Web summative report

a) Background information on the problem area

Multi-document summarisation is an automatic procedure aimed at extraction from multiple texts written about the same topic and encompasses both Natural Language Processing and Information Retrieval. The task is the shortening a text document in order to create a summary with the salient points of the original document. In short, the main idea of summarisation is to find a subset of data which contains the "information" of the entire set.
The main phases of summarisation are: analysis (examining the input information), determining the salient points and synthesis (constructing a concise representation).

There are 2 main techniques to summarisation:

a) Extraction which is the process of selecting original pieces from the source document(s) and concatenating them to create a summarised text. Each text unit is assigned a weighting by certain features such as: unit's location in the source text, how often it occurs, appearance of cue phrases and statistical significance
b) Abstractive methods build an internal semantic representation and then use natural language generation techniques to create a summary that is closer to what a human might express.

Multi-document summarization is challenging due to the thematic diversity within a set of documents and potential problems include:

1. Redundancy; ideally, we would like to extract sentences that both contain the main ideas and are "diverse" (i.e. they differ from one another).
2. Although not relevant to this assignment, a group of articles may contain a temporal dimension, typical in a stream of news reports about an unfolding event. Thus, later information may override early accounts.
3. The compression ratio (size of summary with respect to document set) will be much smaller than for single document summarisation.

Explanation of how code works and the comparison of different techniques

Term Frequency (tf) measures how frequently a term occurs in a document following the intuition that the words that appear most frequently are important! Therefore, the weight of a term that occurs in a document is proportional to this term frequency. Implementation possibilities include:

- Boolean frequencies: 1 if term occurs in document and 0 otherwise
- Logarithmically scaled frequency: $tf(t,d) = \log(1 + f_{t,d})$
- Augmented frequency (to prevent a bias towards longer documents) e.g. raw frequency divided by the raw frequency of the most occurring term in the document:

Algorithm creates a term matrix where each row is a word, each column a document and the entry being the number of times that word appears in that document. Algorithm begins by looping through all the documents in the corpus (collection of documents) and reading them into a *cluster*. This is followed by tokenization of the words (our basic text unit) in a sentence.
Algorithm them removes stops words (*function words which are not important from the view of summary generation and hence are not considered for scoring a sentence*) which are defined at the beginning of the file.
Term matrix is built by looping through all the words to see if the word is already represented by a row in the term matrix. If it is, the count for that document is updated. If not, a new row is created for that word, and the count is set to 0 for all previous documents and set count to 1 for the current document. To split a text file into its sentences, the algorithm employs the use of Python's Natural Language Toolkit and in particular, tokenize.

To get a word's weight, the algorithm counts the number of occurrences of each word in the document(s), and divides by the total number of words. If the word is a keyword, the value/weight is multiplied by a scaling factor. Rapid Automatic Keyword Extraction algorithm is used to generate keywords. This was included to give greater importance/value to those words which are most relevant to the topic.

To calculate sentence weight for a given document, the algorithm sums the weight of all the words in the sentence and divides by the number of words in the sentence to ensure that there is no bias towards longer sentences. Moreover, since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is divided by the document length as a way of further normalization. Sentences that contain lexical / phrasal summary cue words are given greater weighting as # cue words are often used before summarising a point containing crucial info.

The algorithm includes location/structural information when calculating the sentence weights. Sentence reference index gives more weight to a sentence that precedes sentences containing a pronominal reference. Thus, if a sentence contains a pronoun then the weight of the preceding sentence is increased. Moreover, following the *inverted pyramid for news articles*, sentences at beginning of document contain the most important info thus given a higher weighting than the sentences in the middle of a document.

The best sentences may be very long, and if the summary is word limited (<200 words) it is essential to use smaller sentences in the summary to get all the information across. Therefore, when the summary is <200 words, longer sentences are given a reduced/penalised weighting (multiplied by a scaling factor <1)

The algorithm generates a matrix which is a global list of all the sentences in the corpus where elements of the list are of the form [sentence, sentenceWeight, documentNumber, lineNumber]. The best sentence is chosen by looping through matrix are comparing sentence weights. One could now re-run our algorithm with this sentence removed however this added to the time complexity of the algorithm and so this feature was removed. Instead, the chosen best sentence is removed from the list of sentences and algorithm searches for next highest scoring sentence. Sentences may be similar and so to avoid redundancy, every time a 'best sentence' is chosen, it is compared to all previously chosen sentences, testing for similarity. If the sentences are too similar (i.e. similarity score is above a pre-defined threshold value) algorithm ignores this sentence and continues.

