AMATH 482 HW 1
Extended Yale Faces B: Database - Eigenfaces

Ken Lo

January 23, 2018

Abstract

Explore the Yale Faces Database by performing Singular Value Decomposition on its cropped and uncropped face images, and analyzing the decomposition's results.

1 Introduction and Overview

One of the most important computer application in the modern days is computer vision, particularly face recognition. To solve the face recognition problem, L. Sirovich and M. Kirby (1987) developed an approach to characterize human faces by creating low-dimensional representations of the images. This approach results in eigenfaces, which represents a set of eigenvectors that can form a basis set representing the original images.

Another way to create a low-dimensional representation of the image data is to use Singular Value Decomposition (SVD). In this paper, we will perform SVD on cropped and uncropped images of human faces. The cropped images consist of 38 people's faces and 64 images per person. Whereas the uncropped images consist of 15 people's faces, while having 10 images with them having different features, such as winking or wearing glasses.

2 Theoretical Background

A Singular Value Decomposition (SVD) takes the form

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^*$$

where

$$\begin{aligned} \mathbf{U} &\in \mathbb{C}^{m\times m}\text{is unitary} \\ \mathbf{V} &\in \mathbb{C}^{n\times n}\text{is unitary} \\ \mathbf{\Sigma} &\in \mathbb{R}^{m\times n}\text{is diagonal} \end{aligned}$$

Here the matrix $\mathbf{\Sigma}$ is diagonal with entries $\sigma_1 < \sigma_2 < \cdots < \sigma_n$.

By theorem 14.1.77 from class notes, any matrix $\mathbf{A}$ always has an SVD. Thus, we will use this fact to perform SVD on our data matrix, comprised of vectors representing each image we have in our databases. By doing so, we will obtain the three matrices shown above. Our data matrix $\mathbf{A}$ will be diagonalized using two different bases, $\mathbf{U}$ and

$\mathbf{V}^*$, where these bases are also orthogonal. Each diagonal entry in $\mathbf{\Sigma}$ gives us the energy captured by each corresponding mode in $\mathbf{U}$, and the columns in $\mathbf{V}^*$ tells us how each mode projects and describes the respective columns in our data matrix $\mathbf{A}$.

In terms of our face image data, running SVD will give us the proper orthogonal modes of our images, representing the most prominent features of faces. These modes will take form of vectors in the columns of $\mathbf{U}$. The diagonal matrix $\mathbf{\Sigma}$ will give us the singular values for those modes in $\mathbf{U}$, telling us how important each mode is in representing human faces. $\mathbf{V}^*$ will give us information about how each mode in $\mathbf{U}$ is weighted and used to describe each original image.

3 Algorithm Implementation and Development

Each image is read and transformed into a matrix, which we then vectorized, and stored them in our matrix $\mathbf{A}$ as columns.

The cropped database consisted of 38 people and 64 images for each of these people's faces in greyscale. As we read the images using MATLAB, we resized the images to be 120×80 for memory efficiency. After reading each image, we obtained a matrix $\mathbf{A}$ with size 9600×2432. Then we performed SVD on $\mathbf{A}$, and analyzed the resulting matrices $\mathbf{U}$, $\mathbf{\Sigma}$ and $\mathbf{V}$. In addition, we will also attempt low-rank reconstructions by multiplying the columns in $\mathbf{U}$ with their respecting singular values in $\mathbf{\Sigma}$ and columns in $\mathbf{V}$.

Similar steps were also done on the uncropped dataset. The dataset contained images of 15 different people, and for each person, 10 different features shown in each image, such as facial expressions and head accessories. Again, we normalized the size of each image to 120×80 and stored their vectors in another matrix $\mathbf{A}_{uncropped}$ (with size 9600×165). Similar analysis as on the cropped images were then done on the decomposed matrices.

4 Computational Results

After running SVD on our cropped dataset, we first plot the diagonal of matrix $\mathbf{\Sigma}$(i.e. singular value spectrum).

