Research on Short Text Classification Method Based on Keyword Similarity Postprint

Authors: Zhang Zhenhao, excessive shooting, Han Meiqi, Wang Jixiang

Date: 2018-11-29T00:00:00+00:00

Abstract

In traditional text classification, the text vector space matrix suffers from issues such as the "curse of dimensionality" and extreme sparsity, while extracting keywords most relevant to the categories as features for text classification helps address these two problems. Building upon these conclusions, this study proposes a short text classification framework based on keyword similarity. The framework first trains a word2vec word embedding model using a large-scale corpus; then obtains keywords for each category of text via TextRank, and performs deduplication on the keyword set to serve as the feature set. For any given feature, the word embedding model is employed to compute the similarity between each word in the short text and this feature, selecting the maximum similarity as the weight for that feature. Finally, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) are selected as the classifier training algorithms. Experiments are conducted on a Chinese news headline dataset, and compared with traditional short text classification methods, the classification performance improves by approximately 6% on average, thereby validating the effectiveness of the proposed framework.

Full Text

Preamble

Research on Short Text Classification Based on Keyword Similarity

Zhang Zhenhao[1], Guo Yi[1],[2],[3]†, Han Meiqi[1], Wang Jixiang[1]
(1. School of Information Science & Engineering, East China University of Science & Technology, Shanghai 200237, China;
2. School of Information Science & Technology, Shihezi University, Xinjiang Shihezi 832003, China;
3. Business Intelligence & Visualization Research Center, National Engineer-

ing Laboratory for Big Data Distribution & Exchange Technologies, Shanghai 200436, China)

Abstract: Traditional text classification suffers from the "curse of dimensionality" and extreme sparsity in text vector space matrices. Extracting keywords most relevant to categories as features helps address these issues. Building on this conclusion, this paper proposes a short text classification framework based on keyword similarity. The framework first trains a word2vec model on a large corpus, then obtains keywords for each category using TextRank and deduplicates them to form a feature set. For each feature, the similarity between every word in the short text and the feature is computed via the word vector model, with the maximum similarity assigned as the feature weight. Finally, K-Nearest Neighbors (KNN) and Support Vector Machine (SVM) are selected as classifier training algorithms. Experiments on a Chinese news headline dataset demonstrate that classification performance improves by approximately 6% on average compared to traditional short text classification methods, validating the effectiveness of the framework.

Keywords: word embedding; feature selection; short text classification; feature weighting

0 Introduction

With the rapid development of the Internet in recent years, people increasingly rely on obtaining information from online sources. How to quickly and accurately acquire desired information has become a key research topic. The explosive growth of text data, distributed chaotically across the web, has greatly impacted the efficiency and quality of information retrieval. This includes large volumes of short texts such as microblogs, news headlines, and product reviews, making short text classification an increasingly attractive research area.

Traditional text classification typically employs the vector space model approach [1]. However, this method exhibits characteristics of feature sparsity and indistinct semantic features when applied to short texts. Current solutions primarily follow two approaches: (a) expanding short texts into longer documents using search engines [2] before classification, and (b) leveraging large knowledge bases or corpora as background knowledge [3] to discover semantic relationships between words. While both methods can improve short text classification performance, they suffer from computational overhead and inability to handle neologisms.

To address feature sparsity and weak semantic features, reference [4] utilized LDA topic-word distribution matrices to reduce feature dimensions for short text classification. Reference [5] fused lexical category features with semantics for short text classification, achieving improvements over traditional methods, but both relied on the LDA topic model—an unsupervised learning approach that is computationally slow and dependent on optimal topic number selection. Reference [6] proposed extracting document keywords as text features, yielding

good results. However, document keywords increase with document quantity, leading to high-dimensional vector spaces and computational costs. Combining the strengths of different methods, this paper employs TextRank to extract category keywords as new short text features, transforming the feature set from a vocabulary of thousands into a compact, limited keyword collection. Word2vec then computes semantic similarity between words as feature weights, preserving semantic information in short texts for classifier training.

The main contributions are: a) A text feature selection method based on category keywords, where the number of category keywords is far smaller than total document words, effectively addressing high dimensionality. b) Using word similarity as feature weights in representation, compensating for the "bag-of-words" model' s semantic limitations. c) A novel short text classification framework demonstrating improved effectiveness over traditional methods through experiments on real datasets.

