Finding titles representing segments of Wikipedia Articles from keyphrases

Hao Hu Shan Liu Mizuho Iwaihara

Graduate school of Information Production and System, Waseda University
2-7 Hibikino, Wakamatu-ku, Kitakyushu-shi,Fukuoka-ken, 808-0135 Japan
E-mail: waseda-tezuka@fuji.waseda.jp, vickey-liu@toki.waseda.jp, iwaihara@waseda.jp

Abstract: Wikipedia is a free online encyclopedia that aims to allow anyone to edit any article or create them. However, articles tend to become long and complex, so giving appropriate titles or key phrases to untitled segments is necessary for reader assistance. In this paper, we show methods to select titles for representing article segments. Key phrase extraction has been studied for years, but we concentrate on selecting a title phrase for a given target segment from candidate phrases, which needs to reflect local and global contexts. In this paper, we evaluate five features we proposed before, and one new feature which is based on word embedding. These features are combined to produce a ranked list of candidate titles. We construct over a candidate title set consisting of titles of articles, sections and subsections, and anchor texts of inner links (inter-article links) where the hidden title of the target segment is the ground truth. We compare performance of various feature combinations by precision@*K*, reciprocal rank and average precision.

Keyword: Wikipedia, Finding Representative Title, Candidate Ranking

1. Introduction

Wikipedia is a free, web-based, global volunteer collaborative repository of various articles on different aspects. This large and general reference work consists of more than 40 million articles in more than 250 different languages. Due to the large number of visitors and everyone could edit articles in his own opinion, articles tend to become long and complex for readers to capture the content with a clearer view. So, a title which could represent the whole content is so important when author intends to remind readers what he or she is going to emphasize. In Wikipedia, a large number of articles are usually constructed of several sections with subsections inside. These titles vary in forms from sentences to simply single words to refer. In addition, a substantial number of articles are unfinished and changed over time. Like an article for biographies of people, several events in which a person is involved usually do not end up when the article was created. The author would fill his or her work with an ending, while the title could not cover the current content or sometimes leave it as a blank. So, it is necessary for us to find an automatic method to generate a new title for an article and its sections, and subsections.

In conventional methods, most approaches like to extract keyphrases from the target article itself to get close to the objective view of humans. Key-phrase extraction is an intensively studied area in the field of text mining. Extraction methods are usually based on statistical combined with NLP (Natural Language Processing) field. For instance, the basic TF-IDF (term frequency and inverse document frequency) combined with chucking to generate n-gram keyphrases. Another example is the generative topic modeling, which assumes that each document is a mixture of a small number of topics and that each word's creation is attributable to a combination of the document's topics. These words could later build a word set for key phrase generation. In our previous work[7], we concentrated more on the phrase generation part to mine possible keyphrases. Following, we used FP-growth[11] for frequent co-occurring word extraction. Keyphrases are built from a set of words, which has an advantage of flexible phrase combinations no matter where a possible word occurs in the segment. In addition, we incorporate neighboring articles for keyphrase extraction that means similar or linked articles in Wikipedia. Applying phrase generation methods like FP-growth[11] could mine possible word from the neighbor we build.

However, different from our previous work, finding keyphrases most important or fitting as a title of the target segment is still requiring improvements. Keyphrase extraction is so subjective and varies with each individual. In this paper, we integrate five features proposed in [7] with a new feature based on topic vectors utilizing word embedding to estimate phrase quality from a candidate set which is fit as a title of the target segment. We use a linear function to combine these features and train them by gradient descent to obtain appropriate parameters for an article corpus. These features are combined to produce a ranked list of candidate titles set with different combinations. Top-ranked phrases are generated as titles. In evaluation part, we construct over a candidate title set consisting of titles from articles, their sections, subsections, and anchor texts of inner links (inter-article links) where the hidden title of the target segment is the ground truth.

The remainder of this paper is organized as: Section 2 is mainly about related work, Section 3 is the review of our previous work, Section 4 is the candidate object study in our work, Section 5 is about experimental studies, Section 6 is the conclusion and future work.

2. Related Work

For existing approached to supervised and unsupervised phrase extraction job, the methods can be categorized as follows.