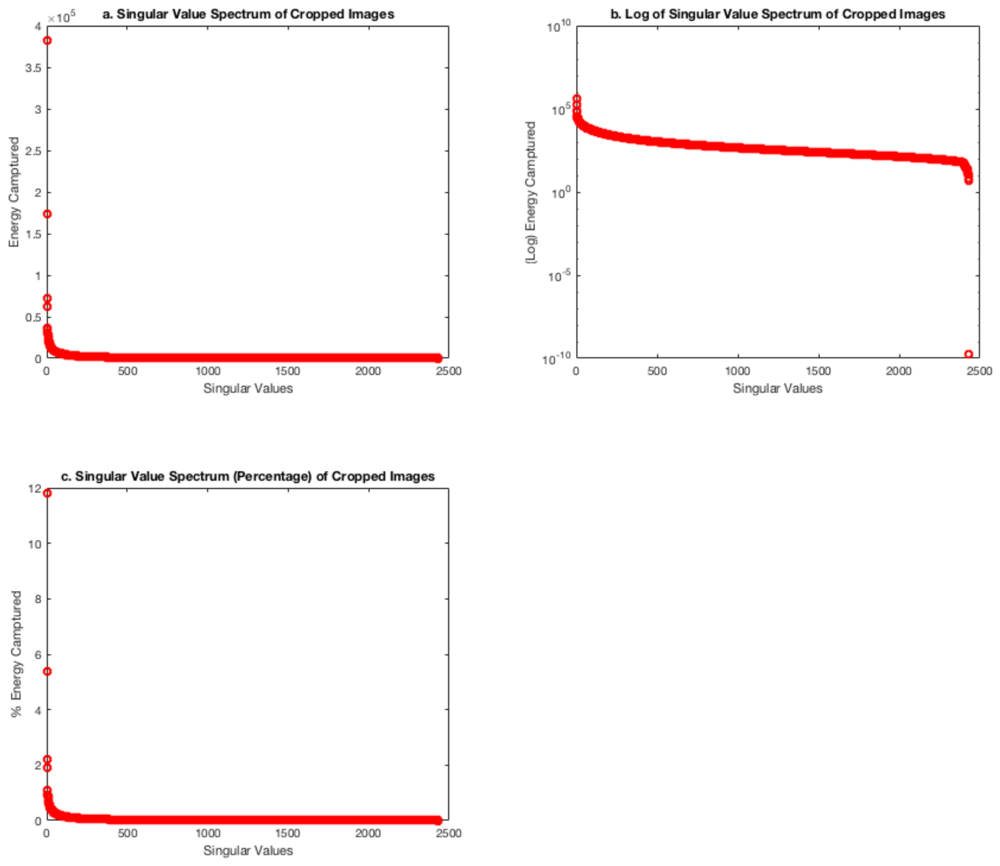

Figure 1: The standard plot (a) and log plot (b) shows the singular values and their respective energy captured in each mode. (c) shows shows the percentage of total energy captured in each mode respective to each singular value.

From figure 1, we see how much energy is captured by each mode. In figure 1c, we converted the energy values into percentage of total energy captured by each mode. From this plot, we see that the first mode captures about 12% of our data, the second mode captures about 5.5%, the third and fourth capture about 2%, with subsequent singular values capturing less and less data, and gradually approach 0%.

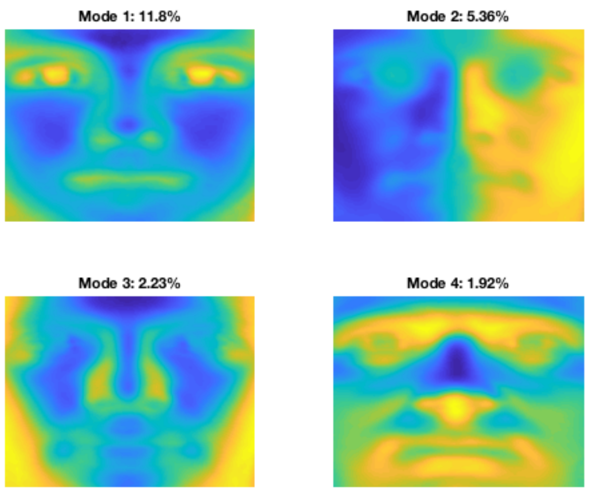

Figure 2: The First Four Modes/Columns in **U**, and the percent energy captured by each of them.

In figure 2, we can see what the four most important modes are. From each of those figures, we can generally see the main/principal components of a human face, including eyes, noses and mouths.

