

## Web Mining Exercises

Mauro Brunato

Elisa Cilia

April 22, 2009

### Exercise 1

A corpus contains the following five documents:

$d_1$	To be or not to be, this is the question!
$d_2$	I have a pair of problems for you to solve today.
$d_3$	It's a long way to Tipperary, it's a long way to go. . .
$d_4$	I've been walking a long way to be here with you today.
$d_5$	I am not able to question these orders.

The indexing system only considers nouns, adjectives, pronouns, adverbs and verbs. All forms are converted to singular, verbs are converted to the infinitive tense, removes all punctuation marks and translates all letters to uppercase. Conjunctions, prepositions, articles and exclamations are discarded as well. Multiple occurrences of the same term within a document are not counted.

For instance, the phrase

*Hey, it's not too late to solve these exercises!*

becomes

IT BE NOT TOO LATE SOLVE THIS EXERCISE

**1.1)** What is the minimum dimension (number of coordinates) of the TDIDF vector space for this collection of documents?

**1.2)** Fill the  $5 \times 5$  matrix of Jaccard coefficients between all pairs of documents.

**1.3)** Apply an agglomerative clustering procedure to the collection. as a measure of similarity between two clusters  $D_1$  and  $D_2$ , consider the highest similarity between  $d_1$  and  $d_2$ , with  $d_1 \in D_1$  and  $d_2 \in D_2$ .

**1.4)** Draw the resulting dendrogram.

**Solution** — The stripped-down documents are the following (the third columns count the number of different terms in each document, just to ease up the calculation of the Jaccard coefficient):

$d_1$	BE NOT THIS QUESTION	4
$d_2$	I HAVE PAIR PROBLEM YOU SOLVE TODAY	7
$d_3$	IT BE LONG WAY TIPPERARY GO	6
$d_4$	I HAVE BE WALK LONG WAY HERE YOU TODAY	9
$d_5$	I BE NOT ABLE QUESTION THIS ORDER	7

**1.1)** The collection includes 20 different terms: ABLE, BE, GO, HAVE, I, IT, HERE, LONG, NOT, ORDER, PAIR, PROBLEM, QUESTION, SOLVE, THIS, TIPPERARY, TODAY, WALK, WAY, and YOU. Therefore, the vector representation requires at least 20 dimensions.

**1.2)** The table of Jaccard coefficients is the following. Only the upper triangular part is shown, since the Jaccard coefficient is symmetrical.

	$d_1$	$d_2$	$d_3$	$d_4$	$d_5$
$d_1$	1	0	1/9	1/12	4/7
$d_2$		1	0	1/3	1/13
$d_3$			1	1/4	1/12
$d_4$				1	1/7
$d_5$					1

**1.3)** The two most similar documents are  $d_1$  and  $d_5$ , so they can be joined in the same partition. The similarity matrix becomes:

	$\{d_1, d_5\}$	$\{d_2\}$	$\{d_3\}$	$\{d_4\}$
$\{d_1, d_5\}$	1	1/13	1/9	1/7
$\{d_2\}$		1	0	1/3
$\{d_3\}$			1	1/4
$\{d_4\}$				1

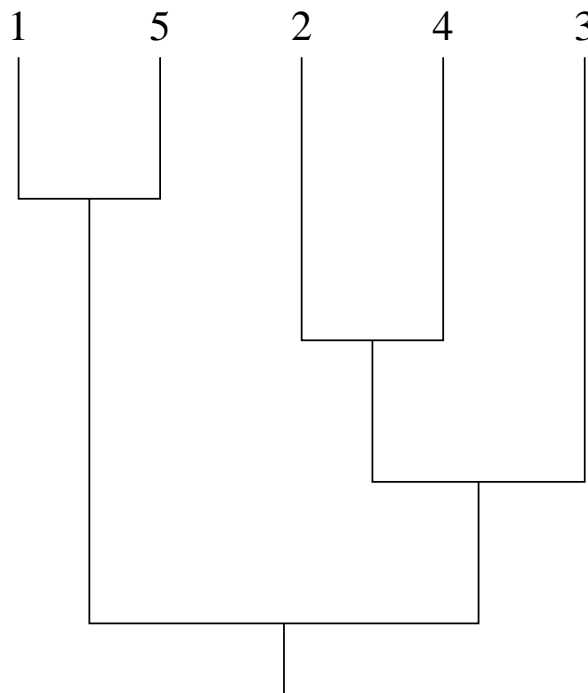
After this step, singletons  $\{d_2\}$  and  $\{d_4\}$  are most similar, and shall be joined:

