Postprint: Keyword Extraction by Fusing Word2vec and TextRank

Authors: Ning Jianfei, Liu Jiangzhen

Date: 2017-10-11T00:00:00+00:00

Abstract

【Objective】 To conduct keyword extraction by fusing intra-document structural information with corpus-level word vector relationships. 【Method】 Word2vec is employed to generate vector representations for all vocabulary within the document collection, enabling computation of inter-word similarities via these vectors. The TextRank algorithm is subsequently enhanced through non-uniform allocation of candidate keyword weights based on both word similarity and adjacency relations, concurrently constructing a corresponding probability transition matrix for iterative computation in the lexical graph model and keyword extraction. 【Results】 Effective integration of Word2vec and TextRank is realized, with notably improved keyword extraction performance when the vocabulary distribution in the training document collection is reasonable. 【Limitations】 The approach necessitates computationally expensive training on the document collection to obtain word vectors and word relationship matrices. 【Conclusion】 Corpus-level word relationships facilitate refinement of intra-document word relationships, thereby enhancing the accuracy of keyword extraction for individual documents.

Full Text

Keyword Extraction by Integrating Word2vec and TextRank

Ning Jianfei, Liu Jiangzhen
(Department of Electronic Information, Luoding Polytechnic, Luoding 527200, China)

Abstract

[**Objective**] This study extracts keywords by combining the internal structure information of individual documents with the word vector relationships of the

entire document corpus. [**Methods**] We used Word2vec to represent all words in the document corpus as vectors and calculated the similarity between words through these vectors. We then improved the TextRank algorithm by non-uniformly assigning weights to candidate keywords based on word similarity and adjacency relations, constructing a corresponding probability transition matrix for iterative calculation of the lexical graph model and keyword extraction. [**Results**] The effective integration of Word2vec and TextRank was achieved, and keyword extraction performance was notably improved when the vocabulary distribution in the training corpus was reasonable. [**Limitations**] The method requires costly corpus training to obtain word vectors and word relationship matrices. [**Conclusions**] Word relationships in the document corpus help correct word relationships within a single document, thereby improving the accuracy of keyword extraction for individual documents.

Keywords: Keyword extraction; Word2vec; TextRank; Graph model; Word vector
Classification Number: TP391; G250

Keyword extraction aims to concisely distill the theme of a text and rapidly capture its core content. It plays an important role in automatic summarization of news and academic papers, social tag annotation, and text topic extraction.

Common keyword extraction steps include: text segmentation, removal of useless stop words, determining whether a word is a keyword, and selecting N words as the document' s keywords. Determining whether a segmented word is a keyword can be accomplished through classification model training on keyword-labeled corpora, or through graphical models that incorporate relationships between words within the text. The TextRank algorithm is a typical representative of the latter approach.

The classical TextRank algorithm does not rely on external training corpora but focuses on studying the internal structural relationships of words in a text, building a graph model for keyword extraction. Xia Tian' s research indicates that the inherent importance differences among words affect the propagation of influence between adjacent nodes. Gu Yijun et al. combined TextRank with LDA, non-uniformly transferring the importance of candidate word nodes according to the topic distribution of the document corpus.

To fully investigate relationships between words and leverage external information provided by documents and document collections, this paper integrates Word2vec with the TextRank algorithm. Word2vec represents external document corpora as word vectors to obtain similarity between words, which is used to improve TextRank by reasonably allocating weights to candidate word nodes based on the similarity of adjacent words. Through iterative calculation of each word' s weight, the final ranking by weight yields keyword extraction results.

Keyword extraction can be categorized into supervised and unsupervised approaches based on whether the corpus is labeled. Supervised keyword extraction typically treats the task as a binary classification problem, where each word in a

text is classified as either a keyword or not. This method requires advance manual keyword labeling of the document corpus for classification model training, demanding substantial manual intervention.

In the unsupervised keyword extraction domain, numerous relevant studies have been conducted in China. Geng Huantong et al. automatically extracted document topic words using topic information and inter-topic connection relationships formed by word co-occurrence graphs. Liu Fei et al. proposed using association rule mining algorithms for topic word extraction. Jiang Changjin et al. considered semantic information of words and proposed a topic word extraction algorithm based on compound words and synonyms.

Currently, there are three mainstream unsupervised keyword extraction methods: TF-IDF model based on word frequency statistics, keyword extraction based on topic models, and keyword extraction based on lexical graph models. Numerous other optimization algorithms have been built upon these three mainstream approaches.

TF-IDF is a simple yet classic keyword extraction method that increases the weight of important words through term frequency while reducing the weight of common words through inverse document frequency. However, this method relies on word frequency and performs poorly on short texts, ignoring relationships between words within the text.

