Graph-based Summarization

Survey, Reproduction, and Improvement

Digital Speech Processing
Final Project
2017 Fall
Nation Taiwan University

Authors:
B04902012 資工三 劉瀚聲
B04902077 資工三 江緯璿

1 Introduction

When we want to buy a new product, such as an electronic gadget, an automobile, or book a restaurant, book a hotel, we tend to surf the Internet first to find some comments made by other customers. These comments is relatively convincing compared to the official advertisement since these are real user feedbacks. However, the amount of total information is too large. Take a famous hotel-booking website "*Agoda*" (https://www.agoda.com/zh-tw/) for example, most of the hotels is commented by at least 500 customers, while some of them has even thousands of evaluation records. We cannot consume such large amount of comments and feedbacks in a short period of time. Despite that there is an average score of all comments for every hotels, it does not provide any reasons or details about why this hotel deserve such grade. We may want to know more concrete facts about its advantages, and its shortages. What if there is a summarization robot, which can consume all the contents and accordingly output an easy-to-read brief summary of the comments? In that way, we can know the main characteristics and facts about the hotel instantly! As a consequence, we would like to research on the summarization technique.

In this project, we focus on summarizing the comments of a single product or service, such as Windows 7, Apple iPod, or a hotel service. We begins with some surveys and paper reading. Next, based on some existing work, we implemented a toolkit to summarize the comments of some certain products on the Internet. In addition, we tried to improve the summarization algorithm proposed by the original

author, in order to generate a better summary. Finally, since not each of our improvement methods are useful, and there are still a lot to be improved, we proposed some points as our future work. Hoping we can make this summarization algorithm better and better in the future.

2 Paper Survey and Reading Report

In order to find some feasible first steps of our implementation and sketch, we surveyed several papers and literatures mentioned on the course handout (Reference Part) and from the Internet.

Overall, we have surveyed three papers related to this technique. The paper list is attached in the reference [1][2][3]. We will summarize each paper in the following paragraphs and discuss their characteristics.

Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. [1]

The technique introduced in this paper is the main part of our implementation:

- The dataset to be summarized a first converted into a directed graph.

