Automatic text summarization (https://en.wikipedia.org/wiki/Automatic_summarization) is a process that takes a source text and presents the most important content in a condensed form in a manner sensitive to the user or task needs. The importance of having a text summarization system has been growing with the rapid expansion of textual information available on-line (web pages, news articles, e-mail messages, tweets, or even reviews of products and services). One major differentiation between the different summarization approaches is the input, which can either be a single document or multiple documents. In this project we are interested in the automatic summarization relies on multiple input documents that are somehow related (e.g., about the same topic). Another key differentiation of existing approaches is how the summary is created: this can either be a concatenation of extracted sentences or an artificially generated text, based on the sentences and information included in the input documents.

Text summarization using MapReduce

In this project, you are asked to implement a MapReduce version of a multi-document summarization system based on the extractive approach. Specifically, you are asked to implement the system that is described in [1]. A textual corpus of around 4000 legal cases for automatic summarization is selected for performing the experiments, the dataset is available on UCI machine learning repository (https://archive.ics.uci.edu/ml/datasets/Legal+Case+Reports). The dataset contains Australian legal cases from the Federal Court of Australia (FCA) all files from the year 2006, 2007, 2008 and 2009.

![Methodology of multi-document summarization](image)
As depicted in Figure 1 the summarization task is performed in four different stages and provides a modular system of multiple documents summarization:

(a) The first stage is the document **clustering** (https://en.wikipedia.org/wiki/Cluster_analysis) stage where a text clustering algorithm (e.g., k-means) is applied on the multi document text collection to create the document clusters. The purpose of this stage is to group the similar text document for making it ready for summarization and ensures that all the similar set of documents participates as a group in summarization process.

(b) In the second stage, **Latent Dirichlet Allocation (LDA)** (https://en.wikipedia.org/wiki/Latent_Dirichlet_allocation) topic modeling technique is applied on each individual text document cluster to generate the cluster topics and important terms belonging to each cluster topic.

(c) In the third stage, **global frequent terms** are identified across all document clusters, as well as **Semantic Terms**, i.e., synonyms of the frequent terms using lexical ontologies or thesauri like Wordnet.

(d) In the last stage, **sentence filtering** is performed from each individual input text document on the basis of frequent and semantic similar terms generated from previous stage. For each document the sentences which are containing the frequent terms and semantic similar terms to the frequent terms are selected for participation in the summary document. Finally, the duplicate and near duplicate sentences are identified and kept only once in the final summary document generated.

In this project, you will implement those stages in increasing order of complexity (c), (a), (b) and (d).

**Phase 1 (5%): Frequent and Semantic Terms (Stage (c)) – Deadline: October 10**

The first phase of the project is almost as easy as the word count example in MapReduce. You are given a set of documents, as a collection of sentences, and you are asked to find the $k$ most frequent words (assume $k = 15$) overall. Once you find the $k$ most frequent words, you can use the Java API of Wordnet, to find the synonyms of those words. The set of frequent words (terms), along with their synonymous words will be the output of this phase (order is not important). You must count only the words in the contents of the <sentence> elements, in the files of “/corpus/fulltext/”.

**LINKS:** The Java API of Wordnet can be found here: https://sourceforge.net/projects/jwordnet/.


**Bibliography**