1. An unsupervised method for keyphrase extraction is utilizing TF-IDF [8] to rank candidate keyphrases (here chucks as candidates) and select top-ranked ones.
2. TextRank [9] is one of graph-based ranking methods for extraction intended to build a word graph in which edges represent semantic relatedness between two co-occurring words.
3. KEA [12] (Keyphrase Extraction Algorithm) KEA algorithm is a supervised extraction algorithm for long documents. A Naïve-Bayes classifier on term frequency and term position features is trained o produce phrase ranking lists.
4. LSA(Latent Semantic Analysis) [13] & LDA(Latent Dirichlet Allocation) [2]:The former is a topic modeling technique, learning word and document representations by applying singular value decomposition to a words-by-documents co-occurrence matrix. LDA is a generative topic model assuming words in each document were generated by a mixture of topics, where a topic is represented as a multinomial probability distribution over words.
5. FP growth(frequent pattern mining) [11] This method intends to extract frequently occurring word sets to obtain an order-free word set. In our previous work, we adopted this method, as our target texts are varying in length, meaning that certain segments are just a few sentences over dozens of sentences.In addition, FP-growth can be applied to articles which are related to the target article, to discover more candidate phrases.

The above methods show different approaches to phrase generation. However, TF-IDF prefers high frequency term, KEA method is a supervised method and a large number of efficient training segments are necessary. LDA could be used to discover topically-related, but not appearing in the target segment can be discovered, but its training corpus should be sufficiently large. Related Wikipedia articles have to be carefully selected for topic extraction.

3. Phrase Generation

In this section, we discuss finding keyphrases for the target segment as a candidate title set. We could consider the usage of existing methods described in Section-2. In our previous work[7], we apply frequent pattern mining on the target segment. Below we briefly describe the method.

3.1 Related articles

Here we all related articles as articles either linked from the target article, or having significant overlaps with the title words of the target article. We can sample candidate phrases from the these related articles. Such related articles can provide candidate phrases which may not occur in the text of the target segment. We apply standard preprocessing, such as stop words, to the corpus of the related articles.

3.2 Frequent Patterns

Frequent patterns on words are order-free sets of words which frequently co-occur frequently in documents[11]. FP-growth can extract frequent word sets be from the corpus, then they can be utilized as candidate phrases. This approach allows us to find phrases having different word orders and containing unrelated words in the middle. But for finding the most popular word orders on phrases, we can utilize query results on search engines, and n-gram corpus such as Google-Ngram.

4. Phrase Quality Estimation

To compare the fitness of candidate phrases as the title of the target segment text, we utilize existing features and introduce a new feature. The five existing features are Coverage[10], Phraseness[7], Uniqueness[7], Potentialness[7], Sim-PF[7]. We then introduce the new feature Embedding-Vector, which evaluate semantic relatedness between the word vectors, generated by paragraph2vec[4] of the target and related articles. In the following, the corpus is denoted by C.

4.1 Quality Phrase Features

Coverage [10]**:**

A representative keyphrase should cover many articles. Coverage gives a high score to phrases that occur frequently in the corpus C of the related articles:

$$S_{cov}(p) = \frac{f(p)}{|D|} \qquad (1)$$

Where $f(p)$ refers to the frequency of phrase p occurring in corpus C and $|D|$ refers to the number of articles in C.

Uniqueness [7]**:**

A representative phrase should be more frequent in the target segment rather than the other segments in the same article.

Uniqueness captures such locality of the phrase within the target article. In (2), here $f(s)$ is the number of sections in the target article. $|S|$ is the number of sections in the target article. $f_s(w)$ is the frequency of word w in segment s.

$$S_{uni}(w) = \log\left(\frac{|S|}{f(s)} * \frac{f_s(w)}{\sum_{s' \in S, s' \neq s} f_{s'}(w)+1}\right) \qquad (2)$$

The uniqueness of phrase p is defined as the average of the uniqueness scores of the words in p:

$$S_{uni}(p) = \frac{\sum_{w \in p} S_{uni}(w)}{|p|} \qquad (3)$$

Potentialness [7]**:**

The feature potentialness is to evaluate relatedness of a phrase to a segment by cosine similarities on latent topic vectors. We construct topic vectors on the corpus C by Gibbs LDA[2], with a given topic number k. Potentialness can capture relatedness of phrase words even when the words are not appearing in the segment.

$$S_{pot}(w|s) = \sum_{j=1}^{k} p(w|t_j) * p(t_j|s) \qquad (4)$$

In (4), k is the latent topic number, $p(w|t_j)$ is the word-topic distribution computed by Gibbs LDA. The potentialness for phrase p is defined as the average of the potentialness values of the words in p:

$$S_{pot}(p) = \frac{\sum_{w \in p} S_{pot}(w|s)}{|p|} \qquad (5)$$

Phraseness [7]**:**

A word set is considered as a good phrase if the words in the set often co-occur in an identical sentence. The phraseness defined in (6) evaluates how often the words of a phrase p appear in one sentence:

$$S_{phr}(p) = \sum_{i=1}^{n} \frac{f_i(p)}{\Pi_{w \in p} f_i(w)} \qquad (6)$$

Here, i denotes the i-th article in corpus C, n denotes the number of the articles in C, $f_i(p)$ denotes the times phrase p occurs in an identical sentence of article i, and $f_i(w)$ denotes the times word w occurs in article i.

