A Semi-supervised Approach for Generating a Table-of-Contents

Viet Cuong Nguyen, Le Minh Nguyen, and Akira Shimazu
School of Information Science
Japan Advanced Institute of Science and Technology
1-1 Asahidai, Nomi, Ishikawa 923-1211, Japan
{cuongnv,nguyenml,shimazu}@jaist.ac.jp

Abstract

This paper presents a semi-supervised model for generating a table-of-contents as an indicative summarization. We mainly focus on using word cluster-based information derived from a large amount of unannotated data by an unsupervised algorithm. We integrate word cluster-based features into a discriminative structured learning model, and show that our approach not only increases the quality of the resulting table-of-contents, but also reduces the number of iterations in the training process. In the experiments, our model shows better results than the baseline model in generating a table-of-contents, about 6.5% improvement in terms of averaged ROUGE-L score.

Keywords

text generation, text summarization, semi-supervised learning

1 Introduction

In our research, to help people quickly having an overview of a topic or an event, we automatically collect articles from online newspapers related to that topic or event, and summarize them. We should be able to make a summarized document from those articles by extracting important sentences. However, there is a lot of articles with many useful information which makes the document too long as well as does not contain the deep comments of the authors of those articles about the topic or event. Our aim is to make a concise summary in the form of table-of-contents, automatically.

A table-of-contents is a hierarchical structure of titles and locations of segments in a very long text such as books or a set of texts such as multiple documents which describe the same topic [2]. It is a type of indicative summarization that is especially suited for locating and accessing information. With a table-of-contents as a navigation tool, a reader can quickly get not only an overview of the content of a very long text or multiple documents via the titles of segments, but also the location of needed information via the locations of segments.

The task of automatically generating a table-of-contents involves two subtasks: (1) separating every article into a hierarchical structure of segments (a tree of segments) and merge them to make a unique tree of segments [6, 11, 12, 15] and (2) generating a title for each segment in that tree to make a table-of-contents [1, 2]. In this paper, we focus on subtask two with the assumption that the tree of segments are easily got from existing methods such as *C99* [6] or *TextTiling* [11] with a text structuring method [5].

The subtask two is previously mentioned in [1, 2]. In [1], for each segment, they used the most important noun phrase based on its frequency to make the title of that segment. This method made the title too short and having very low quality. In [2], they made a better table-of-contents with a supervised learning method which accounts for a number of features at word level and word sequence level. However, in their experiments, a large amount of titles in the result table-of-contents were not related to the content of the corresponding segments. The lack of semantic information might be a reason. Moreover, their model required a large number of iterations in the training process.

In this paper, we propose a model that tries to integrate semantic information into the learning model which is based on [2]. With the support of semantic information, the new model could make the meaningful titles with strong relation to the content of the corresponding segments. Another motivation of our approach is to reduce the number of iterations in the training process [13, 16].

Our learning model is a two-stage semi-supervised learning model [16]. The semantic information used in our model is derived from word clusters which are, in turn, built from a large unannotated data by an unsupervised learning algorithm, the Brown algorithm [3]. After that, we use those word cluster-based information in a supervised learning model. The key of our approach is the way of integrating the word clustering information into the learning model. We encode word cluster-based information as features in a discriminative learning model. The learning algorithm used in this research is based on the Perceptron learning algorithm because of its simplicity, powerful and high speed. Especially, it is appropriate for structured learning tasks.

Our experiments show that, by incorporating word clustering information and using the same corpus, our model not only produces a better table-of-contents than baseline model, but also reduces the number of iterations in the training process. Our model even generated titles which is the same with original titles in the test data.

The remainder of this paper is structured as follows: Section 2 presents our approach on using a

International Conference RANLP 2009 - Borovets, Bulgaria, pages 312–317

structured learning model for generating a table-of-contents. Section 3 describes word clustering and the reason of using the word clustering information to support a supervised learning model. Section 4 designs features for the learning model and the way of incorporating word clustering information in our model. Section 5 presents our experimental results with discussion about the results. Section 6 gives some conclusions and future works.

2 The learning model

In this section, we mainly focus on the subtask of generating a table-of-contents from a tree of segments. In this task, automatically generating a table-of-contents can be seen as a structured learning task, in which the trained model produces a tree of titles $\mathcal{T}$ as the output from a tree of segments $\mathcal{S}$ as the input. $\mathcal{T}$ contains $\mathcal{T}_i$ that is the title of a segment $\mathcal{S}_i$ in $\mathcal{S}$.

