Project 29: Automatic Summarization 2

We shall consider structured document containing a title, abstract and a set of subsections. We would like to build a text summarizer such that tracks important keywords in the document. For this purpose, the first step is identify these keywords.

1) Assume the initial input is given as html document (choose an example of your own), we hypothesize that important keywords are initially contained in the words of titles, abstract and possibly titles of subsections of the document. Suggest a simple python script that inputs an html document and outputs the lists of words in the title, abstract and title of section/subsections.
2) Write down a simple python script that allows you to output the histogram of word frequency in the document, excluding the stopwords (see examples in online NLTK book). Use SpaCy named-entity tagger to identify person-named entities and organization-named entities in the document.
3) We would like the summarizer to contain frequent wording (excluding stopwords) and as many named-entities as possible. For this purpose, use the following heuristic to construct the summarizer. First we shall assume each sentence of the document as individual sub-document. Use TfIdf vectorizer to output the individual tf-idf score of each word of each sentence (after initial preprocessing and wordnet lemmatization stage). Then consider only sentences that contain person or organization named-entities and use similar approach to output the tfidf score of the named-entities in each sentence. Finally construct the sentence (S) weight as a weighted sum:

$$S_{weight} = \sum_{w \epsilon S} w_{TfiDf} + 2 \sum_{NM \epsilon S} NM_{TfiDf} + POS_S$$

where NM_{TfiDF} stands for the TfIdF of named-entity NM in sentence S. POS_S corresponds to the sentence weight associated to the location of the sentence. So that the sentence location weight will be maximum (1) if located in the title of the document, 0.5 if located in the title of one of the subsection, 0.25 if located in the title one of the subsubsection, 0.1 if located in one representative object of the document, and 0 if located only in the main text. Make sure to normalize the term tfidf and Nm tfidf weights and suggest a script to implement the preceding accordingly, so that the summarizer will contain the 10 sentences with the highest S_{weight} scores.

4) Test the above approach with Opinosis dataset available at https://kavita-ganesan.com/opinosis-opinion-dataset/#.YVw6J5ozY2x, and record the corresponding Rouge-2 and Rouge-3 evaluation score.
5) We would like to improve the summarization by taking into account the diversity among the sentence in the sense that we would like to minimize redundancy among sentences. For this purpose, we shall use the sentence-to-sentence semantic similarity introduced in the NLP lab. Next, instead of recording only the 10 sentences with highest S_{weight} scores, we shall record the 20 top sentences in terms of S_{weight} scores. Then the selection of the top 10 sentences among the 20 sentences follows the following approach. First, order the 20 sentences in the decreasing order of their S_{weight} scores, say S1, S2, …, S20 (where S1 is the top ranked and S20 the 20^{th} ranked sentence). Second, we shall assume that S1 is always included in the summarizer, we

shall then attempt to find the other sentences among S2 till S20 to be included into the summarizer. Calculate the sentence-to-sentence similarity Sim(S1,Si) for i=1 to 20, the Sentence Sj that yields the minimum similarity with S1 will therefore be included in the summarizer. Next, for each of the remaining sentences Sk (with k different from 1 and j), we calculate the sentence similarity with Sj. Therefore the sentence Sp that yields minimum value of "Sim(Sp, S1)+Sim(Sp,Sj)" will be included in the summarizer (Note: the quantity Sim(Sp, S1) is already calculated in previous step). Similarly in the next phase, we should select a sentence Sl (l different from 1, j and k) so that "Sim(Sl, S1)+Sim(Sl,Sj)+Sim(Sl,Sp)", Etc.. You then stop once you reached 10 sentences included in the summarizer. Suggest a script that includes this process.. and illustrate its functioning in the example you chosen in 1).

6) We would like to make the choice of keywords not based on histogram frequency but using the open source RAKE https://www.airpair.com/nlp/keyword-extraction-tutorial. Repeat the previous process of selecting the sentences that are associated to the ten first keywords generated by RAKE. Comment on the quality of this summarizer based on your observation
7) It is also suggested to explore alternative implementations with larger number of summarization approaches implemented- https://github.com/miso-belica/sumy. Show how each of the implemented summarizer behaves when inputted with the same document you used in previous case. You may also explore the implementation provided in the Summarizer toolkit provided in Moodle section of the course.
8) Now we would like to compare the above summarizers and those in 3), 5) and 7) on a new dataset constructed as follows. First select an Elsevier journal of your own and select 10 papers highly ranked in the journal according to citation index (The journal papers should be well structured to contain Abstract, Introduction and Conclusion). For each of the ten papers, consider the introduction as the main document to seek to apply summarizer, and consider the Abstract and Conclusion as two golden summary of the document that you can use for assessment using ROUGE-1 and ROUGE-2 evaluation. Report in a table the evaluation score of each summarizer.
9) Design a simple GUI that allows the user to input a text or a link to a document to be summarized and output the summarizer according to 3), algorithms implemented in 7)