Keyword extraction based on the LDA latent topic model has gradually gained attention. The LDA topic model is obtained through corpus training to acquire a "document-topic" probability matrix and a "topic-word" probability matrix, from which the "document-word" probability sum matrix is derived for keyword extraction. The effectiveness of keyword extraction depends on the topic distribution of the training corpus.

Lexical graph model-based keyword extraction does not require additional document corpora for training, relying solely on the structural information of the text's own vocabulary for keyword extraction. This approach is simple, effective, and widely applied, with TextRank being a typical representative.

With the rise of deep learning, Liu Jun et al. used the deep learning tool Word2vec for keyword extraction, representing all words in the training document corpus as K-dimensional vectors and calculating similarity between words based on these vectors to achieve word clustering and obtain document keywords.

Xia Tian' s research demonstrates that the inherent importance differences among words affect the propagation of influence between adjacent nodes. LDA-based keyword extraction requires large amounts of training data at high cost and cannot meet the needs of single-document keyword extraction. Gu Yijun et al. improved TextRank' s probability transition matrix by combining LDA' s latent topic model analysis to calculate the overall influence of words while maintaining PageRank' s uniform jump assumption, but this method does not

consider the overall distribution relationships between words. Li Yuepeng et al. used deep learning combined with word clustering for keyword extraction, showing good results for longer articles but failing to meet accuracy requirements for shorter texts.

Building upon the research 思路 of Gu Yijun et al., which leverages both single-document internal structure information and overall document information for topic word extraction, this paper proposes that word node relationships are influenced by the distribution of relationships among vocabulary in the document corpus. By combining the word similarity matrix obtained through Word2vec training, we improve TextRank' s initial weight calculation and probability transition matrix for word nodes, simultaneously considering both single-document internal vocabulary structure and corpus-level vocabulary structure information to enhance keyword extraction effectiveness.

3 Research Framework and Methods

Drawing on the research 思路 of Gu Yijun et al., this study integrates single-document internal structure information with overall corpus information for topic word extraction, investigating how the distribution of relationships among vocabulary nodes in document collections affects the vocabulary structure of individual documents.

The core idea of the TextRank algorithm originates from the famous PageRank algorithm. TextRank splits text into minimal components—words—as network nodes to form a lexical network graph model. When iteratively calculating word weights, TextRank theoretically requires edge weight calculation like PageRank, but for simplified computation, it typically assumes uniform initial weights and equal distribution of weights to adjacent words.

This paper uses the Word2vec algorithm to compute document corpus word vectors and obtain a similarity matrix between words, which is used to improve TextRank' s initial weight calculation and iterative probability transition matrix, ultimately obtaining weights for all effective words within a document for keyword extraction.

Word2vec is an open-source tool released by Google in 2013 that represents words as spatial vectors, primarily using the Continuous Bag-Of-Words (CBOW) and Skip-gram models. As a deep learning model based on artificial neural networks, it combines initial low-level features into more abstract high-level features through multi-layer perceptrons, using these high-level features in conventional machine learning methods to achieve better results. Through training, Word2vec simplifies text content processing to vector operations in K-dimensional vector space, where similarity in vector space can represent semantic similarity in text.

The CBOW model aims to predict the probability of the current word appearing through its context. As shown in [Figure 1: see original paper], the CBOW

network structure includes three layers: Input Layer, Projection Layer, and Output Layer. The training sample is (Context(w), w), where Context(w) is constructed from c words before and after w.

As shown in [Figure 2: see original paper], Skip-gram also consists of a three-layer network model: Input Layer, Projection Layer, and Output Layer. The Input Layer receives the word vector W(t) R, and the Projection Layer aims to maximize the value of formula (2):

$$\log p(W|W)$$

where c is the vocabulary window size (the n value in n-Skip-gram), and T is the size of the training document corpus.

In the Skip-gram model, the conditional probability of words is calculated as shown in formula (3):

$$p(w|w) = \exp(v)$$

where wv and wv' are the input and output vectors of word w, respectively. Like the CBOW model, the Output Layer of Skip-gram is also a Huffman tree.

The Skip-gram model was proposed to address training corpus selection issues. When selecting the Word2vec training document corpus, we require high coverage and sufficient accuracy. The limitation of fixed window size in N-gram models is that relationships between words outside the window cannot be correctly reflected in the model. While increasing the window size N can mitigate this effect, it also increases training complexity. The Skip-gram model effectively solves this problem.