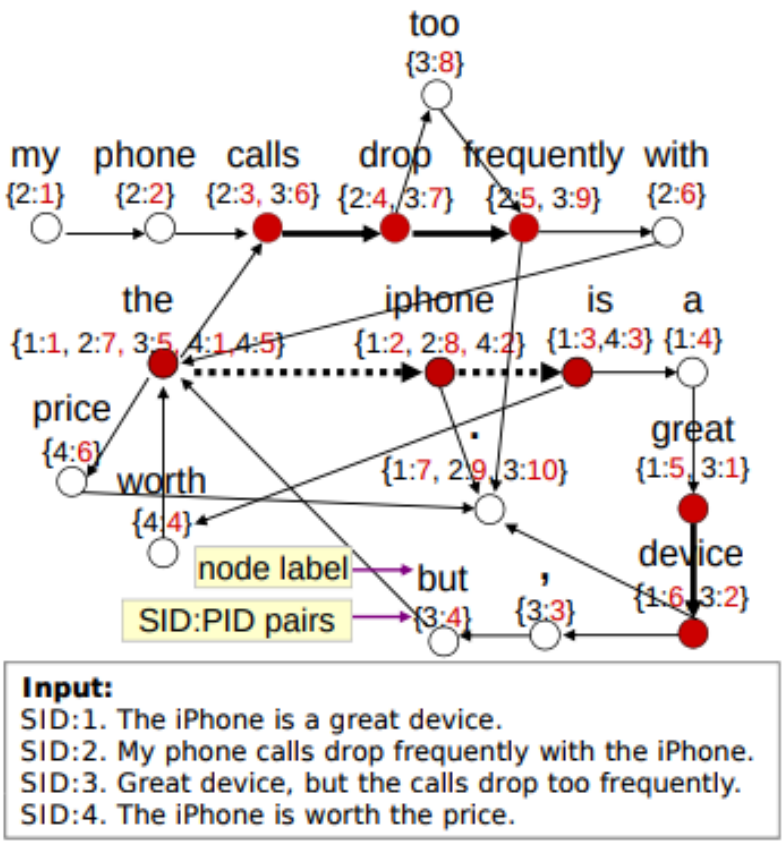

Figure 1. A sample converted graph given in the paper

- Each node in the graph represents an appeared word, storing the list of occurrence position and sentence id.
- Each edge from word u to word v implies that word v comes after word u in some sentences.
- This graph-like data structure preserves some intuitive properties of the dataset:
  - Redundancy: Redundant phrase can be captured by heavy edge weight.
  - Gapped Subsequence Capture: A gapped subsequence can be captured by observing position of adjacent nodes.
  - Each subpath in the graph is a relatively reasonable phrase, since each 2-gram of the phrase appeared in the sentences set.
- Enumerate every possible phrase by traversal every possible path, validate and evaluate them.
  - Validation:
    A valid path $(w[0], w[1], ..., w[n-1])$ should satisfies the following properties:
    - $w[0]$ is in VSN, where VSN is a collection of words whose average position in the sentences set $< \sigma_{vsn}$.
    - $w[n-1]$ is in VEN, where VEN is a collection of punctuations and coordinating conjunctions.
    - w satisfies some POS constraints.
  - Evaluation:
    The score of a path $(w[0], w[1], ..., w[n-1])$ can be chosen from $score_{basic}$, $score_{wt\_len}$, or $score_{wt\_loglen}$, and are defined as follows:
    - $r(w)$=$|\{s|s$ is a sentence from the dataset, and w is a gapped subsequence in s with gap not exceeding $\sigma_{gap}\}|$
    - $score_{basic}(w)$=$\Sigma_{w'}(r(w'))/|w|$ over all subpath w'.
    - $score_{wt\_len}(w)$=$\Sigma_{w'}(r(w')|w'|)/|w|$ over all subpath w'.
    - $score_{wt\_loglen}(w)$=$\Sigma_{w'}(r(w')\lg|w'|)/|w|$ over all subpath w'.
- Create stitched sentences by combining high-score paths with common prefix
- Eliminate similar sentences, and output the top n sentences as the summary.

Algorithm 1 (A1): *OpinosisGraph*(Z)

Input: Topic related sentences to be summarized: $Z = \{z_i\}_{i=1}^{n}$
Output: $G = (V, E)$
for $i = 1$ to n **do**
  $w \leftarrow Tokenize(z_i)$
  $sent\_size \leftarrow SizeOf(w)$
  for $j = 1$ to $sent\_size$ **do**
    $LABEL \leftarrow w_j$
    $PID \leftarrow j$
    $SID \leftarrow i$
    if $ExistsNode(G, LABEL)$ **then**
      $v_j \leftarrow GetExistingNode(G, LABEL)$
      $PRI_{v_j} \leftarrow PRI_{v_j} \cup (SID, PID)$
    else
      $v_j \leftarrow CreateNewNode(G, LABEL)$
      $PRI_{v_j} \leftarrow (SID, PID)$
    end if
    if not $ExistsEdge(v_{j-1} \rightarrow v_j, G)$ **then**
      $AddEdge(v_{j-1} \rightarrow v_j, G)$
    end if
  end for
end for

Algorithm 2 (A2): *OpinosisSummarization*(Z)

Input: Topic related sentences to be summarized: $Z = \{z_i\}_{i=1}^{n}$
Output: $\mathcal{O}$ ={Opinosis Summaries}
$g \leftarrow OpinosisGraph(Z)$
$node\_size \leftarrow SizeOf(g)$
for $j = 1$ to $node\_size$ **do**
  if $VSN(v_j)$ **then**
    $pathLen \leftarrow 1$
    $score \leftarrow 0$
    $cList \leftarrow CreateNewList()$
    Traverse$(cList, v_j, score, PRI_{v_j}, label_{v_j}, pathLen)$
    $candidates \leftarrow \{candidates \cup cList\}$
  end if
end for
$\mathcal{C} \leftarrow EliminateDuplicates(candidates)$
$\mathcal{C} \leftarrow SortByPathScore(\mathcal{C})$
for $i = 1$ to σ_{ss} **do**
  $\mathcal{O} = \{\mathcal{O} \cup PickNextBestCandidate(\mathcal{C})\}$
end for