1 Related Work and Process

Let the short text collection be $D = \{d_1, d_2, d_3, \dots, d_n\}$, short text category labels $Y = \{y_1, y_2, y_3 \dots y_n\}$, and feature set $W = \{w_1, w_2, w_3, \dots, w_m\}$. The primary objective is to transform D into a vector space matrix $X = \{x_1, x_2, x_3, \dots, x_n\}^T$ using W based on keyword similarity, where each $x_i (1 \leq i \leq n)$ is an m-dimensional vector, then train a classifier to correctly categorize elements in D. The process is illustrated in [Figure 1: see original paper].

Traditional short text feature selection computes statistical measures for each feature in a text collection and applies a threshold. Features below the threshold are filtered; others are considered effective. Typical methods include document frequency, information gain, and mutual information. However, these cannot handle new texts—if a text contains no features from the effective feature set, it cannot be properly represented.

This paper proposes extracting category keywords as text features. Current mainstream keyword extraction methods are unsupervised, including statistical, graph model, and semantic approaches. Many effective algorithms derive from term frequency-inverse document frequency (TF-IDF) [7], but TF-IDF only considers word frequency, ignoring semantics and word relationships, and cannot extract keywords from single documents—making it unsuitable for category keyword extraction.

The TextRank algorithm [8] extracts document keywords based on word co-occurrence chains using a graph model, performing well and enabling single-document keyword extraction. In TextRank, a candidate keyword graph $G = (V, E, W)$ is first constructed, where node set $V = \{v_1, v_2, v_3, \dots, v_n\}$ consists of candidate keywords (typically non-stop words with noun, adjective, or verb parts of speech). $W = \{w_{ij} | 1 \leq i \leq n \wedge 1 \leq j \leq n\}$ is the weight set, and $E = \{(v_i, v_j) | v_i \in V \wedge v_j \in V \wedge w_{ij} \in W \wedge w_{ij} \neq 0\}$ is the non-empty finite

set of edges between nodes. An edge exists between two nodes only when their corresponding words co-occur within a window of length K, where K represents the window size (maximum K words co-occurring). From G, the corresponding similarity matrix $S_{n\times n}$ is obtained, as shown in equation (1):

$$S_{n\times n} = \begin{pmatrix} w_{11} & w_{12} & \cdots & w_{1n} \\ w_{21} & w_{22} & \cdots & w_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n1} & w_{n2} & \cdots & w_{nn} \end{pmatrix}$$

From G and $S_{n\times n}$, the weight of each node is iteratively computed using equation (2):

$$WS(v_i) = (1-d) + d \times \sum_{v_j \in In(v_i)} \frac{w_{ji}}{\sum_{v_k \in Out(v_j)} w_{jk}} WS(v_j)$$

where $WS(v_i)$ is the weight of node v_i; d is the damping coefficient (typically 0.85), representing the probability of jumping from any node to any other node; $In(v_i)$ denotes the set of all nodes pointing to v_i; and $Out(v_i)$ denotes the set of all nodes v_i points to. Generally, after 20-30 iterations with a threshold of 0.0001, nodes are ranked in descending order by weight. Based on this, category keywords are extracted from background knowledge corpora for each text category—for example, sports category keywords: {"game", "team", "player", ···} and finance category keywords: { "China" , "company" , "market" , ···}.

2 Feature Selection and Feature Representation

This chapter introduces TextRank-based short text feature selection and word2vec-based feature representation.

2.1 TextRank-Based Short Text Feature Selection

The fundamental idea of traditional short text feature selection is to compute statistical measures for each feature in a text collection and set a threshold. Features below the threshold are filtered; otherwise, they are considered effective. However, such methods cannot handle new texts—if all features in a text are absent from the effective feature set, the text cannot be effectively represented.

To address this, we propose extracting category keywords as text features. Current mainstream keyword extraction methods are primarily unsupervised, including statistical, graph model, and semantic approaches. While TF-IDF [7] has spawned many effective keyword extraction algorithms, it only considers word frequency, ignoring semantics and word relationships, and cannot extract keywords from single documents—making it unsuitable for category keyword extraction.

TextRank [8] extracts document keywords using a graph model based on word co-occurrence chains, achieving good results and enabling single-document keyword extraction. By combining different method characteristics, we use TextRank to extract category keywords as new short text features, transforming the feature set from a vocabulary of thousands into a compact, limited keyword collection. Word2vec then computes semantic similarity between words as feature weights, preserving semantic information for short text classification.

