Word Vector Clustering Weighted TextRank for Keyword Extraction Postprint

Authors: summer

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

Abstract

【Purpose】To incorporate world knowledge embedded in Wikipedia into the TextRank model through word embeddings for improved single-document keyword extraction.

【Method】Utilizing the Word2Vec model trained on Chinese Wikipedia data to generate word embeddings, we cluster the word vectors of TextRank word graph nodes to adjust intra-cluster node voting importance. By incorporating node coverage and position factors, we calculate random transition probabilities between nodes to generate the transition matrix. Node importance scores are finally obtained through iterative computation, and the top TopN words are selected as keywords.

【Results】The proposed word vector clustering weighting method outperforms comparison methods when TopN$ $7. The F-measure reaches its maximum at TopN=3, achieving a 3.374% incremental improvement over the previous best result. When TopN>7, results become 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 trained a Word2Vec model on Chinese Wikipedia data to generate word embeddings, performed clustering on the word vectors of TextRank word graph nodes to adjust voting importance within clusters, and combined coverage and position factors to compute random jump probabilities between nodes. After generating the transition matrix, we obtained node importance scores through iterative computation and selected the top N words as keywords. **Results:** The word vector clustering weighted method outperformed comparison methods when TopN ≤ 7. At TopN $= 3$, the F-measure reached its maximum value, representing a 3.374% improvement over the previous best result. When TopN > 7, results were similar to position-weighted methods. **Limitations:** Clustering analysis increases computational overhead. **Conclusions:** Word vector clustering weighting can effectively improve keyword extraction performance.

Keywords: Keyword Extraction; Word Embedding; TextRank; Word2Vec

1 Introduction

Keyword extraction automatically identifies representative words or phrases that reflect the main semantic information of a given text, with broad applications in library and information science. For instance, constructing term frequency matrices based on extracted keywords enables co-word analysis at the keyword level, revealing thematic evolution in literature and supporting content mining and analysis of large-scale library data. Implementation strategies for keyword extraction can leverage either intrinsic content and structural features of the text or training on large corpora. The former approach has attracted widespread attention in recent years due to its simplicity and satisfactory performance without requiring prior training, with TextRank being a typical representative of such algorithms.

Traditional TextRank algorithms utilize only information within the document itself. Theoretically, incorporating external knowledge could improve keyword extraction effectiveness. Since 2013, word vector representations have been able to project word semantics into a low-dimensional continuous space while preserving semantic characteristics from the corpus. This study leverages Wikipedia, currently the largest online open knowledge base, to train word vectors through the Word2Vec model. We then cluster these word vectors and perform non-uniform weighting of TextRank word graph nodes based on cluster distribution, thereby integrating external world knowledge into the TextRank computation process and 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 implementing importance ranking through iterative graph com-

putation. When using words as the basic unit, it can extract keywords; when using sentences, it can generate text summaries. Due to its superior performance over traditional TF-IDF and its simple implementation, TextRank has been widely adopted.

The original TextRank constructs word graphs without considering edge weights. To further improve keyword extraction, previous research has weighted words based on position, adjusting edge transition weights from three aspects: coverage influence, position influence, and frequency influence. Other studies have integrated TextRank with LDA topic models, combining single-document structural information with overall document theme information, finding improvements when datasets exhibit clear thematic distributions. Tag-TextRank utilizes social tags from web pages to enhance keyword extraction. More recent work has incorporated inverse document frequency alongside position weighting for keyword extraction in automatic paper review recommendation systems. With the rise of Word2Vec, researchers have begun applying it to keyword extraction, with some using word vector similarity for lexical clustering and selecting words closest to cluster centroids as keywords, while others incorporate word vector similarity matrices into the TextRank computation process.

In summary, the key to improving TextRank keyword extraction lies in incorporating external document information into its computation process. Existing methods such as topic weighting and inverse document frequency weighting require preprocessing of the dataset containing the target documents, yielding results that vary significantly across different datasets. Word2Vec training data is independent of the documents being processed, suggesting that using its trained word vectors to improve TextRank could produce more stable results. Unlike approaches that directly adjust transition probabilities based on word vector similarity, this study first performs word vector clustering on individual documents, then weights term importance based on distance to cluster centroids to construct a new probability transition matrix, achieving optimal results.

3 Methodology

Keyword extraction with TextRank transforms the problem into ranking the importance of words comprising the document. We first construct a candidate keyword graph (word graph) to represent structural relationships between words. We then perform cluster analysis on word vectors to determine clustering importance based on spatial position within clusters, implementing cluster weighting for TextRank. Finally, we construct a complete probability transition matrix between words and obtain node importance through iterative computation to achieve keyword ranking and extraction.