Algorithm 3 (A3): $Traverse(...)$

Input: $list, v_k \subseteq V, score, PRI_{overlap}, sentence, len$
Output: A set of candidate summaries
$redundancy \leftarrow SizeOf(PRI_{overlap})$
if $redundancy \geq \sigma_r$ **then**
  if $VEN(v_k)$ **then**
    if $ValidSentence(sentence)$ **then**
      $finalScore \leftarrow \frac{score}{len}$
      $AddCandidate(list, sentence, finalScore)$
    end if
  end if
  for $v_n \in Neighbors_{v_k}$ **do**
    $PRI_{new} \leftarrow PRI_{overlap} \cap PRI_{v_n}$
    $redundancy \leftarrow SizeOf(PRI_{new})$
    $newSent \leftarrow Concat(sentence, label_{v_n})$
    $L \leftarrow len + 1$
    $newScore \leftarrow score + PathScore(redundancy, L)$
    if $Collapsible(v_n)$ **then**
      $C_{anchor} \leftarrow newSent$
      $tmp \leftarrow CreateNewList()$
      for $v_x \in Neighbors_{v_n}$ **do**
        Traverse$(tmp, v_x, 0, PRI_{new}, label_{v_x}, L)$
        $CC \leftarrow EliminateDuplicates(tmp)$
        $CCPathScore \leftarrow AveragePathScore(CC)$
        $finalScore \leftarrow newScore + CCPathScore$
        $stitchedSent \leftarrow Stitch(C_{anchor}, CC)$
        $AddCandidate(list, stitchedSent, finalScore)$
      end for
    else
      Traverse$(list, v_n, newScore, PRI_{new}, newSent, L)$
    end if
  end for
end if

Figure 2. Pseudo-code from the paper[1]

Spoken Lecture Summarization by Random Walk over a Graph Constructed with Automatically Extracted Key Terms.[2]

As the course suggested, Key Term Extraction is also an important approach in summarization. This paper proposed a summarization algorithm utilizing the automatically extracted key terms. As for the generation process, the authors decided to use “Probabilistic Latent Semantic Analysis” (PLSA) with Expectation-

Maximization Algorithm Optimization (EM Algorithm) to extract the key terms. These two techniques are both mentioned in our class and in our handout.

Next, a graph g is constructed with the following rules.

- A node S represents a complete sentence.
- The edges E are directional
- The weight W on a edge E from node i to node j is the *Topical Similarity* between the two vertex sentences of the certain edge. The value is NOT symmetrical
- Topical Similarity $p(i,j)$ is defined as the following formulas, where the LTS is the Latent Topic Significance

$$p(i,j) = \frac{sim(S_i, S_j)}{\sum_{S_k \in A_i} sim(S_i, S_k)} \quad sim(S_i, S_j) = \sum_{t \in S_j} \sum_{k=1}^{K} \mathrm{LTS}_t(T_k) P(T_k \mid S_i)$$

Figure 3. Topical Similarity formula extracted from the original paper[2]

- For each node, only top N outgoing edges are valid, while other edges are eliminated. N is a parameter.

The following is an example graph extracted the original paper.

