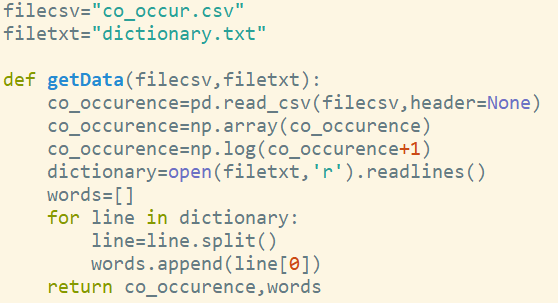
#minipro5

16337266 徐原

Part1

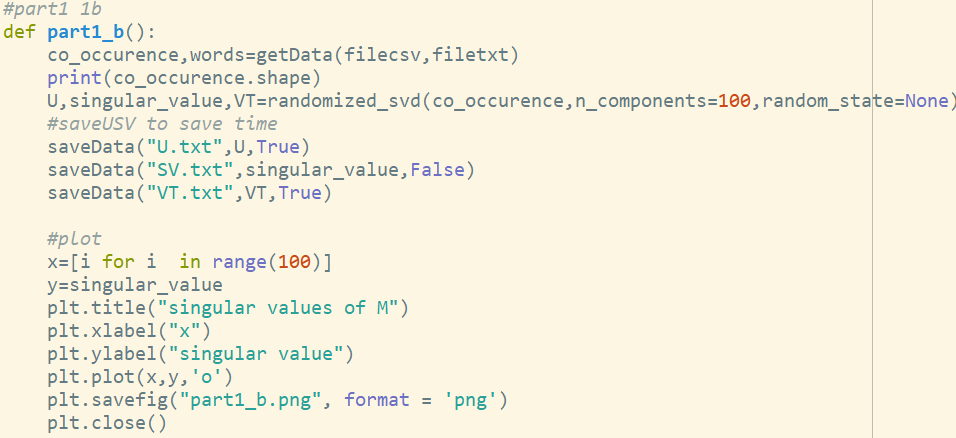
1. Make sure you can import the given datasets into whatever language you’re using. If you’re using  
   MATLAB, you can import the data using the GUI. Also, make sure you can interpret the entries of  
   the co-occurrence matrix using the dictionary, you can try to find the co-occurrence of a few pairs of common words to make sure of this.

Code:

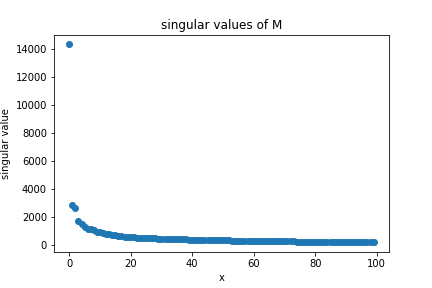


1. (3 points) Let matrix M denote the 10000×10000 matrix of word co-occurrences. In light of the power law distribution of word occurrences, it is more suitable to work with the normalized matrix M~ such that each entry M~ ij = log(1 + Mij). For the remainder of this problem, we will work with this scaled matrix, M: ~ Compute the rank-100 approximation of M~ by computing the SVD M~ = UDV T . Plot the singular values of M: ~ Does M~ seem to be close to a low rank matrix? [Hint: Computing the full SVD will take a bit of time instead you should compute just the top 100 singular values/vectors...and save this decomposition rather than recomputing every time you work on this miniproject!]

Code：



Result:



**Solution:**

**It is truly a low rank matrix, because just first few singular values are higher than others. (And others are similar to zero).**

1. c. (5 points) Note that as the matrix M~ is symmetric, the left and right singular vectors are the same, up to flipping signs of some columns. We will now interpret the singular vectors (columns of U or V ). For any i, denote vi as the singular vector corresponding to the ith largest singular value. Note that the coordinates of this vector correspond to the 10,000 words in our dictionary. For a given vector vi, you can see which words are most/least relevant for that vector by looking at the words corresponding to the coordinates of vi that have the largest or smallest values. This allows you to get a rough sense for the semantic information that is captured in each of the singular vectors. Find 5 interesting/interpretable singular vectors, and describe what semantic or syntactic structures they capture. For each of the 5 vectors you choose, provide a list of the 10 words corresponding to the coordinates with the largest values and the 10 words corresponding to the coordinates of that vector with the smallest values. Not all of the singular vectors have easy-to-interpret semantics; why would you expect this to be the case?