Sim-PF(Similarity-weighted Phrase Frequency) [7]**:**

This feature gives a high score to a phrase if p is frequently occurring in segments having high similarities with the target segment. We rank the articles in C, from i=1 to i=n, by the TF-IDF cosine values with the target segment. Then we compute the product of the ranking and log-frequency of p in article i.

$$S_a(p) = \sum_{i=1}^{n} \frac{n-i}{n} * \log\big(fi(p)\big) \qquad (7)$$

Utilizing similarity ranking instead of raw similarity scores gives us an ideal weighting on articles.

Embedding-Vector:

Inspired by [4], we transform the text of a segment s into a fixed-length feature vector. The previous five features mainly focus on distributions of words in the target segment and related articles. On the other hand, paragraph vectors allow us to measure semantic relatedness between the target segment and candidate phrases in a coherent manner, where vectors can be trained over the whole English Wikipedia articles. In this feature, we evaluate by cosine similarities between the vectors of the phrase and segments, where segments are weighted by similarity ranking to the target segment:

$$S_{vec}(p) = \sum_{i=1}^{n} \frac{n-i}{n^2} * Cos\left(Vp\ ,\ Vs_i\right) \qquad (8)$$

Here, Vp is the vector for phrase p, Vs_i is the segment vector for the i-th segment in the top-n similar articles, where the first segment is the target segment itself, and the second and following segments are the remaining segments in the corpus C, ranked by the TF-IDF scores with the target segment. Finally, the score is normalized to 1 by dividing by n. In this paper, we choose n to be 10.

4.2 Linear function for ranking

The six features are combined with a linear function to calculate scores for candidate phrases. The candidate phrase having the highest score will be selected as the title of the target segment.. The score function is defined as follows:

$$S(p) = \theta_0 S_{cov}(p) + \theta_1 S_{phr}(p) + \theta_2 S_{uni}(p) + \theta_3 S_{pot}(p) + \theta_4 S_{sim}(p) + \theta_5 S_{vec}(p) \qquad (9)$$

Here, $\boldsymbol{\theta} = [\theta_0, \theta_1, \theta_2, \theta_3, \theta_4, \theta_5]$ denotes the weight vector on the six measurement features.

To find an optimum weight vector $\boldsymbol{\theta}$, we apply gradient descent [7].

4.3 Classifiers of features

Besides linear function, we can also consider utilizing various binary classifiers for combing the features. Since our goal is to find a title for the target segment, we try to rank all candidates according to the fitness of each classifier.

Four classifiers here are applied in this paper: Logistic Regression, Naïve Bayes, Support Vector Regression and Random Forest.

In Logistic Regression and Naïve Bayes, we rank candidates by possibility score only when the candidate is a quality one (candidate label is 1).

Support Vector Regression is a variation of SVM (Support Vector Machine) for regression. We choose RBF kernel for our task and rank candidates according to regression scores.

Except Random Forest, the other classifiers are trained by 63 different feature combinations to obtain the best result. Random Forest generates a number of decision trees, where each tree is formed by the random feature combination. Random Forest is also trained here for the regression task to rank candidates.

5. Experiment

In this part, we perform experiments to test the performances of various feature combinations by precision @K, reciprocal rank and AP (Average Precision).

5.1 Dataset:

We extend the dataset in our previous work[7], choosing typical English Wikipedia articles which could represent different aspects like biography of people, historic event, Science, Politic.

Data set corpus is filled with a large number of segments and these segments have subsection titles and their upper titles, inner link titles in the segment. In addition, for each article in this corpus, we would download its related articles to form the neighbor. Usually, the related article's size is thousands to support. Corpus C represents for the neighbor Top-K most similar article to target segment. In this experiment, we set the K value 10. The following table shows our dataset details:

Corpus	Quantity
Segments	266
Candidate titles	7,179

Table.1 Dataset Corpus

In this experiment, we construct a Corpus set consisting of 3 parts.