	$\{d_1, d_5\}$	$\{d_2, d_4\}$	$\{d_3\}$
$\{d_1, d_5\}$	1	1/7	1/9
$\{d_2, d_4\}$		1	1/4
$\{d_3\}$			1

Next, singleton  $d_3$  joins cluster  $\{d_2, d_4\}$ :

	$\{d_1, d_5\}$	$\{d_2, d_3, d_4\}$
$\{d_1, d_5\}$	1	1/7
$\{d_2, d_3, d_4\}$		1

Finally, the two remaining clusters can be merged together. The corresponding dendrogram is the following:

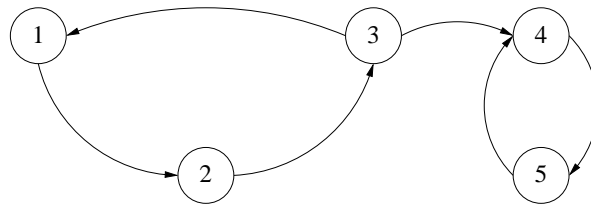


### Exercise 2

In the same setting as in the previous exercise, estimate the Jaccard coefficient for all document pairs based on the application of five random permutations.

**Exercise 3**

The network of references for a set of five hypertexts is given in figure:



Compute the first 5 iterations of the PageRank and HITS algorithms in the following hypotheses:

- No damping factor.
- Initial PageRank vector gives probability 1 to node 1.
- Initial hub and authority vectors are uniformly 1 over all nodes.
- No normalization required.

**Exercise 4**

Let  $D$  be a set of documents over the set  $T$  of terms,  $n_{td}$  counts the number of occurrences of term  $t$  in document  $d$ .

4.1) Consider the following term frequency measures:

$$A_1(t, d) = n_{td}, \quad A_2(t, d) = \begin{cases} 1 & \text{if } n_{td} \neq 0 \\ 0 & \text{otherwise,} \end{cases} \quad A_3(t, d) = \frac{n_{td}}{|d|}, \quad A_4(t, d) = \log(1 + n_{td}).$$

Consider each measure according to each of the following criteria *separately*:

1. The size of a document should not matter (e.g., concatenating two copies of the same document should not change the measure).
2. The number of occurrences of the term should not matter, only its presence is important.
3. Increasing the number of occurrences of a term should have a lesser impact on the measure if the term is already frequent.

4.2) Which of the following are suitable IDF functions, and why?

$$B_1(t) = -\log \left( 1 - \left( \sum_{d \in D} A_1(t, d) \right)^{-1} \right), \quad B_2(d) = \left( 1 + \sum_{t \in T} A_2(t, d) \right)^{-1},$$

$$B_3(t) = \sum_{d \in D} \frac{1}{1 + A_1(t, d)}, \quad B_4(d) = \left( \sum_{t \in T} A_4(t, d) \right)^{-1}$$

**Exercise 5**

A document retrieval system must be implemented in a structured programming language (Java, C, C++). Documents and terms are represented with their numeric IDs.

5.1) Define the appropriate array and record structures to efficiently store the matrix  $n_{td}$  counting the number of occurrences of each term  $t$  in each document  $d$ , considering that it is very sparse. Define the structure to store inverse document frequency values.

5.2) Write a function `retrieve(q)` which, given the array `q` of term indices, returns an array with the IDs of the five nearest documents according to the cosine measure in the TFIDF space.

**Exercise 6**

Describe the hard  $k$ -means algorithm in terms of the Expectation-Maximization framework.

**Solution** — Let  $X = (\mathbf{x}_1, \dots, \mathbf{x}_n)$  be the vector of observables (each being a vector in the TFIDF space), i.e., the  $n$  documents that we want to cluster into  $m$  partitions.