3.1 Word Graph Construction Following the TextRank principle, a document can be represented as a word graph based on adjacency relationships between words, with importance calculated according to structural characteristics within the graph. To construct the candidate keyword graph, we segment

text by sentences, perform word segmentation and part-of-speech tagging, and retain nouns, verbs, and adjectives that are not single characters to form the node set V. All adjacency relationships between words constitute the edge set E, creating candidate keyword graph G = (V, E). When constructing edges, if word a is followed by word b, we add two directed edges a → b and b → a, making G a directed graph, as shown in [Figure 1: see original paper].

Given this word graph G(V, E), let WS(u) denote the TextRank value of node u, calculated using formula (1):

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

where d [0,1] is the damping coefficient, representing the probability that any node will randomly jump to other nodes in the graph, ensuring convergence of TextRank iteration (typically set to 0.85); adj(u) denotes the set of adjacent nodes to u; and p(v → u) represents the random jump probability from node v to u.

Traditional TextRank employs a uniform transition strategy between adjacent nodes, with jump probability p(v → u) calculated using 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 the degree of node u. For example, in [Figure 1: see original paper], the probability of node v jumping to any adjacent node (v → i, i adj(v)) is 1/3.

To improve TextRank, previous research proposed non-uniform transition strategies based on node importance. However, these approaches calculated jump probabilities using only document-internal information, such as position and frequency of target words. To optimize jump probability assignment using external information, we propose a word vector clustering weighting algorithm.

3.2 Word Vector Clustering Weighting In 2013, Mikolov et al. released Word2Vec, a tool that uses shallow neural network models to automatically learn word occurrence patterns in corpora, embedding words into a moderate-dimensional space R^n (typically with dimension n between 100 and 500). The representation in this new space is called word vectors. Compared to traditional text representation methods, Word2Vec generates lower-dimensional vectors where semantic and syntactic relationships are well preserved: semantically similar words are closer in space, and linear operations on word vectors align with human understanding. Word vectors thus encode semantic information from large-scale datasets, enabling us to weight TextRank word graph node transition probabilities based on word vector relationships.

This study assumes that word vectors reflect global information, and a document can be clustered into several groups based on word vector similarity. A word farther from its cluster centroid reflects different aspects of information compared to words near the centroid, and thus has higher voting importance as a TextRank node, with higher transition probability to adjacent nodes.

Given document d and its candidate keyword set {w_i}, let $\vec{w}_i$ denote the word vector for term w_i from the trained Word2Vec model. Let C = {C_1, C_2, ⋯ , C_k} represent clustering results from K-means clustering on the document' s word vector set. We propose formula (3) to calculate the voting importance of any word u in its belonging cluster C_u:

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

where $\vec{u_c}$ is the centroid vector of cluster C_u, and dist($\vec{v_c}$, $\vec{v}$) denotes Euclidean distance between vectors. Formula (3) indicates that a cluster' s total voting score equals its node count, with each node' s voting weight distributed proportionally according to its Euclidean distance from the centroid—farther distances yield higher voting importance.

After representing semantic relationships between nodes as cluster-weighted influence and calculating each word' s voting importance, we propose formula (4) to compute cluster-influenced transition probabilities between nodes:

$$p_{\text{cluster}}(v \to u) = \begin{cases} \frac{\text{VoteWeight}(u)}{\sum_{w \in adj(v)} \text{VoteWeight}(w)} & \text{if } u \in adj(v) \\ 0 & \text{otherwise} \end{cases}$$

3.3 Transition Matrix Computation and Keyword Extraction According to link analysis theory, node importance can be computed iteratively given a transition probability matrix between graph nodes. Let matrix M represent the transition probability matrix between word graph nodes, as shown in formula (5):

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

where column j represents the probability distribution of jumps from word node j to other nodes, with each column summing to 1. Correspondingly, p_{uv} denotes the transition probability from node u to node v, i.e., p_{uv} = p(u → v).

To improve TextRank keyword extraction, we integrate word vector clustering weighting with previously proposed position weighting to assign appropriate

random jump probabilities p(u → v). In previous work, p_{cov}(v → u) represents the coverage influence of u ← v, calculated using formula (2), representing traditional TextRank' s voting contribution. Let p_{pos}(v → u) denote the position influence of u ← v, calculated using formula (6):

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

where I(v) represents node v' s position importance. Following previous experimental results, when v appears in the title, I(v) = 30; otherwise, I(v) = 1.