2.2 Word2vec-Based Feature Weighting

After determining text features, each sample's features must be assigned weights, which significantly impacts classification performance. Traditional weighting methods include binary weighting (1 if present, 0 otherwise), term frequency (tf), inverse document frequency (idf), and TF-IDF. Reference [9] computed feature weights for present words using absent words, showing improvement over TF-IDF and information entropy under equivalent conditions. Reference [10] improved classification by computing feature-weighted frequencies from training data for naive Bayes classifiers. However, most feature weighting research still considers only feature frequency at the "bag-of-words" level, ignoring semantic relationships between features and texts.

Therefore, this paper uses word similarity as feature weights, preserving semantic content during text-to-vector conversion. To accurately compute word similarity, we select the word2vec model [11]. Since its proposal, word2vec has been widely applied—reference [12] used it for opinion classification with better results than bag-of-words models, and reference [13] leveraged its semantic information for Chinese comment sentiment analysis, achieving good performance. Word2vec is essentially a neural network with hidden layers, with input and output layers as vocabularies. By learning word-context relationships, the vector from input to hidden layer represents each word' s vector after convergence. Word2vec includes two training models: CBOW (continuous bag-of-words) and Skip-Gram. CBOW is more suitable for larger corpora, so we adopt the CBOW model to represent words in short texts.

In the CBOW training model, a word' s context serves as input while the word itself is output, as shown in [Figure 2: see original paper]. The input layer consists of the m words before and after the i-th word, with the output layer being the i-th word w_i. Both input and output vectors use one-hot encoding with dimensions equal to vocabulary size. The core idea predicts the current word based on its context. After training, all word vectors are obtained. Let the word vector for word i be s_i and for word j be s_j; their similarity can be computed using cosine similarity as shown in equation (3):

$$sim(s_i, s_j) = \frac{s_i \cdot s_j}{\|s_i\|\|s_j\|}$$

3 Short Text Classification Method Based on Keyword Similarity

This chapter details the framework and specifics of the short text classification method based on keyword similarity. Symbol definitions are provided in .

TABLE:1 Definition of symbols in short text classification

Symbol	Definition
$P = \{p_1, p_2, \dots, p_k\}$	Background knowledge corpora for different categories
$D = \{d_1, d_2, \dots, d_n\}$	Short text collection
$Y = \{y_1, y_2, \dots, y_n\}$	Short text category labels
$K = \{k_1, k_2, \dots, k_k\}$	Keyword sets for all categories
k_i	Keyword set for category i
$W = \{w_1, w_2, \dots, w_m\}$	Text feature set
$d_i = \{q_1, q_2, \dots, q_m\}$	Word set for text i
$V_i = \{w_1, w_2, \dots, w_q\}$	Feature vector representation of text i
$Model$	Word vector model
$sim(a, b)$	Similarity between vectors a and b

3.1 Short Text Feature Selection

Based on TextRank, category keywords are obtained using Algorithm 1 to form the short text feature set.

Algorithm 1: Short Text Feature Selection Based on Keywords

Input: Text corpora for different categories P, number of keywords n
Output: Category keyword set Dic

1. For each p_i in P:
2. `$p_i$ = Text pre-process($p_i$)  // Preprocess background knowledge corpus`
3. `$k_i$ = TextRank($p_i$, topk=n)  // TextRank ranks keywords; select top n for category`
4. End for
5. For each k_i in K:
6. `For each word in $k_i$:`
7. `    If word appears only once:`
8. `        $Dic$.append(word)  // Deduplicate category keyword sets, keeping words appeari`
9. `    End if`
10. End for
11. End for

3.2 Short Text Feature Vector Representation

Before classification, texts must be converted to feature vectors for classifier training. Our feature vector representation method is described in Algorithm 2.

Algorithm 2: Feature Vector Representation for Short Text Collection

Input: Text feature set Dic, word vector model $Model$, short text collection D
Output: Feature vector set V for short texts

1. For each d_i in D:
2. `For each $w$ in $Dic$:  // Number of elements in $Dic$ equals feature count`
3. `    $max\_sim = -1$`
4. `    For each $q_j$ in $d_i$:`
5. `        $s = sim(Model(q_j), Model(w))$  // Convert words to vectors via word2vec for s`
6. `        If $(s > max\_sim)$: $max\_sim = s$`
7. `    End for`
8. `    $w_j = max\_sim$  // Feature weight for text i's j-th feature`
9. `End for`
10. End for

3.3 Short Text Classification

Through the above method, the short text collection' s feature vector matrix $X = \{x_1, x_2, x_3, \dots, x_n\}^T$ is obtained, where n is the dataset size and each $x_i (1 \leq i \leq n)$ is an m-dimensional vector with m as the feature set size. Combined with label set Y, classification algorithms such as KNN and SVM can be applied to validate the feature selection and weighting approach.

