PAPER Special Section on Knowledge Discovery, Data Mining and Creativity Support System

Learning to Generate a Table-of-Contents with Supportive Knowledge

Viet Cuong NGUYEN[†a)], **Le Minh NGUYEN**[†b)], ***Nonmembers***, ***and*** **Akira SHIMAZU**[†c)], ***Member***

SUMMARY In the text summarization field, a table-of-contents is a type of indicative summary that is especially suited for locating information in a long document, or a set of documents. It is also a useful summary for a reader to quickly get an overview of the entire contents. The current models for generating a table-of-contents produced relatively low quality output with many meaningless titles, or titles that have no overlapping meaning with the corresponding contents. This problem may be due to the lack of semantic information and topic information in those models. In this research, we propose to integrate supportive knowledge into the learning models to improve the quality of titles in a generated table-of-contents. The supportive knowledge is derived from a hierarchical clustering of words, which is built from a large collection of raw text, and a topic model, which is directly estimated from the training data. The relatively good results of the experiments showed that the semantic and topic information supplied by supportive knowledge have good effects on title generation, and therefore, they help to improve the quality of the generated table-of-contents.

key words: *text summarization, supportive knowledge, semi-supervised learning*

1. Introduction

A table-of-contents is a list of divisions (chapters or articles) and the pages on which they start [*]. In the text summarization field, a table-of-contents is a kind of indicative summary, which is especially suited for locating information in a long document, or a set of documents. For instance, a table-of-contents could play a role as a navigation tool for accessing information in a long document on a mobile device [1], or understanding a long, unstructured transcript of an academic lecture or a meeting [2]. A reader could use a table-of-contents to locate all the parts in a set of documents that are relevant to his/her interests. In addition to the above functions, a table-of-contents with its meaningful titles, could help a reader get an overview of the entire contents very quickly.

Given a long document, or more generally, a set of documents, our goal is to generate a tree, wherein a node represents a title that summarizes the content of a segment (in a long document), or segments that have similar content (in a set of documents). That tree—a hierarchical structure—can be seen as a table-of-contents [2]. This process involves three tasks: (1) separating every document into a hierarchical structure of segments (a tree of segments), (2) merging all the tree of segments to form a unique tree, and (3) generating titles for every segment in the above tree to form a table-of-contents. In this research, we mainly focus on the third task, with the assumption that the first and second tasks could be done using one of a series of existing methods, such as *TextTiling* [3], *C99* [4], *MinCutSeg* [5], *SenOrder* [6], *CST* [7].

So far, the literature on the title generation task is relatively sparse [2], [8], [9]. Angheluta, et al. [8] presents an unsupervised approach to generating a title for a segment. The title is the best noun phrase extracted from the segment. Their method uses some simple grammatical rules to identify the noun phrases, and uses term frequencies to score the noun phrases. This approach usually produces well-formed titles. However, the extracted titles are usually too short, with very low quality in reflecting the meaning of the segments. Furthermore, every title is made independently, and therefore, the generated table-of-contents lacks coherence. Branavan, et al. [2] presents a supervised learning model for generating a table-of-contents. Their model intends to address most of the problems of the previous approaches. Their model captures most important features at the word level and the word sequence level in generating titles, such as position of a word, its TF*IDF, part-of-speech information, language model score, and so on. They also propose an additional model, which captures the relations between titles, to generate a more coherent table-of-contents. However, in their experimental results, in spite of better scores in comparison to other baseline models, a large number of titles in the generated table-of-contents are meaningless or not related to the content of the corresponding segments.

In this research, we try to generate a table-of-contents with meaningful titles with the following idea: *a good title should have a topic relation to the text*. Therefore, we should use as much semantic or topic information as possible to help the model to generate a meaningful title that has strong relation to the content. The semantic and topic information, which are supportive knowledge, used in this research are derived from a hierarchical clustering of words [10], and a topic model [11]. We follow a two-stage semi-supervised approach to use supportive knowledge in a discriminative learning model, which is a perceptron-based learning model [12]–[14]. The supportive knowledge is derived from un-annotated texts using unsupervised learning

Manuscript received May 11, 2010.
Manuscript revised September 30, 2010.

[†]The authors are with School of Information Science, Japan Advanced Institute of Science and Technology, Nomi-shi, 923–1292 Japan.
a) E-mail: cuongnv@jaist.ac.jp
b) E-mail: nguyenml@jaist.ac.jp
c) E-mail: shimazu@jaist.ac.jp

DOI: 10.1587/transinf.E94.D.423

[*]Definition in WordNet 3.0, ©2006 by Princeton University

Finding a dynamic programming
    Shortest paths in the Floyd Warshall
    Running time of the Floyd Warshall algorithm
    Constructing a recursive algorithm
Bellman Ford algorithm for the running
    Equation for a given function
    Bellman Ford algorithm from section

Fig. 1 A portion of a table-of-contents generated by our model.

algorithms, which are Brown clustering algorithm [10] and Latent Dirichlet Allocation [11].

Our experimental results on the public dataset show that our approach could help the model to produce a table-of-contents with well-formed and meaningful titles. The evaluation results also show that the titles generated by our model have higher quality than those of the baseline model [2]. Figure 1 shows a portion of a table-of-contents generated by our model.

