SummPip

Unsupervised Multi-Document Summarization with sentence graph compression

From: gradboost

Marmik Patel - 2021202009
Mayur Kumar - 2021202028
Devansh Avasthi - 2021201027

Introduction

Summpip is a unsupervised summarizer of multiple documents

The main idea here is to create the summary from sentence graph created based on sentence similarity

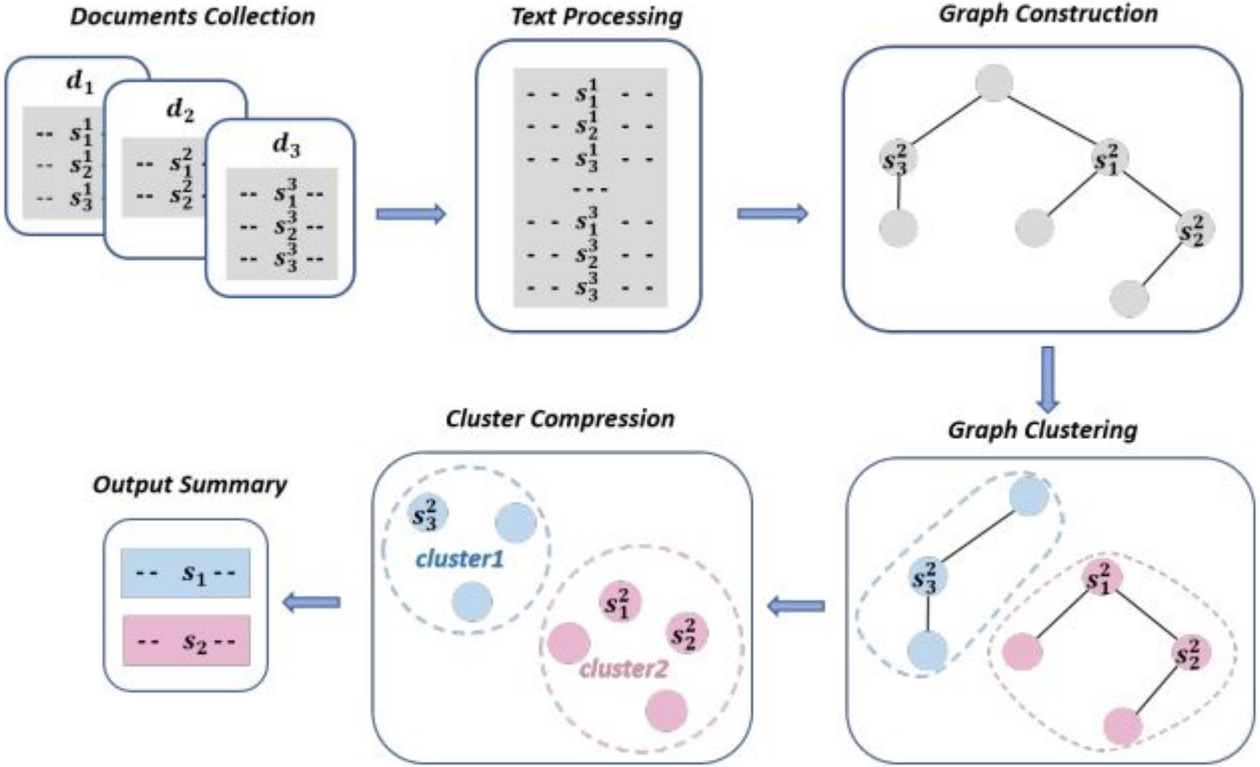
Why unsupervised summarizer ?

- Getting training data for supervised summarizer is time consuming and resource intensive
- Because of this, recent neural models can only be trained for limited domains

Pipeline to get summary

To get the summary, SummPip follows a pipeline that consists of:

1. Text processing
2. Graph construction
3. Graph clustering
4. Cluster compression


Documents Collection
d_1
-- s_1^1
-- s_2^1
-- s_3^1
d_2
-- s_1^2
-- s_2^2
d_3
-- s_1^3 --
-- s_2^3 --
-- s_3^3 --
Text Processing
- - s_1^1 - -
- - s_2^1 - -
- - s_3^1 - -
- - -
- - s_1^3 - -
- - s_2^3 - -
- - s_3^3 - -
Graph Construction
s_3^2
s_1^2
s_2^2
Graph Clustering
s_3^2
s_1^2
s_2^2
Cluster Compression
s_3^2
cluster1
s_1^2
s_2^2
cluster2
Output Summary
-- s_1 --
-- s_2 --

Text Processing

Text processing consists of:

- Concatenating all the document into single document.
- Getting a sentence embeddings for all the sentence inside the concatenated document

Thus, after this process, we will get the sentence embeddings of all the sentences

Graph construction

After getting the sentence embeddings, we construct the graph based on similarity

For a graph G(V,E), with following configuration:

- Each node v_i belongs to V represents a sentence
- Nodes v_i and v_j is connected i.e., $e_{i,j} = 1$, if we get a cosine similarity between respective two sentences is greater than a certain set threshold.

Thus, after this step, we will get a graph constructed based on sentence similarity

Spectral clustering

After the graph is been constructed, we perform following task on that:

- Create a laplacian matrix based on above sentence graph
- Compute first ‘k’ eigenvectors of the laplacian matrix (feature vector for each sentence)
- Vectorize sentences based on these ‘k’ eigenvectors.
- Separate the sentences by clustering them using K-means clustering

This step will return the graph clusters for the sentence graph

Cluster compression

After getting the graph clusters, we use multi-sentence compression on clusters to get the summary

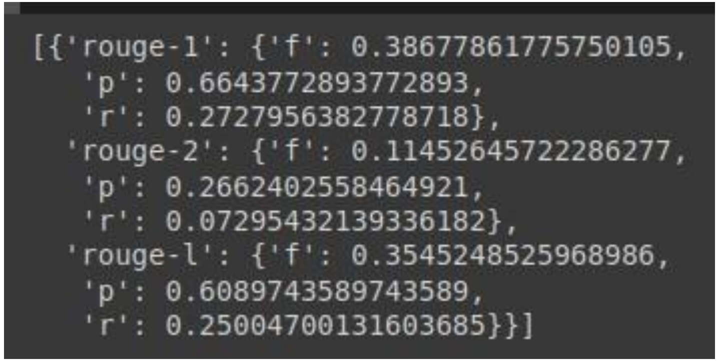
Multi-sentence compression generates a single summary sentence from each cluster by using following steps:

- Get the first sentence in the cluster and prepend start token and postpend end token into it
- From second sentence onwards, append the sentence to the word inside the linear graph created in the previous step, if same word is present with similar POS tagging
- Create a new node only if the word is not present in the previous linear graph with similar POS tagging
- Get the summary by getting the shortest path between start and end token

After this step, we select the summary with highest score as our final output.

Evaluation

- We evaluated our model on Multi-News Dataset and its target summary.
- We are using ROGUE score to evaluate the performance of SummPip.
- Upon summarizing the multi-document dataset into 200 sentences summary, our implementation achieved following scores:



```
[{'rouge-1': {'f': 0.38677861775750105,
   'p': 0.6643772893772893,
   'r': 0.2727956382778718},
  'rouge-2': {'f': 0.11452645722286277,
   'p': 0.2662402558464921,
   'r': 0.07295432139336182},
  'rouge-l': {'f': 0.3545248525968986,
   'p': 0.6089743589743589,
   'r': 0.25004700131603685}}]
```

Thank you