# Multi-Document Abstractive Summarization using Chunk-graph and Recurrent Neural Network

Jianwei Niu*, Huan Chen*, Qingjuan Zhao*, Limin Sun†, Mohammed Atiquzzaman‡
*State Key Laboratory of Virtual Reality Technology and Systems,
School of Computer Science and Engineering, Beihang University, Beijing 100191, China
†Beijing Key Laboratory of IOT Information Security, Institute of Information Engineering, CAS, China
‡School of Computer Science, University of Oklahoma, Norman, OK 73019, USA
Email: niujianwei@buaa.edu.cn

***Abstract*—Automatic multi-document abstractive summarization system is used to summarize several documents into a short one with generated new sentences. Many of them are based on word-graph and ILP method, and lots of sentences are ignored because of the heavy computation load. To reduce computation and generate readable and informative summaries, we propose a novel abstractive multi-document summarization system based on chunk-graph (CG) and recurrent neural network language model (RNNLM). In our approach, A CG which is based on word-graph is constructed to organize all information in a sentence cluster, CG can reduce the size of graph and keep more semantic information than word-graph. We use beam search and character-level RNNLM to generate readable and informative summaries from the CG for each sentence cluster, RNNLM is a better model to evaluate sentence linguistic quality than n-gram language model. Experimental results show that our proposed system outperforms all baseline systems and reach the state-of-art systems, and the system with CG can generate better summaries than that with ordinary word-graph.**

## I. Introduction

Automatic multi-document summarization system aims to generate informative and readable summarization from multi-document. Recent approaches can be classified into two types, extractive multi-document summarization systems and abstractive multi-document summarization systems. The former scores the sentences from source documents and directly extract high score sentences as the summaries. This kind of systems, which can be easily implemented, are able to get higher linguistic quality summaries. While it does have several limits compared with the latter [1]. As for abstractive systems, they need to understand the source documents firstly and then generate new sentences as summaries which are more informative than that of extractive systems. While it is hard for abstractive systems to generate readable sentences. We describes recent works about document summarization system in Section II.

In order to generate readable and informative summaries for documents, we propose a novel abstractive multi-document summarization system based on chunk-graph and recurrent neural network language model (RNNLM) [2]. The details of our approach are described in Section III. Chunk-graph is based on word-graph [3], it can compress similar sentences and generate a directed graph based on chunks and their relations, then gets compressed sentences from short paths on the graph. RNNLM is able to evaluate the linguistic quality of summaries. Our approach consists of the following four steps:

- Generating sentence clusters from source documents. We can extract several topics of source documents and eliminate redundancies among documents after this step.
- Generating chunk-graph for each cluster. To reduce size of the graph, we use chunks instead of words as the basic units on the graph (we denote the graph as chunk-graph (CG)). As for those chunks expressing the same meaning but in different ways, co-reference resolution (CR) has been applied to merge them.
- Using beam search to find candidate paths in each CG. During the search process, there are several rules to decide which node to be searched. We use the average probability score calculated by RNNLM to evaluate the linguistic quality of every candidate path so that we can get readable summaries.
- Considering the most informative sentence should be put in the first place, so we use Lexrank to get the informative score for each summary of cluster, outputting them in descending order according to their score as the whole summaries.

Our approach is mainly inspired by the works of [3] and [5]. Compared with them, the most important contribution of our work is the CG and the process of searching best path as summary in CG. Our CG method is based on the word-graph approach of [3], and it can keep more semantic information in source sentences than word-graph so that we can get better summaries. Our search process can find more paths in CG than the random selection in the work of [5], making sure that good sentence paths wont be ignored. Besides, The RNNLM applied in our approach is a better way to evaluate sentence linguistic quality than the Tri-gram language model used in [5].

We evaluate our proposed method on the DUC2004 dataset[1] by ROUGE scores [4], the ROUGE-2 and ROUGE-SU4 scores obtained by our method outperforms many baseline and reach the state-of-art systems which show the effectiveness of our approach. Section IV describes our experiments and results. We also calculate the average size of CG and word-graph on the DUC2004 dataset, the result shows that with the help of

[1] duc.nist.gov/duc2004/

978-1-4673-8999-0/17/$31.00 ©2017 IEEE

CG and CR, graph size has been greatly reduced instead of using word as the basic unit. Besides, CG is able to filter lots of sentence paths in low linguistic quality.