1. The segment title and its upper title if possible

2. The inner link title appears in the segment which is pretty related to the target segment itself.
3. Other segment titles in the same article, they are like bad phrases.

Average number	Quantity
Segment title and upper title	2
Other titles in the same article	9
Inner link titles	15
Candidate Titles	26

Table.2 Candidate Title Set and Average Number of candidates.

5.2 Golden Standard

These three parts showed in Table.2 construct a candidate title set for a target segment. We treat part 1 titles(segment titles and their upper titles) as golden standard and test if our work could find these original titles at Top-K in the ranking list.

5.3 Evaluation

Since our work is evaluated by the position of correct answers in the ranking list, here we introduce 3 conventional methods to estimate performance.

Precision@ K: Precision at k documents (P @K) is still a useful metric (e.g. P@5 or "Precision at 5" corresponds to the number of relevant results on the first search results page), but fails to take into account the positions of the relevant documents among the top k. Another shortcoming is that on a query with less relevant results than k, even a perfect system will have a score less than 1. It is easier to score manually since only the top k results need to be examined to determine if they are relevant or not.

In this experiment, we are concerned about Precision@ K result and set K value at 1, 3, 5.

Reciprocal Rank: Considering the position of the first relevant document position, Reciprocal rank is the inverse rank number

$$Reciprocal\ Rank = \frac{1}{Rank\ Position\ number} \quad (13)$$

AP(Average Precision): Average precision for a set of queries is the mean of the average precision scores for each query.

$$AP = \frac{Ave \sum_{q=0}^{Q} AveP(q)}{Q} \quad (14)$$

Here in formulation 13, q query in our experiment means every segment, $P(q)$ denotes for Precision @K. Q is the number of total segments.

Like in table.3, we have a ranking list for segment “Early life” in the article “Hillary Clinton”. This table would be shown as the following example: Precision@1, 3, 5. Average Precision, Reciprocal Rank. The original title is listed at position second and we could get the data as table.4 shows.

Candidate title	Score
Westchester County	0.1107
2006 re-election campaign	0.1040
United States Senate	0.0793
Jeanine Pirro	0.0704
Jonathan Tasini	0.0670
John Spencer	0.0450
District Attorney	0.0442

Table.3 Candidate Ranking of Segment: “2006 re-election campaign” in “Hillary Clinton” article by five features.

Early life	Precision @1	Precision @3	Precision @5	Average Precision	Inverse Rank
Cov	1/1	1/3	1/5	0.51	1/1
Cov Phr	1/1	1/3	1/5	0.51	1/1
Cov PhrPot	0/1	1/3	1/5	0.17	1/2
Cov PhrSim	1/1	2/3	2/5	0.68	1/1
Cov PhrUni	0/1	1/3	1/5	0.17	1/2
Covpot	1/1	1/3	1/5	0.51	1/1
SixFeatures	0/1	2/3	2/5	0.35	1/2

Table.4 Evaluation table on segment “2006 re-election campaign” in article “Hillary Clinton”.

5.4 Training part

For dataset which consists of 266 segments, we split them into two parts.

The First part, we have 196 segments for training. They are selected from the article "Greek mythology", "Barack Obama", "Bryan Gunn", "John Sherman", "Wood Badge", "General relativity", "Society of the Song dynasty", "Richard Nixon", etc.

These segments are used for Gradient Descent training to obtain the fixed parameters for each feature.

In addition, we leave 70 segments to test the performance of our work. They are from the article "Hillary Clinton", "Attachment Theory", "History of Minnesota", "Domitian", "Political integration of India", etc.

At last, We have the six features in our experiment which means 63 different combinations. Table 3 would show the final fixed parameters for 8 examples.

	Cov	Phr	Sim	Uni	Vec	Pot
Six	-0.04	-0.01	0.11	0.06	0.05	0.05
NoCov		-0.01	0.08	0.06	0.04	0.04
NoPhr	-0.04		0.1	0.06	0.05	0.05
NoUni	0.01	0.03		0.06	0.06	0.06
NoSim	-0.04	0.00	0.11		0.06	0.05
NoVec	-0.03	-0.01	0.11	0.07		0.06
NoPot	-0.02	-0.02	0.12	0.07	0.08	

Table.5 Seven examples of different feature combination parameters after training from linear model.