We formulate this task as a two-steps structured learning. First, our algorithm learns a model for generating a title $\mathcal{T}_i$ from a segment $\mathcal{S}_i$. Second, our algorithm learns another model for generating a tree of titles $\mathcal{T}$ from a tree of segments $\mathcal{S}$.

A trivial algorithm for the second model is that we construct $\mathcal{T}$ from all $\mathcal{T}_i$ generated by the first model, separately. However, due to our experiments, this trivial algorithm produces a $\mathcal{T}$ with many duplicating titles at the same level in its hierarchical structure and this makes reader have confused. Therefore, we must use another learning algorithm to construct $\mathcal{T}$ from $\mathcal{T}_i$ which makes $\mathcal{T}$ more coherent, for example, without duplicating titles. In the Section 5, we will show the experimental results of the both algorithms.

To keep the learning process as simple as possible and to make the model easier in incorporating new features, we use an instance of the discriminative structured learning algorithm with the *LaSO* technique [10]. Due to this technique, the process of learning for generating a table of contents is as follow:

- First, for each text segment $\mathcal{S}_i$ in the tree of segments, our model learns to generate a list of candidate titles which are ranked by the scores which are the products of the weight vector of the model and the feature vectors of the $\mathcal{S}_i$ and the candidate titles.

- Second, our model learns to generate a table-of-contents from the candidate titles of segments which are produced by the above step. The scores are also computed by using another weight vector.

In this paper, the model used for the first learning step is called the local model and the second one is called the global model. The details of learning algorithm and decoding algorithm of the above two models are described in next sections.

2.1 The local model

In the learning step of the local model, a vine-growth strategy [10] is used to learn a model for generating a title for a text segment. The learning process simulates the process of building a title $\mathcal{T}_i$ incrementally from words inside a segment $\mathcal{S}_i$ as in Algorithm 1. To reduce the size of searching space, it maintains a beam $\mathcal{B}$ which contains partial titles. This strategy has been successfully applied in other tasks such as parsing [7], chunking [10], and machine translation [9].

Algorithm 1 Training algorithm for the local model

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 → N do
2:    foreach (s, t) ∈ D do
3:      for j = 1 → |t| do
4:        B = GetTop(PartialGen(s, B),k)
5:        if t[1..j] ∉ B then
6:          w_l = w_l + f(s, t[1..j]) − Σ_{z∈B} f(s, z)
7:          B = {t[1..j]}
8:        end if
9:      end for
10:   end for
11: end for
12: w_l = w_l/(N * |D|)
```

In Algorithm 1, $\mathcal{D}$ contains a list of *(segment, title)* pairs (s,t) which is provided as the training data of the learning process. N is the number of iterations of the perceptron algorithm. At the line 4, $\mathcal{B}$ is a beam of partial titles. By using *PartialGen*, $\mathcal{B}$ is grown by appending every word in s into every title in $\mathcal{B}$ to make a list of titles of length j. After that, by using *GetTop*, $\mathcal{B}$ is pruned to contain top k ranked titles based on the scores $w_l \cdot f(s,z)$, $\forall z \in \mathcal{B}$. w_l is a weight vector of the perceptron model and the f is a function which produces a feature vector of a segment s and a partial title $t[1..j]$ which is a prefix of the title t with length of j words.

In the decoding step of the local model, Algorithm 2 will produce a list of candidate titles by incrementally generating titles from the words in the text segment. It uses a same strategy, beam search, in Algorithm 1 to reduce the size of searching space.

In Algorithm 2, s is the sequence of words w_i in the text segment, l is length of desired title of that segment. $\mathcal{B}$ is a beam containing partial titles which is similar to $\mathcal{B}$ in Algorithm 1. $\mathcal{Q}$ is a sorted list of the titles made by appending every word s_i into every title z_i of $\mathcal{B}$. The output of this algorithm is a list of k candidate titles that are the input of the global model.

2.2 The global model

In the learning step of the global model, the input is a tree of candidate titles, in which each node contains a list of k candidate titles, and the output is a tree of titles, in which each node contains a title. The input and the output of this model are hierarchical structure (a tree) which are different from the local model, a flat structure of words (a title).

Algorithm 2 Generating a list of candidate titles for a text segment

Input:
- s is the sequence of words in the text segment
- l is the length of desired title
- k is the beam size

Output:
- A list of k candidate titles.

