CS242 assignment solution

February 2018

1 Exercise A

Consider the following document D, taken from a collection C.
"The University of California, Riverside is one of 10 universities within the prestigious University of California system, and the only UC located in Inland Southern California. Widely recognized as one of the most ethnically diverse research universities in the nation."

Consider the following two queries:

1. Q1: university Riverside
2. Q2: diverse university

Characteristics of collection C are as follows:

1. # docs in collection C: 1000
2. # docs in C that contain "Riverside": 100
3. # docs in C that contain "university/ies": 200
4. # docs in C that contain "diverse": 150

Compute the scores of Q1 and Q2 for D, using (a) BM25, and (b) Unigram Language Model (with smoothing method of your choice). Make and state any assumptions necessary, e.g., about the constants in BM25.

Assumptions: $k_1 = 1.2$, $b = 0.75$, $k_2 = 100$, $avdl = 40$.

As $dl = 40$, thus $K = 1.2$ and the number of terms in collection C ($|C|$) = 40 * 1000 = 40000.
Term frequency:

1. "Riverside": 1
2. "university": 4 (with stemming, or else 2)
3. "diverse": 1

$$BM25(Q1, D) = log(\frac{1000-200+0.5}{200+0.5}) \times \frac{4(1.2+1)}{1.2+4} \times \frac{100+1}{100+1} + log(\frac{1000-100+0.5}{100+0.5}) \times \frac{1(1.2+1)}{1.2+1} \times \frac{100+1}{100+1}$$

$$BM25(Q2, D) = log(\frac{1000-200+0.5}{200+0.5}) \times \frac{4(1.2+1)}{1.2+4} \times \frac{100+1}{100+1} + log(\frac{1000-150+0.5}{150+0.5}) \times \frac{1(1.2+1)}{1.2+1} \times \frac{100+1}{100+1}$$

Note:

1. make sufficient assumptions
2. clarify log_2, log_{10}, or log_e if not given.
3. the *log* part is for the IDF

Using J-M smoothing with $\lambda = 0.9$. Further assume that the # of occurrence of each query term is the same as number of documents that contain it.

$$LM(Q1, D) = (0.9 \times \frac{4}{40} + 0.1 \times \frac{200}{40000}) \times (0.9 \times \frac{1}{40} + 0.1 \times \frac{100}{40000})$$

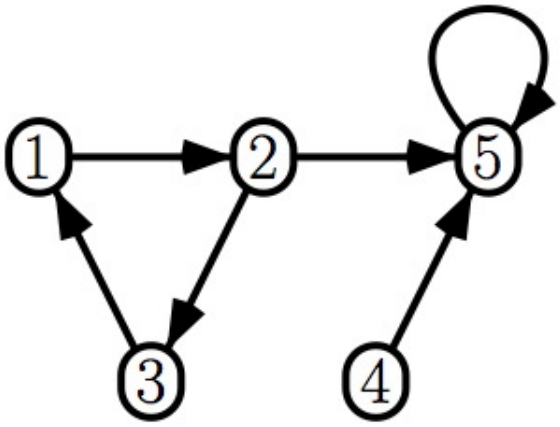
$$LM(Q2, D) = (0.9 \times \frac{4}{40} + 0.1 \times \frac{200}{40000}) \times (0.9 \times \frac{1}{40} + 0.1 \times \frac{150}{40000})$$

Note:

1. make sufficient assumptions.
2. clarify which smoothing is used with setting of parameters .
3. You could also take the *log*, which would turn product into sum.

2 Exercise B

Compute the PageRank score of each node in the graph below. Show your work. In how many iterations does the computation converge?



Factor $d = 0.85$
$PR(D) = \frac{(1-d)}{N} + d \times (\frac{PR(T_1)}{C(T_1)} + ... + \frac{PR(T_n)}{C(T_n)})$

Initiation (Iter 0):
$PR_0(1) = \frac{1}{N} = 0.2$; $PR_0(2) = 0.2$; $PR_0(3) = 0.2$; $PR_0(4) = 0.2$; $PR_0(5) = 0.2$;
(RECOMMENDED)

Iter 1:
$PR_1(1) = 0.2 \times 0.15 + 0.85 \times (\frac{PR_0(3)}{1})$;
$PR_1(2) = 0.2 \times 0.15 + 0.85 \times (\frac{PR_0(1)}{1})$;
$PR_1(3) = 0.2 \times 0.15 + 0.85 \times (\frac{PR_0(2)}{2})$;
$PR_1(4) = 0.2 \times 0.15 + 0.85 \times 0$;
$PR_1(5) = 0.2 \times 0.15 + 0.85 \times (\frac{PR_0(2)}{2} + \frac{PR_0(4)}{1} + \frac{PR_0(5)}{1})$;
Iteratively calculate it, until it converges.
Note:

1. give initiation values.
2. clarify which formula you are using.
3. converge is based on your defined threshold.