4 Experiments and Results

To validate the effectiveness of the proposed method, experiments were conducted using recent Sina hot news data. This section introduces the dataset, comparison algorithms, and validates the method through 10-fold cross-validation.

4.1 Dataset and Experimental Settings

The experiments used hot news from the past two years, collecting news headlines and full texts across seven categories: technology, sports, society, entertainment, military, international, and finance. Headlines were used for short text classification, while full texts served as background knowledge for keyword

extraction. For uniform distribution, each category contained 2,000 news headlines, as shown in . The word2vec model was trained on additional corpora (Baidu Baike 20G, news corpus 12G, novels 90G). For classification algorithms, SVM used the radial basis function (RBF) kernel, which outperforms linear kernels for non-linear cases while being less computationally expensive than polynomial kernels. Through experimental comparison, KNN with $N = 15$ neighbors proved most suitable.

TABLE:2 News headline dataset

Category	Technology	Sports	Society	Entertainment	Military	International	Finance
Number of news headlines	2000	2000	2000	2000	2000	2000	2000

4.2 Experimental Design

To validate our method' s effectiveness, we first examined the impact of different keyword quantities on classification performance under the same algorithms, then selected an optimal keyword count for comparison with two baseline methods:

a) **TF-IDF**: After standard preprocessing (punctuation removal, stopword removal, etc.), compute TF-IDF feature vectors for short texts, then train and validate using KNN and SVM.

b) **Sum-CBOW**: Since short texts contain few words, resulting TF-IDF vectors are high-dimensional and extremely sparse. As word vectors themselves possess good semantic features, reference [14] proposed summing all word vectors in a short text as its feature vector (with dimension equal to word vector dimension rather than category keyword count). This approach yields low-dimensional vector spaces, fast classification, and relatively good performance, referred to herein as Sum-CBOW.

To verify semantic contribution to classification improvement, a third experiment **Key-Frequency** was designed. With the same feature set, feature weights were replaced by term frequency instead of similarity, using KNN as the classifier. This validates whether semantic similarity enhances classification.

4.3 Experimental Evaluation

Four evaluation metrics were used:

a) **Precision**: For category c_i, precision p_i is calculated as:

$$p_i = \frac{\text{Number of correctly classified samples}}{\text{Number of all classified samples}}$$

b) **Recall**: For category c_i, recall r_i is calculated as:

$$r_i = \frac{\text{Number of correctly classified samples}}{\text{Actual number of samples in the category}}$$

c) **F1-score**: For category c_i, F1-score $f1_i$ is calculated as:

$$f1_i = \frac{2 \times p_i \times r_i}{p_i + r_i}$$

d) **Macro-averaged F1-score**: The arithmetic mean of F1-scores across all categories:

$$F1_{macro} = \frac{1}{k} \sum_{i=1}^{k} f1_i$$

4.4 Experimental Results and Analysis

[Figure 3: see original paper] validates the impact of category keyword quantity on short text classification performance using the $F1_{macro}$ metric with keyword counts $N = \{20, 40, 60, 80, 100, 120, 140, 160\}$. Overall, SVM outperforms KNN because features are category keywords with good inter-category separability, while KNN relies more on sample similarity than feature discriminability. Performance is suboptimal with few keywords but improves as the feature set expands, stabilizing after reaching a certain threshold where $F1_{macro}$ no longer increases. To avoid overfitting and underfitting, subsequent experiments use 100 keywords.

Comparison results are shown in . Sum-CBOW performs relatively poorly—directly summing short text word vectors as features, while preserving some semantic information, weakens category feature discriminability. Traditional TF-IDF achieves good results but suffers from slightly imbalanced precision and recall, high dimensionality, sparsity, and computational overhead. Our method selects category keywords as features and uses maximum similarity between short texts and features as weights, effectively mining core semantic associations between short texts and different categories while preserving semantics in the vector conversion process. For the test dataset, although some metrics don't reach maximum values, our method balances precision and recall while improving both average precision and average recall by approximately 6%. [Figure 4: see original paper] provides a visual comparison of F1 values across categories for the three methods, confirming that our keyword similarity-based approach effectively enhances classification performance.