## II. RELATED WORK

There are several abstractive summarization approaches in recent years. Filippova et al. [3] denoted the word-graph method to compress multiple sentences. The word-graph uses a directed graph to represent the sentences and find the shortest path in the graph as a compressed sentence. While in the approach of Filippova, the rank of those sentence paths is only based on informativeness. So the summaries generated by the approach of Filippova are not in high linguistic quality. Banerjee et al. [5] proposed a method, taking both informativeness and readability into consideration when they selected sentences in word-graph. They used an integer linear programming (ILP) model to maximizing information content and readability of the final summary. To reduce the computation, the ILP based model randomly selects 200 paths for evaluation during the ILP process. While a large amounts of paths are neglected because of this kind of random selection, which has a negative effect on the quality of final summaries. Li [6] introduced an BSU Semantic Link Network, using the Basic Semantic Unit (BSU) to describe an event or an action. Not like the word-graph, BSU Semantic Link Network is constructed based on BSU rather than words. Summaries are generated from the constructed network.

There are also some abstractive summarization systems based on deep neural network. Hu et al. [7] constructed a large corpus of Chinese short text summarization dataset named LCSTS. An end-to-end recurrent neural network approach was used to generate summaries. The approach of Rush et al. [8] is similar to Hu's recurrent neural network, while he applied an Attention Model [9] on the network architecture to further improve the ability of recurrent neural network in modeling long documents. Gu et al. [10] followed the work of Hu and used the copy mechanism to handle the problem of out-of-vocabulary word, which greatly improved the performance of their system. Deep neural network based methods are promising, while they need a mass of alignment corpus data which is difficult to get to train the models.

## III. PROPOSED METHOD

Given a document set $D$, that consists of several similar documents $\{d_1, d_2, d_3...d_n\}$, and each document $d$ contains several sentences $\{s_1, s_2, s_3...s_n\}$. Our approach takes all sentences together and clusters them into $k$ clusters $\{c_1, c_2, c_3...c_n\}$, where $k$ is preset manually or decided by clustering algorithms.

After that, sentences in the same cluster $c$ are in high similarity so that they can be compressed to one sentence. We proposed the chunk-graph(CG) which is based on word-graph to compress sentences in each cluster. To construct a chunk-graph, we apply NLTK [11] toolkit to label the chunks in each sentence in a cluster, then take chunk units as nodes and relations among them as edges.

After the CG is constructed, we apply Stanford core-nlp co-reference resolution toolkit [12] to label the chunks which refer to the same entities. Next merge nodes correspond to them into one node in the CG. After that, we apply beam search and RNNLM to find the best path in the CG as the summary of a cluster. In the end, we use Lexrank [13] to score all summaries generated from CGs. According to these scores, we are able to decide the order of each sentence cluster's summary in the final summary. Fig. 1 shows the overall architecture of our approach.

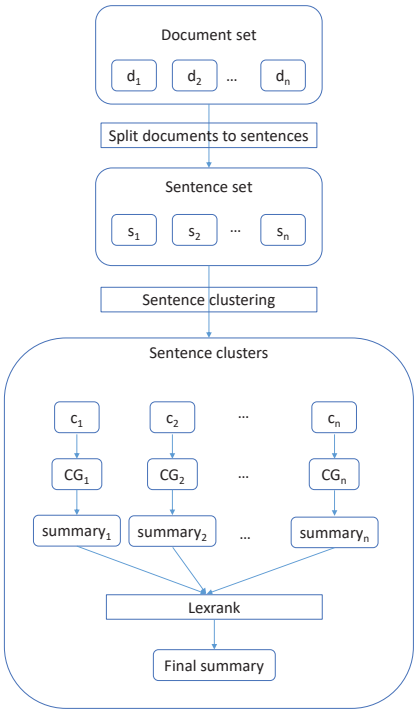


Fig. 1. The overall architecture of our method

### A. Sentence Clustering

Sentence clustering aims to divide sentences into several groups in an unsupervised way. The most important part of that is how to define the distance among sentences. The usual method is to use one-hot encoding to encode words into one-hot vectors and encode sentences into vectors based on important words' vectors, then apply clustering algorithms to those sentence vectors. One problem of this method is that the similarity among words is ignored. In our approach, Word2Vec [14] [15] is used to encode words to word vectors and we calculate the average word vector as the sentence vector.

