Readme

CSE508: Information Retrieval

Assignment 2

Question 1

Methodology:
1. Used the same data given in assignment 1 and carried out the same preprocessing steps as Q2 in Assignment 1.
2. Built the matrix of size no. of document x vocab size using Counters.
3. Found the tf-idf values in the matrix of each word of the vocab.
4. Preprocessed query
5. Computed the TF-IDF score for the query and reported the top 5 relevant documents based on the score.
6. Use all 5 weighting schemes for term frequency calculation and report the TF-IDF score and results for each scheme separately

Preprocessing steps:
(i) Convert the text to lower case
(ii) Perform word tokenization
(iii) Remove stopwords from tokens
(iv) Remove punctuation marks from tokens
(v) Remove blank space tokens

Assumptions:
Words are spelled correctly
Queries are separated by space

Question 2

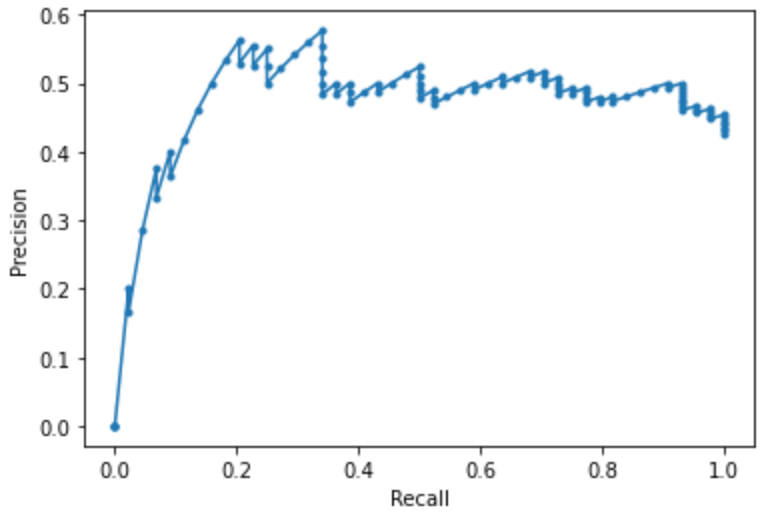
Methodology:
Follow the same steps as given in the assignment.
Every part is explained in the ipynb file.

Analysis:

Total States =
19893497375938370599826047614905329896936840170566570588205180312704857992695193482412686565431050240000000000000000000000000

NDGC At 50 0.5717260627203818
NDCG For the whole dataset 0.6357153091990775

Graphs



Question 3

Preprocessing steps:

- First, remove the special character from the string
- Remove the white spaces from the string
- Remove numbers from the string
- Tokenize the word using nltk word tokenzer().
- Convert letter to lower case
- Applying stemming and lemmatization. (Not included in demo code)
- Remove the stopwords from the result

Methodology:
create TF-ICF for top k feature selection then implement Gaussian Naive Bayes to predict the class.
Follow the same steps as given in the assignment

Analysis:-

- Results heavily depend on the value of k. If k is smaller then better is the accuracy but it will overfit our model.
- Results depend upon the split size. The bigger is the training data better is the model
- Results are not exactly the same for every iteration as training data is different.

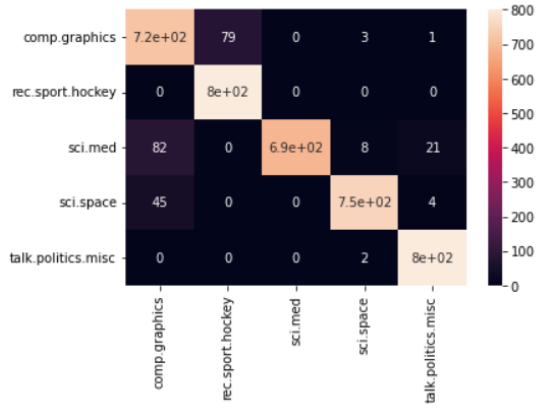
Results for split_size = 0.8 @ topk = 150

Test results on training data

In [27]:
```
stats(ytrain , result_x)
```

```
class-wise precision [0.84952607 0.91012514 1.         0.98298429 0.9684466 ]
class-wise recall [0.89625 1.      0.86125 0.93875 0.9975 ]
classise f1score [0.87226277 0.95294818 0.92545332 0.96035806 0.98275862]
class-wise support [800 800 800 800 800]

Overall Accuracy:- 0.93875
Rows are actual value and cols are predicted labels of Confusion matrix
```

