CAB431 Tutorial (Week 9): Information Filtering

****

Information filtering systems are used to remove irrelevant information (or unwanted information) from the information flow based on user information needs. The difficult problem is how to acquire knowledge of user information needs or topics of interest.

A popular solution to this dilemma is to use text mining to gain knowledge from relevant user feedback, with a training set D (or labelled dataset) consisting of a set of relevant documents D^+ and a set of irrelevant (non-relevant) documents D^-.

For example, in week 8 workshop, you can find a training set D which includes documents in "Training_set" folder, and relevance judgments (benchmark) in file "Training_benchmark.txt". The set of relevant documents D^+ and the set of irrelevant documents D^- are as follows:

$$D^+ = \{d \mid d \in D, \text{its label in Training\_benchmark.txt is "1"}\}$$
$$D^- = \{d \mid d \in D, \text{its label in Training\_benchmark.txt is "0"}\}$$

The following procedure (see lecture notes) shows how to select features from training set D.

procedure *BM25termWeighting*(D, D^+, D^-)

(1) Let T = $\emptyset$, $N = |D|$, $R = |D^+|$
$T = \{t_k \in d_i,\ d_i \in D^+\}$ # a set comprehension to find all terms in D^+

(2) for all $t_k \in T$ do
{ $n(t_k)=0$, $r(t_k)=0$ }
(3) for all $t_k \in T$ do
for all $d_i \in D$ do
if $\tau(t_k, d_i)=1$ then $n(t_k) = n(t_k) + 1$
(4) for all $t_k \in T$ do
for all $d_i \in D^+$ do
if $\tau(t_k, d_i)=1$ then $r(t_k) = r(t_k) + 1$
(5) for all $t_k \in T$ do

$$w_5 = \left[\frac{r(t_k) + 0.5}{R - r(t_k) + 0.5}\right] \div \left[\frac{n(t_k) - r(t_k) + 0.5}{\left(N - n(t_k)\right) - \left(R - r(t_k)\right) + 0.5}\right]$$

Task 1. (Feature selection from training set D)

Design a python program to evaluate terms weights (define function $w5$) using the above procedure; and then select top-terms (e.g., their weights are great than the *mean* of the $w5$ weights of terms + θ), where θ is an experimental parameter.

You can firstly design a function $w5$ using the following
Inputs: Training_set folder, Training_benchmark.txt and θ
Outpout: a dictionary of features with their $w5$ weights

Then in the main program, save the features to a text file (Model_w5_R102.dat) with the following outcomes:

rapist 106.20000000000002
multipl 39.77987421383647
unemploy 37.857142857142286
site 35.98159509202454
convict 31.774193548387096
featur 30.62130177514793
mistak 22.48603351955307
unknown 20.96685082872928
present 19.48087431693989
grief 18.027027027027028
defend 16.604278074866308
father 15.8064
ground 15.21164021164021
bungl 14.627906976744185
inquiri 14.337634408602149
cabinet 13.848167539267015
debat 13.848167539267015
plus 13.848167539267015
wive 13.848167539267015
bewild 13.848167539267015
earli 13.404968944099378
fate 13.029350104821802
nine 12.991111111111111
…

Task 2. Rank documents in Test_set folder (*U*) based on *Features* to test the BM25 model.

(1) Design an algorithm, "**procedure** *BM25Testing*(*Features*, *U*)", to calculate BM25 ranking score for all documents in *U* and return a dictionary of {doc1: rank1, …}.

(2) Design a Python function to implement **procedure** *BM25Testing*(*U*, *Features*), and save all documents in a file name "rankBM25.txt" as follows:

3827 520.5045275357625
3833 516.1318289439776
4306 492.4045372037915
9703 488.95511782538625
8333 445.94029629025107
7118 408.65233456451114
3828 404.2271828695591
7502 394.9948992699487
9790 386.9413050322148
6498 385.8329976942741
…

Task 3. Read a ranking result file (e.g., “rankBM25.txt”), calculate its Average Precision. Design a python program to

(1) Calculate Recall and Precision at rank positions where a relevant document was retrieved.
(2) Calculate the average precision.
Print out the results as follows:

At position 1, precision= 1.0, recall= 0.034482758620689655
At position 2, precision= 1.0, recall= 0.06896551724137931
At position 3, precision= 1.0, recall= 0.10344827586206896
At position 4, precision= 1.0, recall= 0.13793103448275862
At position 5, precision= 1.0, recall= 0.1724137931034483
At position 6, precision= 1.0, recall= 0.20689655172413793
At position 7, precision= 1.0, recall= 0.2413793103448276
At position 8, precision= 1.0, recall= 0.27586206896551724
At position 9, precision= 1.0, recall= 0.3103448275862069
At position 10, precision= 1.0, recall= 0.3448275862068966
…

---The average precision= 1.0