```
%Retrieve data
path = "dataset";

load 'dataset/wordVecV.mat';
```

Which two articles are closest in Euclidean distance?

Which two articles are closest in angle distance?

```
%In V, each column refers to a different document
%We can compute Euclidean distance using pdist
%pdist computes distance between vectors in a matrix where
%each row is a different vector
distances = pdist(transpose(V));
%use squareform for easier file location
distances = squareform(distances);

%loop thorugh matrix to get minimum distance
min_dist_row = -1;
min_dist_col = -1;
min_dist = 1000000000;
for row=1:size(distances,1)
    for col=1:size(distances,2)
        if (distances(row,col) < min_dist) && (row ~= col)
            min_dist = distances(row,col);
            min_dist_row = row;
            min_dist_col = col;
        end
    end
end

fprintf('Smallest distance between: %d and %d.', min_dist_row, min_dist_col)
```

```
Smallest distance between: 7 and 8.
```

```
%find distance using rearranged dot product <-> angle formula
min_angle_row = -1;
min_angle_col = -1;
min_angle = 100;
for row=1:size(distances,1)
    for col=1:size(distances,2)
        angle = acos((distances(row,col)^2)/(norm(V(:,row))*norm(V(:,col))));
        if (angle < min_angle) && (row ~= col)
            min_angle = angle;
            min_angle_row = row;
            min_angle_col = col;
        end
    end
end

fprintf('Smallest angle between: %d and %d.', min_angle_row, min_angle_col)
```

```
Smallest angle between: 1 and 6.
```

The pair with smallest distance (7 & 8) is not the same as the one with smallest angle (1 & 6). They could be different because the vectors are not normalized. As a result, while the angle between vectors is independent of length, the distance is. Vectors with more similar length may be closer in distance than vectors with dissimilar lengths but similar angles.

b) Normalize vectors and recompute smallest distance and smallest angle.

```
%normalize vectors
%Divide each column in V by the sum of the column
V_norm = V;
for col=1:size(V,2)
    V_norm(:,col) = V(:,col)./sum(V(:,col));
end

norm_distances = pdist(transpose(V_norm));
norm_distances = squareform(norm_distances);

%find smallest angle and smallest neighbour
%loop thorugh matrix to get minimum distance
min_dist_row = -1;
min_dist_col = -1;
min_dist = 1000000000;
for row=1:size(norm_distances,1)
    for col=1:size(norm_distances,2)
        if (norm_distances(row,col) < min_dist) && (row ~= col)
            min_dist = norm_distances(row,col);
            min_dist_row = row;
            min_dist_col = col;
        end
    end
end

fprintf('Smallest distance between: %d and %d.', min_dist_row, min_dist_col)
```

```
Smallest distance between: 9 and 10.
```

```
%find distance using rearranged dot product <-> angle formula
min_angle_row = -1;
min_angle_col = -1;
min_angle = 100;
for row=1:size(norm_distances,1)
    for col=1:size(norm_distances,2)
        angle = acos((norm_distances(row,col)^2)/(norm(V(:,row))*norm(V(:,col))));
        if (angle < min_angle) && (row ~= col)
            min_angle = angle;
            min_angle_row = row;
            min_angle_col = col;
        end
    end
```

```
end

fprintf('Smallest angle between: %d and %d.', min_angle_row, min_angle_col)
```

```
Smallest angle between: 5 and 8.
```

No, my answers are not the same as the previous part. Previously, the pair of smallest distnace was 7 & 8 whereas now it's 9 & 10. The pair of smallest angle was 1 & 6 and now is 5 & 8.

The reason for this normalizing is to express the i-th dimension of each word vector as the relative frequency of the word instead of the absolute frequency. The frequency of each word is now expressed as a fraction of all the words in the article. This eliminates discrepencies between different lengths of articles.

c) Find closest distance documents using TF-IDF

```
%compute TF-IDF for each document
%first section of TF-IDF (excluding log part) is the same as normalizing
%w.r.t document length
%after computing the normalized frequency, multiply each row (word) by the
%log term
V_TFIDF = V_norm;

for row=1:size(V,1)
    V_TFIDF(row,:) = V_TFIDF(row,:).*(sqrt(log10(size(V,1)/nnz(V(row,:)))));
end

%Find smallest distance
TFIDF_distances = pdist(transpose(V_TFIDF));
TFIDF_distances = squareform(TFIDF_distances);

%find smallest angle and smallest neighbour
%loop thorugh matrix to get minimum distance
min_dist_row = -1;
min_dist_col = -1;
min_dist = 1000000000;
for row=1:size(TFIDF_distances,1)
    for col=1:size(TFIDF_distances,2)
        if (TFIDF_distances(row,col) < min_dist) && (row ~= col)
            min_dist = TFIDF_distances(row,col);
            min_dist_row = row;
            min_dist_col = col;
        end
    end
end

fprintf('Smallest distance between: %d and %d.', min_dist_row, min_dist_col)
```

```
Smallest distance between: 9 and 10.
```

The smallest distance is now between documents 9 & 10.

d) What might be a reason for using the “inverse document frequency” adjustment? What is the adjustment doing geometrically?

The inverse document frequency adjustment adds an additional scaling term to penalize words that occur frequently in all documents. These terms are often non-descriptive such as "the" and "it" which don't help in describing the similarity of articles as they are common to all articles. For terms with high document frequency, the argument of the log term is small (small but >1 as document frequency <= number of documents) and the square root reduces it further, thus the scaling factor reduces the magnitude of the dimension. Notably, for words with document frequency equal to the number of documents, the TF-IDF of the word is zero. Ultimately, this is another form of normalization that penalizes common words.

Geometrically, the adjustment is increasing vectors' magnitude in dimensions that are uncommon while decreasing vectors' magnitude in dimensions that are common.