According to literature [2-3,10], the keyword extraction problem is transformed into a document word importance ranking problem, where words are ranked by weight to obtain the TopN words as document keywords. Based on literature [1], we construct a keyword graph with words as network nodes and obtain each word' s weight through iterative calculation. The word similarity matrix trained by Word2vec is integrated into the iterative calculation to optimize weight computation results.

First, we need to construct a keyword graph. Before graph construction, we preprocess the training document corpus and test documents through four steps:

(1) Use the ICTCLAS segmentation tool from the Institute of Computing Technology, Chinese Academy of Sciences to segment the training corpus of N documents and single test documents, and filter stop words using a stop word list to obtain vocabulary sets S1 and S2, where S1 consists of N sub-vocabulary sets, each corresponding to one training document.

(2) Perform part-of-speech tagging on S1 and S2, retaining important words such as nouns, verbs, and adjectives to obtain vocabulary sets S1' and S2' , with S1' consisting of N sub-vocabulary sets.

(3) Deduplicate vocabulary sets S1' and S2' to obtain dictionary D' , i.e., candidate keywords.

(4) Train S1' using Word2vec to obtain word vectors and subsequently the word similarity matrix for dictionary D.

Through CBOW and Skip-gram model training on the sample document corpus, each word in dictionary D is represented as a K-dimensional word vector. Cosine similarity is then calculated to obtain similarity between each word and all other words, as shown in formula (4):

$$Sim(e, f) = \frac{||e|| \cdot ||f||}{||e|| \cdot ||f||}$$

where ei is the ith word in the source document sentence, fj is the jth word in the target document sentence, Sim(ei, fj) represents the similarity between the ith and jth words, and ei, fj are word vector representations.

Assuming the total dictionary size is n, Word2vec training on the document corpus yields an n×n word similarity matrix, from which the similarity between any two words in the dictionary can be obtained, as shown in formula (5):

$$M(Sim(w, w)) = \begin{pmatrix} \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \end{pmatrix}$$

where M(Sim(wi, wj)) represents the dictionary' s similarity matrix, and wij denotes the similarity between words i and j. Note that in the matrix, values with identical subscripts represent a word' s similarity with itself (e.g., wii for word i), which is typically 1 and has no reference significance, thus can be ignored.

After all preprocessing, we construct the test document' s candidate keyword graph. The core idea of TextRank is that a word node' s importance depends on how many adjacent nodes point to it and the weights of those adjacent nodes. The weight calculation for word nodes is shown in formula (6):

$$R(w_i) = e(w_j, w_i) \cdot O(w_i) \cdot R(w_j) + (1 - \gamma) \cdot |V|$$

where R(wi) is the weight of word wi, O(wj) is the out-degree of word wi, e(wj, wi) is the edge weight, V is the set of word nodes, and [0,1] is the smoothing factor (Damping Factor), typically valued at 0.85.

In traditional TextRank, each word node' s weight defaults to 1, and weights are updated through iterative calculation based on adjacency relationships. When

calculating weight contributions, nodes distribute weights equally to adjacent nodes. For example, [Figure 3: see original paper] shows the initial state of a candidate keyword graph composed of six word nodes {V, V1, V2, V3, V4, V5}, where each node has an initial default weight of 1. Node V distributes its weight equally to the other five nodes with edge weights set to 0.2, while the other five nodes point to V with edge weights of 1. Subsequent iterations similarly distribute weights equally to adjacent nodes.

Based on this candidate keyword graph, we discuss how to optimize the initial weights of word nodes and improve the influence propagation method to enhance final word weight ranking for keyword extraction. For graph initialization, a more reasonable approach than defaulting to 1 is to use mutual influence between nodes as the initial state, quantified through word similarity. The initial weight calculation for word nodes is shown in formula (7):

$$S(w_j) = e(w_j, w_i) \cdot \gamma \cdot B \cdot M(T(w, w)) \cdot B$$

where S(wj) is the initial weight of word node wj, and e(wj, wi) represents the similarity between words.

In the improved transition matrix, word similarity is introduced for iterative calculation. Word node weight allocation is influenced by two factors: one is the node' s own importance, representing the influence of internal document structure, typically adjusted through adjacent nodes. The initial value can be calculated via formula (7) and subsequently obtained through iteration, denoted as TP(wi). The other factor is the rigid relationship influence score between words, obtained through Word2vec training as shown in formula (5), representing external document influence on word relationships, denoted as M(Sim(wi, wj)).

