Automatic Text Generation via Text Extraction Based on Submodular

Lisi Ai, Na Li, Jianbing Zheng, and Ming Gao(✉)

School of Data Science and Engineering,
East China Normal University, Shanghai 200062, China
irisinsh@163.com, nali0606@foxmail.com, zhengjb@js.chinamobile.com,
mgao@dase.ecnu.edu.cn

Abstract. Automatic text generation is the generation of natural language texts by computer. It has many applications, including automatic report generation, online promotion, etc. However, the problem is still a challenged task due to the lack of readability and coherence even there are many existing works studied it. In this paper, we propose a two-phase algorithm, which consists of text cleanup and text extraction, to automatically generate text from multiple texts. In the first phase, we generate paragraphs based on the topic modeling and clustering analysis. In the second phase, we model the text extraction as a set covering problem after we find the keywords in terms of the scores of TF-IDF, and solve the problem via employing the tool of submodular. We conduct a set of experiments to evaluate our proposed method and experimental results demonstrate the effectiveness of our proposed method by comparing with some comparable baselines.

Keywords: Automatic text generation · Massive information · K-Means · Submodular

1 Introduction

Automatic text generation is the generation of natural language texts by computer, which is a hot topic in both academia and industry. It can be applied to intelligent Question-Answer System, news report and online promotion, etc.

There are many state-of-art methods in this area and most of them already worked in the practice. Although great achievements have been made in this field, for example, the Newsblaster system [1] is a successful system from Columbia university, which is a news tracking tool to summarize the important news every day, it is still a challenged task due to the some problems, such as readability, coherence, a high-level complexity and so on.

We propose an automatic text generation algorithm, which consists of two phases: text cleanup and text extraction. Inspired by Lin [3], we employ the tool of submodular to solve the set covering problem. In summary, the main contributions of this paper are threefold:

S. Song et al. (Eds.): APWeb-WAIM 2017 Workshops, LNCS 10612, pp. 237–246, 2017.
https://doi.org/10.1007/978-3-319-69781-9_23

- Based on topic model, we cluster documents into several clusters, which makes the generated text more readable and accurate.
- We model text extraction as a set covering problem which is solved by the tools of integer programming and submodular. Furthermore, we figure out the lower bound of performance of our proposed algorithm.
- To illustrate the promising results of our algorithm, we conduct extensive experiments on real data sets. Not only do experiments verify the feasibility of our method, but also it reveals our method outperforms the baselines.

Moreover, we describe the related work and preliminaries in Sect. 2. Section 3 and Sect. 4 show the problem formulation and algorithms. Experimental results are analysed in Sect. 5. Finally, we conclude the paper and discuss the future work in Sect. 6.

2 Related Work and Preliminaries

2.1 Related Work

Automatic text generation has been an important research field in the past few years. The methods of automatic text generation can be divided into four categories: text-to-text generation [10–17], meaning-to-text generation [5], data-to-text generation [6–8], image-to-text generation [9]. Text-to-text generation is the major research method and in this paper, we propose a text-to-text method to generate the new text.

Due to extracting the original sentences, text-to-text generation has a more stable semantic and grammatical structure. Li [10] proposed a method which generates new text by extracting sentences from original text, which has the information redundancy problem. Mihalcea and Tarau [16] proposed a sorting algorithm based on graphs, TextRank. They first build a graph associated with the text, where the graph vertices are representative for the units to be ranked. The sentences with high score are taken as the summary. In addition, submodular function is also used in text generation. Lin [3] proposed a method that given a set of objects $V = v_1, \ldots, v_n$ and a function $F : 2^V \rightarrow R$ which returns a real value for any subset $S \subseteq V$, this method is to find the subset of bounded size $|S| \leq k$ that maximizes the function, e.g., $argmax_{S \subseteq V} F(S)$. This method is similar to MMR [4], which compute $F(S)$ by sentence similarity. Although we also address the problem by submodular function, our approach is different from Lin [3]'s method since we compute $F(S)$ by set-covering ratio.