The next section presents the structured learning model for generating a table-of-contents. Section 3 describes the supportive knowledge used in this research, how to derive items, and how to use them. Section 4 explains features used for the learning model. Section 5 presents our experiments on the public dataset, with results and some discussions. Section 6 gives some conclusions.

2. Structured Learning Model for Generating a Table-of-Contents

In this research, we mainly focus on the third task: generating a table-of-contents from a hierarchical structure of segments. In other words, we try to generate a tree of titles from a tree of segments. To formalize this task, we employ the approach used in [2] with some modifications in the learning algorithm.

This task is formalized as a structured learning problem, in which the learning algorithm produces a model that will be used to generate a tree of titles $\mathcal{T}$ from a tree of segments $\mathcal{S}$. This model is, in turn, decomposed into two components, the local model and the global model. The local model is used to generate a list of candidate titles for every segment $s \in \mathcal{S}$. If the size of that list is one, we have a one-best model that generates a title $t \in \mathcal{T}$ for a segment $s \in \mathcal{S}$. The global model is mainly used for capturing the relations between the candidate titles generated by the local model to form a coherent table-of-contents.

We begin the presentation of the models with some common notations. A tree of segments $\mathcal{S}$ and its tree of titles $\mathcal{T}$ are provided as training data. Every segment $s \in \mathcal{S}$ has a corresponding title $t \in \mathcal{T}$ to form a pair *(segment, title)*. All the pairs $\mathcal{D} = \{(s, t)\}$ are provided as the training data for the local model. $|\mathcal{D}|$ is the size of $\mathcal{D}$, and $|t|$ is the length of t. $f(s, z)$ is feature vector of a segment s and a partial title z. w_l and w_g are the weight vectors of the local model and the global model, respectively.

Input:
- $\mathcal{D} = \{(s, t)\}$ is a set of *(segment, title)*;
- N is the number of iterations;
- k is the beam size.

Output:
- w_l is the weight vector of the local model.

1: **for** $i = 1 \rightarrow N$ **do**
2: **foreach** $(s, t) \in \mathcal{D}$ **do**
3: **for** $j = 1 \rightarrow |t|$ **do**
4: $\mathcal{B} = \mathrm{GetTop}(\mathrm{PartialGen}(\mathcal{B}, s), k)$
5: **if** $t[1..j] \notin \mathcal{B}$ **then**
6: $w_l = w_l + f(s, t[1..j]) - \frac{\sum_{z \in \mathcal{B}} f(s,z)}{|\mathcal{B}|}$
7: $\mathcal{B} = \{t[1..j]\}$
8: $w_l = w_l / (N * |\mathcal{D}|)$

Fig. 2 Training algorithm for the local model.

2.1 The Local Model

The local model aims to generate a list of candidate titles given a segment of text. As is common in text summarization, it assumes that a title could be generated from the words inside the text [15]. There are two common approaches based on this assumption: extraction and generation. The former approach normally extracts the most important clause or the most important noun phrase from the text to form a title. The latter approach normally chooses words inside the text to form a title in a generational style. In this model, the latter approach is employed.

In the training step, a vine-growth strategy [16] is employed to learn a model for generating a title of a segment of text by a perceptron-based algorithm. The training process simulates the process of building a title t incrementally by appending words inside the given segment s at each iteration (as in Fig. 2). By following this strategy, the size of the search space is exponential to the length of the desired title, therefore, a beam search algorithm is used. At each iteration, the beam $\mathcal{B}$ keeps up to the k most promising partial titles. This strategy has been successfully applied in other NLP tasks, such as parsing [17] and chunking [16]. In comparison to the original version, we use an averaged perceptron model, and update the parameters of the model at the end of each iteration.

In the algorithm in Fig. 2, N is the number of iterations of the perceptron-based learning algorithm. At each iteration, by using function *PartialGen*, $\mathcal{B}$ is grown by appending every word in s to each partial title in $\mathcal{B}$ to make a list of partial titles of length j. After that, by using function *GetTop*, $\mathcal{B}$ is pruned to contain k top-ranked titles based on the score $w_l \cdot f(s, z), \forall z \in \mathcal{B}$. In this score, w_l used in iteration i is the weight vector of the iteration $i - 1$. The weight vector is updated whenever $\mathcal{B}$ does not contain the prefix $t[1..j]$ of the true title t. At the same time, $\mathcal{B}$ is pruned to contain only $t[1..j]$.

In the decoding step of the local model, the algorithm in Fig. 3 produces a list of candidate titles by incrementally generating titles from the words inside the segment of text.

Input:
- w_l is the weight vector of the local model;
- s is the set of words inside the segment of text;
- l is the length of the desired title;
- k is the beam size.

Output:
- A list of k candidate titles.

1: $\mathcal{B} = \{$a set containing an empty string$\}$
2: **for** $i = 1 \rightarrow l$ **do**
3: $\quad \mathcal{Q} = \emptyset$
4: $\quad$ **foreach** $z \in \mathcal{B}$ **do**
5: $\quad\quad$ **foreach** $w \in s$ **do**
6: $\quad\quad\quad \mathcal{Q} = \mathcal{Q} + \{z + w\}$
7: $\quad \mathcal{Q}$ is sorted by $w_l \cdot f(s, z), \forall z \in \mathcal{Q}$
8: $\quad \mathcal{B} = \{$top k partial titles of $\mathcal{Q}\}$

Fig. 3 Generating a list of candidate titles for a text segment.

