Word Vector Clustering Weighted TextRank for Keyword Extraction (Postprint)

Authors: summer

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

Abstract

【Purpose】 Incorporate world knowledge embedded in Wikipedia into the TextRank model through word vectors to improve single-document keyword extraction effectiveness. 【Method】 Utilize the Word2Vec model based on Chinese Wikipedia data to generate a word vector model, cluster the word vectors of TextRank graph nodes to adjust intra-cluster node voting importance, incorporate node coverage and position factors, calculate random jump probabilities between nodes, generate the transition matrix, and finally obtain node importance scores through iterative computation to select the top TopN words as keywords. 【Results】 When TopN$ $7, the word vector clustering weighting method outperforms comparison methods; when TopN=3, the F-score reaches its maximum value, showing a 3.374% incremental improvement over the previous best result; when TopN > 7, the results are similar to the position weighting method. 【Limitations】 Clustering analysis increases computational overhead. 【Conclusion】 Word vector clustering weighting can improve keyword extraction effectiveness.

Full Text

Extracting Keywords with Word Vector Clustering Weighted TextRank

Xia Tian
(Key Laboratory of Data Engineering and Knowledge Engineering of Ministry of Education, Renmin University of China, Beijing 100872, China)
(School of Information Resource Management, Renmin University of China, Beijing 100872, China)

Abstract

[**Objective**] This study aims to improve single-document keyword extraction by incorporating world knowledge from Wikipedia into the TextRank model through word vector representations.

[**Methods**] We first trained a word embedding model using the Word2Vec algorithm on Chinese Wikipedia data. Then, we clustered the word vectors of TextRank word graph nodes to adjust the voting importance within each cluster. Combining coverage and position factors of nodes, we calculated the random jump probabilities between nodes to generate a transition matrix. Finally, we obtained node importance scores through iterative computation and selected the top N words as keywords.

[**Results**] When TopN $\leq$ 7, the word vector clustering weighted method outperformed all comparison methods. At TopN = 3, the F-measure reached its maximum value, representing a 3.374% improvement over the previous best result. When TopN $>$ 7, the results were similar to the position-weighted method.

[**Limitations**] The clustering analysis increases computational overhead.

[**Conclusions**] Word vector clustering weighting can effectively improve keyword extraction performance.

Keywords: Keyword Extraction; Word Embedding; TextRank; Word2Vec
Classification Number: G353

1. Introduction

Keyword extraction automatically identifies representative words or phrases from given text to reflect its main semantic information, with wide applications in library and information science. For instance, constructing term frequency matrices from extracted keywords enables co-word analysis at the keyword level, revealing thematic evolution in literature and supporting content mining of large-scale library data. Implementation strategies include methods using text' s intrinsic content and structural features, and those based on training from large corpora. The former has attracted considerable attention due to its simplicity and satisfactory performance without requiring prior training, with TextRank being a typical representative.

Traditional TextRank algorithms utilize only document-internal information. Theoretically, incorporating external knowledge could improve extraction effectiveness. Since 2013, word vector representations have projected word semantics into low-dimensional continuous spaces while preserving semantic characteristics from corpora. This study leverages Wikipedia—the largest online open knowledge base—to train word vectors via Word2Vec, performs word vector clustering, and applies non-uniform weighting to TextRank word graph nodes based on cluster distribution. This approach integrates external world knowledge into TextRank computation, achieving improved keyword extraction results.

TextRank introduces the PageRank algorithm from link analysis into text processing, representing textual units and their co-occurrence relationships as graph structures and computing importance through iterative graph calculations. When using words as basic units, it enables keyword extraction; when using sentences, it enables text summarization. Its superior performance over traditional TF-IDF and simple implementation have led to widespread adoption.

The original TextRank constructs word graphs without edge weights. To improve extraction, previous research weighted words by position, adjusting edge transfer weights from three perspectives: coverage influence, position influence, and frequency influence. Other studies integrated TextRank with LDA topic models, combining document structure and overall topic information, finding improvements when datasets exhibited clear topic distributions. Tag-TextRank utilized social tags to enhance web keyword extraction. Some researchers incorporated inverse document frequency alongside position weighting for keyword extraction in paper review recommendation systems.