*1) Calculating Sentence Distance:* The most important step of sentence clustering is to define the distance among sentences. We should encode words into vectors before that.

Words are usually considered as category feature and encoded to high dimensional sparse vectors by one-hot encoding, which totally ignores the semantic information of words and is hard to store and compute because of its high dimensional.

A better way is to encode words into dense vectors with fixed dimension, which is called the distribution representation of words.

Word2Vec is a common way to generate distributed word vectors. It can generate word vectors from large documents in an unsupervised way by using a shallow neural network. The obtained vectors can keep the words' semantic information. If there is high relevance between two words, their Euclidean or Cosine distance of their corresponding vectors will be small.

We set the word vectors in $100$ dimensions then trained a Word2Vec model in a large English Wikipedia corpus [2] which contains $1.9$ billion words in more than $4.4$ million articles. Finally, we get a look-up table containing words and their vectors.

For the dataset $D$, we combine all sentences into a sentence set $S$ which contains $n * m$ sentences $\{s_1, s_2, ..., s_{n*m}\}$. We denote the sentence vector as the average of word vectors in it. In detail, for a sentence $s$ which is consists of $l$ words $\{w_1, w_2, ..., w_l\}$, each word can be encoded into a 100 dimensions vector $V_{wi}$ $\{v_{w(i,1)}, v_{w(i,2)}, ..., v_{w(i,100)}\}$ according to the look-up table(For the words not in the look-up table we set their vectors to zero), where $i$ is the index of a word in the sentence, $v_{w(i,j)}$ is the $j-th$ dimension value of the $i-th$ word's vector. Then the sentence vector $V_s$ can be defined as:

$$Vs = \{\frac{\sum_{i=0}^{l} v_{w(i,1)}}{l}, \frac{\sum_{i=0}^{l} v_{w(i,2)}}{l}, ..., \frac{\sum_{i=0}^{l} v_{w(i,100)}}{l}\} \quad (1)$$

The distance $dis(s_i, s_j)$ between two sentences $s_i$ and $s_j$ can is defined the cosine distance of their vector:

$$dis(s_i, s_j) = \frac{V_{s(i)} \cdot V_{s(j)}}{\|V_{s(i)}\|\|V_{s(j)}\|} \quad (2)$$

where $V_{s(i)}$ and $V_{s(j)}$ are the sentence vectors of the $i-th$ and $j-th$ sentences.

*2) Clustering Algorithm:* We adopt the fast search-and-nd of density peaks approach [16] which can determine the cluster number $k$ and is able to recognize non-spherical clusters. In this algorithm, we need to compute the local density $p$ for each sentence vector. The local density $p(x)$ of the $x-th$ sentence vector can be calculated by Parzen-Window Density Estimation [17]:

$$p(x) = \frac{1}{n}\sum_{i=0}^{n} \frac{1}{h_n^d} K(\frac{x - x_i}{h_n}) \quad (3)$$

Where $K(x)$ is the window function or kernel in the $d-dimensional$ space, $h_n$ is the window width corresponding to the width of the kernel, $n$ is the total number of points(sentences). Then we need to compute the distance of the closest data point with higher density for each point. The minimum distance $md(i)$ for the i-th sentence is denoted as:

$$md(i) = \min_{j:p_j > p_i} dis(s_i, s_j) \quad (4)$$

[2]corpus.byu.edu/wiki

Where $dis(s_i, s_j)$ is the distance of two sentence vectors which we denoted in Eq. 2. The whole algorithm for sentence clustering is showed in the Algorithm.1:

**Algorithm 1** Sentence Clustering

Input: sentence set $S$, look-up table $D$ for words and word vectors
Output: sentence cluster set $C$
**for all** sentence $s \in S$ **do**
  get sentence vector $V_s$ from $D$
  **add** $V_s$ **to** sentence vector set $S_v$
**end for**
$P \leftarrow \{\}$
**for all** sentence vector $V_s \in S_v$ **do**
  caculate density $p$
  $p(V_s) \leftarrow \frac{1}{n}\sum_{i=0}^{n} \frac{1}{h_n^d} K(\frac{V_s - V_{si}}{h_n})$
  **add** $p(V_s)$ **to** $P$
**end for**
$D \leftarrow \{\}$
**for all** $p_i \in P$ **do**
  $TEMP \leftarrow \{\}$
  **for all** $p_j \in P \,\&\, j! = i$ **do**
    **if** $p_j > p_i$ **then**
      **add** $dis(V_{s(i)}, V_{s(j)})$ **to** $TEMP$
    **end if**
  **end for**
  **add** $\min(TEMP)$ **to** $D$
**end for**
$centers \leftarrow \{\}$
plot $D$ as function of $P$ in $G$
**for all** $point \in G$ **do**
  **if** $point$ is outlier **then**
    **add** $point$ **to** $centers$
  **end if**
**end for**
$C \leftarrow \{\}$
**for all** $point \in G \,\&\, point \notin centers$ **do**
  **find** cluster $c$ of point's nearest neighbor of higher density
  $c \leftarrow point$
**end for**
**add** all $c$ **to** $C$
**return** $C$

### B. Chunk-graph (CG)

For each sentence cluster, we construct a CG from all sentences within it. Our CG is based on word-graph approach [3], which is able to maintain more semantic structures than word-graph approach. CG can also generate smaller size of graph than word-graph so that it reduces computational load in the summaries generation step. We also apply CR to further reduce the size of the graph.

*1) Constructing CG:* To clarify our approach, we take the following three sentences as an example to generate a CG:

- Eg.1 In order to saving time, Harry chose an apple as his lunch today.
- Eg.2 At noon, Harry ate an apple due to his busy work.
- Eg.3 Harry often eats an apple during his lunch time.

We firstly analyse each sentence and get the chunks lists for them by applying NLTK toolkit [11]. The following lists are the chunks lists of Eg.1, Eg.2 and Eg.3:

- Chunks list of Eg.1: [In order to] [saving] [time], [Harry] [chose] [an apple] [as] [his lunch] [today].
- Chunks list of Eg.2: [At] [noon], [Harry] [ate] [an apple] [due to] [his busy work].
- Chunks list of Eg.3: [Harry] [often] [eats] [an apple] [during] [his lunch] [time].

Then we can generate a CG based on the chunk lists for a sentence cluster with three rules that are similar to the rules in [3]:

- Take each chunk as node and only one node in the graph.
- All sentences start with a dummy 'start' node and end with a dummy 'end' node.
- A directed edge is added to the graph between two chunks. The direction of the edge is from a chunk to its successor chunk in the same sentence.

After the above steps, each path starts from a dummy node 'start' and ends in a dummy node 'end' in the CG. All paths in the CG are possible to be a summary for this sentence cluster. Fg.2 is the CG constructed from the above three chunk lists based on the rules mentioned above.

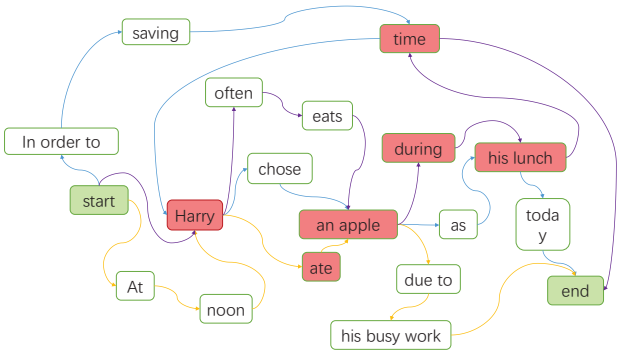


Fig. 2. The CG constructed from Eg.1, Eg.2 and Eg.3. We can see a short sentence path in the graph with red color: Harry ate an apple during his lunch time.

*2) Finding best path in CG:* The size of word-graph for dozens of sentences in a cluster can be quite large. Our CG is much smaller than original word-graph. But in terms of computation time, it is still impossible to search all paths in the graph. So we apply beam search to find informative paths on the graph. During each level of the search process, we use two rules to determine heuristic values for each available node and only store nodes of high values for the next level search. We adopt the following two factors to calculate the heuristic value $h$ for a node.

The first factor is the average TF-IDF value of the chunks of this node. The score of TF-IDF value for a chunk node $tf{-}idf_{s(i)}$ is the average TF-IDF value of all words except stop words in the chunk.