2.2 Preliminaries

A set function that satisfies the rule of diminishing marginal efficiency is called submodular function. For a set $V = \{v_1, v_2, v_3, \ldots, v_n\}$ and a function $f : 2^V \rightarrow R$. If $A \subseteq B \subseteq V$ and $e \in V - B$, then:

$$f(A \cup \{e\}) - f(A) \geq f(B \cup \{e\}) - f(B) \tag{1}$$

For a submodular function f and a limit condition C, we need to find the set S which satisfies the limit of C and can maximize the value of $f(S)$. The most basic greedy algorithm [18] for this NP hard is to add the largest incremental value which satisfies the conditional C at each iteration, so in i-th iteration:

$$S_i = S_{i-1} \cup \{argmax_e \Delta(e \mid S_{i-1})\} \tag{2}$$

In this equation, $\Delta(e \mid S_{i-1}) = f(S_{i-1} \cup \{e\}) - f(S_{i-1})$, where $f(S)$ satisfies the property of submodular. Obviously, a submodular function is monotonous and non-negative, i.e., for $\forall e \notin S, f(S \cup \{e\}) \geq f(S)$ or $\forall e \notin S, f(S \cup \{e\}) \leq f(S)$ and $\forall S \subseteq V, f(S) \geq 0$. The lower bound performance of this algorithm is 63.21% of optimal solution. From the property of submodular, we can define coverage function as weights of sentences so that automatic text generation satisfies the property of submodular and can be solved by this algortihm.

3 Problem Setup

3.1 Problem Formulation

Assume we have document cluster $D = \{d_1, d_2, \ldots, d_n\}$ and keyword set K. The goal is to find the sentence set $S = \{s_i \mid s_i \in d_j, d_j \in D\}$, which S can cover all keywords from K. Considering the applicability and effectiveness, we choose text extraction method. We model text extraction as a set covering problem, the definition as follows:

Definition 1 ***(Text Extraction).*** *Given keyword set $V = \{w_1, w_2, \ldots, w_{n1}\}$, the customized keyword set $U = \{u_1, u_2, \ldots, u_{n2}\}$ and sentence set $C = \{S_1, S_2, \ldots, S_m\}$. Assume that $S_j = \{w_k | w_k \in U \cup V\}$, the purpose is to find the minimal subset $C^{'}$ that $V \in U_{S \in C^{'}} S$ and $\exists u_i \in U_{S \in C^{'}} S$.*

3.2 Automatic Text Generation Process

The automatic text generation process is divided into two parts: corpus preprocessing, text generation.

1. Corpus Preprocessing.
   Because the high topic confusion of corpus, we classified it into different clusters. LDA (Latent Dirichlet Allocation) is a topic generation model proposed by Blei [15], which can generate the topic probability distribution of a document. In this paper, we use LDA model to predict distribution of topics in corpus and cluster documents by topic distribution. Finally, we obtain several document sets with lower confusion as candidate set of documents.
2. Text Generation.
   In order to generate a beautiful text structure, we classified the candidate set again. After clustering the candidate set into several clusters, we use each cluster to generate a paragraph.

According to the framework shown in Fig. 1, this approach contains two-phase: text cleanup and text extraction. In text cleanup phase, we obtain topic distribution of candidate documents by LDA model and use it to represent documents. Then, we cluster the documents into different topics. In text extraction phase, we use the scores of TF-IDF to define the keywords, we model the text extraction as a keyword set covering problem. Then, we solve the problem via employing the tool of submodular. The new text is composed of sentences extracted from each cluster.