Our clustering model uses the  $m$  cluster centroids as parameters:

$$\Theta = (\theta_1, \dots, \theta_m).$$

Our hypothesis is that document  $\mathbf{x}_i$  belongs to the cluster having the nearest centroid. To this purpose we introduce the hidden variable

$$Y = (\gamma_1, \dots, \gamma_n)$$

where  $\gamma_i \in \{1, \dots, m\}$  defines the true cluster of document  $\mathbf{x}_i$ .

The E-M algorithm works in steps, at the  $s$ -th step we have an initial guess of parameter values  $\Theta_s$ , and we refine it by building the expectation function  $Q(\Theta, \Theta_s)$  and optimizing it. Given the  $s$ -th guess at parameter values

$$\Theta_s = (\theta_1^s, \dots, \theta_n^s),$$

we can compute the corresponding clustering:

$$Y_s = (\gamma_1^s, \dots, \gamma_n^s) \quad \text{where} \quad \gamma_i^s = \arg \min_{j=1, \dots, m} d(\mathbf{x}_i, \theta_j^s). \quad (1)$$

Given this new clustering hypothesis, we can improve our centroid positions:

$$\Theta_{s+1} = (\theta_1^{s+1}, \dots, \theta_n^{s+1}) \quad \text{where} \quad \theta_i^{s+1} = \frac{\sum_{j:\gamma_j^s=i} \mathbf{x}_j}{|\{j : \gamma_j^s = i\}|}. \quad (2)$$

In particular, being centroids the average of the positions of the documents, they minimize the sum of the squares of their distances, so we can reformulate (2) as follows:

$$\theta_i^{s+1} = \arg \min_{\theta} \sum_{j:\gamma_j^s=i} d^2(\theta, \mathbf{x}_j).$$

Note that the  $m$  centroids are determined independently, so that minimization can be done simultaneously on all  $m$  clusters. Let us define

$$Q(\Theta, \Theta_s) = \sum_{i=1}^m \sum_{j:\gamma_j^s=i} d^2(\theta_i, \mathbf{x}_j)$$

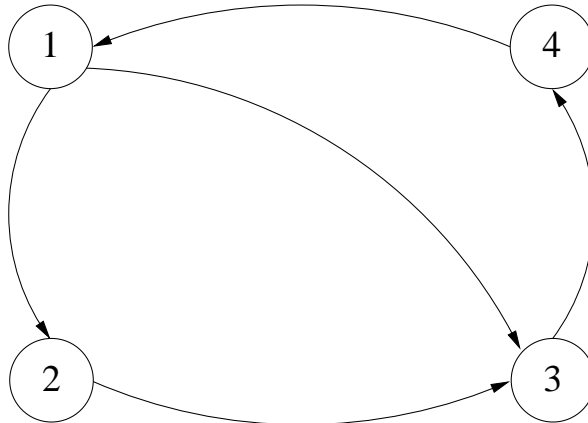
where dependence on  $\Theta_s$  comes from (1). Then the parameter guess for step  $s + 1$  is

$$\Theta_{s+1} = \arg \min_{\Theta} Q(\Theta, \Theta_s)$$

as required by the E-M algorithm. Note that we are *minimizing*, rather than maximizing as in the original E-M formulation.

### Exercise 7

The network of references for a set of four hypertexts is given in figure:



**7.1)** Execute the first four steps of the PageRank algorithm starting from user being with certainty at node 1 (no damping factor).

**7.2)** Compute the stationary PageRank scores of the documents.

**Exercise 8**

Suppose that a query, executed on the same network as Exercise 7, returns nodes 1 and 2, and that we want to use the HITS algorithm in order to rank the pages.

- 8.1) Define the root set and the extended set for the given query.
- 8.2) Compute the first five iterations of the HITS algorithm for the extended set.
- 8.3) Which hub and authority values will asymptotically dominate?

**Exercise 9**

An IR system manages a corpus of six documents. Given the query  $q$ , the system computes the following probabilities for the documents to be relevant:

$i$	1	2	3	4	5	6
$p_i$	100%	80%	20%	80%	0	100%

9.1) What strategy can the system adopt in order to maximize its recall score? What strategy can maximize its precision score?

