TF.IDF & Cosine Similarity Programming Assignment

In this assignment, you will implement the TF.IDF and Cosine similarity to compute the similarity between documents. You can use Python to implement your work. In your final submission, you need to provide the solutions to the following questions and the source code of your implementation.

Data:
You can find 10 txt files in the Data folder. Those are the documents in your corpus that you need to process.

Step 1: Tokenization
Apply tokenization on all the documents in your corpus, and answer the following questions: how many tokens and types (unique tokens) in each document, and in the entire corpus (collections)?

Step 2: Stopword Removal
Remove the stop-words from the documents. You can use the list of stop words defined by Python NLTK library, and answer the following questions: how many tokens and types (unique tokens) in each document after removing all the stop words?

Step 3: Stemming/Lemmatization
Now you can apply stemming or lemmatization to reduce words into their root/base word. You can use porter stemmer or wordnet lemmatizer implemented in Python NLTK library, and answer the following questions: how many terms (size of vocabulary) are left in each document after stemming/ lemmatization and what's the total vocabulary size for the entire corpus?

Step 4: Compute TF-IDF
Now you can computer the TF-IDF features for each document in your corpus. (You can use the TfidfVectorizer function provided by python scikit-learn library, or you can have your own implementation from scratch.)

Step 5: Compute the Cosine Similarity
Now you can compute the cosine similarity for every pair of two documents in your corpus and generate the corresponding similarity matrix (You can use the scikit-learn cosine similarity function, or use the numpy's dot and norm functions to compute the cosine similarity). You need to provide the similarity matrix in your final submission.