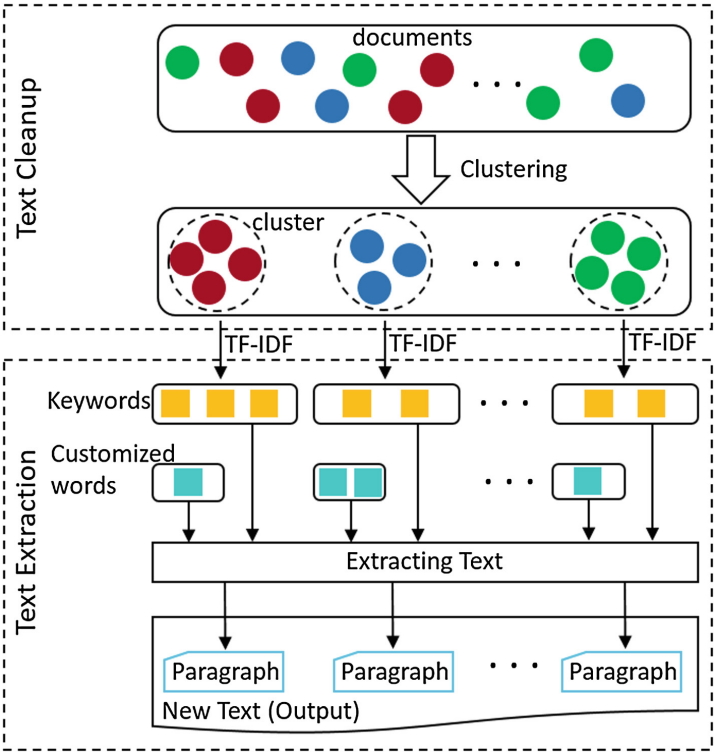

Fig. 1. The framework for automatic text generation

4 Algorithms

1. Keyword Set Generation
   TF-IDF (Term Frequency-Inverse Document Frequency) is a statistical method to evaluate the importance of a word in one document or one corpus. TF represents the frequency of one word appearing in the document. For one specified word t_i, the value of TF is denoted as $T_{i,j}$. IDF is a measure of the

importance of one word, for a specified word t_i, the IDF value is denoted as I_i. Then, we use variable $T - I_{i,j} = T_{i,j} \times I_i$ to obtain the value of TF-IDF. All in all, a high-frequency word has a larger TF-IDF value, so we can use this algorithm to obtain most important words and to filter out high-frequency non-keywords.

2. Text Extraction
After obtaining the set of keywords from the above algorithm and defining customized words, we model text extraction as a set covering problem, the formal description is as follows:

$$
\begin{aligned}
& minimize|C'|,\ that\ is,\ minimize \sum_{j=1}^{m} X_j \\
s.t.\ & X_j \in \{0,1\}, j = 1,2,\ldots,m \\
& Y_{ij} \in \{0,1\}, i = 1,2,\ldots,n1.j = 1,2,\ldots,m \\
& Z_{ij} \in \{0,1\}, i = 1,2,\ldots,n2.j = 1,2,\ldots,m \\
& \sum_{j=1}^{m} X_j Y_{i,j} \geq 1, i = 1,2,\ldots,n1.(1*) \\
& \sum_{i=1}^{n2} \sum_{j=1}^{m} X_j Z_{i,j} \geq 1.(2*)
\end{aligned}
$$

Among this, C' is a set of the extracted sentences, which composes the generated new text; $X_j = 1$ represents sentence S_j is contained in the sentence set C', $X_j = 0$ is opposite; $Y_{i,j} = 1$ means the i-th keyword w_i is in sentence S_j, $Y_{i,j} = 0$ is opposite; $Z_{i,j} = 1$ means the i-th customized keyword u_i is in sentence S_j, $Z_{i,j} = 0$ is opposite. Also, constraint (1*) is to ensure generated text covers all keywords; constraint (2*) is to ensure generated text at least covers one customized keyword.

To solve this set covering problem, one method is to employ the linear programming to approximate the integer programming, which is a NP-hard problem and we cannot theoretically figure out the lower bound of performance of it. Another method is using submodular (See Sect. 2.2) if we can define a set function satisfying the property of submodular. That is the focus of this paper.

In order to solve the set covering problem, we use the following objective function:

$$
f(S) = \sum_{w_i \in \{w|w \in S\} \cap K / \{w \mid w \in C'\}} W_{w_i} + \sum_{w_j \in \{w \mid w \in S\} \cap K_U / \{w \mid w \in C'\}} W_{w_j} \quad (3)
$$

where w_i is a generated keyword, which occurs in sentence set S but not in extracted sentence set C', W_{w_i} is the weight value of keyword w_i. w_j is a customized keyword, which occurs in sentence set S but not in extracted sentence set C', W_{w_j} is the weight value of keyword w_j. Obviously, this function is a submodular. The algorithm is shown in Algorithm 1.

