Problem 1: Document Vectors

Write a MATLAB function `DocSimilarity(D, E)` which computes the “similarity” of text documents D and E using the vector model of documents. Specifically, the arguments D and E are each cell arrays of strings, each string being a word of the document, normalized to lower case. The function returns a number between 0 and 1, 0 meaning that the two documents have no two significant words in common, 1 meaning that they have the identical significant words with the same frequency.

A word is considered “significant” if it has at least three letters and is not in the list of stop words provided at SampleCode/GetStopwords.m on the course website.

A stop word is a very common word that should be ignored.

Your function should execute the following steps.

- Let LargeOdd be any reasonably large odd number that is not very close to a power of 256. 10,000,001 will do fine.
- Load in the cell array of stop words from GetStopwords.m
- Create three sparse vectors \( \vec{S}, \vec{D}, \vec{E} \) of size LargeOdd, as follows: For every word \( W \), let \( i = \text{hash}(W, \text{LargeOdd}) \). You can find a hash function at SampleCode/hash.m. Then
  - \( \vec{S}[i] = 1 \) if \( W \) is on the list of stop words.
  - \( \vec{D}[i] = \text{the number of occurrences of } W \text{ in } D, \text{ if } W \text{ is significant.} \)
  - \( \vec{E}[i] = \text{the number of occurrences of } W \text{ in } E, \text{ if } W \text{ is significant.} \)

  (Create \( \vec{S} \) first, then use it for a quick test for whether words in the documents are significant.) \( \vec{D} \) and \( \vec{E} \) are the document vectors (we omit the inverse document frequency).
- Return the quantity \( \vec{D} \cdot \vec{E} / ||\vec{D}|| \cdot ||\vec{E}|| \)

For instance,

```matlab
>> D = { 'how', 'much', 'wood', 'could', 'a', 'woodchuck', 'chuck', ...
      'if', 'a', 'woodchuck', 'could', 'chuck', 'wood' };
>> E = { 'all', 'the', 'wood', 'that', 'a', 'woodchuck', 'could', ...
      'if', 'a', 'woodchuck', 'could', 'chuck', 'wood' }
>> DocSimilarity(D, E)
ans =
   0.9245
```

Note that the only significant words in these two texts are ”chuck”, ”much”, ”wood”, and ”woodchuck”.

You don’t have to worry about hash collisions, because they are very infrequent, and the technique is completely imprecise in any case.
Problem 2: Evaluating a linear classifier

Consider a classification problem of the following kind: You are given \( n \) numeric features of an entity and you want to predict whether or not the entity belongs to a specific category. For example, based on measurable geometric features, is a given image a picture of a camel? Based on test results, does a patient have diabetes? Based on financial data, is a prospective business a good investment?

A linear classifier consists of a \( n \)-dimensional weight vector \( \vec{W} \) and a numeric threshold \( T \). The classifier predicts that a given entity \( \vec{X} \) belongs to the category just if \( \vec{W} \cdot \vec{X} \geq T \).

Suppose, now, that you have a data set consisting of collection \( D \) of \( m \) entities together with the associated correct labels \( \vec{L} \). Specifically, \( D \) is an \( m \times n \) matrix, where each row is the feature vector for one entity. Vector \( \vec{L} \) is a \( m \)-dimensional column vector, where \( \vec{L}[i] = 1 \) if the \( i \)th entity is in the category and 0 otherwise.

There are a number of different ways of evaluating how well a given classifier fits a given labelled data set. The simplest is the overall accuracy which is just the fraction of the instances in the data set where the classifier gets the right answer.

Overall accuracy, however, is often a very unhelpful measure. Suppose you are trying to locate pictures of camels in a large collection of images collected across the images. Then, since images of camels constitutes only a very small fraction of the collection, you can achieve a high degree of overall accuracy simply by rejecting all the images. Clearly, this is not a useful retrieval engine.

In this kind of case, the most commonly used measures are precision/recall. Let \( C \) be the set of entities which are actually in the category (i.e. labelled so by \( \vec{L} \)); let \( R \) be the set of entities which the classifier predicts are in the category; and let \( Q = C \cap R \). Then precision is defined as \( |Q|/|R| \) and recall is defined as \( |Q|/|C| \). In the camel example, the precision is the fraction of images that are actually camels, out of all the images that the classifier identifies as camels; and the recall is the fraction of the images of camels in the collection that the classifier accepts as camels.

A. Write a Matlab function \texttt{evaluate}(D,L,W,T) which takes as arguments \( D, L, W, \) and \( T \), as described above, and which returns the overall accuracy, the precision, and the recall.

For example let \( m = 6, n = 4 \),

\[
D = \begin{bmatrix}
1 & 1 & 0 & 4 \\
2 & 0 & 1 & 1 \\
2 & 3 & 0 & 0 \\
0 & 2 & 3 & 1 \\
4 & 0 & 2 & 0 \\
3 & 0 & 1 & 3
\end{bmatrix} \quad \quad \quad L = \begin{bmatrix}
1 \\
1 \\
0 \\
0 \\
0 \\
0
\end{bmatrix} \quad \quad \quad W = \begin{bmatrix}
1 \\
2 \\
2 
\end{bmatrix} \quad \quad \quad T = 9
\]

Then the classifications returned by the classifier are \( [1,0,0,1,0,1] \); the first, third, and fifth rows are correctly classified and the rest are misclassified. Thus, the accuracy is \( 3/6 = 0.5 \). The precision is \( 1/3 \); of the three instances identified by the classifier, only one is correct. The recall is \( 1/2 \); of the two actual instances of the category, one is identified by the classifier.

B. Write a function \texttt{evaluate2}(D,L,W,T). Here the input arguments \( D \) and \( L \) are as in part (A). \( W \) and \( T \), however, represent a collection of \( q \) classifiers. \( W \) is a \( n \times q \) matrix; \( T \) is a \( q \)-dimensional vector. For \( j = 1 \ldots q \), the column \( W[:,j] \) and the value \( T[j] \) are the weight vector and threshold of a classifier. \( \texttt{E=evaluate2}(D,L,W,T) \) returns a \( 3 \times q \) matrix, where, for \( j = 1 \ldots q \), \( \texttt{E}[1,j], \texttt{E}[2,j] \) and \( \texttt{E}[3,j] \) are respectively the accuracy, precision, and recall of the \( j \)th classifier.
For example, let D and L be as in part (A). Let $q = 2$ and let

$$W = \begin{bmatrix} 1 & 0 \\ 2 & 0 \\ 1 & 0 \\ 2 & 1 \end{bmatrix} \quad T = [9, 2]$$

Then evaluate $2(D, L, W, T) =$

$$\begin{bmatrix} 0.5 & 0.6667 \\ 0.3333 & 0.5 \\ 0.5 & 0.5 \end{bmatrix}$$

**Problem 3: Plotting the harmonic series**

The harmonic function $H(n)$ is defined as $H(n) = \sum_{i=1}^{n} 1/i$. For instance $H(4) = 1/1 + 1/2 + 1/3 + 1/4 = 25/12 = 2.1666$. For large $n$, $H(n)$ is approximately equal to $\ln(n) + \gamma$, where $\ln(n)$ is the natural logarithm and $\gamma = 0.5772$, known as Euler’s constant.

Write a MATLAB program `PlotHarmonic(N)` that shows both $H(k)$ and $\ln(k) + \gamma$ for $k = 1 \ldots N$. 