The redefined node importance iteration process is shown in formula (8):

$$TP(w_i) = \gamma \cdot \alpha + (1 - \gamma) \cdot |V| \cdot \frac{M(Sim(w, w))}{O(M(Sim(w, w)))} \cdot \frac{TP(w_j)}{O(w_j)} \cdot R(w_j)$$

where [0,1] is the smoothing factor, and are weight factors for the two influence types (with + = 1, both set to 0.5 in experiments, meaning internal node influence and external corpus word relationship influence each account for 50%), M(Sim(wi, wj)) is the similarity between words from external documents (reference formula (5)), O(wj) is the out-degree of wj, R(wj) is the weight of word wj, and V is the set of word nodes.

Before iterative calculation, we construct the probability transition matrix between words as shown in formula (9):

$$M(T(w, w)) = \begin{pmatrix} \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \end{pmatrix}$$

where element wij represents the probability of transferring influence from node wj to word wi, specifically manifested as the distribution of edge weights between adjacent words, calculated via formula (10):

$$M(Sim(w,w)) \cdot O(M(Sim(w,w))) \cdot O(w_j)$$

where e is a vector of dimension k with all components equal to 1. Iteration stops when the difference between two consecutive calculations becomes sufficiently small, indicating convergence. After convergence, all word node weights are sorted in descending order, and the TopN words are selected as the document' s keywords.

The entire integration process of Word2vec and TextRank consists of two steps: (1) training the document corpus with Word2vec to obtain a word relationship matrix as shown in formula (5); (2) incorporating external word relationship influence into formula (10) to iteratively calculate word node weights for ranking and keyword extraction.

Experiments

We selected the publicly available Sogou Lab corpus as both training and test document sets, covering multiple domains including military, education, economy, and entertainment. For each domain, 10 documents were selected as the test set (90 test documents total), with the remaining 4,500 documents serving as the training corpus. Word2vec training on the training corpus took 38 minutes on a 4GB memory computer, producing a word similarity matrix model file of approximately 120MB.

Using the word similarity matrix model to optimize TextRank calculations, we automatically extracted 3, 5, 7, and 10 keywords from each of the 90 test documents. Multiple human annotators performed cross-validation on the test set to reduce subjectivity bias, with 3, 5, 7, and 10 keywords extracted as comparative validation results.

Additionally, we implemented TF-IDF-based keyword extraction, traditional TextRank keyword extraction, and Word2vec word clustering-based keyword extraction using the same training and test sets, analyzing and comparing the results of these four algorithms.

Current evaluation metrics for keyword extraction algorithms include precision (P), recall (R), and F-measure, calculated as follows:

$$Precision = \frac{\text{Number of keywords matching manual annotation}}{\text{Total number of extracted keywords}}$$

$$Recall = \frac{\text{Number of keywords matching manual annotation}}{\text{Total number of manually annotated keywords}}$$

After introducing the transition probability matrix, each iteration result can be calculated through M(T(wiwj)). Let Bi represent one iteration result; the iterative calculation process transforms from formula (6) to formula (11).

To ensure evaluation validity, multiple experimenters cross-annotated keywords for the test document set, with each test document annotated with 3, 5, 7, and 10 keywords. We then calculated precision, recall, and F-measure for the four algorithms.

As shown in –, TF-IDF performance gradually deteriorates as the number of keywords increases, with overall poor performance. TextRank performance remains relatively stable with little fluctuation. Word2vec word clustering performance gradually improves as the number of keywords increases. The integrated Word2vec-TextRank method also improves with more keywords and achieves the best overall performance.

**** Precision comparison of four algorithms
**** Recall comparison of four algorithms
**** F-measure comparison of four algorithms

Through in-depth analysis of the extraction process, we conclude:

(1) Traditional TF-IDF-based keyword extraction performs moderately.

(2) Traditional TextRank-based keyword extraction demonstrates stable performance.

(3) Word vector clustering-based keyword extraction is suitable for longer documents.

(4) The integrated Word2vec-TextRank method inherits TextRank' s stability while achieving further improvement, also inheriting Word2vec clustering' s characteristic of improving performance with more keywords.

The integrated approach uses Word2vec for word vector training to compute word similarity matrices. We propose the following improvements: (1) use larger training corpora for more accurate similarity matrices; (2) employ more precise stopword dictionaries to further eliminate invalid word interference; (3) improve Word2vec by adding neural network layers to enhance semantic abstraction and similarity accuracy; (4) utilize distributed architectures like Spark in-memory computing to improve algorithm speed.

In summary, the integrated Word2vec-TextRank method primarily benefits from combining internal document structure with external corpus word relationships, inheriting the advantages of both algorithms to achieve relatively good keyword extraction performance, though results and analysis still have considerable room for improvement.