With the rise of Word2Vec, researchers began applying it to keyword extraction. One approach clustered words by vector similarity, selecting words nearest to cluster centroids as keywords. Another integrated Word2Vec similarity matrices into TextRank computation to improve extraction.

In summary, incorporating external document information into TextRank computation is key to improving keyword extraction. Existing methods like topic weighting and inverse document frequency weighting require preprocessing of the target document' s dataset, yielding varying results across different datasets. Word2Vec training data is independent of target documents, theoretically providing more stable extraction results. Unlike direct similarity-based probability adjustment, this study first clusters word vectors within single documents, then weights term importance based on distance to cluster centroids, constructs a new probability transition matrix, and achieves optimal extraction performance.

2. Methodology

Our approach transforms keyword extraction into a term importance ranking problem. We first construct a candidate keyword graph to represent term relationships, then perform cluster analysis on word vectors to determine clustering importance based on spatial positions within clusters, enabling cluster-weighted TextRank. Finally, we construct a complete probability transition matrix between terms and obtain node importance through iterative computation for keyword ranking and extraction.

2.1 Word Graph Construction

Following TextRank principles, a document can be represented as a word graph based on term adjacency relationships, with term importance computed from

structural graph features. To construct the candidate keyword graph, we segment text by sentences, perform word segmentation and POS tagging, and retain nouns, verbs, and adjectives with more than one character as node set V. All term adjacency relationships form edge set E, creating candidate keyword graph G = (V, E). When constructing edges, if word a is adjacent to word b, we add two directed edges a→b and b→a, making G a directed graph [Figure 1: see original paper].

Given graph G(V, E), let t(u) denote the TextRank value of node u, computed using formula (1):

$$t(u) = (1-d) + d \times \sum_{v \in adj(u)} p(v \to u) \times t(v)$$

where d [0,1] is the damping factor representing the probability of random jumps to other nodes, ensuring convergence (typically 0.85); adj(u) denotes u' s neighbor set; and p(v→u) represents the random jump probability from v to u.

Traditional TextRank uses uniform jump strategies between adjacent nodes, with transition probabilities p(v→u) calculated by formula (2):

$$p(v \to u) = \begin{cases} \frac{1}{\deg(v)} & \text{if } u \in adj(v) \\ 0 & \text{otherwise} \end{cases}$$

where deg(u) is node u' s degree. For example, in Figure 1, node v jumps to any neighbor with probability p(v→i) = 1/3.

To improve TextRank, previous research proposed non-uniform jump strategies based on node importance. However, these calculations used only document-internal information—position and frequency of target words. To leverage external knowledge for optimizing jump probability assignment, we propose a word vector clustering weighting algorithm.

2.2 Word Vector Clustering Weighting

In 2013, Mikolov et al. released Word2Vec, a tool that uses shallow neural networks to learn word occurrences in corpora, embedding words into a moderate-dimensional space R^n (typically 100-500 dimensions). The resulting word vectors exhibit both lower dimensionality and preserved semantic/syntactic relationships: semantically similar words have closer distances, and linear operations on vectors align with human understanding. Word2Vec-trained vectors encode semantic information from large-scale datasets, enabling us to weight TextRank transition probabilities based on word vector relationships.

This study assumes that word vectors reflect global information: a document can be clustered into several groups by vector similarity, where words farther from cluster centroids represent more distinctive aspects of that cluster compared to

centroid-proximate words. As TextRank nodes, such words have higher voting importance and greater jump probabilities to adjacent nodes.

Given document d with candidate keyword set {w_i} and trained Word2Vec model, let vec(w_i) denote word w_i' s vector. Let C = {C_1, C_2, ···, C_k} represent K-means clustering results of the document' s word vector set. We propose formula (3) to compute voting importance VoteWeight(u) for any word u in its cluster C_u:

$$\text{VoteWeight}(u) = \frac{|C_u| \times \text{dist}(\vec{u}, \vec{c}_u)}{\sum_{v \in C_u} \text{dist}(\vec{v}, \vec{c}_u)}$$

where vec(c_u) is the centroid vector of cluster C_u; dist(vec(v), vec(c_u)) denotes Euclidean distance between vectors; and |C_u| is the number of words in cluster C_u. Formula (3) allocates each cluster' s total voting score (equal to its node count) proportionally based on Euclidean distance from the centroid—farther nodes receive higher voting importance.