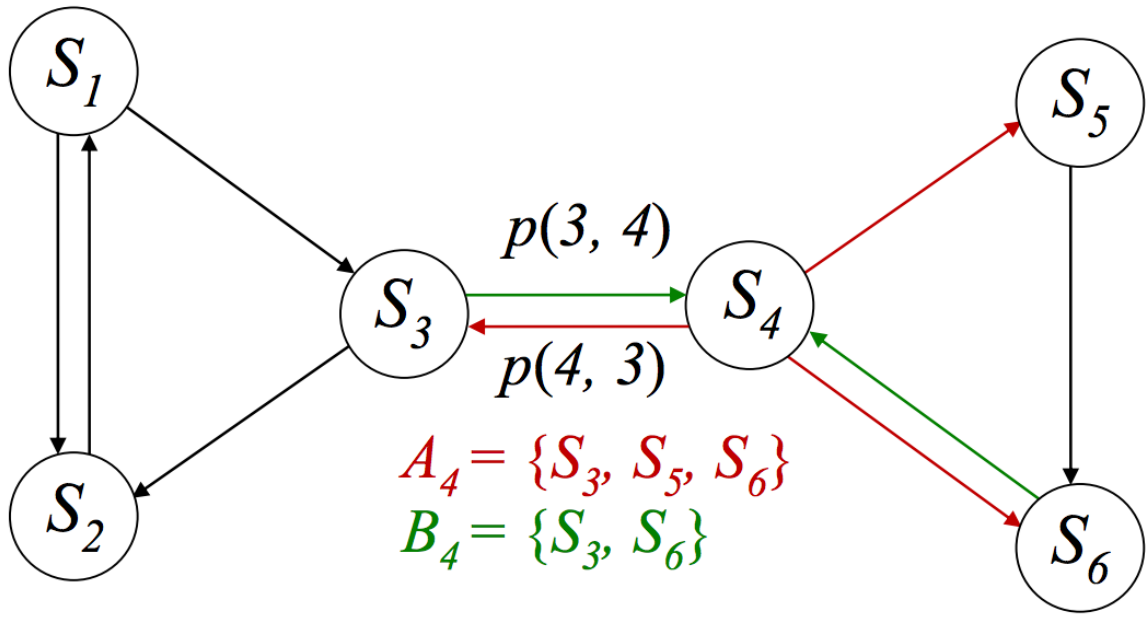

Figure 4. A simplified example of the graph extracted from the paper[2]. Each sentence is represented as a node on the graph. Ai and Bi are the neighbors of the node Si connected respectively by outgoing and incoming edges.

The elimination of weak edges guaranteed the quality of each transition. As a result, to some extent, the noise is reduces.

As for the traversal, the authors proposed a random walk algorithm. It gives a new score to each of the nodes. The score is composed of two parts. One is the Normalized Initial Importance, the other is the score of other neighbors weighted by the Topical Similarity, $p(i,j)$. The two parts are merged by a interpolation weight α.

The following is the scoring function $v(i)$.

$$v(i) = (1-\alpha)r(i) + \alpha \sum_{S_j \in B_i} p(j,i)v(j)$$

Figure 5. Scoring function formula extracted from the original paper[2]

After several transition and equation/eigenvectors solving which is similar to the PageRank problem, we can know the importance of each sentence, and therefore generate proper summary.

The authors conduct several experiments based on the previous Digital Speech Processing course speech content, and uses *Rouge*[3] to evaluate the effectiveness and correctness of the automatically generated summary. In addition, as for the content, the authors tested the same content with two different sources. One is the transcript made by human, and the other is the transcript made by Automatic Speech Recognition (ASR) [7]. The baseline achievable score is evaluated through the LTE-based summarization.

The results are astonishing under all cases. Despite that under some cases such as Key-term based summarization under low summarization ratio, the random walk algorithm can hardly make the results better, it did improve the performance under most cases. Random walk algorithm can really help improve the accuracy of existing summarization methods.

Rouge: A package for automatic evaluation of summaries.[3]

ROUGE is the abbreviation of *Recall-Oriented Understudy for Gisting Evaluation*. It is mainly used to measure the quality of an automatically generated summary by comparing it with a man-made summary. It covers lots of aspects to evaluate the similarity, inclusive of N-gram, word sequences, word pairs between the ideal summary made by human and the auto-generated summary to be evaluated.

There are multiple types ROUGE scores. Each of them emphasize different approaches for checking the similarity.

- $ROUGE_N$: Based on the N-gram matching. The score is evaluated by the ratio of matched N-gram. The following is the formula. It can have multiple references. In this way, the score will be the maximum among all reference sets.

$$\text{ROUGE-N} = \frac{\sum_{S \in \{ReferemceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)}$$

Figure 6. ROUGE-N formula extracted from the original paper[3]

- $ROUGE_L$: Based on Longest Common Subsequence (LCS). LCS is a classic string matching problem in computer science. The key idea is that this problem can evaluate the similarity between two contexts. It utilizes normalized pairwise LCS to evaluate, and $ROUGE_L$ takes the union score of the results.
- $ROUGE_W$: Improvement of $ROUGE_L$. Based on Weighted LCS with dynamic programming.
- $ROUGE_S$: Based on the N-gram matching and allows arbitrary gaps.
- $ROUGE_{SU}$: Extension of $ROUGE_S$ since $ROUGE_S$ does not consider a candidate sentence with no word pair co-occurring with the reference sets.

3 Implementation and Experiment

We implement such procedure by ourselves using Python. (Git Repository Link: https://github.com/b04902012/DSP_Final)

We utilized some existing toolkit, including *NLTK[4]*, and *Pattern.en[5]*, and both of them are used for their POS tagging modules.

We evaluate our program by running on some sentence set in *Opinosis Dataset[6]*.

Our program is highly modulized. Most of the program is the implementation of [1], but we did some modification.

- During traversing, we add a threshold σ_r. If $r(\text{traversed path}) < \sigma_r$, return immediately. This pruning is implemented for efficient issue, since a naive algorithm took more than half an our to summarize a dataset containing about 100 sentences.

