Text Representations

Objective: Generate document representations using vector space models (TF/TF-IDF) and semantic vector models (word embeddings) to analyze a subset of the "20 Newsgroups" dataset.

This practice is publicly available at
https://github.com/alvaro-francisco-gil/text-mining/tree/main/02_text_representation

Extract Message Body

1. **Remove Headers**: Identify and remove everything before the first blank line in the email.
2. **Filter Out Email Addresses**: Remove lines containing email addresses using a regex pattern.
3. **Exclude Proper Nouns**: Remove lines with only 2-3 capitalized words.
4. **Remove Signatures**: Detect and exclude lines matching common signature patterns (e.g., "--", "Kind regards", "Sent from my iPhone") and any subsequent lines until a blank line is encountered.
5. **Preserve Quoted Lines**: Retain quoted lines (starting with ">") unless they are part of a signature or irrelevant.

These first steps ensure the extracted message body focuses on semantically meaningful content by removing irrelevant or noisy elements such as headers, email addresses, signatures, and isolated proper nouns. This reduces noise, enhances dataset consistency, and avoids skewing vector representations in models like TF/TF-IDF or word embeddings, as proper nouns or email addresses provide little to no semantic relevance to the document's topic.

Preprocessing

6. **Remove Punctuation**: All punctuation is stripped from the text.
7. **Remove Numbers**: Numbers are removed if the remove_numbers flag is set to True.H
8. **Convert to Lowercase**: The entire text is converted to lowercase for uniformity.
9. **Tokenization**: The text is split into individual words.
10. **Remove Stop-Words**: Common English stop-words are removed using an NLTK stop-word list.
11. **Lemmatization or Stemming**: Words are normalized by applying lemmatization (default) using WordNetLemmatizer or stemming using PorterStemmer, based on the use_lemmatization flag.

These preprocessing steps refine the text for more effective vector representation and analysis by ensuring uniformity and removing irrelevant elements. Stripping punctuation and numbers eliminates unnecessary symbols and data that do not contribute to semantic meaning, while converting text to lowercase standardizes it, avoiding duplication caused by case sensitivity. Tokenization splits the text into individual words for granular processing, and removing stop-words filters out commonly used terms that carry little contextual value. Stemming was applied in all cases to reduce words to their root forms, ensuring consistency and simplicity in representations. Stemming was preferred over lemmatization to achieve more accurate results in identifying word roots, as speed was not a constraint.

TF and TF-IDF Representations

12. **TF (Term Frequency)**: A TfidfVectorizer is used to compute the raw term frequency matrix.
13. **TF-IDF (Term Frequency-Inverse Document Frequency)**: A standard TfidfVectorizer computes the TF-IDF matrix for the documents.

Computing the Term Frequency (TF) matrix captures how often terms appear in each document, reflecting their importance within that document. Using TF-IDF adjusts these frequencies by downweighting common terms across documents and upweighting rare but significant terms, enhancing the representation's focus on terms that are most discriminative for a document's content.

Word2Vec Embeddings

14. For each preprocessed document, word embeddings are retrieved from a pre-trained Word2Vec model.
15. Two representations are generated:

    - **Average Vector**: The mean of all word vectors in the document.
    - **Sum Vector**: The sum of all word vectors in the document.

Using pre-trained Word2Vec embeddings captures the semantic meaning of words by representing them as dense vectors. For each document, generating an average vector aggregates the overall semantic context of the document, while the sum vector emphasizes the cumulative semantic contribution of all words. These representations enable capturing both the general meaning and the document's emphasis on specific terms.

Output

1. **TF Matrix**: A sparse matrix representing term frequencies for each document.
2. **TF-IDF Matrix**: A sparse matrix representing weighted term frequencies (TF-IDF) for each document.
3. **Word2Vec Average Vectors**: Dense vectors representing the average of word embeddings for each document.
4. **Word2Vec Sum Vectors**: Dense vectors representing the sum of word embeddings for each document.

Conclusions

- This practice demonstrates how to preprocess text data and generate multiple types of document representations (TF, TF-IDF, and Word2Vec embeddings). Each representation captures different aspects of textual information.
- **TF and TF-IDF** are effective for sparse representations and are widely used in traditional machine learning models. However, they do not capture semantic relationships between words.
- **Word2Vec embeddings**, on the other hand, provide dense and semantically meaningful representations by leveraging pre-trained word vectors. These are particularly useful in deep learning applications or tasks requiring semantic understanding.
- Combining these representations can enhance downstream tasks like classification, clustering, or similarity analysis by leveraging both lexical and semantic features.