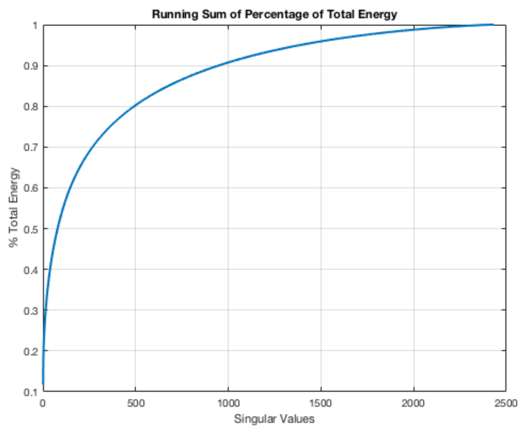

Figure 3: Cumulative energy captured by the modes.

To determine how many modes are necessary to reconstruct the face images, first we plot the running total of the percent energies of the respective singular values (figure 3). In this plot, we can see roughly the number of modes needed to make up different percentages of images. For instance, 70% of the images can be described using about 250 modes, 80% using about 500 modes, 90% using about 900 modes, and so on.

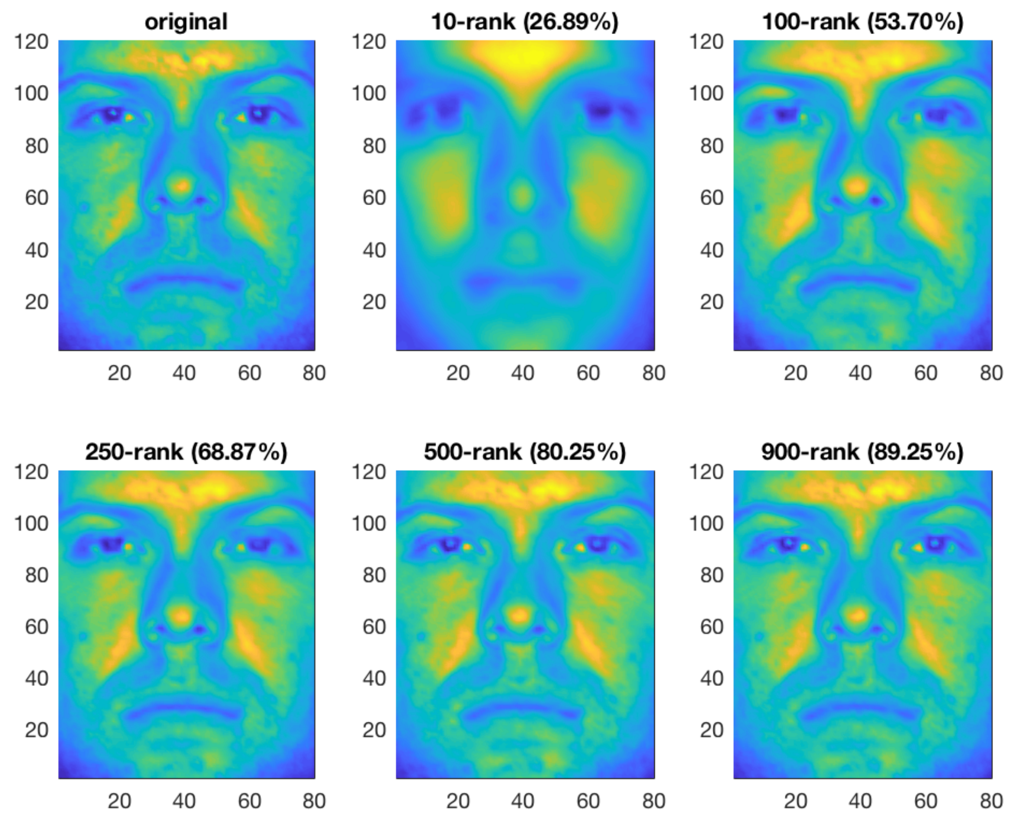

Figure 4: Low rank(r) reconstructions, with r = 10, 100, 250, 500, 900, and the original image for comparison.

In figure 4, the 10-rank reconstruction, which captures 27% of our data, still looks rather "generic". However, starting from rank 100(54%) and above, we can see how our reconstructed images resemble the person in the original image. With a 900-rank reconstruction (89.25%), the reconstructed image appears to be nearly the same as the original image, despite the differences in lightnings.