The length of the desired title is provided as a parameter of the algorithm [2]. This algorithm uses the same strategy as in training to reduce the size of the search space.

In the algorithm in Fig. 3, s is the set of words w of the input text segment; l is the length of the desired title of that segment; and $\mathcal{B}$ is a beam containing the k top promising partial titles, which is similar to $\mathcal{B}$ in the training algorithm. $\mathcal{Q}$ is a sorted list of the titles made by appending every word $w \in s$ to each partial title $z \in \mathcal{B}$. The output of this algorithm is the top k candidate titles in the last beam $\mathcal{B}$, which are used in the global model.

2.2 The Global Model

Normally, we can use the best title from among the candidate titles, which are produced by the local model, to form a title of a node in the table-of-contents. However, the process of generating candidate titles for a segment is independent of the other segments. Therefore, we need a *global model* to build a coherent table-of-contents, which can capture the relations between titles.

In the global model, the input is a tree $\mathcal{T}$, wherein a node contains a list of k candidate titles of the corresponding segment $s \in \mathcal{S}$. The output of this model is a tree with the same structure, wherein a node contains only one title. In this model, the input and the output are hierarchical structures (trees). This is different from the local model, in which the input and the output are sequences of words (a segment of text and a title). However, we can still employ the learning and decoding algorithms used in the local model. The technique used here is to traverse the tree of titles in pre-order—the order of titles will appear in the table-of-contents. By using this technique, we can also incrementally build the output tree using beam search, as in the local model. The differences here are the elements used in extending and pruning the beam. In the global model, at a node in the pre-order traversing, a beam, which contains a list of K partial trees, is grown by appending every candidate title of that node. After that, the beam is pruned to contain a list of the K top-ranked partial trees. Similar to the local model, partial trees are ranked by score, which is the output value of the perceptron-based model.

The output of the decoding algorithm of the global model is a tree of titles, which is also the desired table-of-contents.

3. Supportive Knowledge

In this research, we aim to use supportive knowledge to make the learning model take into account semantic and topic information. Supportive knowledge could help the model to improve the quality of the generated titles, and therefore, the generated table-of-contents. To reduce the cost of the training process, the supportive knowledge should be easily derived from un-annotated text in an unsupervised way. After that, we incorporate that knowledge as features in a supervised learning model, which is described in Sect. 2. This approach is called a two-stage semi-supervised approach, which was previously used in [12]–[14].

To achieve this goal, we, firstly, try to use word classes, which have two advantages in our task [18]. First, because the classes are created for use in a language model, they should be useful in predicting subsequent words. This is useful in title generation. However, at the same time, it is natural to think of such classes as semantic in nature. This could reduce the effects of the data sparsity. To get word classes, which could help the model to exploit both the above advantages, the word clustering algorithm should use a similarity measure based on contextual properties of words. In this research, we use the Brown clustering algorithm [10], which is detailed in Sect. 3.1. The Brown algorithm gets a large collection of raw text as an input and produce a hierarchical clustering of words. This hierarchical structure allows us to choose an arbitrary number of clusters with an arbitrary level of abstraction. This is the main advantage of this algorithm in comparison to K-Means based algorithms.

We, secondly, try to use topic modeling, which could provide topic information in title generation. This is based on the idea: a good title should reflect most important topics mentioned in the given segment of text. This relation could be modeled by the similarity between topic distribution of the candidate titles and the text. Thereby, our model could choose the title that best reflects topic information in the text. In this research, we choose the most popular topic modeling method, Latent Dirichlet Allocation [11], [19], to estimate and infer the topics from the data. This method is briefly introduced in Sect. 3.2.

Both word clustering and topic modeling are used to produce clusters of words from a large collection of raw text. However, word clustering normally produces hard clusters, in which a word can belong to only one cluster. It is different from topic modeling, in which a word can belong to many clusters with different probabilities.

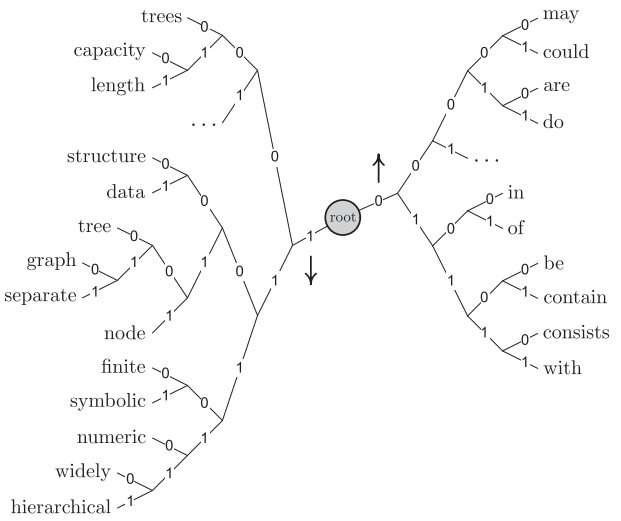

Fig. 4 An example of a hierarchical clustering. Each word at a leaf is encoded by a bit string with respect to the path from the root, where 0 indicates an "*up*" branch and 1 indicates a "*down*" branch.