```
1: B = a set that contains an empty string
2: for i = 1 to l do
3:    foreach w_i ∈ s do
4:       Q = {}
5:       foreach z_i ∈ B do
6:          Q = Q + {(z_i + w_i, w_l * f(s, z_i))}
7:       end for
8:       B ={top k partial titles of Q}
9:    end for
10: end for
```

In this model, we can still use a learning algorithm which is similar to the one used in the local model by traversing the tree of titles in pre-order. With this technique, we can also incrementally build the tree of titles.

In the training process, at each node of the tree in the traversing process, our algorithm create a beam $\mathcal{B}$ containing a list of partial trees ranked by the scores which are similarly computed as in the local model. Because the global model use those candidate titles returned by the local model, therefore, at some nodes, the true title may not be among the candidate titles. In this cases, our algorithm chooses the best title in those candidate titles which is closest to the title of the corresponding segment in the training data by using ROUGE-1 score[1]. The output of this learning step is a weight vector w_g.

In the decoding step, we use the same strategy as in the local model. We incrementally generate the tree of titles by traversing the tree in pre-order with beam search strategy. However, the output of this step is a tree of titles which is called a table-of-contents.

3 Word Clustering

In this research, we use word clustering information to make the learning model take into account semantic information, in terms of the word similarity. With this information, we can choose better words which are semantically related for generating titles of segments in the table-of-contents. For example, normally, "tree" is no more similar to "graph" than "plant". However, by using word clusters derived from a large amount of text in computer science, "graph" and "tree" may have semantic relations. This approach is successful on other natural language processing tasks such as name-entity recognition [16] and dependency parsing [13].

To get the word clusters, we use the Brown algorithm described in [3]. This is a bottom-up agglomerative clustering algorithm which is used to produce a hierarchical cluster of words. The input of this algorithm is a large sequence of words and the output is a binary tree, in which leaves of the tree are words. Every word in this tree is uniquely identified by a path from a root. This path is encoded by a binary string, where 0 indicates a left branch and 1 indicates a right branch. For example, in Figure 1, "tree" is encoded by "001", "node" is encoded by "011" and so on.

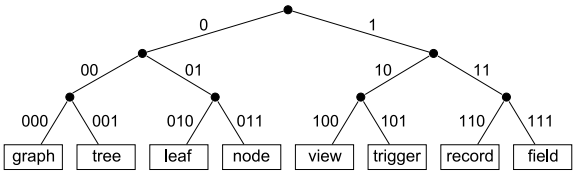

Fig. 1: *An example of a Brown word cluster tree. Each word at the leaf is encoded by a binary string with respect to the path from the root, where 0 indicates a left branch and 1 indicates a right branch*

In the word cluster tree, by selecting an inner node of the tree, we have a set of leaves of the corresponding subtree. Therefore, we have a set of words semantically related which forms a word cluster. This cluster is also identified by a path from the root to the chosen inner node. For example, in Figure 1, we might select three clusters {graph, tree} encoded by "00", {leaf, node} encoded by "01" and {view, trigger, record, field} encoded by "1".

4 Feature Design

The features are divided into two sets. The first set is used for the local model. It contains the features that capture selection constrains at word level and contextual constrains at word sequence level. The selection features in the model captures word information of a text segment, which are:

- The position of the word;
- The TF*IDF value of the word;
- The part-of-speech tag of the word;
- Whether the parent segment contains that word and its position;
- Whether the sibling segments contain that word and its position;
- The word cluster information of the word.

To use the word cluster-based information, we encoded each word cluster in a unique code with respect to the binary string described in previous section. A word cluster-based feature is an indicative function which has a value 1 if the current word is in the corresponding cluster and a value 0 otherwise. It is an interesting point of our approach.

The contextual features record bi-gram and tri-gram language model scores, both for words and POS tags. To eliminate generic phrases from the generated titles, such as "*the following section*", it also captures

[1] ROUGE-1 computes the number of overlapping words of two word sequences.