Algorithm 1. Text Extraction Algorithm with Submodular

Input:

The set of sentences, V;
The set of keywords, K;
The set of customized keywords, K_U;

Output:

The set of generated sentences, $C' \subseteq V$;

1: Initialize $C' = \emptyset$
2: **while** $(K! = \emptyset) or (C' \cap K_U = \emptyset)$ **do**
3: **select** s, s.t. $s = arg \max_{v \in V - C'} [f(C' \cup \{v\}) - f(C')]$
4: $C' = C' \cup \{s\}$
5: $\forall_{w \in s} K.remove(w)$
6: $\forall_{u \in s} K_U.remove(u)$
7: **end while**
8: **return** C'

In the Algorithm 1, w is a word in the set of K, v is a word in the set of K_U. The statement of 2 row is the termination conditions of the algorithm, $K! = \emptyset$ means the keywords is not covered completely, $C' \cap K_U = \emptyset$ means that no customized keywords is covered, when satisfying both of them, program is terminated. After finding optimal sentence, we update K by removing keywords in this sentence and update K_U by removing customized words in this sentence. we employ hill-climbing algorithm to solve the set covering problem, so that we can use the least sentences to cover all keywords.

In this algorithm, we find unit s with the largest ratio of objective function gain to scaled cost. If adding s increases the objective function value and not violates the budget constraint, it is then selected and otherwise bypassed. In each iteration, we add the sentence with greatest weight into coverage set. The coverage set becomes the final output.

5 Experiments

5.1 Datasets and Parameters

The corpus which contains 1703 documents. After classifying those documents, we choose 4 classifications as candidate sets. The number of documents in each classification are 529, 245, 649 and 280 respectively. We pick 10 documents from each classification as experimental data and other documents are trained by LDA model to cluster by different topics.

Owning to no standard evaluation data set, we generate standard summarization by manual annotation. For each candidate set which contains 10 documents, we choose 3 volunteers to extract one summarization and there are 12 standard summarizations in all. In this paper, the default parameters of number of topics in LDA topic cluster, K value of K-Means and number of keywords in one classification are set as 50, 4 and 2 * #documents per cluster respectively.

5.2 Baselines

We compare our method with TextRank method based on graphs proposed by Mihalcea [16] and LinearPro method which approximates the set covering problem with a linear programming.

1. TextRank
   This algorithm builds graph by using sentences as nodes and similarities between sentences as weights of edges. TextRank algorithm is an unsupervised text generation extraction method with better results. Different methods have different computing method if sentence similarity.
2. LinearPro
   In Sect. 4, we have mentioned that two methods can solve the set covering problem, the one is integer programming which is a NP-hard problem. We can employ the linear programming to approximate the problem. However, we cannot figure out the lower bound of performance of this method, we use this method as baseline and we donate it as LinearPro. In this method, after getting the weights of each sentence, we need to define a threshold and sentence is extracted when the weight of the sentence is greater than the threshold, in this experiment, we set threshold as 0.8.

5.3 Evaluation Metrics

The method proposed in this paper is evaluated by Edmundson coincidence rate [17] and ROUGE evaluation standard [19].

- Edmundson
  The basic unit of Edmundson is sentence, the coincidence rate C is calculated by the following equation:

$$C = \frac{|S_g \cap S_s|}{|S_s|} \times 100\% \tag{4}$$

  In this equation, S_g represents the set of extracted sentences and S_s represent standard text, generally, we use average value of coincidence rate (Ave) to reduce error.
- ROUGE
  In Edmundson method, we take sentences as compared objects, in this paper, we use ROUGE-N and ROUGE-S metrics. ROUGE-N can reflect the occurrence order of words, the equation is defined as:

$$ROUGE - N = \frac{\sum_{S \in \{ST\}} \sum_{ng \in S} C_M(ng)}{\sum_{S \in \{ST\}} \sum_{ngw \in S} C(ng)} \tag{5}$$

  In this equation, ST represents standard text, $C_M(ng)$ represents the number of n-grams coexisted in new text and standard text. $C(ng)$ represents the occurrence number of n-grams in standard text.