3.1 The Brown Algorithm

The Brown algorithm is a hierarchical agglomerative word clustering algorithm [10]. The input of this algorithm is a large sequence of words $w_1, w_2, \ldots, w_n$, which are extracted from raw texts. The output of this algorithm is a hierarchical clustering of words—a binary tree—wherein a leaf represents a word, and an internal node represents a cluster containing the words in the sub-tree, whose root is that internal node.

This algorithm uses contextual information—the next word information—to represent properties of a word. More formally, $C(w)$ denotes the vector of properties of w (or w's context). We can think of our vector for w_i as counts, for each word w_j, of how often w_j followed w_i in the corpus:

$$C(w_i) = (|w_1|, |w_2|, \ldots, |w_n|)$$

$C(w_i)$ is normalized by the count of w_i, and then we would have a vector of conditional properties $P(w_j|w_i)$. The clustering algorithm used here is HAC-based, therefore, at each iteration, it must determine which two clusters are combined into one cluster. The metric used for that purpose is the minimal loss of average mutual information [18].

Figure 4 shows a portion of a hierarchical clustering, which is derived from a small portion of text, which contains 11 sentences and 116 words. This portion of text is about tree and graph data structures in computer science. From this tree, we can freely get a cluster of words by collecting all words at the leaves of the sub-tree, whose root is a chosen internal node. For instance, some clusters are: {*trees, capacity, length*}, {*structure, data, tree, graph, separate, node*}, {*in, of*}, {*widely, hierarchical*}, and so on.

To use word clustering information in our model at several levels of abstraction, we encode each word cluster by a bit string that describes the path from the root to the chosen internal node. The path is encoded as follows: we start from the root of the hierarchical clustering, "0" is appended to the binary string if we go up, and "1" is appended if we go down. For instance, to encode above four clusters, we use the following bit strings "100", "110", "010", and "1111", respectively. If we want to use a higher level of abstraction, we can simply combine the clusters that have the same prefix. For instance, if we need only two clusters, we can use the prefix with the length of 1. In that situation, all the words in the left sub-tree are in a cluster encoded by "1", and all the words in the right sub-tree are in another cluster encoded by "0".

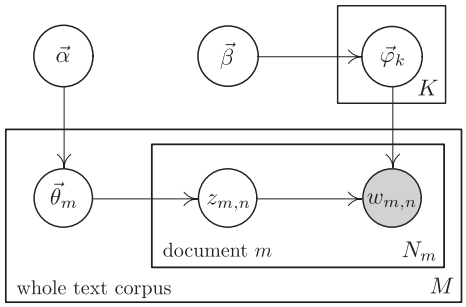

Fig. 5 The generative graphical model of LDA.

3.2 Topic Modeling with LDA

Latent Dirichlet Allocation (LDA) [11], [19] is a probabilistic generative model that can be used to estimate the properties of multinomial observations by unsupervised learning. With respect to text modeling, LDA is a method to perform so-called Latent Semantic Analysis (LSA). It is shown that the co-occurrence structure of terms in text documents can be used to recover this latent topic structure, notably without any usage of background knowledge. Latent-topic representations of text, in turn, allows modeling of linguistic phenomena like synonymy and polysemy. This could help our model to capture the relations between the titles and the text at the semantic level, rather than by lexical overlapping.

The generative graphical model of LDA is shown in Fig. 5. This generation process can be interpreted as follows [11]: a document containing N_m words $\vec{w}_m = \{w_{m,n}\}_{n=1}^{N_m}$ is generated by first picking a distribution over topics $\vec{\theta}_m$ from a Dirichlet distribution $\mathrm{Dir}(\vec{\alpha})$, which determines topic assignments for words in that document. Then the topic assignment for each word placeholder $[m, n]$ is performed by sampling a particular topic $z_{m,n}$ from multinomial distribution $\mathrm{Mult}(\vec{\theta}_m)$. Finally, a particular word $w_{m,n}$ is generated for the word placeholder $[m, n]$ by sampling from multinomial distribution $\mathrm{Mult}(\vec{\varphi}_{z_{m,n}})$. The topics $\vec{\varphi}_k$ are sampled once for the entire corpus. K is the number of topics, M is the number of documents in the corpus, and N_m is the number of words in document m.

In the title generation process of our model, the topic distribution of a segment of text is used as a type of feature. The topics of words $\vec{z}_m$ inside that segment are used to com-

Table 1 Baseline features of the local model for capturing selection constraints at the word level and contextual constraints at the word sequence level.

Features	Type
Word level	
Is it a stop word or an auxiliary word?	binary
Its TF*IDF score	real
Its part-of-speech	binary
Its first occurrence in segment by word	real
Its first occurrence in segment by sentence	real
Does it occur in the sibling or the parent segments?	binary
Word sequence level	
Uni-gram, bi-gram and tri-gram language model scores	real
The frequency of noun phrases in the word sequence at segment level and corpus level	real

pute the topic distribution of every candidate title. The topic distribution information helps our model capture the topic relations between the segment of text and the candidate titles.

4. Features

The model is decomposed into two components, the local model and the global model. Therefore, the feature set is also divided into two subsets for use in the local model and the global model, respectively.