Based on coverage influence, position influence, and clustering weighted influence between nodes, we propose formula (7) to calculate the jump probability from node u to v:

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

where $\alpha + \beta + \gamma = 1$. Using formula (7), we generate the final transition matrix M. For the document graph represented by candidate keywords, we set initial node scores as in formula (8):

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

We then perform iterative computation using formula (9):

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

where e is an n-dimensional vector of all ones, and B_i represents node scores after the i-th iteration. When the difference between successive iterations B_i and B_{i-1} becomes sufficiently small (approaching 0), we stop iteration. At this point, each node' s score represents its importance in the graph. We sort nodes by score in descending order and select the top TopN nodes as the keyword extraction result.

4 Experiments and Analysis

4.1 Experimental Data We selected the Chinese Wikipedia export dataset "zhwiki-20150602-pages-articles-multistream.xml.bz"released in June 2015. This dataset contains 2,648,029 pages, including 1,480,963 article pages (55.93% of the total). After data cleaning to filter redirect pages and short articles, we retained 516,695 pages. Using the Chinese word segmentation tool Ansj, we segmented the text to create a training corpus for Word2Vec. We then used

Gensim's Word2Vec module with default parameters (CBOW model, dimension = 100, window size = 5) to train word vector model files.

For the keyword extraction test dataset, while previous research extracted 1,000 news articles with titles, bodies, and META keywords, the annotation quality was suboptimal, with issues like using titles themselves as keywords and low relevance between keywords and content. Therefore, we collected 1,524 articles from the Southern Weekend website, extracted their titles and bodies, and used explicitly marked tags as corresponding keywords to construct a new test dataset. This dataset averages 2,629.101 characters and 3.565 keywords per document.

4.2 Experimental Results and Analysis For comparative analysis, we used precision (P), recall (R), and macro-averaged F-measure as evaluation metrics. Let AK denote the keyword set provided in the test dataset and BK denote the algorithm-extracted keyword set. P, R, and F are calculated using formula (10):

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

Consistent with previous studies, we extracted 3, 5, 7, and 10 keywords for comparison with the dataset' s provided keywords. Compared methods include:

- M1: Original TextRank keyword extraction
- M2: Word2Vec-based word vector clustering keyword extraction
- M3: Word2Vec and TextRank fusion keyword extraction
- M4: Word position weighted TextRank keyword extraction
- M5: Our proposed word vector clustering weighted TextRank

All clustering analyses used K-means with 20 iterations, with other parameters set to optimal values from respective studies. All data and code are publicly available for reproducibility.

Results are shown in . For single-document keyword extraction, TextRank methods based on article structure and voting mechanisms significantly outperformed word vector clustering (M2). Although word vector clustering can group semantically related words, selecting centroid-proximal words as keywords proved ineffective. Consistent with previous findings, non-uniform weighting of TextRank transition probabilities improves performance. However, method M3, which directly computes cosine similarity between word vectors to bias transitions toward more similar words, performed poorly on our test dataset. Unlike M3, our approach first determines semantic clusters through word vector clustering, then assigns voting importance based on node-centroid distance within clusters. When retaining $ $7 keywords, our method outperformed all others, demonstrating that word vector clustering weighting enhances ranking of important keywords.

To comprehensively observe differences across methods, [Figure 2: see original paper] shows overall variations in precision, recall, and F-measure for TopN values in [1,10]. Overall, word vector clustering weighted TextRank and word position weighted TextRank significantly outperformed the other three methods. When TopN ≤ 5, word vector clustering weighting showed clear improvements over position weighting, though differences diminished as TopN increased. At TopN = 3, methods M4 and M5 simultaneously achieved maximum F-measure, with M5 showing a 3.374% improvement over M4.

To examine specific output when extraction performed poorly, we selected documents with completely mismatched results and organized various method outputs in , retaining the same number of keywords as original tags. For these complete mismatch cases, methods M1, M4, and M5 (all graph-based iterative methods) produced highly overlapping results that still represented main document content to some extent, while M2 showed some relevance and M3 performed relatively poorly.

In summary: (1) Direct application of word vector clustering analysis to single documents, selecting representative words from each cluster as keywords, is ineffective. (2) TextRank demonstrates stable performance for single-document keyword extraction, with word position weighting and word vector clustering weighting further improving accuracy.

This study incorporates Wikipedia' s world knowledge into TextRank' s keyword extraction process through word vector clustering weighting. Unlike IDF or LDA-based improvements, word vector training is independent of the target dataset, yielding more objective and stable results. Experiments show that word vector clustering weighting improvements are more significant when fewer keywords are retained, with no substantial difference from pure position weighting when TopN > 7.

Future research will explore more reasonable weighting methods for word vector clustering results and comprehensive evaluation of keyword extraction from an ordering perspective.

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 author declares no conflict of interest.

Supporting Data: The test article collection for keyword extraction is 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.