Latent Semantic Indexing

Julia L. Wang

wordVecV.mat: a 1651x10 vector V (t x d) of *term frequencies* $f_{\text{term}}(t,d)$: the number of times word w_t appears in d

wordVecArticles.txt: set of text from 10 wikepedia articles (D), 1 article / line (d)

wordVecTitles.txt: corresponding titles of the articles

wordVecWords.txt: pre-processed set for the union of words in all articles W, let each word be w with index t

```
% initialization
clear
load 'wordVecV.mat' %t row and d column
D = size(V, 2); %there are 10 articles/documents
W = size(V, 1);
```

Latent semantic indexing to discover text documents similar to each other. From a set of documents D_1 to D_m, use a bag of words technique where W is the total number of disctinct words. M = matrix `wxd` is the normalized "raw" term-by-document matrix. An interpretation of vectors u_l and v_l (from U, V, S, of SVD) from $l = 1 \ldots r$:

$M = \sum_{l=1}^{r} a_l u_l v_l^T$ where u provides a weighting for each word, v assigns the weighting for each document.

```
M = zeros([W, D]); % size of M is wxd
for i = 1:W
    for j = 1:D
        if V(i,j) > 0
            M(i,j) = 1;
        end
    end
end
M_norm = M; % normalize M
for i = 1:D
    M_norm(:,i) = M(:,i) / norm(M(:,i));
end
```

SVD decomposition of the normalized matrix M, list the 10 largest singular values

```
[U,S,V] = svd(M_norm); % svd

fprintf('The 10 largest singular values are:')
```

```
The 10 largest singular values are:
```

```
% 10 largest singular values
for i = 1:10
    disp(S(i,i));
end
```

```
    1.5366

    1.0192

    0.9587

    0.9539

    0.9413

    0.9289

    0.8977

    0.8919

    0.8687

    0.8161
```

M is close to a low-rank matrix in real life. In the latent semantic indexing approach, we project the documents and query onto the subspace generated by u_1 to u_k with cos similarity to the projected vectors. We measure similarity via:

1. Projecting the coordinates of a document in the orthonormal basis formed by all the u's onto the subspace spanned by the first k vectors of the basis
2. Projecting the query onto the subspace spanned by u1 to uk
3. Calculating the angle between these projections

Assuming the rank of the approximation is k=9. Let distance $d(i, j)$ be the distance between the ith and jth documents, write the titles of the 2 most similar documents

```
file = fopen('wordVecTitles.txt');
page = textscan(file, '%s', 'delimiter', '\n');

% 2 most similar documents when k = 9
k=9;
[doc1, doc2] = similarityApprox(W, D, k, M_norm, U);
fprintf('2 most similar documents when k= %d:', k)
```

```
2 most similar documents when k= 9:
```

```
disp(page{1}{doc1})
```

```
Barack Obama
```

```
disp(page{1}{doc2})
```

```
George W. Bush
```

```
% 2 most similar documents from k=8 to 1
docs = zeros([8,2]);
for i = 1:8
    [docs(i,1), docs(i,2)] = similarityApprox(W, D, i, M_norm, U);
```

```
    fprintf('2 most similar documents when k= %d:', i)
    disp(page{1}{docs(i,1)})
    disp(page{1}{docs(i, 2)})
end
```

```
2 most similar documents when k= 1:
Jessica Feshbach
Susie Au
2 most similar documents when k= 2:
B. J. Cole
John Holland (composer)
2 most similar documents when k= 3:
Barack Obama
George W. Bush
2 most similar documents when k= 4:
Barack Obama
George W. Bush
2 most similar documents when k= 5:
Barack Obama
George W. Bush
2 most similar documents when k= 6:
Barack Obama
George W. Bush
2 most similar documents when k= 7:
Barack Obama
George W. Bush
2 most similar documents when k= 8:
Barack Obama
George W. Bush
```

Function

Implementation of the document similarity approximation described above.

```
function [doc1, doc2] = similarityApprox(W, D, k, M_norm, U)
U_k = U(:, 1:k); % subspace to project onto (u1, ..., uk)

% projecting the document
proj_M_k = zeros(W,D);
for j = 1:D
    d_j = M_norm(:,j);
    for i = 1:k % projmk = projection of d_j onto U_k
        proj_M_k(:,j) = proj_M_k(:,j) + dot(d_j, U_k(:,i)) * U_k(:,i) / (norm(U_k(:,i))^2);
    end
end
% second projection
similarityMatrix = zeros(D,D);
for i = 1:D
    proj_i = proj_M_k(:,i); % take previous result
    for j = i+1:D
        proj_j = proj_M_k(:,j);
        similarityMatrix(i,j) = dot(proj_i, proj_j) / (norm(proj_i) * norm(proj_j));
    end
end
% finding the row (doc 1) and col (doc2) with the highest similarity
[maxValue, ~] = max(similarityMatrix(:));
[doc1, doc2] = find(similarityMatrix == maxValue);
doc1 = doc1(1);
```

```
doc2 = doc2(1);
end
```