5.4 Experimental Results Analysis

Based on the evaluation metrics of Edmundson coincidence rate, ROUGE1, ROUGE2, ROUGE S and ROUGE SU, we compare our algorithm with baselines.

Comparison in Edmundson Coincidence Rate. Table 1 is the comparison with TextRank, LinearPro and Submodular in Edmundson Coincidence Rate.

Table 1. Comparison with baselines in edmundson coincidence rate

Algorithm	Classification 1	Classification 2	Classification 3	Classification 4
TextRank(BM25)	0.0606	0.0	0.0192	0.0093
TextRank(Sim)	0.0303	0.0	0.0192	0.0093
LinearPro	0.0455	0.0213	0.0288	0.0093
Submodular	**0.09**	**0.0851**	**0.1346**	**0.0370**

From Table 1, we can find that for four classifications, Edmundson coincidence rate of our algorithm is higher a lot than other algorithms. For classification 2, TextRank(BM25) and TextRank(Sim) have not worked and the coincidence rate of LinearPro is only 0.0213, but the coincidence rate of our method is 0.0851, which is higher than LinearPro. Obviously, these experimental data indicates that our method outperforms than baselines.

Comparison in ROUGE. In the progress of experiment, we find the performance of TextRank(Sim) is better than TextRank(BM25), so in this part and next part, we use TextRank(Sim) and LinearPro to compare with Submodular algorithm in ROUGE-1, ROUGE-2, ROUGE-S, ROUGE-SU. The experimental results are shown in Fig. 2.

From Fig. 2, Submodular outperforms TextRank and LinearPro generally. Though LinearPro is better than Submodular in C1, but for C2, C3, C4, Submodular method have absolute superiority. Submodular is worse than TextRank(Sim) in R-2 for C3, but it is better than TextRank(Sim) in R-1, R-S and R-SU. From the perspective of sentence extraction, R-S and R-SU can reflect the sequence of sentences consistent with standard text more effectively and have more practical significance than R-2. Also, from the perspective of probability, our method has better results in most classifications. Therefore, our method is more effective than TextRank and LinearPro.

Comparison in Mean Value of ROUGE. Table 2 shows the mean values of R-1, R-2, R-S, R-SU of four classifications for TextRank(Sim), LinearPro and Submodular methods respectively. The mean value of Submodular in R-1, R-2, R-S, R-SU is higher than that of TextRank(Sim) and LinearPro algorithms.

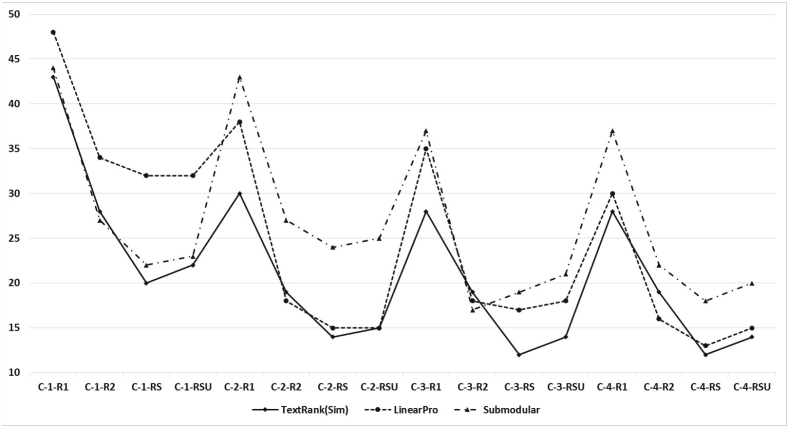

Fig. 2. Comparison with baselines in ROUGE

Thus, from the perspective of average effect, Submodular is superior to baselines. Even in the worst case, the average R-S value of Submodular is improved about 1.5 than LinearPro, the average R-2 value of Submodular is improved about 2 than TextRank(Sim).

