**Document clustering**

Here are two very short texts to compare:

1. Julie loves me more than Linda loves me
2. Jane likes me more than Julie loves me

We want to know how similar these texts are, purely in terms of word counts (and ignoring word order). We begin by making a list of the words from both texts:

me Julie loves Linda than more likes Jane

Now we count the number of times each of these words appears in each text:

me 2 2

Jane 0 1

Julie 1 1

Linda 1 0

likes 0 1

loves 2 1

more 1 1

than 1 1

We are not interested in the words themselves though. We are interested only in those two vertical vectors of counts. For instance, there are two instances of 'me' in each text. We are going to decide how close these two texts are to each other by calculating one function of those two vectors, namely the cosine of the angle between them.

The two vectors are, again:

a: [2, 1, 0, 2, 0, 1, 1, 1]

b: [2, 1, 1, 1, 1, 0, 1, 1]

The cosine of the angle between them is about 0.822.

Overview[[edit](https://en.wikipedia.org/w/index.php?title=Document_clustering&action=edit&section=1)]

Document clustering involves the use of descriptors and descriptor extraction. Descriptors are sets of words that describe the contents within the cluster. Document clustering is generally considered to be a centralized process. Examples of document clustering include web document clustering for search users.

The application of document clustering can be categorized to two types, online and offline. Online applications are usually constrained by efficiency problems when compared to offline applications.Text clustering may be used for different tasks, such as grouping similar documents (news, tweets, etc.) and the analysis of customer/employee feedback, discovering meaningful implicit subjects across all documents.

In general, there are two common algorithms. The first one is the hierarchical based algorithm, which includes single link, complete linkage, group average and Ward's method. By aggregating or dividing, documents can be clustered into hierarchical structure, which is suitable for browsing. However, such an algorithm usually suffers from efficiency problems. The other algorithm is developed using the [K-means algorithm](https://en.wikipedia.org/wiki/K-means_algorithm) and its variants. Generally hierarchical algorithms produce more in-depth information for detailed analyses, while algorithms based around variants of the [K-means algorithm](https://en.wikipedia.org/wiki/K-means_algorithm) are more efficient and provide sufficient information for most purposes.[[1]](https://en.wikipedia.org/wiki/Document_clustering#cite_note-manning-1):Ch.14

These algorithms can further be classified as hard or soft clustering algorithms. Hard clustering computes a hard assignment – each document is a member of exactly one cluster. The assignment of soft clustering algorithms is soft – a document’s assignment is a distribution over all clusters. In a soft assignment, a document has fractional membership in several clusters.[[1]](https://en.wikipedia.org/wiki/Document_clustering#cite_note-manning-1):499 [Dimensionality reduction](https://en.wikipedia.org/wiki/Dimensionality_reduction) methods can be considered a subtype of soft clustering; for documents, these include [latent semantic indexing](https://en.wikipedia.org/wiki/Latent_semantic_indexing) ([truncated singular value decomposition](https://en.wikipedia.org/wiki/Truncated_singular_value_decomposition) on term histograms)[[2]](https://en.wikipedia.org/wiki/Document_clustering#cite_note-2) and [topic models](https://en.wikipedia.org/wiki/Topic_model).

Other algorithms involve graph based clustering, ontology supported clustering and order sensitive clustering.

Given a clustering, it can be beneficial to automatically derive human-readable labels for the clusters. [Various methods](https://en.wikipedia.org/wiki/Cluster_labeling) exist for this purpose.

Clustering in search engines[[edit](https://en.wikipedia.org/w/index.php?title=Document_clustering&action=edit&section=2)]

A [web search engine](https://en.wikipedia.org/wiki/Web_search_engine) often returns thousands of pages in response to a broad query, making it difficult for users to browse or to identify relevant information. Clustering methods can be used to automatically group the retrieved documents into a list of meaningful categories, as is achieved by e.g. open source software such as [Carrot2](https://en.wikipedia.org/wiki/Carrot2).

