test set of 670 short texts is used (computer: 145, economics: 130, environment: 135, arts: 120, sports: 140), with no overlap between sets. Chinese segmentation uses NLPIR from the Chinese Academy of Sciences. Word2Vec training uses 50-dimensional vectors (no significant improvement beyond 50). LDA uses Gibbs sampling with $K = 100$ topics, $\alpha = 0.5$, $\beta = 0.1$, and 20 topic words, following GibbsLDA++ guidelines [23]. k-NN uses $k = 20$ (generally not exceeding the square root of training samples).

4.2 Evaluation Metrics

Classification performance is measured by Precision (Pr), Recall (Re), and F1-score [1]:

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}}, \quad \mathrm{Re} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}}, \quad \mathrm{F1} = \frac{2 \times \mathrm{Pr} \times \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}}$$

Parameter meanings are in .

4.3 Results and Analysis

Experiment 1: We modified CBOW by replacing additive averaging with vector concatenation to incorporate word order information. Comparing original CBOW, Skip-gram, and modified CBOW () shows CBOW outperforms Skip-gram for short texts. The modified CBOW shows only slight precision improvement but decreased recall and F1, indicating no significant overall difference. Thus, we adopt original CBOW for its lower complexity.

Experiment 2: We evaluate our method against three single-model baselines (VSM+k-NN, word-vector+k-NN, LDA+k-NN). Results in and show our method achieves satisfactory performance across all domains. Among baselines, LDA performs worst, even below bag-of-words, confirming its unsuitability for short texts. Our method improves precision by $ $3.7%, recall by $ $4.1%, and

F1 by $ $3.9% over all baselines. The fusion approach provides finer semantic representation, overcoming LDA' s poor topic focus and bag-of-words' sparsity.

5. Conclusion

This paper proposes a novel short text classification model fusing word embedding with LDA, simultaneously modeling at "word" and "text" granularities. The semi-supervised approach requires only unlabeled data for representation learning. Experiments demonstrate its superiority over single-model methods. Future work will extend the method to other classifiers.

References

[1] Yang Y, Liu X. A Re-examination of Text Categorization Methods. In: Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. ACM, 2003:42-49.
[2] Di Peng, Duan Liguo. New Naive Bayes Text Classification Algorithm[J]. Journal of Data Acquisition and Processing, 2014, 29(1): 71-75.
[3] Joachims T. Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms[M]. Springer Berlin, 2002.
[4] Wang Zhongyuan, Cheng Jianpeng, Wang Haixun, et al. Short Text Understanding: A Survey[J]. Journal of Computer Research and Development, 2016, 53(2): 262-269.
[5] Lebanon G. Metric Learning for Text Documents[J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2006, 28(4): 497-508.
[6] Zhu Zhengyu, Sun Junhua. Improved Vocabulary Semantic Similarity Calculation Based on HowNet[J]. Journal of Computer Applications, 2013, 33(8): 2276-2279,2288.
[7] Wang Rongbo, Chen Zhiqun, Zhou Jianzheng, et al. Short Texts Semantic Relevance Computation Method Based on Wikipedia[J]. Computer Applications and Software, 2015, 32(1): 82-85,92.
[8] Deerwester S, Dumais S T, Furnas G W, et al. Indexing by Latent Semantic Analysis[J]. Journal of the Association for Information Science and Technology, 1990, 41(6): 391-407.
[9] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet Allocation[J]. Journal of Machine Learning Research, 2003, 3: 993-1022.
[10] Yao Quanzhu, Song Zhili, Peng Cheng. Research on Text Categorization Based on LDA[J]. Computer Engineering and Applications, 2011, 47(13): 150-153.
[11] Rubin T N, Chambers A, Smyth P, et al. Statistical Topic Models for Multi-label Document Classification[J]. Machine Learning, 2012, 88(1-2): 157-208.
[12] Hu Yongjun, Jiang Jiaxin, Chang Huiyou. A New Method of Keywords Extraction for Chinese Short-text Classification[J]. New Technology of Library and Information Service, 2013(6): 42-48.
[13] Chen M, Jin X, Shen D. Short Text Classification Improved by Learning Multi-granularity Topics[C]. In: Proceedings of the 22nd International Joint

Conference on Artificial Intelligence. AAAI Press, 2011: 1776-1781.
[14] Phan X H, Nguyen L M, Horiguchi S. Learning to Classify Short and Sparse Text & Web with Hidden Topics from Large-scale Data Collections[C]. In: Proceedings of the 17th Information Conference on World Wide Web (WWW' 08). New York: ACM, 2008:91-100.
[15] Mikolov T, Sutskever I, Chen K, et al. Distributed Representations of Words and Phrases and Their Compositionality[J]. Advances in Neural Information Processing Systems, 2013, 26: 3111-3119.
[16] Turney P D, Pantel P. From Frequency to Meaning: Vector Space Models of Semantics[J]. Journal of Artificial Intelligence Research, 2010, 37(1): 141-188.
[17] Kim Y. Convolutional Neural Networks for Sentence Classification[C]. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing. Stroudsburg, PA: ACL, 2014: 1746-1751.
[18] Chapelle O, Schlkopf B, Zien A. Semi-Supervised Learning[J]. Journal of the Royal Statistical Society, 2010, 6493(10): 1-3.
[19] Bengio Y, Ducharme R, Vincent P, et al. A Neural Probabilistic Language Model[J]. Journal of Machine Learning Research, 2003, 3(6): 1137-1155.
[20] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[C]. In: Proceedings of Workshop at ICLR. 2013.
[21] Morin F, Bengio Y. Hierarchical Probabilistic Neural Network Language Model[C]. In: Proceedings of Workshop at AISTATS. 2005.
[22] Porteous I, Newman D, Ihler A, et al. Fast Collapsed Gibbs Sampling for Latent Dirichlet Allocation[C]. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Las Vegas, USA.
[23] GibbsLDA++: A C/C++ Implementation of Latent Dirichlet Allocation (LDA) Using Gibbs Sampling for Parameter Estimation and Inference[EB/OL]. [2016-05-15]. https://sourceforge.net/projects/jgibblda/.

Author Contributions: Zhang Qun and Wang Hongjun conceived the research and designed the methodology; Zhang Qun performed experiments, collected and analyzed data, and drafted the manuscript; Wang Hongjun and Wang Lunwen revised the final version.

Conflict of Interest: All authors declare no conflict of interest.

Supporting Data:
[1] Zhang Qun. dataset.zip. Large-scale dataset for training word vectors and LDA.
[2] Zhang Qun. word2vec.zip. Word2Vec source code.
[3] Zhang Qun. GibbsLDA++.zip. Gibbs sampling-based LDA implementation.
[4] Zhang Qun. LDA_{result}.zip. LDA training results.

Note: Figure translations are in progress. See original paper for figures.