Samuel Reade

Professor Hosseinmardi

Communications 188C

May 13th, 2025

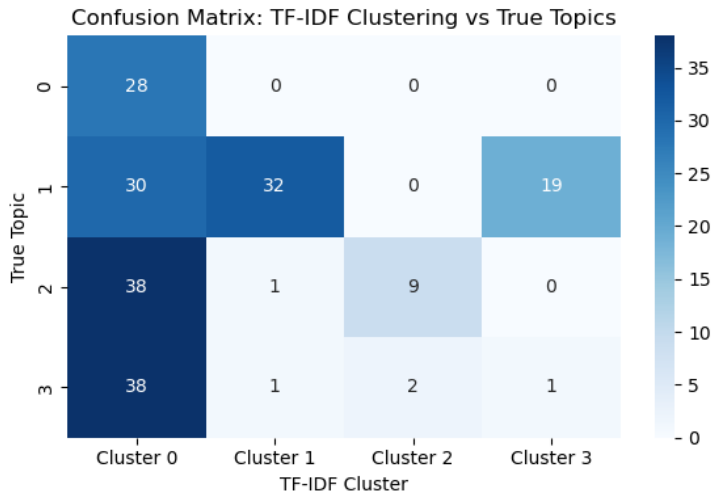
Homework 2 Report

After creating a virtual environment, downloading and calling all the libraries that were needed to complete the assignment, reading in the ‘news_AP.csv’ file was the next step. This was then converted into a pandas dataframe. Once the data was in a pandas data frame the code performs text clustering and evaluation on a dataset of news titles. First, it defines a function ‘preprocess_texts’ that takes a list of text strings and applies several preprocessing steps. This includes converting all letters to lowercase, removing punctuation, tokenizing the text, removing stop words, and filtering out rare words, words that show up only once. The cleaned and filtered tokens are then joined back into strings. This function is applied to the ‘title’ column of the ‘news’ data frame enabling more reliable string data, which would produce better clustering results.

Next, the code vectorizes the preprocessed titles using TF-IDF, ‘TfidfVectorizer’, turning the text data into a numerical matrix which was initialized as ‘titles_tfidf’. In the matrix each row represents a title and each column corresponds to a unique term weighted by its TF-IDF score. The resulting shape and array are printed for inspection.

After vectorizing, the code applies KMeans clustering with four clusters, attempting to accurately categorize the article titles in their correct respective groups. The clustering groups the TF-IDF vectors into clusters based on similarity. The cluster assignments are then added to the ‘news’ data frame and stored in a new column as ‘tfidf_cluster’. To compare the clustering

results with the actual topic labels, the code factorizes the ‘topic’ column and stores them in the ‘topic_label’ column. A confusion matrix is generated to evaluate how well the TF-IDF-based clusters match the true topic labels. The confusion matrix is stored in the ‘conf_df_tfidf’ data frame, and visualized using a heatmap from Seaborn as shown below.

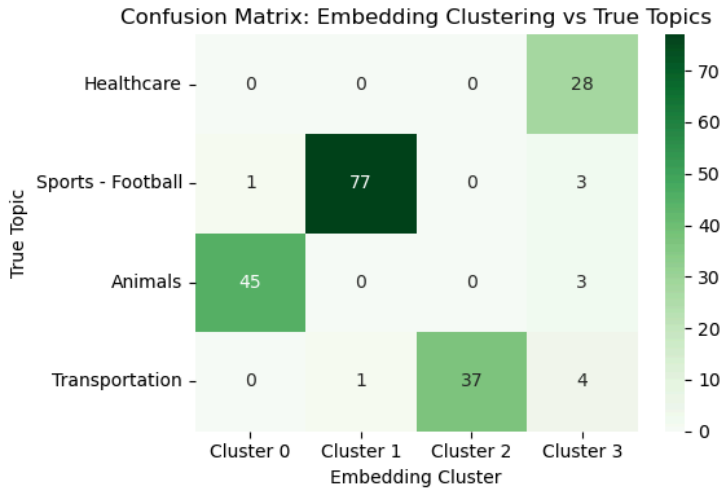


Finally, the code generates a classification report using ‘classification_report’, which includes precision, recall, and F1-scores for each topic, and computes the overall accuracy of the clustering using ‘accuracy_score’. Both techniques are part of the ‘sklearn.metrics package’. This quantifies how well the unsupervised TF-IDF based clustering aligns with the known topic labels.

The next step was to perform text clustering using sentence embeddings by using a pre-trained language model. First ‘SentenceTransformer’ model, ‘all-MiniLM-L6-v2’, is loaded. This model efficiently encodes text into high dimensional vector representations. The same titles used for TF-IDF are passed through this model, producing a matrix of embeddings where each row corresponds to a title, attempting to capture its semantic meaning.

These embeddings are then clustered using the same KMeans algorithm which was previously applied. Four clusters are initialized. The resulting cluster assignments are stored in a new column called ‘embed_cluster’ in the ‘news’ DataFrame.

To evaluate the clustering quality, a confusion matrix is constructed using the same strategy as before. This matrix is converted into a DataFrame ‘conf_df_embed’ with topic names as row labels and cluster indices as column headers. A heatmap is then generated using Seaborn to visually assess how well the embedding clusters performed.



The precision, recall, and F1-scores, and accuracy is then reported. This allows for a direct comparison of how well the embeddings perform compared to TF-IDF. When comparing performance, embeddings outperformed TF-IDF overall, achieving a higher accuracy, 40.7% vs. 35.18%, and a better weighted F1-score, 0.42 vs. 0.36. However, TF-IDF performed better in both precision, 0.50 vs. 0.27, and recall, 0.40 vs. 0.26. Despite this, the higher accuracy and F1-score suggest that embeddings provided a more balanced and effective result.

A Chatgpt generated corpus of titles was also converted to embeddings and clustered. All the same steps were followed for these, which can be seen at the bottom of the .ipynb file.