LAB: IR and text categorization with VSM

- These assignments include different parts: each will be provided with its own score, so that everyone is allowed to choose what best suits her/his preferences.

Document Similarity and Retrieval using the VSM

Document Similarity and Retrieval using the VSM

A. With Sparse Embeddings

- Use the News Category Dataset available on Kaggle:
  https://www.kaggle.com/datasets/rmisra/news-category-dataset
- This dataset contains ~200k news headlines from HuffPost, classified into categories such as politics, tech, and sports.

• Setup and Preprocessing

- Load the dataset and randomly select a subset of headline texts (e.g., 10,000).
- Preprocess the text through tokenization, lowercasing, stopwords and punctuation removal, lemmatization.

Document Similarity and Retrieval using the VSM

- Vector Space Model representation
  - Convert the preprocessed documents to TF-IDF vectors (use `TfidfVectorizer` from scikit-learn)
  - Store the resulting matrix and vocabulary
- Document Retrieval
  - Prompt the user for a query (e.g., "government budget")
  - Preprocess the query using the same pipeline
  - Convert the query to a TF-IDF vector using the same vectorizer
  - Compute the cosine similarity between the query and all documents
  - Return and display the top 5 most relevant documents

Document Similarity and Retrieval using the VSM

- Analysis
- Manually inspect the top results: do they seem relevant?
- Try at least 10 queries from different categories and observe the results

Document Similarity and Retrieval using the VSM

B. With the Rocchio Method

- Data selection
- There are a total of 42 news categories in the dataset. The top-15 categories and corresponding article counts cover above 70% of the whole data, and are as follows
  - Select only news from these classes, and apply a 90/10 split (90% for building profiles, and 10% for testing).

- POLITICS: 35602
- WELLNESS: 17945
- ENTERTAINMENT: 17362
- TRAVEL: 9900
- STYLE & BEAUTY: 9814
- PARENTING: 8791
- HEALTHY LIVING: 6694
- QUEER VOICES: 6347
- FOOD & DRINK: 6340
- BUSINESS: 5992
- COMEDY: 5400
- SPORTS: 5077
- BLACK VOICES: 4583
- HOME & LIVING: 4320
- PARENTS: 3955

Document Similarity and Retrieval using the VSM

- Categorization task

- The task is to categorize the items in the test-set.
- Accuracy over classes must be recorded, as well as the average over the 15 classes.
- split data into 10 folds (90/10), repeat the classification as many times, and record average accuracy.

Document Similarity and Retrieval using the VSM

- The solution should be based on the following steps:
- Preprocessing
- Clean and normalize text (lowercase, punctuation removal, etc.), tokenize and remove stopwords
- Use `TfidfVectorizer` from scikit-learn to convert documents to TF-IDF vectors
- Split data into training and test sets
- For each class compute the centroid based on the set of training documents for that class
- For each test document compute cosine similarity to all class centroids
- Assign the label of the closest centroid

Document Similarity and Retrieval using the VSM

C. With Dense Embeddings

- In this assignment we use dense (Word2Vec) embeddings within the Vector Space Model (VSM) framework.
- We start by training a model, whose vectors will then be employed to represent the documents
  - create a text corpus using the aforementioned News Category Dataset
  - preprocess the text through tokenization, lowercasing, and stopwords and punctuation removal.
  - train a Word2Vec Model (using gensim.models.Word2Vec):
  - choose hyperparameters: vector_size, window, min_count, sg
  - save the trained model for later use

Document Similarity and Retrieval using the VSM

- The Vector Space Model is a general framework where text items (e.g., words, documents, sentences) are represented as vectors in an n-dimensional space, and similarity is computed using vector distances (typically cosine similarity).
- While TF-IDF is sparse and based on word frequency, Word2Vec produces dense and semantic word vectors. These can be used to represent not just words but also larger units like sentences and documents by aggregation.

Document Similarity and Retrieval using the VSM

- Different from the previous version (Part A), we now represent a document as a vector as follows:
  - Average the Word2Vec vectors of all the words in the document (after preprocessing).
  - Use a weighted average, weighting each word vector by its TF-IDF score.
  - Compare the results obtained on the same set of queries employed in Part A.

Word Clustering Based on Word2Vec Representation

Clustering based on W2V

- Train a Word2Vec Model
  - create a text corpus (e.g., Wikipedia dump, or works from the Project Gutenberg, such as https://www.gutenberg.org/ebooks/).
  - preprocess the text through tokenization, lowercasing, and stopwords and punctuation removal.

Clustering based on W2V

- train a Word2Vec Model (using `gensim.models.Word2Vec`):
  - choose hyperparameters: vector_size, window, min_count, sg
  - save the trained model for later use
- explore the Model:
  - use `.most_similar()` to explore semantic neighborhoods
  - visualize some vectors using PCA or t-SNE

Clustering based on W2V

- Words Clustering
  - Choose a list of words
  - Cluster them using scikit-learn's KMeans
  - Visualize the resulting clusters using t-SNE or PCA