The second factor is the smallest cosine distance $D_{min}$ between chunk of this node and chunk of its former nodes. In detail, for a node set with $m$ nodes $\{n_1, n_2, ..., n_m\}$, which have already been added into the a path, the smallest cosine distance $D_{min(i)}$ for node $i$ can be calculated as:

$$D_{min(i)} = \min(dis(n_i, n_1), dis(n_i, n_2), ..., dis(n_i, n_m)) \tag{5}$$

Where $dis(n_i, n_j)$ is the cosine distance between chunks of $n_i$ and $n_j$. The way to calculate distance is the same as calculating sentence distance in Eq. 2. The value of $D_{min}$ should be big enough to big enough to ensure that the summary is in low-redundancy. To keep the summary both in low-redundancy and informative, the total heuristic value $h_{(i)}$ of $n_i$ can be defined as:

$$h_{(i)} = D_{min(i)} + tf - idf_{s(i)} \tag{6}$$

Then we can get the $h$ value for a path by calculating the average $h$ value of all nodes in it. To reduce the computation load during evaluating summaries linguistic quality by a pre-trained RNNLM, we only reserve 200 paths with the highest $h$ values. Besides, they need to be longer than 8 words. Algorithm.2 is the pseudocode of the beam search process.

**Algorithm 2** Beam Search

```
Input: A chunk-graph (CG)
Output: Best sentence path p_best
BEAM ← {}
add 'start' node to BEAM
size ← 5
paths ← {}
while BEAM! = ∅
    searchnode ← BEAM.pop()
    for all node ∈ successor nodes of searchnode and not searched do
        h_(node) = D_min(node) + tf − idf_s(node)
    end for
    add size highest h score nodes to BEAM
end while
p_best ← highest h score sentence path in paths
return p_best
```

We use a pre-trained language model to ensure the linguistic quality of summaries. A statistical language model assigns probabilities to sequence of words as follows:

$$P(w_1, w_2, ..., w_m) = \prod_{i=1}^{m} P(w_i | w_1, w_2, ..., w_{i-1}) \tag{7}$$

Where $w_i$ is the $i{-}th$ word in sentence. While the above formula is too complex to compute with limited computation

resources. The $n-gram$ model is used to assign probabilities to sequence of words approximately. In an $N-gram$ language model, the probability of each word is only affected by the previous $n$ words, so the formula can be rewritten as:

$$P(w_1, w_2, ..., w_m) = \prod_{i=1}^{m} P(w_i | w_{i-(n-1)}, w_{i-(n-2)}, ..., w_{i-1}) \tag{8}$$

In fact, $N-gram$ language model is only able to keep history information of several words for a word sequence, where $n$ is usually smaller than 4 because of the computational limitation. A lot of information will be lost during the modeling process if we adopt a $N-gram$ model. A better way is to use the recurrent neural network language model(RNNLM). RNNLM is based on neural network language model(NNLM) [18] and is able to encode all history information in a word sequence with the help of its recurrent structure in the network. This kind of structure enable the neurons to memorize its last output.

We trained a character-level RNNLM on the English Wikipedia corpus. Then we can apply the RNNLM model to evaluate sentences' linguistic quality according to their average probability. The sentence with max average probability will be outputted as the summary for its sentence cluster. To clarify our way of evaluate sentence linguistic quality by a character level RNNLM, we take three sentences as example. The first sentence is a fluent sentence. Then we shuffled it in word level and we get the second sentence. Shuffle the second sentence in character level to get the third one. Fig.3 shows our method to decide which sentence is the best sentence in linguistic quality. The shuffled sentences will get much lower probability than the ordinary fluent sentence.

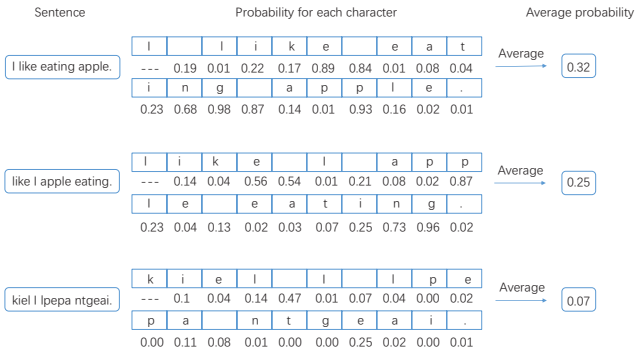


Fig. 3. The process of evaluating sentence probability by RNNLM

*3) Generating final summary:* The final summary is composed of all sentence clusters' summaries. The order of all sentence clusters' summaries in the final summary are depended on their informativeness. The most informative summary should be put in the first place. We use the Lexrank algorithm to assign each sentence cluster's summary a score which represents how many information the summary's contains. Lexrank calculate all sentences' score iteratively. Then we output all sentence clusters' summaries in descending order of their Lexrank scores.