After clustering and computing each word' s voting importance, we propose formula (4) to calculate cluster-influenced transition probabilities between nodes:

$$p_{\text{cluster}}(v \to u) = \frac{\text{VoteWeight}(u)}{\sum_{w \in adj(v)} \text{VoteWeight}(w)}$$

2.3 Transition Matrix Computation and Keyword Extraction

According to link analysis theory, node importance can be computed iteratively given a transition probability matrix. Let matrix M represent transition probabilities between word graph nodes, as shown in formula (5):

$$M = \begin{bmatrix} p(1 \to 1) & p(2 \to 1) & \cdots & p(n \to 1) \\ p(1 \to 2) & p(2 \to 2) & \cdots & p(n \to 2) \\ \vdots & \vdots & \ddots & \vdots \\ p(1 \to n) & p(2 \to n) & \cdots & p(n \to n) \end{bmatrix}$$

Each column j represents jump probability distributions from node j to others, summing to 1. Element p_{uv} denotes transition probability from node u to v, i.e., p_{uv} = p(u→v).

To improve TextRank, we combine position weighting with word vector clustering weighting to assign reasonable values to random jump probability p(u→v). In previous work, p_{cov}(v→u) represents coverage influence computed by formula (2), reflecting traditional TextRank contributions. Position influence p_{pos}(v→u) is calculated by formula (6):

$$p_{\text{pos}}(v \to u) = \frac{I(u)}{\sum_{w \in adj(v)} I(w)}$$

where I(v) denotes node v' s position importance: I(v) = 30 when v appears in titles, otherwise I(v) = 1.

Based on coverage, position, and clustering influences, we propose formula (7) to compute jump probability p(u→v):

$$p(u \to v) = \alpha \times p_{\text{cov}}(u \to v) + \beta \times p_{\text{pos}}(u \to v) + \gamma \times p_{\text{cluster}}(u \to v)$$

with $\alpha + \beta + \gamma = 1$. This generates the final transition matrix M. For document graph G(V, E), initial node scores follow formula (8):

$$B_0 = \left[\frac{1}{n}, \frac{1}{n}, \cdots, \frac{1}{n}\right]^T$$

Iterative computation proceeds via formula (9):

$$B_{i+1} = d \times M \times B_i + (1 - d) \times e$$

where e is an n-dimensional vector of ones; B_i represents node scores after i iterations. When the difference between consecutive iterations becomes sufficiently small, we stop and rank nodes by final scores, selecting the top N as keywords.

3. Experiments

3.1 Experimental Data

We used the Chinese Wikipedia dump "zhwiki-20150602-pages-articles-multistream.xml.bz" released in June 2015, containing 2,648,029 pages (1,480,963 article pages, 55.93% of total). After filtering redirects and short articles, we retained 516,695 pages. Using Ansj for Chinese word segmentation, we created a training corpus for Word2Vec. We employed Gensim' s Word2Vec module with default parameters (CBOW model, 100 dimensions, window size 5) to train the word vector model.

For the test dataset, while previous work provided a public dataset of 1,000 news articles, its keyword annotation quality was suboptimal. Therefore, we collected 1,524 articles from the Southern Weekend website, extracting titles and content with explicitly marked tags as ground-truth keywords. The new dataset averages 2,629.101 characters and 3.565 keywords per document.

3.2 Experimental Results and Analysis

We evaluated performance using precision (P), recall (R), and macro-averaged F-measure. Let AK denote ground-truth keywords and BK denote extracted keywords, with P, R, and F computed as in formula (10):

$$P = \frac{|AK \cap BK|}{|BK|}, \quad R = \frac{|AK \cap BK|}{|AK|}, \quad F = \frac{2PR}{P+R}$$

Following previous work, we extracted 3, 5, 7, and 10 keywords for comparison. Methods compared were:

- M1: Original TextRank
- M2: Word2Vec clustering keyword extraction
- M3: Word2Vec-TextRank fusion
- M4: Position-weighted TextRank
- M5: Our word vector clustering weighted TextRank

All clustering used K-means with 20 iterations; other parameters followed optimal values from respective papers. All data and code are publicly available for reproducibility.