9.2) Suppose that the only documents that are relevant with respect to query  $q$  are 1, 2, 4 and 6 (of course, the system does not know this). The system implements two alternative algorithms:

- 1. let document  $i$  appear in the returned list iff  $p_i = 100\%$ , or
- 2. let document  $i$  appear in the list with probability  $p_i$ .

Compute the expected values of precision and recall assigned by the user (who knows the actual document relevance) to the list of documents returned by each algorithm.

Hint — Note that algorithm (1) is deterministic, only algorithm (2) is stochastic.

**Solution —**

9.1) Let  $\mathbf{r} = (r_i)$ , where  $r_i$  is the “true” relevance of document  $i$  (remember that the query is fixed). Let  $\mathbf{x} = (x_i)$ , where  $x_i = 1$  iff the IR system returns document  $i$  in response to the query. Then,

$$\text{Precision}_r(\mathbf{x}) = \frac{\mathbf{x} \cdot \mathbf{r}}{\sum_{i=1}^6 x_i}, \quad \text{Recall}_r(\mathbf{x}) = \frac{\mathbf{x} \cdot \mathbf{r}}{\sum_{i=1}^6 r_i}.$$

In other words, the “precision” of the answer is the amount of relevant documents within the list provided by the IR system. Its maximum value is attained when all returned documents are relevant, so we need to return only the two documents, 1 and 6, which are certainly relevant to the user. The “recall” of the answer is its property of containing as many relevant documents as possible, and it is maximized by returning all documents (with the possible exception of 5, which is irrelevant for sure).

9.2) In the first case, the IR system provides a deterministic answer, having precision 100% and recall 50%. In the second case, we need to compute precision and recall scores for all possible return strings, and compute their probability-weighted average:

$$E(\text{Precision}) = \sum_{\mathbf{x}} \Pr(\mathbf{x}) \text{Precision}_r(\mathbf{x}), \quad E(\text{Recall}) = \sum_{\mathbf{x}} \Pr(\mathbf{x}) \text{Recall}_r(\mathbf{x}).$$

Note that documents 1 and 6 are always returned, while document 5 is never returned; moreover, documents 2 and 4 are indistinguishable, so we can determine the following table, where precision (left) and recall (right) scores are provided together with their probabilities (in parentheses).

		$x_2 + x_4$					
		0		1		2	
		(.04)		(.32)		(.64)	
$x_3$	0    (.8)	$\frac{2}{5}$	$\frac{2}{4}$	$\frac{3}{5}$	$\frac{3}{4}$	$\frac{4}{4}$	$\frac{4}{4}$
		(.032)		(.256)		(.512)	
	1    (.2)	$\frac{2}{3}$	$\frac{2}{4}$	$\frac{3}{4}$	$\frac{3}{4}$	$\frac{4}{5}$	$\frac{4}{4}$
		(.008)		(.064)		(.128)	

Finally,

$$E(\text{Precision}) = .8 + \frac{2}{3} \cdot .008 + \frac{3}{4} \cdot .064 + \frac{4}{5} \cdot .128 \approx .8 + .005 + .048 + .102 \approx 96\%,$$

$$E(\text{Recall}) = \frac{2}{4} \cdot .04 + \frac{3}{4} \cdot .32 + \frac{4}{4} \cdot .64 = .02 + .24 + .64 = 90\%.$$

**Exercise 10**

With the same data of Exercise 9, suppose that the system uses algorithm (1).

**10.1)** Compute the expected precision and recall scores from the point of view of the IR system, who only knows the probabilities  $p_i$  for document  $i$  to be relevant.

**Solution** — In this case the IR system's answer is known, but the actual document relevance is a random variable with the given probabilities. Therefore, the average values must be computed against probabilities of the unknown  $r$ :

$$E_r(\text{Precision}) = \sum_r \Pr(r) \text{Precision}_r(x), \quad E_r(\text{Recall}) = \sum_r \Pr(r) \text{Recall}_r(x).$$

We know the answer  $x$  of the IR system, which is  $(1, 0, 0, 0, 0, 1)$ , therefore we can compute a table which is similar to that of Exercise 9:

