Nicholas Clement

CSCI 2830 - Search Engine Part Two

Introduction

This is part two of Application two, a Search Engine that uses cosine comparisons to determine similarity between matrices. Our goal in this part of the application is to improve on our previous results through QR decomposition to find and remove redundancy in our vector version of the 'database'.

We know mathematically that redundancy in a matrix occurs when its rank is smaller than its number of columns. This signals that we have at least one linearly dependent column. We can combat these issues by identifying a basis for the column space, and one way to calculate a basis for the column space is through the Gram-Schmidt process (A = QR).

Gram-Schmidt Process in MATLAB:

```
% %this script implements the QR factorization as seen in
% %section 6.4 of the text
%
% %A = QR
% %The vectors v1, . . . , vp are the columns of Q.
n = size(a,1);
k = size(a,2);
U = zeros(n,k);


U(:,1) = a(:,1)/sqrt(a(:,1)'* a(:,1)); %inital case

for i = 2:k
    U(:,i) = a(:,i);
    for j = 1:i-1
        U(:,i) = U(:,i) - ((U(:,i)'* U(:,j))* U(:,j));
    end
    (sqrt(U(:,i)'* U(:,i)));
    if (sqrt(U(:,i)'*U(:,i)))~=0
         U(:,i) = U(:,i)/(sqrt(U(:,i)'* U(:,i)));
    end
end
%get rid of nan

%allow us to pick the rank we want
% for x = r:k
%     U(:,x) = 0;
% end


r = zeros(n,k);
squares = zeros(n,k);
qnorm = zeros(n,k);
U;
```

```
%now we normalize
%first find squares
for x = 1:k
    for y = 1:n

        squares(y,x) = U(y,x)^2;

    end
end
%next sum column vectors
squares;
for x = 1:k
    su = sum(squares(:,x));

    for y = 1:n
        if su ~=0
            qnorm(y,x) = U(y,x)/(sqrt(su));
        end
    end

end


swag = qnorm'* qnorm;
qnorm;
r = qnorm'* a;



for x = z:k
    r(x,:) = 0;
end

r
newA = qnorm*r;
```

Results:

Initial Matrix (Database):

1 0 0 1 0
1 0 1 1 1
1 0 0 1 0
0 0 0 1 0
0 1 0 1 1
0 0 0 1 0

Resulting Matrices:

Q (qnorm):

0.5774 0 -0.4082 -0.0000 0
0.5774 0 0.8165 -0.0000 0

0.5774	**0**	**-0.4082**	**-0.0000**	**0**
0.5774	0	-0.4082	-0.0000	0
0	0	0	0.7071	-0.0000
0	1.0000	0	0	0
0	0	0	0.7071	-0.0000

R:

1.7321	**0**	**0.5774**	**1.7321**	**0.5774**
0	1.0000	0	1.0000	1.0000
-0.0000	0	0.8165	-0.0000	.8165
-0.0000	0	-0.0000	1.4142	-0.0000
0	0	0	0	0

We can see that R is an upper triangular matrix, just like expected.

When we multiply Q*R we get A:

A:

1.0000	**0**	**0.0000**	**1.0000**	**-0.0000**
1.0000	0	1.0000	1.0000	1.0000
1.0000	0	0.0000	1.0000	-0.0000
-0.0000	0	-0.0000	1.0000	-0.0000
0	1.0000	0	1.0000	1.0000
-0.0000	0	-0.0000	1.0000	-0.0000

Q^T*Q:

1.0000	**0**	**-0.0000**	**-0.0000**	**0**
0	1.0000	0	0	0
-0.0000	0	1.0000	0	0
-0.0000	0	0	1.0000	0
0	0	0	0	0

We see above the last 1 in the identity matrix getting cut off. This may be a result of our rank reduction in QR decomposition.

Rank Approximation:

In order to get a rank K approximation of A, we set the last K rows of R equal to 0. We then proceed to carry out our multiplication of A = QR.

The code I used to implement this is as follows:

```
for x = z:k
    r(x,:) = 0;
```

```
end

r
newA = qnorm*r;
```

Where z is the desired rank, and K is the number of rows present in r.

Reduced Databases:

Rank 4 Database:

1.0000	**0**	**0.0000**	**1.0000**	**-0.0000**
1.0000	0	1.0000	1.0000	1.0000
1.0000	0	0.0000	1.0000	-0.0000
-0.0000	0	-0.0000	1.0000	-0.0000
0	1.0000	0	1.0000	1.0000
-0.0000	0	-0.0000	1.0000	-0.0000

Rank 3 Database:

1.0000	**0**	**0.0000**	**1.0000**	**-0.0000**
1.0000	0	1.0000	1.0000	1.0000
1.0000	0	0.0000	1.0000	-0.0000
0	0	0	0	0
0	1.0000	0	1.0000	1.0000
0	0	0	0	0

Rank 2 Database:

1.0000	**0**	**0.3333**	**1.0000**	**0.3333**
1.0000	0	0.3333	1.0000	0.3333
1.0000	0	0.3333	1.0000	0.3333
0	0	0	0	0
0	1.0000	0	1.0000	1.0000
0	0	0	0	0

Rank 1 Database:

1.0000	**0**	**0.3333**	**1.0000**	**0.3333**
1.0000	0	0.3333	1.0000	0.3333
1.0000	0	0.3333	1.0000	0.3333
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Queries and Results

Queries:

Query
(1) Baking bread. = |1|0|1|0|0|0
(2) Baking. = |1|0|0|0|0|0
(3) What is the difference between pastry and bread?|Pastry-Bread
(4) Show me all recipes for pastry.|0|0|0|0|1|0
(5) A query of your devising. (Cake)|0|0|0|1|0|0

- All queries used 0.5 as the cosine cut off for similarity

Rank 4 Database Results:

Query Number:	**1**	**2**	**3**	**4**	**5**
Results	Columns 1 and 4	Column 1	Bread returned column one, Pastry returned columns two and five	Columns 2 and 5	Returned None

Rank 3 Database Results:

Query Number:	**1**	**2**	**3**	**4**	**5**
Results	Columns 1 and 4	Column 1 and 4	Bread returned columns one and four, Pastry returned columns two and five	Columns 2 and 5	Returned None

Rank 2 Database Results:

Query Number:	**1**	**2**	**3**	**4**	**5**
Results	Columns 1, 3, and 4	Column 1, 3, and 4	Bread returned columns one, three, and four, Pastry returned columns two, four, and five	Columns 2, 4, and 5	Returned None

Rank 1 Database Results:

Query Number:	**1**	**2**	**3**	**4**	**5**
Results	Columns 1, 3, 4, 5	Column 1, 3, 4 and 5	Bread returned columns one, three, four, and five, Pastry returned none	Returned None	Returned None

Analysis

After seeing the results of our test cases, I recommend using a cosine cutoff of 0.5, and a rank size of 3 for this particular data set.

Our cosine cutoff is a measure of confidence for the strength of correlation between two documents. In a realistic situation our cutoff would likely be around 0.9, but for our limited data and queries I recommend our cutoff to be 0.5. This cosine cutoff still yields some level of similarity and factors discrepancies out, but allows more results to be associated with our queries.

With using a cosine cutoff of 0.5, I also recommend using a `rank size of three`. We can see in the above charts that there is one subtle difference between our rank four database and our rank three database. When viewing our rank two and rank one results we can see that there are some major differences between these and rank four. For this reason, rank reducing to three will provide our sample database with the best balance of efficiency and reliability/accuracy. Rank one results are very far off base, with nearly every entry in the database returning true or nothing returning true for our q1-q5 queries.