4.1 Local Features

The goal of the local model is to generate a list of candidate titles for a given segment of text. It involves indentifying interest words in the text, and combining them into the title [20]. Therefore, the feature set for this model should capture selection constraints at the word level, and the contextual constraints at the word sequence level. More specifically, the features at word level plays as a filter to select appropriate words to be included in the candidate titles. The features at word sequence level plays as a filter to select and rank the candidate titles via their fluency in language or the relevance between title and the segment of text.

The local features are summarized in Table 1. These features are used in the baseline models. As described in Table 1, some types of features are normalized. For instance, "*Its first occurrence in segment by sentence*" is the relative position of the first sentence containing the word, which is normalized by the number of sentences in the given segment.

4.2 Supportive Knowledge Features

The supportive knowledge is incorporated into the local model in the form of features.

For using the word clustering information, each cluster of words is represented by a bit string as described in Sect. 3.1. To exploit various levels of abstraction of the word clustering, we use corresponding prefixes of the bit string of a word as features. Then, an indicator function is created for each type of prefix and is used as a feature. In experiments, we use three levels of abstraction with three types of the prefix length 4, 6, and 8, respectively. For instance, a indicator function f^4_{0110} can be defined as follows:

$$f^4_{0110}(w) = \begin{cases} 1 & \text{if the 4-bits-prefix of } w \text{ is 0110,} \\ 0 & \text{otherwise.} \end{cases}$$

For the word "language" with bit string "10110111100", the following indicator functions are activated: f^4_{1011}, f^6_{101101}, and $f^8_{10110111}$. With this representation method, we can limit the number of word clustering features regardless of the cluster size.

To exploit topic information, we use it at both word level and word sequence level. At the word level, the topic distribution information of a word is used. For instance, if the vector of topic counts of a word w_i in a given segment of text is $\vec{z}_i = (|z^1_i|, |z^2_i|, \ldots, |z^K_i|)$, we normalize that vector by the number of occurrences of w_i in that segment. The normalized vector is directly incorporated into the feature set as a selection feature.

At the word sequence level, for each partial title at each iteration of the training and decoding algorithms, the topic distribution is easily computed by normalizing the sum of the vectors of topic counts of all the words in that partial title by the total number of occurrences of all the words in the given segment s. For instance, topic distribution of a partial title $t = w_1 w_2 \ldots w_l$ is computed as:

$$p(z^i_t|t, s) = \frac{\sum^l_{j=1} |z^i_j|}{\sum^l_{j=1} |w^i_j|}, \quad i = 1 \ldots K$$

To take into account the relevance between the partial title and the segment of text, we measure the similarity between the topic distribution of the partial title p_t and the topic distribution of the segment of text p_s:

$$\mathrm{sim}(p_t, p_s) = 10^{-\beta \mathrm{IRad}(p_t, p_s)}$$

where β is a scale parameter, which is normally set to 1 in practice, and $\mathrm{IRad}(p, q)$ is the information radius between two distribution p and q [21]. $\mathrm{IRad}(p, q)$, in turn, is defined via Kullback–Leibler divergence $\mathrm{KL}(p, q)$ as follows:

$$\mathrm{IRad}(p, q) = \mathrm{KL}\left(p \middle\| \frac{p+q}{2}\right) + \mathrm{KL}\left(q \middle\| \frac{p+q}{2}\right)$$

where

$$\mathrm{KL}(p\|q) = \sum_i p_i \log \frac{p_i}{q_i}$$

The above similarity score is incorporated into the feature set as a contextual feature.

4.3 Global Features

The goal of the global model is to make a coherent table-of-contents by choosing the most appropriate title for each node in the tree of titles. This model has to account for the relations between titles in a hierarchical structure. Given a

partial tree of titles and a candidate title of the next node in the pre-order traversing, three types of features are used to capture that relation:

1. Whether the title is redundant at various levels of the tree: at the sibling nodes, at the parent node.
2. The rank of the title provided by the local model via its score. With this feature, the global model can exploit the preferences of the local model in the title generation process.
3. The parallel structure of titles that have the same parent node. This phenomena is popular in a table-of-contents. For instance, in the dataset used in this research, a section titled "*Performance of Quicksort*" has three subsections titled "*Worst case partitioning*", "*Best case partitioning*", and "*Balanced partitioning*".

5. Experiments

5.1 Data

In experiments, we use a public dataset [†] for training and testing the model [2]. This dataset is, actually, the table-of-contents of the textbook "*Introduction to Algorithms*" [22]. This book contains 564 sections in 39 chapters. The depth of the table-of-contents is 4. The authors of this dataset treated the fragment of table-of-contents of each chapter as a small table-of-contents with depth of 3. Thereby, we have 39 table-of-contents used for training and testing. We divided this dataset into a development set and a test set at a ratio of 80/20. For tuning the parameters of the model, we divided the development set into a training set and a development test set (dev-test set) for training and testing the model before application to the test set. The ratio was also 80/20. Similar to [2], in our experiments, we use ten different randomizations to compensate for the small number of available trees. For each randomization, we have done a 5-fold cross-validation to get the average score.

At the preprocessing step, the dataset is tokenized and tagged by Stanford Log-linear Part-Of-Speech Tagger [††]. The noun phrases are extracted using regular expressions chunking tool in NLTK [†††]. SRILM Toolkit [††††] is used to train the language model on the training set.