		$x_2 + x_4$			
		0 (.04)	1 (.32)	2 (.64)	
$x_3$	0 (.8)	$\frac{2}{2} \mid \frac{2}{2}$ (.032)	$\frac{2}{2} \mid \frac{2}{3}$ (.256)	$\frac{2}{2} \mid \frac{2}{4}$ (.512)	
	1 (.2)	$\frac{2}{2} \mid \frac{2}{3}$ (.008)	$\frac{2}{2} \mid \frac{2}{4}$ (.064)	$\frac{2}{2} \mid \frac{2}{5}$ (.128)	

Therefore, as expected,

$$E_r(\text{Precision}) = 100\%$$

because we are sure that only relevant documents are returned. On the other hand,

$$E_r(\text{Recall}) = \frac{2}{2} \cdot .032 + \frac{2}{3} \cdot .256 + \frac{2}{4} \cdot .512 + \frac{2}{3} \cdot .008 + \frac{2}{4} \cdot .064 + \frac{2}{5} \cdot .128 \approx .032 + .171 + .256 + .005 + .032 + .051 \approx 55\%.$$

**Exercise 11**

Write in your favorite high-level language a function that implements the FastMap algorithm. In particular, define what input must be provided and which output shall be returned.

**Solution** — Let matrix  $d$  be the input data (mutual distances between couples of items). The matrix is symmetric, so many optimizations are possible. Let  $x$  be the output matrix with one column per document and one row per extracted coordinate. We assume that the number of documents  $n$  and the number of extracted dimensions  $m$  are encoded into matrix sizes; otherwise, we can pass them as two additional integer parameters.

```

1. void FastMap (double d[], double x[])
2. {
3.     int n = d.length, m = x.length;
4.     for (int s = 0; s < m; s++) {
5.         i, j ← arg maxi,j d[i][j];
6.         for (int k = 0; k < n; k++)
7.             x[s][k] ←  $\frac{d[i][k]^2 + d[j][k]^2 - d[i][j]^2}{2d[i][j]}$ ;
8.         for (int i1 = 0; i1 < n; i1++)
9.             for (int j1 = 0; j1 < n; j1++)
10.                d[i1][j1] ←  $\sqrt{d[i1][i1]^2 + d[j1][j1]^2 - (x[s][i1] - x[s][j1])^2}$ ;
11.     }
12. }
```

*Repeat for the desired number of coordinates*  
*Find the two farthest points*  
*Compute the s-th coordinate*

*Recompute the mutual distances*

Note that the term within the square root sign at line 10 might be negative, so a bit of care must be taken when actually implementing the algorithm...

**Exercise 12**

The columns of the following matrix represent the coordinates of a set of documents in a TFIDF space:

$$A = \frac{1}{\sqrt{6}} \begin{pmatrix} 2 & 0 & 2 \\ 0 & 1 & 0 \\ 2 & 1 & 2 \\ 2 & -1 & 2 \end{pmatrix}$$

Let document similarity be defined by the cosine measure (dot product).

**12.1)** Compute the rank of matrix  $A$ .

**12.2)** Let  $\mathbf{q} = (1, 3, 0, -2)^T$  be a query. Find the document in the set that best satisfies the query.

**12.3)** Given the matrices

$$U = \alpha \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad V = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix}$$

determine coefficient  $\alpha$  and the diagonal matrix  $\Sigma$  so that  $U$  is column-orthonormal and  $A = U\Sigma V^T$ .

**12.4)** Project the query  $\mathbf{q}$  onto the LSI space defined by this decomposition and verify the answer to question 12.2. Why isn't the requirement that  $V$  be column-orthonormal important in our case?

**12.5)** Suppose that we want to reduce the LSI space to one dimension. Show how the new approximate document similarities to  $\mathbf{q}$  are computed.

**Solution —**

**12.1)** Notice that  $A$  has two linearly dependent (actually equal) columns (thus  $\text{rk } A < 3$ ), while the first two columns are independent (thus  $\text{rk } A \geq 2$ ), therefore  $\text{rk } A = 2$ .

**12.2)** Similarities are computed by dot products, let's do it in a single shot for all documents:

$$A^T \mathbf{q} = \frac{1}{\sqrt{6}} \begin{pmatrix} -2 \\ 5 \\ -2 \end{pmatrix};$$

The most similar is document 2.