Table 2. Comparison with baselines in mean value of ROUGE

Method	R-1	R-2	R-S	R-SU
TextRank(Sim)	32.25	21.25	14.50	16.25
LinearPro	37.75	21.50	19.25	20.00
Submodular	**40.25**	**23.25**	**20.75**	**22.25**

In summary, for those evaluation metrics, Submodular algorithm is more outstanding than other algorithms.

6 Conclusion and Future Work

In this paper, we propose a two-phase algorithm which consists of text cleanup and text extraction. We generate paragraphs based on the topic modeling and clustering analysis firstly. Then, we model the text extraction as a set covering problem after we find the keywords by TF-IDF. Experiments show the effectiveness of our proposed method by comparing with comparable baselines.

In the future work, we will improve the efficiency of our method. Also, we can consider about the position of sentences or words of title so that the generated text will be more consistent with standard summarization in the next work.

Acknowledgements. This work has been supported by the National Key Research and Development Program of China under grant 2016YFB1000905, NSFC under Grant Nos. U1401256, 61402177, 61672234, 61402180, 61502236, 61462017, and 61363005.

References

1. McKeown, K.R., Barzilay, R., Evans, D.K., et al.: Tracking and summarizing news on a daily basis with Columbia's newsblaster. In: Proceedings of Human Language Technology Conference (2002)
2. Radev, D.R., McKeovwn, K.R.: Generating natural languages summaries from multiple on-line sources. Comput. Linguist. **24**, 21–29 (1998)
3. Lin, H., Bilmes, J.: Multi-document summarization via budgeted maximization of submodular functions. In: Association for Computational Linguistics, pp. 912–920 (2010)
4. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 335–336 (1998)
5. Zhang, Y., Krieger, H.U.: Large-scale corpus-driven PCFG approximation of an HPSG. In: Association for Computational Linguistics, pp. 198–208 (2011)
6. Sripada, S., Reiter, E., Davy, I.: SumTime-Mousam: configurable marine weather forecast generator. Expert Update. **6**, 4–10 (2003)
7. Kukich, K.: Design of a knowledge-based report generator. In: Association for Computational Linguistics, pp. 145–150 (1983)
8. Portet, F., Reiter, E., Gatt, A., et al.: Automatic generation of textual summaries from neonatal intensive care data. Artif. Intell. **173**, 789–816 (2009)
9. Karpathy, A., Fei-Fei, L.: Deep visual-semantic alignments for generating image descriptions. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 3128–3137 (2015)
10. Li, S., Ouyang, Y., Wang, W., et al.: Multi-document summarization using support vector regression. In: Proceedings of DUC (2007)
11. Clarke, J., Lapata, M.: Global inference for sentence compression: an integer linear programming approach. J. Artif. Intell. Res. **31**, 399–429 (2008)
12. Filippova, K.: Multi-sentence compression: finding shortest paths in word graphs. In: Association for Computational Linguistics, pp. 322–330 (2010)
13. Thadani, K., McKeown, K.: Supervised Sentence Fusion with single-stage inference. In: IJCNL, pp. 1410–1418 (2013)
14. Fujita, A., Inui, K., Matsumoto, Y.: Exploiting lexical conceptual structure for paraphrase generation. In: Dale, R., Wong, K.-F., Su, J., Kwong, O.Y. (eds.) IJCNLP 2005. LNCS, vol. 3651, pp. 908–919. Springer, Heidelberg (2005). doi:10.1007/11562214_79
15. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent Dirichlet allocation. J. Mach. Learn. Res. **3**, 993–1022 (2003)
16. Mihalcea, R., Tarau, P.: TextRank: bringing order into texts. In: Association for Computational Linguistics, pp. 404–411 (2004)
17. Edmundson, H.P.: New methods in automatic extracting. J. ACM (JACM) **16**, 264–285 (1969)
18. Nemhauser, G.L., Wolsey, L.A., Fisher, M.L.: An analysis of approximations for maximizing submodular set functions I. Math. Program. **14**, 265–294 (1978)
19. Lin, C.Y.: Rouge: a package for automatic evaluation of summaries. In: Text Summarization Branches Out: Proceedings of ACL-2004 Workshop, pp. 74–81 (2004)