Procedures[[edit](https://en.wikipedia.org/w/index.php?title=Document_clustering&action=edit&section=3)]

In practice, document clustering often takes the following steps:

1. [Tokenization](https://en.wikipedia.org/wiki/Tokenization_(lexical_analysis))

Tokenization is the process of parsing text data into smaller units (tokens) such as words and phrases. Commonly used tokenization methods include [Bag-of-words model](https://en.wikipedia.org/wiki/Bag-of-words_model) and [N-gram model](https://en.wikipedia.org/wiki/N-gram_model).

2. [Stemming](https://en.wikipedia.org/wiki/Stemming) and [lemmatization](https://en.wikipedia.org/wiki/Lemmatization)

Different tokens might carry out similar information (e.g. tokenization and tokenizing). And we can avoid calculating similar information repeatedly by reducing all tokens to its base form using various stemming and lemmatization dictionaries.

3. Removing [stop words](https://en.wikipedia.org/wiki/Stop_words) and [punctuation](https://en.wikipedia.org/wiki/Punctuation)

Some tokens are less important than others. For instance, common words such as "the" might not be very helpful for revealing the essential characteristics of a text. So usually it is a good idea to eliminate stop words and punctuation marks before doing further analysis.

4. Computing term frequencies or [tf-idf](https://en.wikipedia.org/wiki/Tf-idf" \o "Tf-idf)

After pre-processing the text data, we can then proceed to generate features. For document clustering, one of the most common ways to generate features for a document is to calculate the term frequencies of all its tokens. Although not perfect, these frequencies can usually provide some clues about the topic of the document. And sometimes it is also useful to weight the term frequencies by the inverse document frequencies. See [tf-idf](https://en.wikipedia.org/wiki/Tf-idf" \o "Tf-idf) for detailed discussions.

5. Clustering

We can then cluster different documents based on the features we have generated. See the algorithm section in [cluster analysis](https://en.wikipedia.org/wiki/Cluster_analysis) for different types of clustering methods.

6. Evaluation and visualization

Finally, the clustering models can be assessed by various metrics. And it is sometimes helpful to visualize the results by plotting the clusters into low (two) dimensional space. See [multidimensional scaling](https://en.wikipedia.org/wiki/Multidimensional_scaling) as a possible approach.

Clustering v. Classifying[[edit](https://en.wikipedia.org/w/index.php?title=Document_clustering&action=edit&section=4)]

Clustering algorithms in computational text analysis groups documents into grouping a set of text what are called subsets or *clusters* where the algorithm's goal is to create internally coherent clusters that are distinct from one another.[[3]](https://en.wikipedia.org/wiki/Document_clustering#cite_note-3) Classification on the other hand, is a form of [supervised learning](https://en.wikipedia.org/wiki/Supervised_learning) where the features of the documents are used to predict the "type" of documents.

Key Steps

1. **Read data**: read titles, genres, synopses, rankings into four arrays
2. **Tokenize and stem**: break paragraphs into sentences, then to words, stem the words (without removing stopwords) - each synopsis essentially becomes a bag of stemmed words.
3. **Generate tf-idf matrix**: each row is a term (unigram, bigram, trigram...generated from the bag of words in 2.), each column is a synopsis.
4. **Generate clusters**: based on the tf-idf matrix, 5 (or any number) clusters are generated using k-means. The top key terms are selected for each cluster.
5. **Calculate similarity**: generate the cosine similarity matrix using the tf-idf matrix (100x100), then generate the distance matrix (1 - similarity matrix), so each pair of synopsis has a distance number between 0 and 1.
6. **Plot clusters**: use multidimensional scaling (MDS) to convert distance matrix to a 2-dimensional array, each synopsis has (x, y) that represents their relative location based on the distance matrix. Plot the 100 points with their (x, y) using matplotlib (I added an example on using plotly.js).