## IV. EXPERIMENTS AND RESULTS

Our approach was evaluated on the DUC 2004 dataset in 3 parts. In the first part we show that our CG and CR approach can greatly reduce the size of graph on the DUC 2004 dataset. In the second part we compared the average probability score between the CG and word-graph. In the last part we evaluated our summaries quality according to the ROUGE score. There are 50 document sets in the DUC 2004 dataset, with each dataset containing 10 similar short documents within a similar topic.

### A. Graph size

Our CG and CR approach aims to reduce the size of graph so that we are able to get better sentence path as summary and reduce the computation load. We choose the DUC 2004 dataset and generate word-graphs based on the work of [3], CGs are constructed as we described in section.III.

According to the result in Table.I, the average size of CGs are much smaller than that of word-graphs. After we apply CR into the CG, the size of CG becomes much smaller. But there are more available sentence paths in CG with CR. After two nodes $n_i$ and $n_j$ are merged, a node which can only reach $n_i$'s successor nodes before is able to reach all $n_j$'s successor nodes now. So using CR to merge nodes in CG may create more available sentence paths. Apparently, smaller size of graph can reduce the computation load effectively, but whether it can improve the summaries linguistic quality still need to be evaluated in the next section.

TABLE I
THE SIZE OF WORD-GRAPH AND CG

| | word-graph | CG without CR | CG with CR |
|---|---|---|---|
| Average sentence count | | 13.54 | |
| Average node count | 75.04 | 55.91 | **45.12** |
| Average edge count | 107.99 | 75.13 | **71.62** |
| Average path count | 94805.51 | **2820.36** | 24073.45 |

### B. Average probability of sentence paths

We can prove that our approach to reduce graph size has positive effect on summaries linguistic quality by calculating the average probability of all sentence paths in a graph using a pre-trained RNNLM. Our RNNLM is trained on the English Wikipedia corpus in an unsupervised way with a GTX TITAN-Z graphic card. To reduce the evaluation time, we randomly select 80 paths in a graph in four length level and evaluate 11 graphs generated from the first document set.

Table.II shows that the average probability score of our CG approach is much higher than word-graph, the score of CG with CR is a bit lower than the original CG. Our CG approach is able to remove the majority of low linguistic quality paths and keep high linguistic quality paths on the graph, which improves the final summaries' quality effectively.

### C. ROUGE evaluation

ROUGE [4] evaluation is the most important part in all of our evaluation steps. ROUGE is the abbreviation for Recall-Oriented Understudy for Gisting Evaluation, which can measure the summary quality by comparing the similarity between it and summary created by humans. We use the ROUGE-2 and ROUGE-SU4 scores in our evaluation. ROUGE-2 is a $N-gram$ based ROUGE. It counts the same bigrams in two sentences. ROUGE-2 gives higher score to a summary with more bi-grams that are same with the human-write summary. ROUGE-SU4 is the skip-bigrams with unigrams ROUGE, it scores according to the co-occurrence of skip-bigrams.

TABLE II
AVERAGE PROBABILITY SCORE

| graphs | word-graph | CG without CR | CG with CR |
|---|---|---|---|
| less than 20 words | 0.1527 | 0.2683 | **0.2699** |
| 20-30 words | 0.1513 | **0.2727** | 0.2708 |
| 30-40 words | 0.1649 | **0.2812** | 0.2803 |
| more than 40 words | 0.1717 | **0.2885** | 0.2871 |

We evaluate three similar systems, which generate sentence cluster's summary by word-graph, CG without CR and CG with CR respectively. We also compare our systems ROUGE scores to the best baseline system's result in DUC 2004, ILP based abstractive summarization systems in [5] and some systems' results reported in Banerjee's paper including both abstractive and extractive summarization systems.