To build a hierarchical cluster of words, we use an external corpus, the BLLIP corpus [23], which is a collection of raw text with approximately 30 million words. The Brown algorithm implementation of Liang [†††††] [13] ran on that corpus to produce 1,000 word clusters. Some clusters are shown in Table 2. The number 1,000 is the default setting for large corpora and has been widely used in other research [12]–[14]. On the other hand, as described in Sect. 4.2, we only use the prefixes of the bit string representation of the word cluster. Therefore, the number of cluster features is limited regardless of the size of word clustering.

To estimate the topic model, we use the raw data in the training set, in which each section is treated as an independent document. After that, we infer the topic distribution for every section in the training set and the test set. Only the inferred topic information of the training set is used in the learning process as described in Sect. 4. To estimate and infer the topic model, we use the LDA implementation of Blei [††††††] [11]. In each run, we re-estimate the topic model with 100 topics on the training set. Some topics with their most likely words are shown in Table 3. To choose the number of topics, we manually tuned on a development set. Similar to the results in [24], the performance of the model becomes stable when we increase the number of topics, and 100 is a relatively *good* number of topics.

Table 2 Sample word clusters derived from BLLIP corpus and their bit strings.

101010010010	101010010000	101101101011110
sorting	construction	review
partitioning	restoration	journal
labelling	conversion	selection
metering	implementation	survey
clustering	execution	encyclopedia
formatting	installation	encyclopaedia
tunnelling	renovation	timeline
...	...	...

Table 3 Most likely words of some sample topics.

Topic 1	Topic 5	Topic 23	Topic 81
circuit	vertex	tree	section
input	search	minimum	chapter
output	edge	spanning	present
combinational	vertices	edge	method
element	directed	algorithm	application
gate	breadth	weight	basic
boolean	discovered	set	finally
figure	white	edges	material
gates	edges	prim	practical
clock	gray	safe	based
wire	reachable	trees	examine
register	source	section	discusses
...	...	...	...

5.2 Evaluation

The baseline models used in evaluation are the flat discriminative model (Baseline FD) and the hierarchical discriminative model (Baseline HD), which are the best models in [2].

The difference between the flat discriminative (FD) models and the hierarchical discriminative (HD) models is that the hierarchical discriminative models account for global features, which capture the relations between titles in the hierarchical structure. The flat discriminative model omits those relations and simply chooses the highest ranked title from the local model to form the title of a node in the table-of-contents.

We compare the quality of the baseline model to our

[†] http://people.csail.mit.edu/branavan/
[††] http://nlp.stanford.edu/software/tagger.shtml
[†††] http://www.nltk.org
[††††] http://www.speech.sri.com/projects/srilm/
[†††††] http://www.eecs.berkeley.edu/~pliang/software/
[††††††] http://www.cs.princeton.edu/~blei/lda-c/

Reference	Baseline	Our model
hash tables	many dictionaries	dictionary operations
direct address tables	direct address dictionary	direct address table
hash tables	computer address	hash function
collision resolution by chaining	chaining a same slot	chaining all the elements
analysis of hashing with chaining	creating an same chaining hash	hash table with load factor
open addressing	address hash	address hash
linear probing	hash probing	hash function
quadratic probing	quadratic hash	quadratic probing
double hashing	double hash	double hashing

Fig. 6 Fragments of the reference table-of-contents, along with baseline generated table-of-contents and our generated table-of-contents.

Table 4 Results of experiments on public dataset.

	Rouge-1	Rouge-L	Rouge-W	Match
Baseline FD	0.235	0.215	0.169	10.35
Baseline HD	0.246	0.226	0.178	11.75
FD+WC	0.252	0.231	0.182	10.60
HD+WC	0.301	0.290	0.229	12.80
FD+TM	0.302	0.290	0.252	13.40
HD+TM	0.322	0.327	0.269	13.60

four models. The first model denoted by *FD+WC* is a flat discriminative model, which uses the local feature set and word clustering based features. The second model denoted by *FD+TM* is a flat discriminative model, which uses the local feature set and topic model features. The third model denoted by *HD+WC* is a hierarchical discriminative model, which is based on the FD+WC model with the global features set. The last model, denoted by *HD+TM*, is a hierarchical discriminative model, which is based on the *FD+TM* model with global feature set.

The experimental results are evaluated using ROUGE metrics [25], which is commonly used to assess the quality of machine-generated headlines [26]. All the scores are averaged over ten randomizations of the dataset. Table 4 shows the experimental results with three scores ROUGE-1, ROUGE-L, ROUGE-W, and the number of matched titles, which is the number of generated titles having the same word sequence as original titles.

5.3 Discussion

Table 4 indicates that HD models achieve higher quality than FD models. The reason is that HD models use the global model that captures some useful information about candidate titles, such as the rank of the most matched title generated by the local model, the relations between titles in the same tree (duplication, parallel structure). It is difficult for the local model to take into account that information. On the other hand, when the supportive knowledge is used, the candidate titles are much more relevant to the content of the segment of text. Specifically, 10-best candidate titles contain about 50% matched titles, and 5-best candidate titles contain about 20% matched titles. This means the global model has bigger chance of choosing *good* title. Therefore, the HD models generally have higher quality than the FD models.