```
111011011101    microcomputer
111011011101    peripheral
111011011101    minicomputer
111011011101    magnetic
111011011101    PC
111011011101    computer
.........
01101010    Dijkstra
01101010    Clanton
01101010    Rooney
01101010    Nakagama
01101010    Shannon
.........
011001011010    Japan
011001011010    Colombia
011001011010    Austria
011001011010    Russia
011001011010    Brazil
.........
```

Fig. 2: *Examples of word clusters derived from BLLIP corpus*

the collocational properties of noun phrases in the title.

In the global model, to avoid duplicating titles of the segments in the same section and to make them more coherent, we use some types of features that describe interaction between different titles. The first type is for title redundancy that includes title duplication and title similarity. The second type is for capturing parallel construction of titles of segments in the same section. For example, in the section describing sorting algorithms, we may see some titles of subsections with the same prefix such as "*Bubble sort algorithm*" and "*Quick sort algorithm*". The last type of features is to take into account the process of selecting the best title in the list of candidate titles at every node of the table-of-contents. It helps to choose a better title in the list of ranked titles produced by the local model.

5 Experiments

5.1 Data

In our experiments, we used two dataset. The first dataset is a large sequence of words used for Brown algorithm to derive word clusters. The second dataset is a table-of-contents readily in a books used for learning model.

To build word clusters, we used the BLLIP corpus [4], which contains a collection of three-year Wall Street Journal (WSJ) from the ACL/DCI corpus with approximately 30 million words. All the text is cleaned, separated into sentences and joined into a large text file. The Brown algorithm ran on that cleaned text to produce 1000 clusters. Figure 2 shows some result word clusters with their code (binary string) and some words in those clusters. In these word clusters, the words related to PC hardware are grouped into a cluster, the countries' name are grouped into a cluster and so on.

We used the content and the table-of-contents of the textbook "*Introduction to Algorithm*" [8] as the corpus for the learning model which is the same as [2]. However, we used Charniak's tool[2] to get part-of-speech information for words in the book.

We considered the table-of-contents of the above book as a collection of smaller table-of-contents. In this book, parts are at level 1, chapters are at level 2, sections are at level 3 and so on. Level 1 is removed to make a set of trees of titles which have root at level 2 of the original tree. With this technique, we had 39 trees and 540 titles to used as training and testing data. The average depth of trees is 4.

We randomly choose 80% of these trees for training and the rest for testing. In our experiments, we choose ten different randomizations and get the average score to make the experiments fairly.

5.2 Evaluation

In our experiments, we used ROUGE scores[3] with the default settings as the measure of performance. It contains ROUGE-1, ROUGE-L and ROUGE-W used to compute similarity score between candidate title and original title [14].

- ROUGE-1 is based on 1-gram overlapping between two titles.
- ROUGE-L is based on longest common subsequence of two titles.
- ROUGE-W is based on weighted longest common subsequence of two titles.

We did two type of experiments. In the first type, the models generated table-of-contents without hierarchical constrains described in Section 2. And in the second type, the models used hierarchical constrains to generate table-of-contents.

In both two type of experiments, we did three experiments. In the first experiment, the baseline model was trained and tested with the best configuration which is published on the web [2]. The local model was trained in 50 iterations and the global model was trained in 200 iterations. This experiment is denoted by BS. The beam size used in the local model and the global model were 50 and 250, respectively.

In the second experiment, we ran our model on the same configuration with the baseline model. This experiment is denoted by EX1.

We ran our model in a series of experiments with different configurations. The best result was reported in the third experiment. In this experiment, the local model was trained in 10 iterations and the global model was trained in 40 iterations. The size of beams used in the local model and the global model were the same to the baseline model. This experiment is denoted by EX2.

Every experiment was done with 10 randomizations. The experimental results were averaged based on three ROUGE scores: ROUGE-1, ROUGE-L and ROUGE-W.

[2] `http://www.cs.brown.edu/~ec/#software`
[3] `http://berouge.com`

Original:	**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. 3: *Fragments of the table-of-contents generated by our model and the baseline model*

	ROUGE-1	ROUGE-L	ROUGE-W
BS	0.235	0.215	0.169
EX1	0.236	0.216	0.169
	+0.001	+0.001	0.000
EX2	**0.292**	**0.281**	**0.222**
	+0.057	**+0.066**	**+0.053**

Table 1: *In these experiments, the features that captures hierarchical constrains were not used*

	ROUGE-1	ROUGE-L	ROUGE-W
BS	0.246	0.226	0.178
EX1	0.252	0.231	0.182
	+0.006	+0.005	+0.004
EX2	**0.301**	**0.290**	**0.229**
	+0.055	**+0.064**	**+0.076**

Table 2: *In these experiments, we used the features that help to avoid duplicate titles and make titles more coherent*