Results are shown in Table 1 .

Table 1 Comparison of keyword extraction algorithms (TopN = 3,5,7,10)

[Table content would appear here with performance metrics for each method at different TopN values]

Table 1 shows that for single-document extraction, TextRank methods based on article structure and voting mechanisms significantly outperform word vector clustering (M2). While M2 can cluster semantically related words, selecting centroid-nearest words as keywords proves ineffective. Non-uniform weighting of TextRank transition probabilities improves performance, as confirmed by previous research. However, M3' s direct use of cosine similarity between word vectors for importance transfer performs poorly on our dataset. Unlike M3, our approach first determines semantic clusters via vector clustering, then assigns voting importance based on distance to centroids within clusters. When retaining $ $7 keywords, M5 outperforms all methods, demonstrating that word vector clustering weighting enhances important keyword ranking.

To observe method differences comprehensively, Figure 2 [Figure 2: see original paper] shows accuracy, recall, and F-measure curves for TopN [1,10].

Figure 2 Accuracy, recall, and F-measure for methods M1-M5 when TopN [1,10]

[Figure would appear here]

Overall, both position-weighted and word vector clustering weighted TextRank significantly outperform the other three methods. When TopN $\leq$ 5, clustering

weighting shows clear improvement over position weighting; as TopN increases, their performance differences diminish. At TopN = 3, both M4 and M5 achieve maximum F-measure, with M5 improving 3.374% over M4.

To examine poor extraction cases, we selected documents with completely mismatched results and compared outputs in Table 2 , where each method' s output length matches the original document' s tag count.

Table 2 Example extraction results completely missing ground-truth keywords

[Table would appear here showing extracted keywords from each method for sample documents]

For these complete mismatch cases, M1, M4, and M5 (all graph-based iterative methods) produce highly overlapping results that still represent main content to some extent. M2' s partial results show some relevance, while M3 performs relatively poorly.

Conclusions:

1. Direct application of word vector clustering to select representative words per cluster as keywords performs poorly for single documents.
2. TextRank provides stable single-document keyword extraction, with position and word vector clustering weighting further improving accuracy.

This study incorporates Wikipedia' s world knowledge into TextRank via word vector clustering weighting. Unlike IDF or LDA-based methods that depend on the target dataset, word vector training is dataset-independent, yielding more objective and stable results. Experiments show that clustering weighting improvements are more significant when fewer keywords are retained; beyond TopN = 7, clustering weighting shows no significant difference from position weighting alone.

Future work includes exploring more reasonable weighting methods for clustering results and comprehensive evaluation from ranking perspectives.

References

[1] Mihalcea R, Tarau P. Textrank: Bringing Order into Texts [C]//Proceedings of Empirical Methods in Natural Language Processing. 2004.

[2] Xia Tian. Study on Keyword Extraction Using Word Position Weighted TextRank [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] Li Peng, Wang Bin, Shi Zhiwei, et al. Tag-TextRank: A Webpage Keyword

Extraction Method Based on Tags [J]. Journal of Computer Research and Development, 2012, 49(11): 2344-2351.

[5] Xie Wei, Shen Yi, Ma Yongzheng. Recommendation System for Paper Reviewing Based on Graph Computing [J]. Application Research of Computers, 2016, 33(3): 798-801.

[6] Li Yuepeng, Jin Cui, Ji Junchuan. A Keyword Extraction Algorithm Based on Word2vec [J]. e-Science Technology & Application, 2015, 6(4): 54-59.

[7] Ning Jianfei, Liu Jiangzhen. Using Word2vec with TextRank to Extract Keywords [J]. New Technology of Library and Information Service, 2016(6): 20-27.

[8] Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space [C]//Proceedings of Workshop International Conference on Learning Representations. 2013.

[9] Ansj Lexical Parser [EB/OL]. [2016-10-01]. https://github.com/NLPchina/ansj_{seg}.

[10] Deep Learning with Word2vec [EB/OL]. [2016-10-01]. http://radimrehurek.com/gensim/models/word2vec.

Conflict of Interest Statement: The authors declare no conflict of interest.

Supporting Data: Available at https://github.com/iamxiatian/x-extractor/tree/master/data/articles.xml.

Received: 2016-10-28
Revised: 2016-12-16

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