Conclusion

A document' s internal structural information and word relationships reflected in external corpora are crucial bases for keyword extraction. This paper uses Word2vec to calculate relationships between dictionary words, improving TextRank' s weight distribution iteration formula by incorporating word similarity influence into edge weight allocation and transfer construction. Through iterative calculation until word node weights converge, we rank word node weights for keyword extraction, comparing different algorithms on the same corpus.

Experimental results show that as the number of keywords increases, our method slightly outperforms traditional TextRank and Word2vec clustering methods, inheriting their respective advantages. Additionally, training corpus scale and the weight ratio between internal document structure and external corpus word relationships significantly affect extraction results. Therefore, further increasing training corpus scale and investigating the impact of internal-external influence weighting on extraction effectiveness will be the focus of subsequent work.

References

[1] Mihalcea R, Tarau P. TextRank: Bringing Order into Texts [C]. In: Proceedings of Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain. 2004: 404-411.

[2] Xia Tian. Study on Keyword Extraction Using Word Position Weighted Text Rank [J]. New Technology of Library and Information Service, 2013(9): 30-34.

[3] Gu Yijun, Xia Tian. Study on Keyword Extraction with LDA and TextRank Combination [J]. New Technology of Library and Information Service, 2014(7-8): 41-47.

[4] Goldberg Y, Levy O. Word2vec Explained: Deriving Mikolov et al.' s Negative-sampling Word-embedding Method [OL]. ArXiv, 2014. arXiv: 1402.3722v1.

[5] Frank E, Paynter G W, Witten I H, et al. Domain-Specific Keyphrase Extraction [C]. In: Proceedings of the 16th International Joint Conference on Artificial Intelligence, Stockholm, Sweden. San Francisco: Morgan Kaufmann Publishers Inc., 1999: 668-673.

[6] Turney P D. Learning Algorithms for Keyphrase Extraction [J]. Information Retrieval, 2000, 2(4): 303-336.

[7] Geng Huantong, Cai Qingsheng, Yu Kun, et al. A Method Based on the Co-occurrence of Automatic Text Keyphrase Extraction Method [J]. Journal of Nanjing University: Natural Science Edition, 2006, 42(2): 156-162.

[8] Liu Fei, Huang Xuanjing, Wu Lide. The Method of Using Association Rule Mining Text Topic Words [J]. Computer Engineering, 2010, 34(7): 81-83.

[9] Jiang Changjin, Peng Hong, Chen Jianchao, et al. Keyword Extraction Algorithm Based on Combination of Words and Synonyms [J]. Computer Application Research, 2010, 27(8): 2853-2856.

[10] Xu Wenhai, Wen Youkui. Chinese Keywords Extraction Based on TFIDF Method [J]. Information Studies: Theory & Application, 2008, 31(2): 298-302.

[11] Blei D M, Ng A Y, Jordan M I. Latent Dirichlet Allocation [J]. Journal of Machine Learning Research, 2003, 3: 993-1022.

[12] Shi Jing, Li Wanlong. Topic Words Extraction Method Based on LDA Model [J]. Computer Engineering, 2010, 36(19): 81-83.

[13] Liu Jun, Zou Dongsheng, Xing Xinlai, et al. Keyphrase Extraction Based on Topic Feature [J]. Application Research of Computers, 2012, 29(11): 4224-4227.

[14] Li Yuepeng, Jin Cui, Ji Junchuan. A Keyword Extraction Algorithm Based on Word2vec [J]. E-science Technology & Application, 2015(4): 54-59.

[15] Zhou Lian. Exploration of the Working Principle and Application of Word2vec [J]. Sci-Tech Information Development & Economy, 2015(2): 145-148.

[16] Page L, Brin S, Motwani R, et al. The PageRank Citation Ranking: Bringing Order to the Web [R]. Stanford InfoLab, 1999.

[17] Tomas M, Kai C, Greg C, et al. Efficient Estimation of Word Representations in Vector Space [OL]. ArXiv, 2013. arXiv: 1301.3781v3.

Author Contributions Statement

Ning Jianfei: Conceived the research idea, designed the study, performed experiments and data analysis, drafted the manuscript.
Liu Jiangzhen: Revised the manuscript.

Conflict of Interest Statement

All authors declare no conflict of interest.

Supporting Data

Supporting data is available in the online version of the journal at http://www.infotech.ac.cn.
[1] Ning Jianfei. train_set.zip. Training corpus—4,500 documents covering multiple domains provided by Sogou Lab.
[2] Ning Jianfei. validation_set.zip. Test corpus—90 documents covering multiple domains provided by Sogou Lab.

Received: 2016-03-01
Revised: 2016-04-19