Automated Summarization System

Haobo Gu, Yuanhe Tian, Weifeng Jin, Haotian Zhu

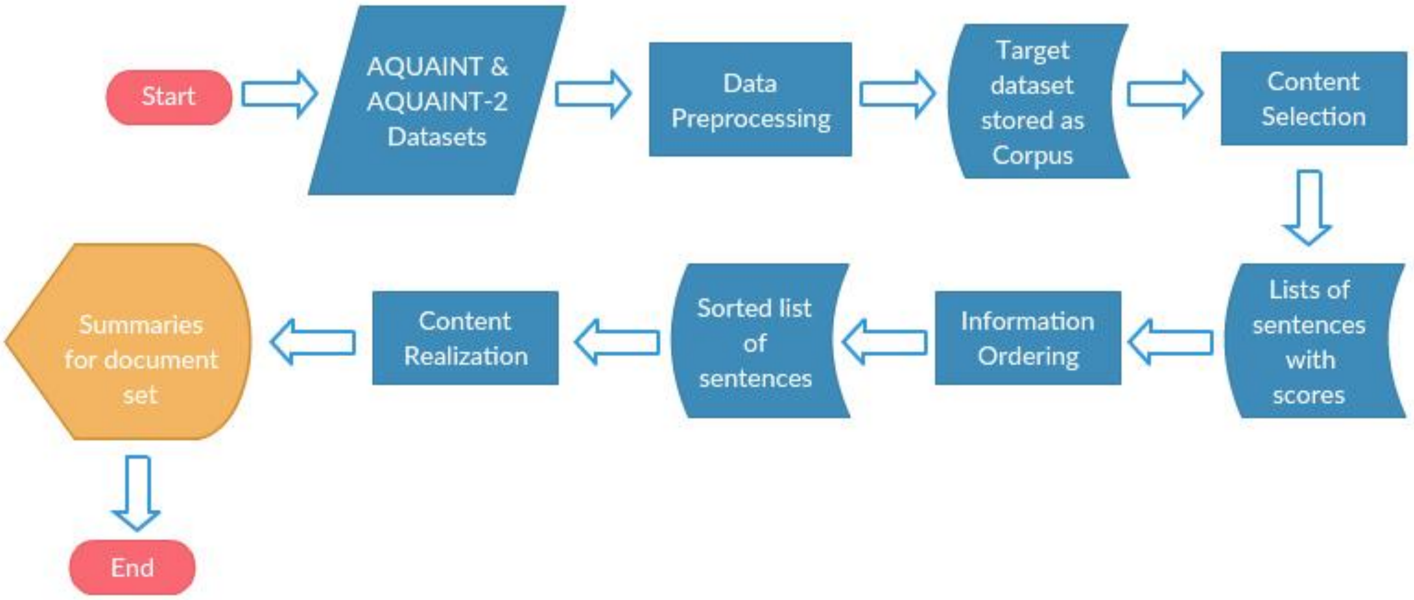
Overview

- Automated multi-document summarization system
- to shorten the input text without losing too much information in the context and to create an abbreviated, informative and consistent summary

Overview: Baseline System

- Unsupervised, no training required
- Extraction-based approach without modifying sentences
- Four parts:
  - Data Preprocessing
  - Content Selection
  - Information Ordering
  - Content Realization

System Flowchart



Data Preprocessing

<u>Input</u>: one document specification (.xml file) and two corpora (AQUAINT and AQUAINT-2)

<u>Output</u>: ***corpus*** object consists of:

- docsetList: list of docSet objects (document set)
- tokenDict: vocabulary dictionary to record word occurrence in the whole dataset.

docSet object consists of:

- idCode: the ID for the document set
- documentCluster: a list of document objects (document)
- hummanSummary: a list of human summaries
- topicID, tokenDict

Data Preprocessing

document object consists of:

- idCode, topicID, tokenDict, time
- sentences: the list of sentence objects (sentences)

sentence object consists of:

- idCode, tokenDict, doctime
- content: the content of the sentence
- index: the position index of the sentence
- score: the score of the sentence
- length: the number of tokens in the sentence

Content Selection

Input: A document set which has *m* sentences and *n* different tokens

Algorithm:

1. For each sentence in the docset, generate its feature vector
2. Calculate sentence similarity between every two sentences based on their feature vector
3. Calculate salient score for each sentence based on LexRank algorithm