TABLE:3 Comparison of classification effects across different algorithms

Category	TF-IDF	Sum-CBOW	Key-CBOW
Technology	0.843	0.801	0.891
Sports	0.912	0.874	0.945
Society	0.796	0.753	0.834
Entertainment	0.823	0.792	0.856
Military	0.879	0.834	0.912
International	0.851	0.813	0.887
Finance	0.834	0.801	0.879
Average	0.848	0.810	0.886

To validate the impact of semantic relationships, the **Key-Frequency** experiment was conducted where feature weights were term frequencies rather than similarities, using KNN classification. F1 results in show TF-IDF significantly outperforms this variant, demonstrating that with limited features and without semantic similarity, frequency-based weighting cannot handle words outside the feature set, confirming that our semantic similarity-based weighting effectively improves classification.

TABLE:4 Influence of semantic similarity on F1 value

Category	Key-Frequency	TF-IDF
Technology	0.723	0.843
Sports	0.801	0.912
Society	0.698	0.796
Entertainment	0.734	0.823
Military	0.789	0.879
International	0.756	0.851
Finance	0.742	0.834

5 Conclusion

To address high sparsity and lack of semantic features in short text classification, this paper proposes a framework based on keyword similarity. By selecting a limited set of category keywords as features and using word-feature similarity as weights, the framework resolves high dimensionality while preserving semantics and improving text discriminability. Comparative experiments using term frequency versus semantic similarity validate the framework' s effectiveness.

Future work will analyze performance variations across different categories and consider optimization methods. For feature weighting via semantic similarity, different similarity computation methods will be compared to further validate the approach.

References

[1] Salton G, Wong A, Yang C S. A vector space model for automatic indexing [M]. New York: ACM Press, 1974.
[2] Sun Aixin. Short text classification using very few words [C]// Proc of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval. New York: ACM Press, 2012: 1145-1146.
[3] Chen Mengen, Jin Xiaoming, Shen Dou. Proceedings of the twenty-second international joint conference on artificial intelligence short classification improved by learning multi-granularity topics [J]. Journal of Shanxi Institute of Economic Management, 2012: 152-157.
[4] 杨萌萌, 黄浩, 程露红, 等. 基于 LDA 主题模型的短文本分类 [J]. 计算机工程与设计, 2016, 37 (12): 3371-3377. (Yang Mengmeng, Huang Hao, Cheng Luhong, et al. Short text classification based on LDA topic model [J]. Computer Engineering and Design, 2016, 37 (12): 3371-3377.)
[5] 马慧芳, 周汝南, 吉余岗, 等. 融合词语类别特征和语义的短文本分类方法 [J]. 计算机工程与科学, 2017, 39 (2): 399-404. (Ma Huifang, Zhou Runan, Ji Yugang, et al. A short text classification method combining lexical category features and semantics [J]. Computer Engineering & Science, 2017, 39 (2): 399-404.)
[6] Onan A, Korukoğlu S, Bulut H. Ensemble of keyword extraction methods and classifiers in text classification [J]. Expert Systems with Applications, 2016, 57 (C): 232-247.
[7] Salton G, Buckley C. Term-weighting approaches in automatic text retrieval [M]. Oxford: Pergamon Press, Inc, 1988.
[8] Mihalcea R. TextRank: bringing order into texts [J]. Empirical Methods on Natural Language Processing, 2004: 404-411.
[9] Sabbah T, Selamat A, Selamat M H, et al. Modified frequency-based term weighting schemes for text classification [J]. Applied Soft Computing, 2017, 58: 193-206.
[10] Jiang Liangxiao, Li Chaoqun, Wang Shasha, et al. Deep feature weighting for naive Bayes and its application to text classification [J]. Engineering Applications of Artificial Intelligence, 2016, 52 (C): 26-39.
[11] Mikolov T, Sutskever I, Chen Kai, et al. Distributed representations of words and phrases and their compositionality [J]. 2013, 26: 3111-3119.
[12] Enríquez F, Troyano J A, López-Solaz T. An approach to the use of word embeddings in an opinion classification task [J]. Expert Systems with Applications, 2016, 66: 1-6.
[13] Zhang Dongwen, Xu Hua, Su Zengcai, et al. Chinese comments sentiment classification based on word2vec and SVM perf [J]. Expert Systems with Applications, 2015, 42 (4): 1857-1863.
[14] 江大鹏. 基于词向量的短文本分类方法研究 [D]. 杭州: 浙江大学, 2015. (Jiang Dapeng, Research on short text classification based on word distributed representation [D]. Hangzhou: Zhejiang University, 2015.)

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