Identifying related bodies of text using TF-IDF vectorization.

Five-part summary of the article.
Assignment 7.

Tania Cajal (MBDS 2021/2022)

1. What is term frequency-inverse document frequency vectorisation?
    a. A simple way to characterise bodies of text, great for large datasets.
    b. TF-IDF vectorization is a useful tool for text analytics. It allows you to quantify the similarity of different documents.
    c. It consists of tokenization - we'll split up the sentences into a list of words, removing capitalization.
2. Definitions
    a. Term frequency: how often a word comes up.
    b. IDF: establishes a scoring mechanism for the usefulness of a given term - higher scores mean more information is present.
3. Applying it
    a. Log-frequency lessens the difference between large term counts (ie. 1 vs 20 is treated as a larger difference than 100 vs 150). Also, this is element-wise multiplication. The TF score for "similarity" is multiplied by the IDF score for "similarity."
    b. The result: a TF-IDF vector for each text sample.
4. Cosine similarity
    a. Even if the magnitudes of two specific vectors are vastly different, they may be considerably more similar than others that are closer together.
    b. Each TF-IDF vector is normalised with respect to its length to adjust for variable document lengths.

c. You may simply take the dot product to obtain the cosine similarity after length-normalizing the two TF-IDF vectors you want to compare.

5. Conclusion

   a. TF-IDF vectors are generally very sparse, as an individual document will only contain a small subset of the vocabulary found across the collection of documents.

   b. SKLearn's TF-IDF vectorizer has lots of features, such as tokenization, n-grams (which creates tokens for collections of words in addition to individual words), and stopword removal.

Example Code

Here's a quick example using SKLearn's TF-IDF vectorizer module.

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

tfidf_vectorizer = TfidfVectorizer()
print "Fitting TF-IDF matrix..."
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
print "Calculating cosine similarity..."
dist = 1 - cosine_similarity(tfidf_matrix)
```