The report was generated by iteratively obtaining the sentence of greatest weight and adding it to a list until the length of the report (number of words) was greater than the required threshold. When deciding in what order the best sentences were in, the algorithm incorporates a structural/location based approach, looking at where each sentence was located in its original document. Sentences were ranked according to their position in the original documents to generate the final, more coherent, summary.

tf–idf (term frequency–inverse document frequency) reflects how important a word is to a document in a collection. The tf-idf value increases proportionally to the number of times a given word appears in a document and is offset by the frequency of the word in the collection, which helps to adjust for the fact that some words appear more frequently in general. In short, the specificity of a term can be quantified as an inverse function of the number of documents in which it occurs. As a result, tf-idf can remove the need for a list of stop words.

The tf-idf program is very similar to tf however when calculating a word weight, the value is multiplied by the word's idf value. This value is calculated by the following equation:
$idf(t, D) = \log(\frac{N}{d \in D : t \varepsilon d})$, where N = total number of documents in corpus and denominator is the number of documents d where term t appears. If a word appears in all documents, the word obtains an idf score of 0.

Evaluation of results

In principle, there two ways to intrinsically evaluate summarisation:
User-based: users are asked to judge the quality of the summary. A typical question could be as simple as "how did you like the summary?", or something more complex regarding the coherence of the summary, or whether it was helpful to understand the full text.
System-based: gold standard summaries are produced by human judges, and the system-generated summaries are compared against them.

Here we focus on the former and the multi-document summary quality criteria are as follows where each is given a rating from 1-5 where 5 is a perfect score for a given metric:

1. Clear structure, including an outline of the main content.
2. Gradual transition from more general to more specific thematic aspects
3. Readability
4. No dangling references to what is not mentioned or explained in the overview
5. Semantic redundancy
6. Key-words included

Metric	200 words	300 words	400 words
1	4	3	3
2	4	4	3
3	4	4	3
4	3	3	3
5	4	4	3
6	4	4	4
Average	3.7	3.5	3.2

Table to left is for the **tf-idf algorithm**. My general analysis is that the general readability of the summary report decreases as the length of the report increases. This is due to the increased potential for redundancy and dangling references. Moreover, for longer summaries, the first sentence which often used to introduce the outline of the content is not always relevant to entire summary. The summaries provide a gradual transition from more general to more specific thematic aspects and they do not end abruptly or in the middle of a description. Finally, the summaries are good at including the relevant keywords.

Example and analysis of 100-word summary (tf)

*Falcon Heavy's successful launch opened a new chapter for SpaceX. Two times now, SpaceX has sent a Dragon cargo capsule to space and back two times. Surreal, yes. The reusability didn't stop with the Dragon capsule for this mission. There's no Dragon capsule that's been used three times - yet. The rocket is carrying Elon Musk's red Tesla Roadster. "We have invested massively in the ISS. For the meantime, SpaceX continues to render services through the Falcon 9 and Falcon Heavy. Wow. So, there's something after Heavy. The Big F***ing Rocket. Mr Musk added: "It's all hands on deck for Crew Dragon."*

Summary produced reads coherently and has no only one grammatical errors (highlighted). The first line works well as an introduction for the topic, introducing the topics (Falcon Heavy and SpaceX) providing an outline of main content and there is a gradual transition from more general to more specific thematic aspects. Moreover, there are no dangling references to what is not mentioned or explained beforehand. In the penultimate sentence, the summary correctly states that the "Big F***ing Rocket" is coming "after Heavy"; the summary flows, has good readability and has no incorrect references. The short sentence "Surreal, yes" fits in very well with the preceding sentence although it does not provide any additional information on the topic. The algorithm has given greater weighting to smaller sentences in an attempt to include as many of the available words as possible as the summary length is small. There is no semantic redundancy as no same point is included twice and out of the major keywords I defined in the program, the summary includes 90% of them, an impressive achievement for a small summary.

Conclusion (How I would adapt algorithm given more time to produce a better-quality summary)

The best sentences may be very long and so, when the summary is word limited, I would implement a box packing algorithm to use as many of the available words as possible. I would improve the semantic similarity between sentences by using Jaccard or Cosine similarity. I would do more work on feature based approach- identify the features that reflects the relevance of that sentence. These features include: position, presence of title word and cue words for example. One idea would be to may convert the selection of 'best' sentences into an optimization problem where a collection of sentences is chosen, considering the constraint that it should maximize overall importance and coherency and minimise the redundancy. Another potential solution to the redundancy problem is by generating ideograms that represent the meaning of each sentence in each document and then evaluates similarity qualitatively by comparing the shape and position of said ideograms. The text summarization technique used above does not consider the semantics of words. Ideally, I would combine the summarization technique with a semantic-based knowledge base. Finally, I would incorporate latent semantic analysis: identify most important topics in each document and then select the best sentence to cover each topic.