3 Exercise C

Show how MapReduce can be used to efficiently solve the following problem: Given a collection of input documents, output all pairs of keywords that co-occur in at least 1000 of the documents.
Write pseudocode for map and reduce functions.
Full points for most efficient implementation.
Hint: is multi-phase MapReduce useful here?

A solution with one iteration is as following:

$Mapper(docId, text)\{$
$List < String > uniqueKeys = uniqueTermsInText(text);$
$for(i = 0; i < uniqueKeys.size(); i++)$
$for(j = i + 1; j < uniqueKeys.size(); j++)$
$emit([uniqueKeys[i], uniqueKeys[j]], 1);$
$\}$

$Reducer([pair, count])\{$
$if(countSUM([pair, count]) > 1000)$
$emit([pair, countSUM([pair, count])]);$
$\}$

In this solution, the number of emitted pairs from the mapper to the reducers is $(n^2 - n)/2$, where n is the number of unique keywords in a document.

To reduce the number of pairs emitted by mappers, we can follow a multiphase solution:

phase 1: reducer emits $< docId, termKey >$ pairs only if the the term appears in at least 1000 documents.

$Mapper < docId, text > \rightarrow Reducer(<< termKey, docId >>) \rightarrow < docId, termKey >$

phase 2: pairs of termKeys are generated per docId by reducer for termKeys generated in first phase. Mapper just emit the input as it is.

$Mapper < docId, termKey > \rightarrow Reducer(<< docId, termKey >>) \rightarrow < docId, pair >$

phase 3: Mapper emits input pair and count(=1) to reducer. Reducer calculates the pair's count and emit the pairs with count greater than 1000,

$Mapper < docId, pair > \rightarrow Reducer(<< pair, 1 >>) \rightarrow < pair, count >$

Using these 3 iterations of MapReduce, the total number of mapper to reducer transmissions reduces. Depending on the cluster configuration and the data properties (e.g., documents length) the first or the second approach may be better.

4 Exercise D

For a specific query Q, suppose that a search engine can produce up to 3 results, where the i-th result has probability 1/(2i) of being relevant. That is, 1st result has probability 1/2, 2nd has 1/4, 3rd has 1/6, and so on. Also, assume Q has a total of 3 relevant results in the collection.
C1: What is the expected average precision (AP) if the engine outputs 2 results?
C2: How many results should the search engine output to maximize the expected AP? Show your calculations and results.
C3: How many results should the search engine output to maximize F (harmonic mean of precision and recall)? Show your calculations and results.

If search engine outputs 1 result:

ExpectedAP@1 = 1/2 * 1/3 = 1/6

ExpectedPrecision@1 = 1/2 * 1 = 0.5

ExpectedRecall@1 = 1/2 * 1/3 = 1/6

ExpectedF@1 = 0.25

If search engine outputs 2 results:

Ranking	Probability	Precision@2	Recall@2	AP@2	F score
r x	1/2 * 3/4	1/2	1/3	(1+0)/3 = 1/3	2/5
r r	1/2 * 1/4	1	2/3	(1+1)/3 = 2/3	4/5
x r	1/2 * 1/4	1/2	1/3	(0+1/2)/3 = 1/6	2/5

ExpectedAP@2 = 3/8 * 1/3 + 1/8 * 2/3 + 1/8 * 1/6 = 11/48 = 0.229
ExpectedPrecision@2 = 3/8 * 1/2 + 1/8 * 1 + 1/8 * 1/2 = 3/8 = 0.375
ExpectedRecall@2 = 3/8 * 1/3 + 1/8 * 2/3 + 1/8 * 1/3 = 1/4 = 0.25
ExpectedF@2 = 3/8 * 2/5 + 1/8 * 4/5 + 1/8 * 2/5 = 0.3

If search engine outputs 3 results:

Ranking	Probability	Precision@3	Recall@3	AP@3	F score
r x x	1/2 * 3/4 * 5/6	1/3	1/3	(1+0+0)/3 = 1/3	1/3
x r x	1/2 * 1/4 * 5/6	1/3	1/3	(0+1/2+0)/3 = 1/6	1/3
x x r	1/2 * 3/4 * 1/6	1/3	1/3	(0+0+1/3)/3 = 1/9	1/3
r r x	1/2 * 1/4 * 5/6	2/3	2/3	(1+1+0)/3 = 2/3	2/3
x r r	1/2 * 1/4 * 1/6	2/3	2/3	(0+1/2+2/3)/3 = 7/18	2/3
r x r	1/2 * 3/4 * 1/6	2/3	2/3	(1+0+2/3)/3 = 5/9	2/3
r r r	1/2 * 1/4 * 1/6	1	1	(1+1+1)/3 = 1	1

ExpectedAP@3 = 0.2615
ExpectedPrecision@3 = 0.305
ExpectedRecall@3 = 0.305
ExpectedF@3 = 0.305

C1: ExpectedAP@2 = 0.229
C2: 3 results
C3: 3 results