Amrita Vishwa Vidyapeetham
Amrita School of Computing, Amritapuri
BTech, Third Year, 6 Semester

---

1 Objective

The objective of this lab is to understand different word embedding techniques, generate word vectors, visualize them, and compute paragraph similarity using various representations such as One-Hot Encoding, Bag of Words (BoW), TF-IDF, Word2Vec, and FastText.

2 Task 1: Data Preparation

1. Select a dataset containing multiple paragraphs (e.g., news articles, Wikipedia entries, or any text corpus).
2. Preprocess the text:
   - Convert to lowercase.
   - Remove special characters and punctuations.
   - Tokenize the text.
   - Remove stopwords.
   - Perform stemming/lemmatization (optional).

3 Task 2: Generating Word Representations

3.1 One-Hot Encoding

1. Implement One-Hot Encoding on the text corpus.
2. Convert each word into its respective binary vector representation.
3. Display an example of the One-Hot Encoding representation for a few words.

3.2 Bag of Words (BoW)

1. Generate a BoW representation using the tokenized text.
2. Construct a document-term matrix.
3. Display the resulting feature vector representation.

3.3 TF-IDF Representation

1. Compute Term Frequency-Inverse Document Frequency (TF-IDF) for the corpus.
2. Construct a TF-IDF matrix using $sklearn.feature_extraction.text.TfidfVectorizer$.
3. Display sample feature vectors for selected words.

3.4 Word2Vec Representation

1. Train a Word2Vec model using the dataset (`gensim.models.Word2Vec`).
2. Generate vector representations for words.
3. Display the vector representation of sample words.
4. Find and display the most similar words for a given word using cosine similarity.

3.5 FastText Representation

1. Train a FastText model (`gensim.models.FastText`).
2. Generate and display word embeddings.
3. Compare results with Word2Vec and note the differences.

4 Task 3: Visualization of Word Embeddings

1. Use Principal Component Analysis (PCA) or t-SNE to reduce the dimensionality of the word embeddings.
2. Visualize the word embeddings using `matplotlib` or `seaborn`.
3. Interpret the cluster formations in the visualization.

5 Task 4: Paragraph Similarity Computation

1. Represent each paragraph using the following:
   - BoW
   - TF-IDF
   - Word2Vec (average word vectors in the paragraph)
   - FastText (average word vectors in the paragraph)
2. Compute pairwise similarity between paragraphs using cosine similarity.
3. Compare results across different representations and analyze which performs best.

6 Task 5: Comparative Analysis & Conclusion

1. Compare the performance of different word vector representations.
2. Discuss which method captures semantic meaning better and why.
3. Note observations from the similarity computation.