5.5 Result

In this part, we would show the final work performance on 63 different feature combinations. The table 4 shows the Top-8 highest AP score in Precision @1 ,3 ,5 and Reciprocal Rank.

Method	P@1	P@3	P@5	AP	Inverse Rank
CovPhr Vec	0.289	0.202	0.156	0.2164	0.4512
CovVec	0.2753	0.183	0.294	0.2119	0.4470
CovVec Pot	0.2753	0.183	0.289	0.2109	0.4475
Cov	0.2318	0.188	0.270	0.1942	0.4046
CovPot	0.2318	0.188	0.270	0.1942	0.4046
VecPot	0.2463	0.173	0.251	0.1903	0.4188

Table.6 Top-6 highest Average Precision from linear function.

Method	P@1	P@3	P@ 5	AP	Inverse Rank
Logistic R CovVecPot	0.275	0.227	0.28	0.2244	0.4689
Linear CovPhr	0.289	0.202	0.26	0.2164	0.4512
Logistic R CovPhrVec	0.289	0.193	0.26	0.2131	0.4470
Naïve B CovVecPot	0.246	0.227	0.27	0.2128	0.4502
Linear CovVec	0.275	0.183	0.29	0.2119	0.4470
SVR VecPot	0.231	0.207	0.32	0.2112	0.4463
Random Forest	0.159	0.154	0.24	0.1539	0.3648

Table.7 Top-7 highest Average Precision across different classifiers

From table 6, we see that our work could find the correct title from the title candidate set in which the highest AP score reaches 67.34% precision. In addition, Embedding-vector, coverage and Phraseness features support the function to obtain the best result.

Reciprocal rank here is the mean average score across different segments. The larger the score is, The higher position golden standard appears in the ranking list. That is, we could find the correct answer quickly. In our experiment Method Embedding-Vector, Potentialness and Coverage feature combination in Logistic Regression shows the best result.

Besides the linear model Logistic Regression, we also study different classifier performances. Performed as Table 7, linear function with Coverage, Phraseness feature combination performs well. However, other classifiers like Naïve Bayes also has good result as its inverse rank score is close to the linear function highest score ranking at 2nd. While other classifier's performance, such as random forest is not very ideal.

Since the limit of pages, we could not show all the 63 combination results.

6. Conclusion and Future Work

Our work aims to find an appropriate title for the target segment is working according to the final result. When adding the appropriate candidate phrase like FP-growth, we could find quality phrases to form a new title automatically. Although, In table.6, feature Embedding-vector may show better performance. We still have to point out that the Embedding-Vector method is not stable. It is necessary to retrain the article corpus until a balanced state.

Referenced Work

[1] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*.

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

[3] S. Liu,M Iwaihara. “Extracting Representative Phrases from Wikipedia Article” 2016 DEIM Forum.

[4] Le, Mikolov “Distributed Representations of Sentences and Documents”. *The 31th International Conference on Machine Learning*,

[5] I.Hulpu, C.Hayes,D.Greene, “Unsupervised Graph-based Topic labelling using DBpedia”, Pro. of ACM WSDM’13, pages 465-474, 2013.

[6] Z. Liu, W. Huang, Y. Zheng, and M. Sun,”Automatic key phrase extraction via topic decomposition”, Proc. Of the 26th Annual Conference on Learning Theory, 2013.

[7] S. Liu, M. Iwaihara, “ Extracting Representative Phrases from Wikipedia Article Sections”, ICIS Conference, Special Sessions, June 2016.

[8] Salton and C. Buckley “Term-weighting approaches in automatic text retrieval.” Information processing and management, pages 513-523 1988

[9] Mihalcea and P. Tarau. “Textrank: Bringing order into texts” In Proceedings of ACL, pages 825-833

[10] M. Danilevsky, C. Wang etc, ”Automatic Constuction and Ranking of Topical Keyphrases on Collections OF Short Documents”, Proc.of 201 4SIAM Int. Conf. on Data Mining (SDM’14),2014

[11] J.Han, J.Shang etc,“Mining Frequent Patterns without Candidate Generation” *Conference on Empirical Methods in Natural Language Processing*(EMNLP), pages 366-376 2010

[12] Ian H.Witten, Gordon W.Paynter etc “KEA: Practical Automatic Keyphrase Extraction” ACM DL 1999 pages 254-255

[13] S.C. Deerwester, S.T. Dumais etc, “Indexing by latent semantic analysis” Journal f the America Society of Information Science pages:391-407,1990