As described in Sect. 3, one of the advantages of word clustering is the prediction the next word, which is naturally appropriate to the title generation mechanism used in this research. The experimental results and logs show that it has good effects on choosing words for the candidate titles. Figure 6 shows a fragment of a table-of-contents generated by the model HD+WC, in comparison to the same fragment generated by the baseline model, and the reference fragment. In that fragment, HD+WC chose the word "*dictionary*" followed by "*operations*" to make a title for the segment describing the hash table rather than "*many dictionaries*" by the baseline model. Another advantage of word clustering is the various levels of abstraction of words, which could help the model to reduce the effects of data sparsity. Table 4 shows that FD+WC model can achieve higher quality than the baseline model without capturing the relations between titles. On the other hand, word clustering is similar to the part-of-speech in terms of grouping words by functionality. Furthermore, word clustering can group words by the category or topic. Some examples are shown in Table 2. These characteristics of the word clustering affects the title generation model in a similar way as the part-of-speech does. In other words, some clusters have the higher impact than others. For instance, the 6-bits-prefix cluster "100100" containing "introduction", "conclusion", "overview"... has a higher weight score than others, because they tend to be occurred in the title more frequently than other words. That is one of the advantages of our approach in comparison to the baseline method. However, it also negatively affects the ranking of candidate titles by promote titles containing common words, especially when the desired length of title is short or the title locate at high level in the table-of-contents. For instance, instead of choosing "recurrences" or some related words as the title of the chapter "Recurrences", the model chose "introduction".

The topic modeling is even better in support of the table-of-contents generation process. Table 4 shows the improvement of 0.083 averaged ROUGE-L score in comparison to the baseline model. This is the effect of topic-based similarity score between the title and the given segment of text. By using this similarity score as a feature in a linear learning model, the model could put a title in a higher position if the topic distribution of that title is closer to the topic distribution of the given segment than the others. For instance, in experiments, our model ranked the candidate title

Table 5 Results of experiments that remove some type of feature.

Remove	Rouge-1	Rouge-L	Rouge-W	Match
Lang. model	0.167	0.143	0.106	1.10
Position	0.173	0.156	0.118	6.10
TF-IDF	0.203	0.185	0.152	8.35
Sibling	0.219	0.200	0.156	9.10
POS Tag	0.220	0.202	0.158	9.15
NP Freq.	0.232	0.212	0.166	10.00
None	0.235	0.215	0.169	10.35

"*computing the minimum cost*" (score: −4.39, position: 1^{st}) in a higher position than the other "*computing the matrix product*" (score: −7.69, position: 4^{th}) in a segment with the reference title "*computing the optimal costs*". The reason is that "*matrix product*" is only an example of computing the optimal costs, and therefore, the candidate title containing that phrase is *further* than the first one in terms of topic. However, the use of topic modeling is also noise sensitive. For instance, in a segment of text discusses on a topic with an example of another topic at the end, titles that are close to the main topic usually have low ranks in comparison to general meaning titles. For example, in segment titled "longest common subsequence", the 2-best candidate titles are "representing the problem" and "the dna strands".

To compare the impacts of word clustering and topic modeling on the title generation task with the other feature types in Table 1, we did some additional experiments, in which we removed some type of feature from the baseline model. Table 5 shows that the most important feature is the language model score with −0.072 point of ROUGE-L score. At the word level, the position of a word and its TF-IDF are very important. The impact of POS tag is approximate the impact of the word clustering in Table 4, which is suitable to our analysis on the use of word clustering. The impact of frequency of noun phrases is very small and not significant. Table 4 and Table 5 also show the high impact of topic modeling on the title generation task with +0.075 point of ROUGE-L score.

6. Conclusion

In this paper, we proposed a supportive knowledge approach for supporting the table-of-contents learning model. We also proposed a method of integrating supportive knowledge into the learning model to help the model capture semantic and topic information at both the word level and the word sequence level. The supportive knowledge used in this research is derived from the word clustering, which is acquired from a large collection of raw text by an unsupervised learning algorithm. Another type of supportive knowledge is derived from a topic model, which is directly estimated from the training dataset. We performed experiments on a public dataset and obtained relatively good results in comparison to the current state-of-the-art model.

Our approach is general enough to be applied to other tasks in text summarization that need semantic or topic information, such as summary generation, sentence extraction, or sentence compression.

Acknowledgements

We would like to thank anonymous reviewers for your useful and constructive comments on the manuscript.

This study was supported by the Graduate Research Program in Computer Science, Japan Advanced Institute of Science and Technology.