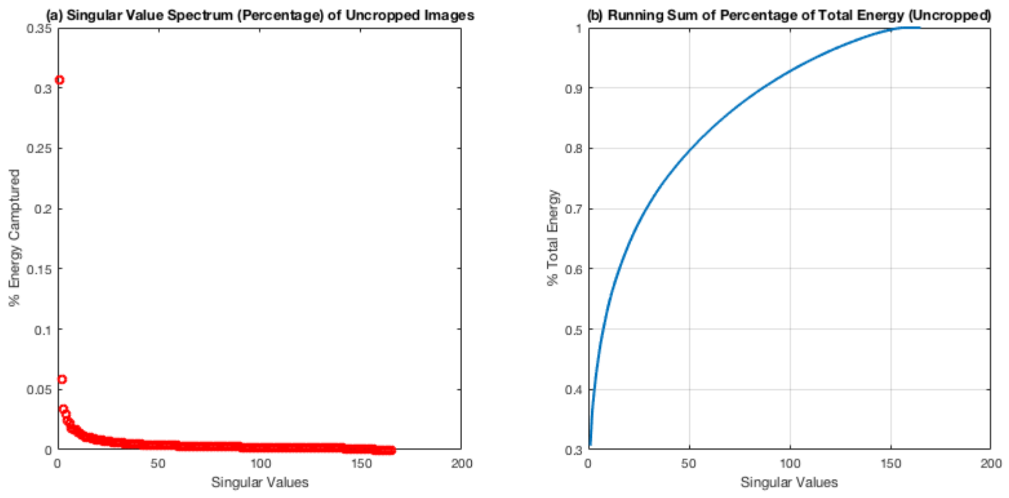

Figure 5: Uncropped Images: Singular Value Spectrum (a) and Cumulative Energy Captured by the Singular Values(b).

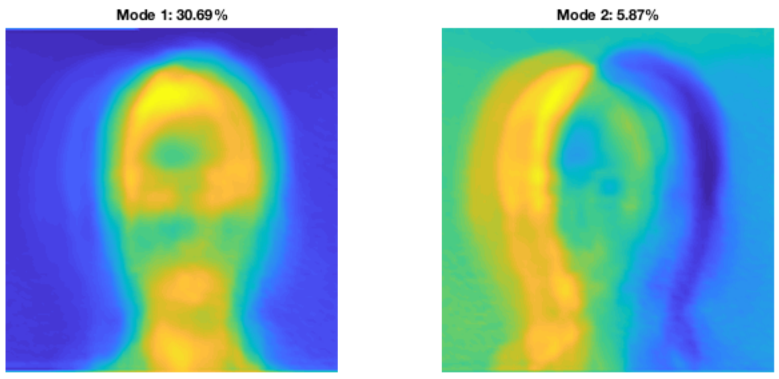

Figure 6: The First Two Modes in **U** for uncropped images.

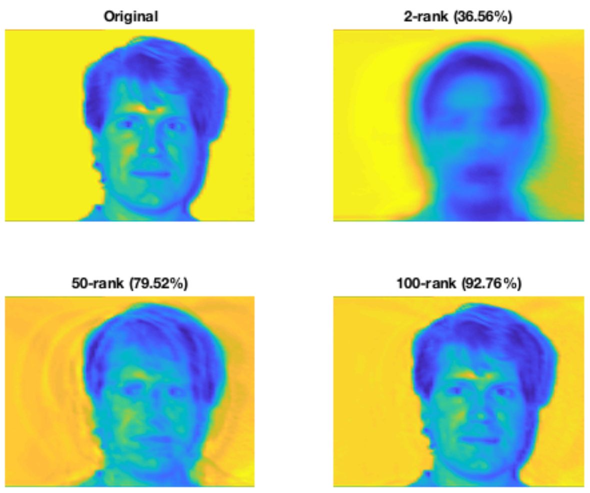

Figure 7: Low-rank(r) reconstructions of the first uncropped image, using $r = 2, 50, 100$, along with the original image for comparison.

