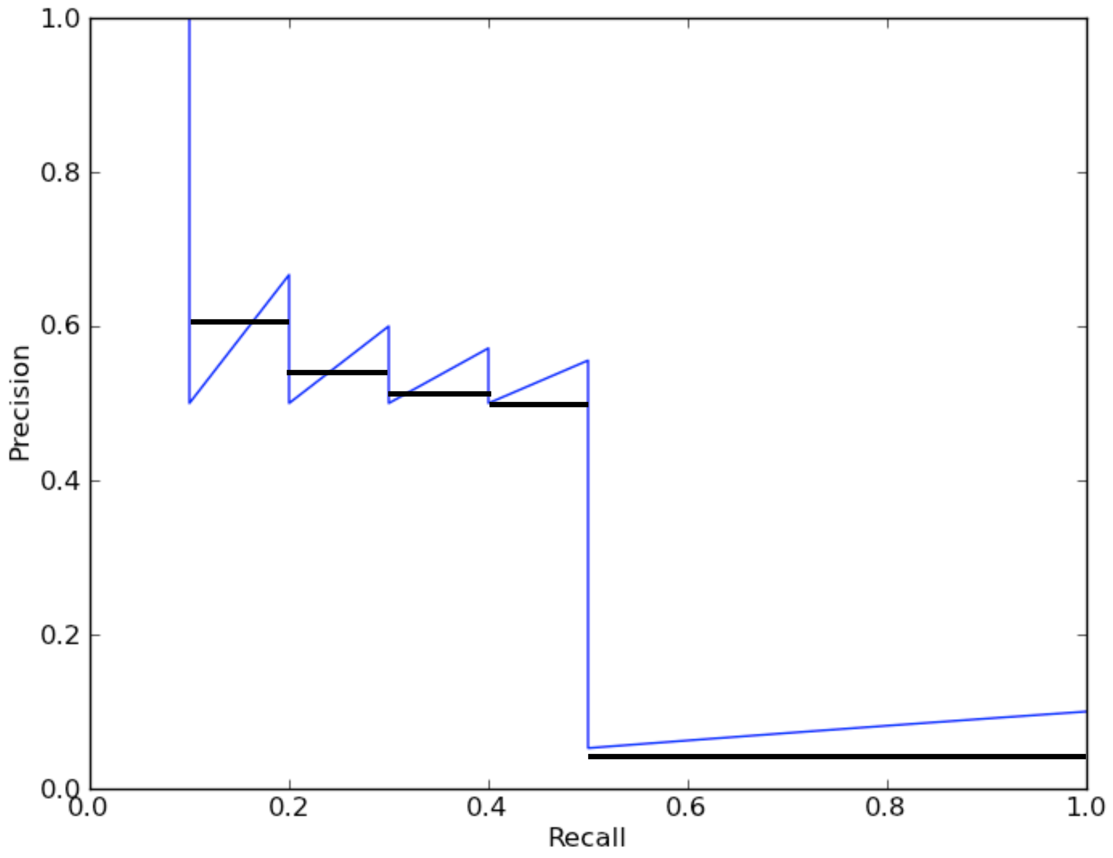
Name : Rahul Arora
NetID : ra487

IA. Precision/Recall exercise

a. Plot (roughly) an interpolated ranked precision/recall graph of this data.



b. What is the value of the 11-point interpolated average precision (average of recall values 0, .1, ... , .9, 1)?

Value = (1.0 + 1.0 + 0.666 + 0.6 + 0.571 + 0.555 + 0.1 + 0.1 + 0.1 + 0.1 + 0.1) * 0.1
= 0.44

IB. A small-scale recommender system

a. What are the singular values from the matrix Σ, and which did you retain in Σ_k?

Σ: [2.68852390e+01 2.38576363e+01 2.16765414e+01 3.87619289e+00
2.60091001e+00 8.04786840e-01 6.63664093e-01 4.96924517e-01
5.52926600e-16 2.32768468e-16]

Σk : [2.68852390e+01 2.38576363e+01 2.16765414e+01 0 0 0 0 0 0 0]
The first three dominate and should be retained.

b. By comparing X_k to the original X, what are the three strongest recommendations you would make to the existing users? (i.e., examine largest values of X_k-X)

user5, song10 = 2.32
user3, song8 = 2.23
user10, song8 = 1.99

c. Now consider four new users who join the system, and start accumulating usage data. Based on the usage patterns of the prior 20 users considered above (i.e., do not perform a new SVD), what recommendations would you make to three users with the following usage vectors:

Recommendations to make to 1st user = Song 1,8,10
Recommendations to make to 2nd user = Song 9,10
Recommendations to make to 3rd user = Song 2,5
Recommendation to make to the new user4 = Song 3,5,7

II. A search engine that indexes full text using latent semantic indexing

I have implemented Problem II using Python. It requires the following dependencies :

```
Numpy
Scipy
Matplotlib
invert.py
```

Installation steps :

```
$ curl -O https://raw.github.com/pypa/virtualenv/master/virtualenv.py
$ python virtualenv.py my_new_env
$ . my_new_env/bin/activate
$ (my_new_env)$ pip install numpy

$ (my_new_env)$ pip install scipy
If the above command fails, go to http://www.scipy.org/Installing_SciPy/ and
read the installation guide or read the steps written below if you are on OSX or
Linux.
      OSX specific installation for scipy :
      $ git clone https://github.com/scipy/scipy.git
      $ python setup.py build
      $ python setup.py install

      RPM-based linux OS installation for scipy :
      $ yum install python-scipy

      Debian-based linux OS installation for scipy :
      $ sudo apt-get install python-numpy python-scipy

$ (my_new_env) pip install matplotlib
```

How to run
1) Unzip the .zip file
2) $ `(my_new_env)` python LSI.py
3) Enter stopping criteria which will generate the stoplist (Stopping criteria should be between 0 and 1. Stopping criteria of 1 implies that the stoplist will include words which are present in all 40 documents)
4) Enter the search query
5) Enter the rank

6) Continue steps 4 and 5 till you want to run the program. To exit, enter zzz in the search query.

How numpy and scipy are used in the program?
Numpy is used to initialize large matrixes using numpy.zeros() as initializing matrixes using array.array() is not efficient. I also used numpy.transpose() for getting the transpose of a matrix.
Scipy is used to calculate svd using linalg.svd(). It is also used to align sigma using linalg.diagsvd(). dot() is used for performing matrix multiplication

Whenever the term-document matrix is too large, we use approximated low rank matrix. We presume that the term-document matrix is noisy and approximated matrix is considered de-noisified. The consequence of rank lowering is that some dimensions are combined and depend on more than one term like car and automobile. This mitigated the problem of identifying synonymy. It also mitigates polysemy.

From the test cases I inferred that as rank is low, the singular decomposition ends up under-representing the original matrix. Alternately, if the choice of rank is high, the singular decomposition over-represent the original matrix by adding in noise components. Also, I inferred the documents represented in the test cases in which the search query is not present. However, they are relevant to the search query for ranks greater than 5.
For e.g Document 33 is retrieved for query “graphene” at rank 5 and 10 but when at rank 25, Doc 33 is not in the top 5 retrieved documents.

In the program, I have taken into consideration that the stopping criteria is between 0 and 1. The user should input positive rank and if the search query after deducting the stopwords is empty, the user is prompted for a new search query.

I have included a **README** file for instructions.