Output: A list of selected k sentences with their salient score

Step 1 - Generate Sentence Feature Vector

- We generate feature vector based on the number of token occurrence in this sentence

		t_1	t_2	t_3	t_4	...	t_n	
s_1	$<$	2	0	1	1		0	$>$
s_2	$<$	1	1	0	3		1	$>$
...	$<$	...	...	...	...	...	...	$>$
s_m	$<$	0	1	2	0		0	$>$

Step 2 - Calculate Sentence Similarity

- Sentence similarity is represented by the **cosine** distance between two sentences:

$$Sim(s_i, s_j) = \frac{s_i * s_j}{|s_i| * |s_j|}$$

- Then generate an *m* * *m* matrix to store similarities and convert this matrix to a transition probability matrix ***M*** (the sum of each row equals to 1)

Step 3 - Calculate Salient Score

- Calculate salient score by Power Method

- Output the top k sentences with highest score, where *k* is determined by the compress rate *r*.

Power Method:

__**input**__: transition probability matrix M
__**initialize**__: n-dimension vector $\mathbf{p}_0 = \mathbf{1}$
$t = 0$
__**do**__:
    $t = t + 1$
    $\mathbf{p}_t = M^T \mathbf{p}_{t-1}$
while ($\mathbf{p}_t - \mathbf{p}_{t-1}$ > delta)   (**delta = 0.001**)
__**return**__: $\mathbf{p}_t$

Information Ordering

- Current consideration:
    - Chronological order
    - Cohesion

Information Ordering

Input: a list of selected sentences with salient scores.

Do:

- Sort the list by salient scores in a descending order.
- Group sentences by the document to which they belong and sort by **publication dates.**
- Within each document group, sort sentences by indices.

Output: a sorted list of sentences.

Content Realization

We implemented two content realization approaches.

- **Naive approach**: simply pick sentences according to sorted order. If the length of summary exceeds the word limitation in current sentence, discard and terminate.
- **Integer Linear Programming approach**: maximize a target function consists of scores of the sentences and diversity of the sentences.
    - **Diversity** of the sentences is measured using the number of bigrams in summary.
    - The more bigrams appear in summary, the larger the diversity is.

Results and Analysis

Table 1: ROUGE Results for `devtest`

ROUGE	Recall	Precision	F-score
ROUGE-1	0.20038	0.25005	0.22145
ROUGE-2	0.04347	0.05310	0.04759
ROUGE-3	0.01218	0.01495	0.01336
ROUGE-4	0.00356	0.00433	0.00389

Table 2: ROUGE Results for `training`

ROUGE	Recall	Precision	F-score
ROUGE-1	0.20231	0.28003	0.23382
ROUGE-2	0.05260	0.07219	0.06062
ROUGE-3	0.01885	0.02640	0.02191
ROUGE-4	0.00847	0.01224	0.00997

- The baseline system does not deliver a satisfying performance.
- NO risk for overfitting since it is an unsupervised learning approach and there is no training required.
- Highest score in ROUGE-1, lowest score in ROUGE-4, indicating that the level of summarization required for each document should be improved.
- Want to try a supervised learning approach in the next deliverable to improve the performance and fully utilize the dataset provided.

Issues and Successes

Issues:

- Low recall/precision/F-score in all ROUGE standards.
- NOT utilizing the training dataset.
- Several aspects of the approach are too straightforward.
- Alternative approaches to consider for Information Ordering to make more use of salient score.

Successes:

- Connectivity is achieved with easy-to-understand data structure and programming scripts.
- Training NOT required, which resulted in a relatively short running time.

Strategies for Group Work

- Weekly meeting on Sunday night, discuss this week’s work and coordinate next week’s tasks.
- Every team member works on a single component. Pros: straightforward task allocation and great efficiency, Cons: have to wait for the other team member to finish his part to start.
- Mainly used github for code sharing and coordination.

References

Dimitrios Galanis, Gerasimos Lampouras and Ion Androutsopoulos. 2012. Extractive Multi-Document Summarization with Integer Linear Programming and Support Vector Regression. Proceedings of COLING 2012, 2012: 911-926.

Ryan A. Georgi. 2018. Lecture of Content Selection and Information Ordering in the course LING 573 Spring 2018 in the University of Washington.