To compare the differences between cropped and uncropped images, first we look at figure 5 (a). Here we see that the most prominent mode captures roughly 30% of the energy, followed by a mode capturing about 5% of the energy, and gradually decreases for the subsequent modes. Plotting these first two modes, as seen in figure 6, we can hardly see any feature that resembles a human face, except the general shape of a human head. In the second mode, it even looks like two people's heads combined. As we try to perform low-rank deconstruction to the first image, we can see from figure 7, using only the first two modes gave us a rough, blurry image of a person, who does not look like the person in the original image. With a 50-rank reconstruction, the person in the image starts to resemble the original image, however, the image is still rather blurry. Using a rank of 100, the reconstructed image is almost identical to the original image, despite some shadings in the background.

Low-rank reconstructions performed poorly using the uncropped images. The different facial expressions, accessories and lightning environments in different uncropped images attributed to such poor performance.

5 Summary and Conclusions

In this report, we have performed analyses on two types of data, cropped images and uncropped images. Through these analyses, images were broken down into modes with which we can reconstruct these images, in a lower-dimensional space.

When comparing the decomposed modes in cropped and uncropped images. We saw that the most prominent mode of the cropped images actually showed basic human faces' features, such as the location of the eyes, the nose and the mouth. In our example reconstructing the first cropped image, we successfully produced an image that looks very much alike the original image, using only 100-rank.

However, even the most prominent mode of the uncropped dataset barely resembled a human's face, human eyes, noses, and other features are all absent. In our reconstruction example, the images were blurry until we used a 100-rank reconstruction.

As mentioned in the previous section, different features and environments in each of the uncropped images were responsible for the poor performance in low-dimensional reconstructions. More generally, these inconsistencies among each uncropped image, are noise of the data. Here we experienced the power of noise which affects the results and performances when we are working with data.

The SVD is a powerful method to filter data and extract the principal features from the data. With these principal features, we can compress data and later reconstruct them with minimal loss to the data.

A MATLAB functions used and brief implementation explanation

- dir() - list the files in a folder
- imresize() - resize image
- double() - convert data to type double
- imread() read image into MATLAB as a matrix
- svd() - perform SVD on a matrix
- plot() - basic plotting
- semilogy() - for log plotting
- diag() - retrieve diagonals of a matrix
- sum() - get the sum of all the values in a vector/matrix
- pcolor() - make a pseudocolor plot using image matrices
- flipud() - flip an arraay upside down
- reshape() - reshape a matrix