- We did not check the POS pattern due to lacking of POS pattern scorer. Instead, we constraint the POS of the beginning and the ending of a valid path.
- $score_{wt\_len}$ can be viewed as weighted sum of $r(w')$ with length be its weight.A POS-pattern checker seems to be a permanent cure, but we also found that the generated summary is more readable when weighting w' by $len(w')^5$, to weight more on longer substring.

The summaries generated by our program fit our expectation on some of the sentences set. For example, the summaries for speed_windows7.txt.data is “faster than vista”, “faster install” and “faster than xp”.

4 Usage

```
python main.py txt/to/path
```

5 Difficulties

We encountered some difficulties in our works. Some of them have not been overcome and might be solved in future works.

- Traversing all paths takes exponential time, so it’s necessary to apply pruning, but we are not sure whether our policy is proper or not. The best summary might be abandoned at the beginning if σ_r is too large, while smaller σ_r will takes longer time to traverse.
- We determine the set VSN by checking the average position of each word, but average positions of common words are often greater than expected. For example, the word “the” is often judged as not in VSN because “the” appears in every part of sentences.
- If two word appears more than once respectively in a single sentence, it’s hard to define the size of the gap between them.
- The best value for those tunable parameters (σ_{vsn}, σ_{gap}, etc) varies greatly, so it’s necessary to set them properly. However, we haven’t find an algorithm to set those tunable parameters automatically.
- It seem that checking POS sequence is the most effective way to improve performance, but we haven’t applied a POS checker into our program yet. Perhaps we’ll train one in the future.

6 Conclusion

We have surveyed some summarization methods proposed by others. In addition, we implemented one of them, and made some modifications. The experiment results become more precise and correct after our tuning and adjustment. Therefore, summarization is really feasible, and is not too difficult to implement a basic summarization program.

Despite that our results are not as good as we had expected, this is an initial attempt. There are still a lot of improvement approaches that we have not applied yet, such as language model, grammar checking, different scoring methods (i.e. $ROUGE$ [3]), etc. We expect that with good modification and justification, our summarizer can work even better than it can currently.

All in all, we have learned a lot from this project, and looking forward to the future improvement of our implementation.

7 Future Work

1. Summary smoothing

   Despite that the result indeed included some high-value key words, the sentence is not quite smooth. There were grammar mistakes, mistaken language usage, etc. These poses significant difficulties to the readers. As a result, we would like to find out a solution to make the generated summary content readable and meaningful. One possible approach is to apply n-gram or any existing language model to the summary content, and accordingly improve the smooth and readability of the automatically generated summary. In addition, checking the POS tagging sequence might also be useful since English has a relatively simple sentence structure. Applying a POS checker may be helpful.
2. Auto Tuning

   We have several parameters in the summarization algorithm. However, currently, we can only adjust those parameters and judge the corresponding output manually. What if we can automatically tune those parameters and let the machine decide the optimal parameters? This can save lots of time for parameter adjustment. In addition, an optimal set of parameters can also improve the summarization result. This might be feasible and hope we can implement it as an improvement in the future.
3. Efficiency

   As mentioned in Section 5, traversal consumes a significant amount of time with an exponential time complexity. We would like to reduce the search space, but how to prune our search tree is a big problem. We should not give up some significant candidate, nor should't we cause a premature interrupt. Consequently, improving the time efficiency and complexity becomes one of our future work.

8 Reference

1. Ganesan, K., Zhai, C., & Han, J. (2010, August). Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 340-348). Association for Computational Linguistics.
2. Chen, Y. N., Huang, Y., Yeh, C. F., & Lee, L. S. (2011). Spoken Lecture Summarization by Random Walk over a Graph Constructed with Automatically Extracted Key Terms. In Interspeech (pp. 933-936).Implementation
3. Lin, C. Y. (2004, July). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop* (Vol. 8).
4. NLTK Official Document: http://www.nltk.org/book/ch05.html
5. Pattern.en Official Document: https://www.clips.uantwerpen.be/pages/pattern-en
6. Opinosis Dataset: https://github.com/kavgan/opinosis/blob/master/OpinosisDataset1.0_0.zip
7. HOVELL, Simon Alexander. *Automatic speech recognition*. U.S. Patent No 5,905,971, 1999.
8. Lee, L. S. (2017). Handouts and slides of Digital Speech Processing course in 2017 fall semester, NTU. (Chapter 11.0)

9 Work Distribution

- 劉瀚聲 Implementation, Baseline Testing, Paper Reading[1], Parameter Tuning, Report Writing
- 江緯璿 Implementation, Baseline Testing, Paper Reading[2][3], Validation, Report Writing