According to the ROUGE scores obtained by automatic multi-document summarization systems in the Table.III, our CG system without CR can generate better summaries than the other two systems. Both the CG system with and without obtained much higher score than the word-graph based system. CG system obtained higher score in ROUGE-SU4 while lower score in ROUGE-2 than CG with CR system. Compared with other systems, our CG system outperforms all baseline systems and many state-of-art systems. Besides, it obtained higher ROUGE-SU4 score than Banerjee's ILP based system.

TABLE III
ROUGE SCORES OF DUC 2004 DATASET

| Methods | ROUGE-2 | ROUGE-SU4 |
|---|---|---|
| Baseline system in DUC 2004 (ID:67) | 0.09226 | 0.13349 |
| DPP | 0.10079 | 0.14556 |
| Submodular | 0.09602 | 0.14227 |
| RegSum | 0.09712 | 0.13812 |
| ILP based method [5] | **0.11992** | 0.14765 |
| Our system(WG) | 0.98311 | 0.13420 |
| Our system(CG without CR) | 0.10444 | **0.14820** |
| Our system(CG with CR) | 0.10621 | 0.14013 |

## V. CONCLUSION AND FUTURE WORK

We introduced an abstractive multi-document summarization system based on chunk-graph and recurrent neural network language model. We apply beam search with some rules to find informative paths in chunk-graph. A character-level recurrent neural language model is used to ensure the summaries are readable. Our results on the DUC 2004 dataset show that chunk-graph approach outperforms all baseline systems and reach the state-of-art systems. Our results also show that the chunk-graph based summarization system can generate better summaries than word-graph based summarization system. We plan to adopt word-level RNNLM to improve our summaries linguistic quality in the future, using more semantic information to construct better CG.

## VI. ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (61572060, U1536107, 61472024), and CERNET Innovation Project (NGII20151104, NGII20160316).

## REFERENCES

[1] G. Carenini and J. C. K. Cheung, "Extractive vs. nlg-based abstractive summarization of evaluative text: The effect of corpus controversiality," in *Proceedings of the Fifth International Natural Language Generation Conference*. Association for Computational Linguistics, 2008, pp. 33–41.

[2] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *Interspeech*, vol. 2, 2010, p. 3.

[3] K. Filippova, "Multi-sentence compression: finding shortest paths in word graphs," in *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 2010, pp. 322–330.

[4] C.-Y. Lin, "Rouge: A package for automatic evaluation of summaries," in *Text summarization branches out: Proceedings of the ACL-04 workshop*. Barcelona, Spain, 2004.

[5] S. Banerjee, P. Mitra, and K. Sugiyama, "Multi-document abstractive summarization using ilp based multi-sentence compression," in *Proceedings of the 24th International Conference on Artificial Intelligence*. AAAI Press, 2015, pp. 1208–1214.

[6] W. Li, "Abstractive multi-document summarization with semantic information extraction," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 2015, pp. 1908–1913.

[7] B. Hu, Q. Chen, and F. Zhu, "Lcsts: A large scale chinese short text summarization dataset," *arXiv preprint arXiv:1506.05865*, 2015.

[8] A. M. Rush, S. Chopra, and J. Weston, "A neural attention model for abstractive sentence summarization," *arXiv preprint arXiv:1509.00685*, 2015.

[9] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[10] J. Gu, Z. Lu, H. Li, and V. O. Li, "Incorporating copying mechanism in sequence-to-sequence learning," *arXiv preprint arXiv:1603.06393*, 2016.

[11] S. Bird, "Nltk: the natural language toolkit," in *Proceedings of the COLING/ACL on Interactive presentation sessions*. Association for Computational Linguistics, 2006, pp. 69–72.

[12] K. Toutanova, D. Klein, C. Manning *et al.*, "Stanford core nlp," *The Stanford Natural Language Processing Group. Available: http://nlp. stanford. edu/software/corenlp. shtml. Accessed*, 2013.

[13] G. Erkan and D. R. Radev, "Lexrank: Graph-based lexical centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–479, 2004.

[14] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[15] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv preprint arXiv:1301.3781*, 2013.

[16] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[17] E. Parzen, "On estimation of a probability density function and mode," *The annals of mathematical statistics*, vol. 33, no. 3, pp. 1065–1076, 1962.

[18] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, Feb 2003.