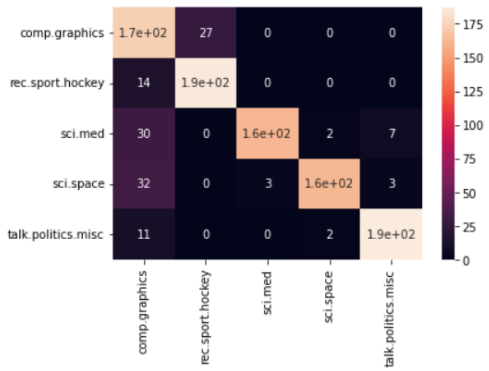


Test results on testing data

In [28]:
```
stats(ytest , result_y)
```

```
class-wise precision [0.66538462 0.87323944 0.98170732 0.97590361 0.94923858]
class-wise recall [0.865 0.93  0.805 0.81  0.935]
classise f1score [0.75217391 0.90072639 0.88461538 0.8852459  0.94206549]
class-wise support [200 200 200 200 200]

Overall Accuracy:- 0.869
Rows are actual value and cols are predicted labels of Confusion matrix
```



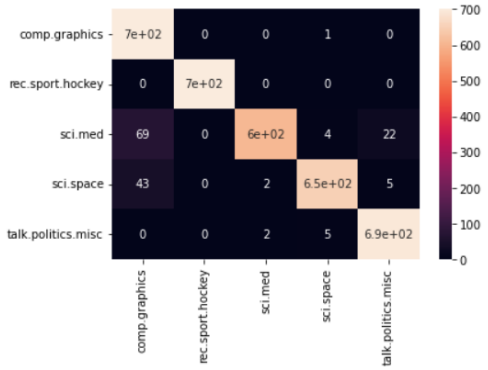
Results for split_size = 0.7 @ topk = 150

Test results on training data

```
In [24]: stats(ytrain , result_x)
```

```
class-wise precision [0.86189889 1.         0.99343186 0.98484848 0.9625    ]
class-wise recall [0.99857143 1.         0.86428571 0.92857143 0.99      ]
classise f1score [0.92521509 1.         0.92436975 0.95588235 0.97605634]
class-wise support [700 700 700 700 700]

Overall Accuracy:- 0.9562857142857143
Rows are actual value and cols are predicted labels of Confusion matrix
```

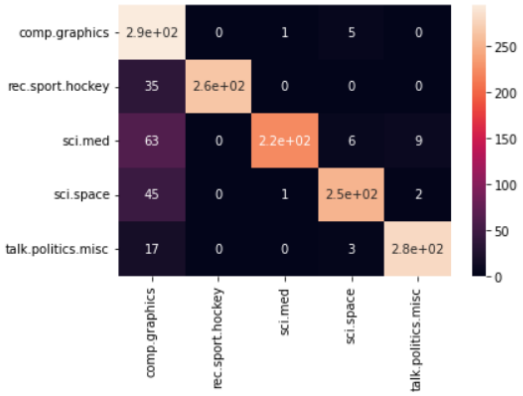


Test results on testing data

```
stats(ytest , result_y)
```

```
class-wise precision [0.64757709 1.         0.99107143 0.94736842 0.96219931]
class-wise recall [0.98       0.88333333 0.74       0.84       0.93333333]
classise f1score [0.77984085 0.9380531  0.84732824 0.89045936 0.94754653]
class-wise support [300 300 300 300 300]

Overall Accuracy:- 0.8753333333333333
Rows are actual value and cols are predicted labels of Confusion matrix
```



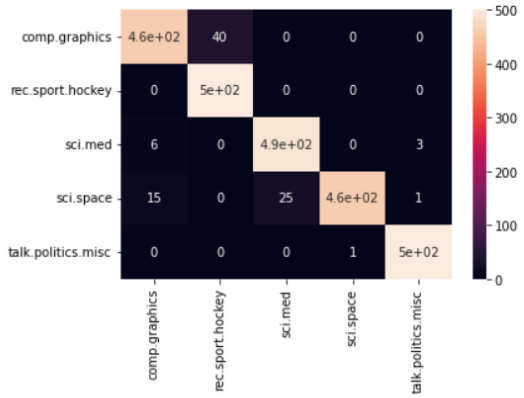
Results for split_size = 0.5 @ topk = 150

Test results on training data

```
stats(ytrain , result_x)
```

```
class-wise precision [0.95634096 0.92592593 0.95155039 0.99782609 0.99204771]
class-wise recall [0.92  1.    0.982 0.918 0.998]
classise f1score [0.93781855 0.96153846 0.96653543 0.95625    0.99501496]
class-wise support [500 500 500 500 500]

Overall Accuracy:- 0.9636
Rows are actual value and cols are predicted labels of Confusion matrix
```



Test results on testing data

```
stats(ytest , result_y)
```

```
class-wise precision [0.62196862 0.89344262 0.93377483 0.97051597 0.96895787]
class-wise recall [0.872 0.872 0.846 0.79  0.874]
classise f1score [0.72606162 0.88259109 0.88772298 0.87100331 0.9190326 ]
class-wise support [500 500 500 500 500]

Overall Accuracy:- 0.8508
Rows are actual value and cols are predicted labels of Confusion matrix
```