CS69201: Computing Lab-1
Assignment 7 (Part 2) - Text Summarization
August 28, 2024
Time - 60 minutes

========== Instructions==========

1. In the case of user input assume only valid values will be passed as input.
2. You will use Python for this assignment.
3. Regarding Submission: Please create one notebook (.ipynb) file for your submission with one section per assignment segment. Kindly name the notebook file as **<ROLL_NO>.ipynb**.
4. Input Document should be read from a file (input_doc.txt) **and output summary sentences should be printed in a file named output.txt.**

==========

In this assignment, you will work on text summarization using NLTK, networkx, and scikit-learn. The assignment is segmented in X parts, following which sequentially will cause the desired summarized output.

Task-1: Reading and Segmenting Documents into Sentences.

In this task, you will read a document from a file (input_doc.txt) and store it into a variable. Then using nltk sent_tokenize utility, segment the document into individual sentences.

Task-2: Cleaning Sentences and Removing Stopwords.

In this task, you will process each of the sentences which removes punctuation, stem words in the sentences and remove stopwords from the sentences, yielding clean sentences.

Task-3: Generating Representations on Cleaned Sentences.

In this task, you will encode each of the sentences using the fit_transform method of scikit-learn's TfIdfVectorizer class (sklearn.feature_extraction.text.TfidfVectorizer). The method takes as input the list of sentences and yields an array with each row corresponding to a sentence. Additionally, compute the cosine similarity between all the sentence pairs (using sklearn.metrics.pairwise.cosine_similarity).

Task-4: Generating Graph on Sentences

In this task, you will generate a graph of sentences. First generate an empty undirected graph. Then add nodes in the graph one per sentence such that node 'i' will correspond to i^{th} sentence. For all i,j such that $i \neq j$, you will add edges in the graph. The weight of the edge between node i and j: w_{ij}, will be the cosine similarity between sentence i and j.

Task-5: Identifying Important Sentences using PageRank

After constructing the graph, the next step is to identify the important sentences. To do so, you will run PageRank on this weighted graph and select top-K (K=5) node ids in the ranked order. These ids will identify the sentences that will be used in the summary. Kindly note that this sentence ids should identify original raw sentences and not the cleaned sentences.

Task-6: Generating Final Summary

The top-K sentences you get may be in any order (Eg. 1,5,0,6,3). However, while summarizing a document the order of sentences matters. Therefore, you should output the sentence in the order of the occurrence (i.e., using the id of the sentence) with new-line character as the separator between the sentences.

To clarify, the following is to be done.

1. You start with a document of N sentences {Sent:0, Sent:1, Sent:2, …, Sent:N-1}.
2. You clean them and then obtain the following {CleanSent:0, CleanSent:1,....}.
3. You then encode clean sentences using TfIdfVectorizer().fit_transform and get a matrix of dimension N*M, where N is #Sentences and M is the #unique words across sentences.
4. You compute a pairwise cosine similarity matrix of dimension N*N where each entry is the cosine similarity between the tf-idf representation of the sentences.
5. Then you create a graph as stated and run the PageRank algorithm.
6. You identify top-5 important nodes by sorting based on the PageRank scores.
7. You then sort the top-5 nodes based on the node index.
8. Finally the summary is generated by retrieving the sentences corresponding to each of the top-5 nodes and concatenating them by new-line (‘\n’) character.