A MATLAB codes

```
close all; clear all; clc
%% load cropped data
folders = dir('./CroppedYale/*');

A = [];
for i = 4:size(folders) % ignore '.', '..', '.DS_Store'
    foldername = folders(i).name;
    imgs = dir(['./CroppedYale/' foldername '/*']);
    for j = 3:size(imgs) % ignore '.', '..'
        img_mat = imresize(double(imread([imgs(j).folder '/' imgs(j).name]))

        vectorized = img_mat(:);
        A = [A vectorized];
    end
end
%% save A
save('A.mat', 'A')

%% run SVD on A
[u, s, v] = svd(A);
%% plot singular value spectrum
figure(1)
subplot(2,2,1), plot(diag(s), 'ro', 'Linewidth', 2);
title('a. Singular Value Spectrum of Cropped Images');
xlabel('Singular Values');
ylabel('Energy Camptured');
subplot(2,2,2), semilogy(diag(s), 'ro', 'Linewidth', 2);
title('b. Log of Singular Value Spectrum of Cropped Images');
xlabel('Singular Values');
ylabel('(Log) Energy Camptured');
subplot(2,2,3), plot(diag(s)*100/sum(diag(s)), 'ro', 'Linewidth', 2);
title('c. Singular Value Spectrum (Percentage) of Cropped Images');
xlabel('Singular Values');
ylabel('% Energy Camptured');
%% plot running sum
run_sum = 0;
sums = [];
vals = diag(s)/sum(diag(s));
for i=1:size(diag(s))
    run_sum = run_sum + vals(i);
    sums = [sums run_sum];
end
figure
plot(sums, 'Linewidth', 2), grid on;
title('Running Sum of Percentage of Total Energy');
xlabel('Singular Values');
ylabel('% Total Energy');
%% Get Percentages for Singular Values
percentS = diag(s)/sum(diag(s));
%% reconstruction first image
figure
ff10=u(:,1:10) * s(1:10, 1:10) * v(:,1:10).';
ff100=u(:,1:100) * s(1:100, 1:100) * v(:,1:100).';
ff250=u(:,1:250) * s(1:250, 1:250) * v(:,1:250).';
ff500=u(:,1:500) * s(1:500, 1:500) * v(:,1:500).';
ff900=u(:,1:900) * s(1:900, 1:900) * v(:,1:900).';
subplot(2,3,1), pcolor(flipud(reshape(A(:, 1), 120, 80))), shading interp; % original
title('original')
subplot(2,3,2), pcolor(flipud(reshape(ff10(:,1), 120, 80))), shading interp;
title('10-rank (26.89%)')
```

```
subplot(2,3,3), pcolor(flipud(reshape(ff100(:,1), 120, 80))), shading interp;
title('100-rank (53.70%)')
subplot(2,3,4), pcolor(flipud(reshape(ff250(:,1), 120, 80))), shading interp;
title('250-rank (68.87%)')
subplot(2,3,5), pcolor(flipud(reshape(ff500(:,1), 120, 80))), shading interp;
title('500-rank (80.25%)')
subplot(2,3,6), pcolor(flipud(reshape(ff900(:,1), 120, 80))), shading interp;
title('900-rank (89.25%)')
%% plot first 4 in u
figure
grid off
subplot(2,2,1)
pcolor(flipud(reshape(u(:, 1), 120, 80))), shading interp, axis off
title('Mode 1: 11.8%')
subplot(2,2,2)
pcolor(flipud(reshape(u(:, 2), 120, 80))), shading interp, axis off
title('Mode 2: 5.36%')
subplot(2,2,3)
pcolor(flipud(reshape(u(:, 3), 120, 80))), shading interp, axis off
title('Mode 3: 2.23%')
subplot(2,2,4)
pcolor(flipud(reshape(u(:, 4), 120, 80))), shading interp, axis off
title('Mode 4: 1.92%')
%% load uncropped data
imgs = dir('./yalefaces_uncropped/yalefaces/*');
Auncropped = [];
for i=3:size(imgs)
    img_mat = imresize(double(imread([imgs(i).folder '/' imgs(i).name])), [120 80]);
    vectorized = img_mat(:);
    Auncropped = [Auncropped vectorized];
end
%% SVD on uncropped images
[uun, sun, vun] = svd(Auncropped);
%% Calculate Running Sum of Singular Values Percentage (uncropped)
run_sum = 0;
sums = [];
vals = diag(sun)/sum(diag(sun));
for i=1:size(diag(sun))
    run_sum = run_sum + vals(i);
    sums = [sums run_sum];
end
%% uncropped low-rank reconstruction
ff2=uun(:,1:2) * sun(1:2, 1:2) * vun(:,1:2).';
ff50=uun(:,1:50) * sun(1:50, 1:50) * vun(:,1:50).';
ff100=uun(:,1:100) * sun(1:100, 1:100) * vun(:,1:100).';
%% plots

figure
```

```
subplot(1,2,1), plot(diag(sun)/sum(diag(sun)), 'ro', 'Linewidth', 2);
title('(a) Singular Value Spectrum (Percentage) of Uncropped Images');
xlabel('Singular Values');
ylabel('% Energy Camptured');
subplot(1,2,2), plot(sums, 'Linewidth', 2), grid on;
title('(b) Running Sum of Percentage of Total Energy (Uncropped)');
xlabel('Singular Values');
ylabel('% Total Energy');

figure
subplot(1,2,1), pcolor(flipud(reshape(uun(:, 1), 120, 80))), shading interp
, axis off;
title('Mode 1: 30.69%');
subplot(1,2,2), pcolor(flipud(reshape(uun(:, 2), 120, 80))), shading interp
, axis off;
title('Mode 2: 5.87%');

figure
subplot(2,2,1), pcolor(flipud(reshape(Auncropped(:, 1), 120, 80)))
, shading interp; %original image
axis off;
title('Original');
subplot(2,2,2), pcolor(flipud(reshape(ff2(:,1), 120, 80))), shading interp;
axis off;
title('2-rank (36.56%)')
subplot(2,2,3), pcolor(flipud(reshape(ff50(:,1), 120, 80))), shading interp;
axis off;
title('50-rank (79.52%)')
subplot(2,2,4), pcolor(flipud(reshape(ff100(:,1), 120, 80))), shading interp;
axis off;
title('100-rank (92.76%)')
```