References

[1] J. Otterbacher, D. Radev, and O. Kareem, "Hierarchical summarization for delivering information to mobile devices," Inf. Process. Manage., vol.44, no.2, pp.931–947, 2008.
[2] S.R.K. Branavan, P. Deshpande, and R. Barzilay, "Generating a table-of-contents," Proc. ACL 2007, pp.544–551, Prague, Czech Republic, 2007.
[3] M.A. Hearst, "Texttiling: Segmenting text into multiparagraph subtopic passages," Computational Linguistics, vol.23, no.1, pp.33–64, 1997.
[4] F.Y.Y. Choi, "Advances in domain independent linear text segmentation," Proc. NAACL '00, pp.26–33, Seattle, USA, 2000.
[5] I. Malioutov and R. Barzilay, "Minimum cut model for spoken lecture segmentation," Proc. CICLing and ACL 2006, pp.25–32, Sydney, Australia, 2006.
[6] E. Chen, B. Snyder, and R. Barzilay, "Incremental text structuring with online hierarchical ranking," Proc. EMNLP 2007, pp.83–91, Prague, Czech, 2007.
[7] D. Radev, "A common theory of information fusion from multiple text sources step one: Cross-document structure," Proc. ACL SIGDIAL Workshop on Discourse and Dialogue, pp.74–83, Hong Kong, 2000.
[8] R. Angheluta, R.D. Busser, and M.F. Moens, "The use of topic segmentation for automatic summarization," Proc. Workshop on Text Summarization in Conjunction with the ACL 2002, pp.11–12, Philadelphia, Pennsylvania, USA, 2002.
[9] V.C. Nguyen, L.M. Nguyen, and A. Shimazu, "A semisupervised approach for generating a table-of-contents," Proc. RANLP 2009, pp.313–318, Borovets, Bulgaria, 2009.
[10] P.F. Brown, P.V. Desouza, R.L. Mercer, and J.C. Lai, "Class-based n-gram models of natural language," Computational Linguistics, vol.18, no.4, pp.467–479, 1992.
[11] D.M. Blei, A.Y. Ng, and M.I. Jordan, "Latent dirichlet allocation," Machine Learning Research, vol.3, pp.993–1022, 2003.
[12] S. Miller, J. Guinness, and A. Zamanian, "Name tagging with word clusters and discriminative training," Proc. HLT-NAACL 2004, pp.337–342, Boston, Massachusetts, USA, 2004.
[13] P. Liang and M. Collins, Semi-supervised learning for natural language, Master's Thesis, MIT, 2005.
[14] T. Koo, X. Carreras, and M. Collins, "Simple semisupervised dependency parsing," Proc. ACL08: HLT, pp.595–603, Columbus, Ohio, USA, 2008.
[15] K.S. Jones, "Automatic summarising: The state of the art," Inf. Process. Manage., vol.43, no.6, pp.1449–1481, 2007.
[16] H. Daumé III and D. Marcu, "Learning as search optimization: Approximate large margin methods for structured prediction," Proc. ICML 2005, pp.169–176, Bonn, Germany, 2005.
[17] M. Collins and B. Roark, "Incremental parsing with the perceptron algorithm," Proc. ACL '04, Main Volume, pp.111–118, Barcelona, Spain, 2004.
[18] E. Charniak, Statistical Language Learning, The MIT Press, 1996.
[19] T.L. Griffiths and M. Steyvers, "Finding scientific topics," Proc. National Academy of Sciences, vol.101, no.Suppl. 1, pp.5228–5235, April 2004.

[20] M. Banko, V.O. Mittal, and M.J. Witbrock, "Headline generation based on statistical translation," Proc. ACL 2000, pp.318–325, Hong Kong, 2000.
[21] C.D. Manning and H. Schütze, Foundations of Statistical Natural Language Processing, MIT Press, 1999.
[22] T.H. Cormen, C.E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms, second ed., The MIT Press, Cambridge, Massachusetts, 2001.
[23] E. Charniak, D. Blaheta, N. Ge, K. Hall, J. Hale, and M. Johnson, BLLIP 1987-89 WSJ Corpus Release 1, Linguistic Data Consortium, 2000.
[24] H.M. Wallach, D. Mimno, and A. McCallum, "Rethinking lda: Why priors matter," Proc. Wordshop on Topic Models: Text and Beyond in Neural Information Processing Systems Conference (NIPS), pp.1973–1981, Vancouver, B.C., Canada, 2009.
[25] C.Y. Lin, "Rouge: A package for automatic evaluation of summaries," Proc. WAS 2004, pp.25–26, Barcelona, Spain, 2004.
[26] R. Wang, J. Dunnion, and J. Carthy, "Machine learning approach to augmenting news headline generation," Proc. IJCNLP 2005, pp.155–160, Korea, 2005.

Viet Cuong Nguyen received his B.S. and M.S. degrees in Information Technology from College of Technology, Vietnam National University, Hanoi in 2005 and 2007, respectively. Since January 2008, he has been a PhD student in Natural Language Processing Laboratory, School of Information Science, Japan Advanced Institute of Science and Technology (JAIST).

Le Minh Nguyen has been an assistant professor in Natural Language Processing Laboratory, Japan Advanced Institute of Science and Technology (JAIST) since 2008. He received his B.S. degree and M.S. degree in Information Technology from Vietnam National University, Hanoi in 1998 and 2001, respectively. He received the Ph.D. degree in Computer Science in 2004 from JAIST.

Akira Shimazu has been a professor in the School of Information Science, Japan Advanced Institute of Science and Technology (JAIST) since 1997. He received his B.S. and M.S. degrees in Mathematics from Kyushu University in 1971 and 1973, respectively. He received the Ph.D. degree in Natural Language Processing from Kyushu University in 1991. From 1973 to 1997, he worked at Musashino Electrical Communication Laboratories and Basic Research Laboratories of NTT Corporation. From 2002 to 2004, he was the president of the Association for Natural Language Processing, Japan.