# Part2

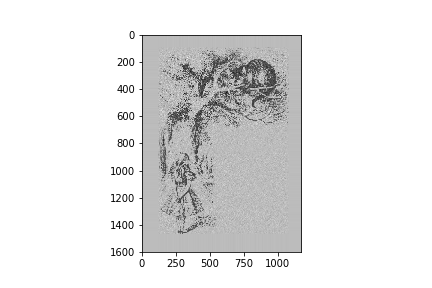
(4 points) Before running SVD on Alice, think about what you expect the rank 1 approximation given  
by SVD to look like. To guide your thinking, consider the following simpler picture, where black pixels  
have value 0. Qualitatively describe the rank 1 approximation (given by SVD) of the following picture  
of the moon. Explain your reasoning. [Hint: it might be helpful for you to sketch what you expect the  
answer to be|and make sure it is rank 1!]

Solution：

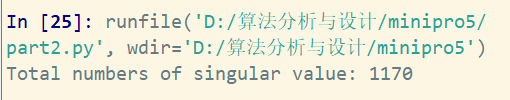
Suppose we got the rank 1 approximation of this moon picture, and we have known black pixels is 0,white pixels is 1. So the value of singular vector between [0,1], the parts where black background correspondent on are always equal to zero. The column and row where include the moon can be valued in (0,1]. The bigger value a point has, the more moon part it represents.

b. (6 points) Run SVD and recover the rank k approximation for the image of Alice, for k 2 f1; 3; 10; 20; 50; 100; 150; 200;400; 800; 1170g. In your assignment, include the recovered drawing for k = 150. Note that the recovered drawing will have pixel values outside of the range [0; 1]; feel free to either scale things so that the smallest value in the matrix is black and the largest is white (default for most python packages and matlab), or to clip values to lie between 0 and 1.



When K=150:  


1. (2 points) Why did we stop at 1170?



Solution:

From SVD we can get the total numbers of singular value is 1170. So we at most use 1170 singular vector. If we use more, this compression will don’t make sense.

1. (3 points) How much memory is required to efficiently store the rank 150 approximation? Assume each floating point number takes 1 unit of memory, and don’t store unnecessary blocks of 0s. How much better is this than naively saving the image as a matrix of pixel values?

Solution:

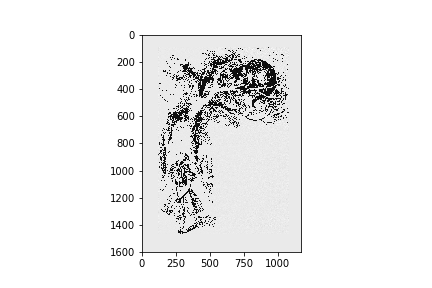
Original matrix size: 1170\*1600=1 872 000 units

Compression matrix size: 1170\*150+ 1600\*150=415 500 units

Save times : 1 872 000 / 415 500 = 4.51084337

1. e. (Bonus 3 points) Details of the drawing are visible even at relatively low k, but the gray haze / random background noise persists till almost the very end (you might need to squint to see it at k = 800).Why is this the case? [For full credit, you need to explain the presence of the haze and say more than just "the truncated SVD is an approximation of the original image."]

when k=800:



Solution:

Less singular values can express most of the information in the image, and some singular values can be discarded to realize compression. On image noise reduction: noise generally exists in the high-frequency part of the image, which means in the part with small singular value. Therefore, SVD can be used to achieve noise reduction. when K=800, there truly occur some noise.