به نام خدا

دانشگاه تهران

دانشکدگان فنی

دانشکده مهندسی برق و کامپیوتر

درس پردازش زبان طبیعی

پاسخ تمرین شماره دو (سوال اول)

نام و نام خانوادگی: امیرحسین شاه قلی

شماره دانشجویی: ۸۱۰۱۹۹۴۴۱

اسفند ماه ۱۴۰۲

Table of Contents

Answer of the question one

Answer of the first part:

I started preprocessing with lowercasing to unify the representations of words, then I tokenized and lemmatized using Spacy to reduce words to their base forms and ensure consistent features for modeling. I choose lemmatization becouse it considers the full morphological analysis of words, resulting in proper dictionary forms that improve model accuracy. Particularly for sentiment analysis, using lemmas over stems is advantageous. Subsequently, all tokens representing punctuation, numerals, and URLs are filtered out to eliminate potential noise and restrict the dataset to purely textual content. This entire preprocessing pipeline is calibrated to enhance model performance by emphasizing the meaningful content of the tweets while reducing computational overhead and potential sources of misclassification. (**Notice**: I did not delete stopwords which provide clearer differentiation for TF-IDF over TF)

Answer of the second part:

The matrix is saved in a file named "TF_matrix.csv"

Answer of the third part:

The matrix is saved in a file named "TF-IDF_matrix.csv"

In implementing the TF-IDF calculation, I start by converting the sparse matrix of term frequencies into a dense array to manipulate the term occurrences within the documents. For each document, I calculate term frequency by dividing the number of occurrences of each term by the total number of terms in that document, this normalizes the term counts and represents their relative importance within individual documents. Next, I compute the document frequency for each term across all documents, which counts how many documents contain a particular term. The inverse document frequency is then derived using the logarithm to scale down the impact of frequently occurring words across the corpus offset the log with a +1 to prevent division by zero and add an additional +1 to the result so that terms that appear in all documents are not completely ignored. I calculate the TF-IDF score by multiplying TF by IDF, effectively weighting the frequency of terms by their tailored importance. To ensure that document length does not bias the results, I normalize each document's vector to unit norm by dividing it by its Euclidean length, which provides a consistent scale for all documents.

Answer of the fourth part:

The matrix is saved in a file named “PPMI_matrix.csv”

In implementing the compute_ppmi_matrix function, I start by converting the sparse matrix of term frequencies into a dense array. I calculate the total count of all term occurrences across the entire corpus. Then, I sum the occurrences of each word across all documents and the word counts for each document to establish the overall word frequency and document lengths. With these, I calculate an expected co-occurrence frequency matrix by taking the outer product of the document sums and word sums, divided by the total count of the corpus. I then determine the Pointwise Mutual Information by comparing the actual word occurrences against the expected occurrences, deriving a log ratio that quantifies the association strength between words and contexts. As PMI can be negative values for less frequent than expected word-context pairs, I set all negative values to zero to focus on meaningful associations, thus obtaining the PPMI.

Answer of the fifth part:

results:

TF Model Metrics:

Accuracy: 0.7245, Precision: 0.7485970819304153, Recall: 0.6710261569416499, F1-score: 0.7076923076923077

TF-IDF Model Metrics:

Accuracy: 0.727, Precision: 0.751685393258427, Recall: 0.6730382293762576, F1-score: 0.7101910828025477

PPMI Model Metrics:

Accuracy: 0.685, Precision: 0.6952789699570815, Recall: 0.6519114688128773, F1-score: 0.6728971962616822

The TF-IDF model performs a bit better than the plain TF model, with slightly higher correct predictions and balance between precision and recall. The PPMI model doesn't do as well as TF or TF-IDF. This might be because PPMI is focused on word relationships and doesn't capture the overall importance of words across different tweets as effectively as TF-IDF, especially in short texts like tweets where context is limited. As a result, TF-IDF's approach to giving weight to key words helps it more accurately identify the sentiment of tweets.