**12.3)** The column normality condition for matrix  $U$  implies  $\alpha = 1/\sqrt{3}$ . By expliciting the calculation of some entries of matrix  $A$ , we obtain

$$\Sigma = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}.$$

**12.4)** Projection onto the document LSI space is achieved via  $\Sigma^{-1}U^T$ :

$$\hat{\mathbf{q}} = \Sigma^{-1}U^T \mathbf{q} = \frac{1}{\sqrt{3}} \begin{pmatrix} -1/2 \\ 5 \end{pmatrix}.$$

Similarity to the documents is computed via the  $V\Sigma^2$  matrix. If all computations are right,

$$V\Sigma^2 \hat{\mathbf{q}} = A^T \mathbf{q}.$$

**Exercise 13**

Specify in the MapReduce framework the Map and Reduce functions to find the number of occurrences of one/more given pattern/s in a collection of documents.

**Solution —** Let us define the two functions.

$$\begin{array}{lll} \text{Map:} & \mathbb{N} \times T^* & \longrightarrow (T \times \mathbb{N})^* \\ & (\text{offset, line}) & \mapsto [(\text{match}, 1)] \\ \text{Reduce:} & T \times \mathbb{N}^* & \longrightarrow (T \times \mathbb{N})^* \\ & (\text{match}, [n_1, \dots, n_k]) & \mapsto [(\text{match}, \sum n_i)] \end{array}$$

Function *Map* receives a key (related to the document ID or line offset), which we can disregard, and a sequence of terms (a line or a full document). It gives as output a list of pairs (match, 1), one for each match of the pattern in the received value.

Function *Reduce* takes as input a pair (match, [n<sub>1</sub>, ..., n<sub>k</sub>]) where the value part is a list of previously computed occurrences (originally all 1's) and returns the list of matching patterns (only one element in this case) with the number of occurrences for each match.

The pseudo-code for the Map and Reduce functions is the following:

```

1. map (offset, line)
2.   [ while pattern.matches (line)
3.     emit (pattern, 1);

```

```

1. reduce (match, values)

```

*values is an iterator over counts*

```

2.   [ result = 0;
3.     for each v in values
4.       result += v;
5.     emit (match, result);

```

### Exercise 14

Consider a document corpus with  $m = 6$  documents,  $n = 5$  terms. Suppose that documents have been clustered into  $m' = 2$  clusters and terms have been clustered into  $n' = 2$  clusters. The following document-term matrix and cluster attribution has been determined:

		1	2	3	4	5
		1	1	2	1	2
1	1		X			
2	2			X		X
3	1	X	X		X	
4	1		X		X	
5	2			X	X	
6	2		X	X		

**14.1)** Consider the Jaccard index as similarity measure. Suppose that all we know about a document is that it contains term 2. Which other term is most likely to occur in the same document?

**14.2)** Compute the following probabilities for all suitable index values:

- the probability  $p_{i'}$  that a random document belongs to cluster  $i'$ ;
- the probability  $p_{j'}$  that a random item belongs to cluster  $j'$ ;
- the probability  $p_{i'j'}$  that a document in cluster  $i'$  contains a term in cluster  $j'$ .

**14.3)** Perform a step of the Gibbs Sampling technique on document 4 by computing the posterior probabilities  $\pi_{4 \rightarrow i'}$  for  $i' = 1, 2$ . Was the proposed cluster attribution likely, or will it be probably changed?

### Exercise 15

Given the following three documents (each row is a document and each cell corresponds to a term and contains its term id)

1	1	2	1	5	2	2
2	4	3	3	1	2	1
3	2	2	5	4	3	3

assume a multinomial model for the document generation and estimate the parameters of the term distribution by using the maximum likelihood estimation method. (*Show all the steps to obtain the best parameter estimation*) As all the documents have the same length, assume  $P(L = l_d | \Theta) = 1$  in the multinomial model  $P(l_d, n(d, t) | \Theta)$ .

**Solution** — The multinomial model for a document generation is the following:

$$P(d|\Theta) = P(l_d, n(d, t)|\Theta) = P(L = l_d|\Theta) \binom{l_d}{\{n(d, t)\}} \prod_{t \in d} \theta_t^{n(d, t)} \quad (3)$$

where  $\Theta = (\theta_t \forall t \in T)$  and  $T$  is our vocabulary.