The results of the first type of experiments are shown on Table 1. Table 2 contains the results of the second type of experiments.

In Figure 3, we show a subtree of table-of-contents of the original table-of-contents and the table-of-contents generated by the baseline model to compare with the table-of-contents generated by our model. In EX2, the averaged number of exact match titles is 13.

5.3 Discussion

Our experiments show that word clustering information is useful in generating a table-of-contents task. It could be also useful in the title generation task in general. By using this information as features in a discriminative learning model, we can not only improve the quality of generated table-of-contents and reduce the number of iterations in training process.

In terms of quality, by using word clustering information, it reduces the sparsity of data, thereby, it makes our model to be better at setting the parameter values. Moreover, word cluster-based features help the model choose the words that are semantically related even those words did not occur in the training data. In our experiments, our model gets higher results than the baseline model, about 6.5%. A fragment of the generated table-of-contents in Figure 3 is an example of the better quality of our model. For example, "*dictionary operations*" is close to "*hash tables*" than "*many dictionaries*". It also has some generated titles that match with the original titles.

In terms of speed, with additional semantic information derived from word clustering, the model converges faster, therefore, the number of iterations are reduced in the training process, about 5 times in our experiments.

The word cluster in our experiments was derived from the BLLIP corpus that is a general corpus (multi-domains). Therefore, its could be used without regenerate word clusters. However, the training data, which is a book about algorithm, should be replaced by suitable data. For example, we can use Reuters corpus, which contains articles and their titles, to generate a table-of-contents for a set of news articles.

In this research, the Brown word clustering algorithm uses only correlation of words at the sentence level. However, to generate a title for a document, we need the correlation of words at the document level. This problem could be solved by using topic information by using topic modeling. This should be an extension of this research.

6 Conclusion

We presented a two-stage semi-supervised learning approach for generating a table-of-contents automatically. The main contribution is to use semantic information derived from word clusters in a discriminative learning model to generate titles which have strong relations to the corresponding segments. It helps our model not only makes a better table-of-contents but also reduces the number of iterations in training process. This method could be successfully applied in other NLP tasks which requires semantic information of the text, such as summarization, machine translation and so on.

References

[1] R. Angheluta, R. D. Busser, and M.-F. Moens. The use of topic segmentation for automatic summarization. In *In Workshop on Text Summarization in Conjunction with the ACL 2002 and including the DARPA/NIST sponsored DUC 2002*

Meeting on Text Summarization, pages 11–12, Philadelphia, Pennsylvania, USA, 2002.

[2] S. R. K. Branavan, P. Deshpande, and R. Barzilay. Generating a table-of-contents. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 544–551, Prague, Czech Republic, 2007.

[3] P. F. Brown, P. V. Desouza, R. L. Mercer, and J. C. Lai. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

[4] E. Charniak, D. Blaheta, N. Ge, K. Hall, J. Hale, and M. Johnson. *BLLIP 1987-89 WSJ Corpus Release 1.* Linguistic Data Consortium, 2000.

[5] E. Chen, B. Snyder, and R. Barzilay. Incremental text structuring with online hierarchical ranking. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 83–91, Prague, Czech, 2007.

[6] F. Y. Y. Choi. Advances in domain independent linear text segmentation. In *Proceedings of NAACL '00*, pages 26–33, Seattle, USA, 2000.

[7] M. Collins and B. Roark. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118, Barcelona, Spain, 2004.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* The MIT Press, Cambridge, Massachusetts, second edition, 2001.

[9] B. Cowan, I. Kučerová, and M. Collins. A discriminative model for tree-to-tree translation. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 232–241, Sydney, Australia, 2006.

[10] H. Daumé III and D. Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *International Conference on Machine Learning (ICML)*, Bonn, Germany, 2005.

[11] M. A. Hearst. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64, 1997.

[12] X. Ji and H. Zha. Domain-independent text segmentation using anisotropic diffusion and dynamic programming. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 322–329, Toronto, Canada, 2003.

[13] T. Koo, X. Carreras, and M. Collins. Simple semi-supervised dependency parsing. In *Proceedings of ACL-08: HLT*, pages 595–603, Columbus, Ohio, USA, 2008.

[14] C.-Y. Lin. Rouge: A package for automatic evaluation of summaries. In *Proceedings of the Workshop on Text Summarization Branches Out (WAS 2004)*, pages 25–26, Barcelona, Spain, 2004.

[15] I. Malioutov and R. Barzilay. Minimum cut model for spoken lecture segmentation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 25–32, Sydney, Australia, 2006.

[16] S. Miller, J. Guinness, and A. Zamanian. Name tagging with word clusters and discriminative training. In *HLT-NAACL 2004: Main Proceedings*, pages 337–342, Boston, Massachusetts, USA, 2004.