We have a set  $D = (d_1, d_2, d_3)$  of iid observations, thus  $P(D|\Theta) = \prod_{d \in D} P(d|\Theta)$

In this case, as we assume  $P(L = l_d|\Theta) = 1$ , then:

$$P(D|\Theta) = \left( \binom{l_d}{\{n(d, t)\}} \right) \prod_d \prod_t \theta_t^{n(d, t)} \quad (4)$$

This model corresponds to the likelihood function  $L(\Theta|D)$ .

We want to estimate the  $\Theta$  parameters which maximize the likelihood function. We can do that by computing the partial derivatives with respect to each one of the parameters  $\theta_t$  and putting them equal to zero.

From now on we will consider the log likelihood function which is easier to derive:

$$\log L(\Theta|D) = \log \left( \binom{l_d}{\{n(d, t)\}} \right) + \sum_d \sum_t n(d, t) \log(\theta_t) \quad (5)$$

moreover there is one constraint to the maximization, namely  $\sum_t \theta_t = 1$ , thus we perform a standard Lagrangian optimization:

$$\frac{\partial}{\partial \theta_t} \left[ \log \left( \binom{l_d}{\{n(d, t)\}} \right) + \sum_d \sum_t n(d, t) \log(\theta_t) - \lambda \left( \sum_t \theta_t - 1 \right) \right] = 0 \quad (6)$$

$$\frac{\partial \log L}{\partial \theta_t} = \frac{\sum_d n(d, t)}{\theta_t} - \lambda = 0 \quad (7)$$

then the estimation of our parameters is:

$$\theta_t = \frac{\sum_d n(d, t)}{\lambda} \quad (8)$$

In order to compute the Lagrangian multiplier  $\lambda$ , we consider the constraint  $\sum_t \theta_t = 1$  which becomes  $\sum_t \frac{\sum_d n(d, t)}{\lambda} = 1$  and thus  $\lambda = \sum_d \sum_t n(d, t) = \sum_d l_d = n \cdot l_d$ , where  $n = |D|$ .

The parameters are:

$$\theta_t = \frac{\sum_d n(d, t)}{n \cdot l_d} \quad (9)$$

Substituting the values we obtain  $\theta_1 = \frac{4}{3 \times 6} = \frac{2}{9}$ ,  $\theta_2 = \frac{7}{18}$ ,  $\theta_3 = \frac{5}{18}$ ,  $\theta_4 = \frac{1}{9}$ ,  $\theta_5 = \frac{1}{9}$ .

### Exercise 16

Solve the previous exercise by using the least square method. (Show all the steps to obtain the best parameter estimation)

### Exercise 17

Given the following three documents (each row is a document and each cell corresponds to a term and contains its term id)

1	1	2	1	5	2	2	3	2
2	1	3	1	5	2	2		
3	2	2	5	4	3	3	2	

assume a multinomial model for the document generation and estimate the parameters of the term distribution by using the least square method. (Show all the steps to obtain the best parameter estimation)

**Exercise 18**

Given the following relevance ranking vector in response to a query  $q$ :

$$d_1, \underline{d_2}, \underline{d_3}, d_4, \underline{d_5}, \underline{d_6}$$

(the underlined documents are exactly all the relevant ones)

**18.1)** Determine the interpolated precision at level  $\rho = 0.5$  of recall,

**18.2)** Determine the “global”  $F_1$  – measure (for the system returning all the six documents),

**18.3)** Determine the Break Even Point (BEP), which is the point of equivalence between (interpolated) precision and recall.

**Solution — 18.1)** We are given a ranked list of documents returned in response to a query  $q$  with their associated relevance values. In this ranked retrieval context, precision and recall can be computed by considering as the set of retrieved documents the top  $k$  ranked documents:

<b>k</b>	$r_k$	<b>R</b>	<b>P</b>
0	0	0	1
1	0	0	0
2	1	0.25	0.5
3	1	0.5	0.66
4	0	0.5	0.5
5	1	0.75	0.6
6	1	1	0.66

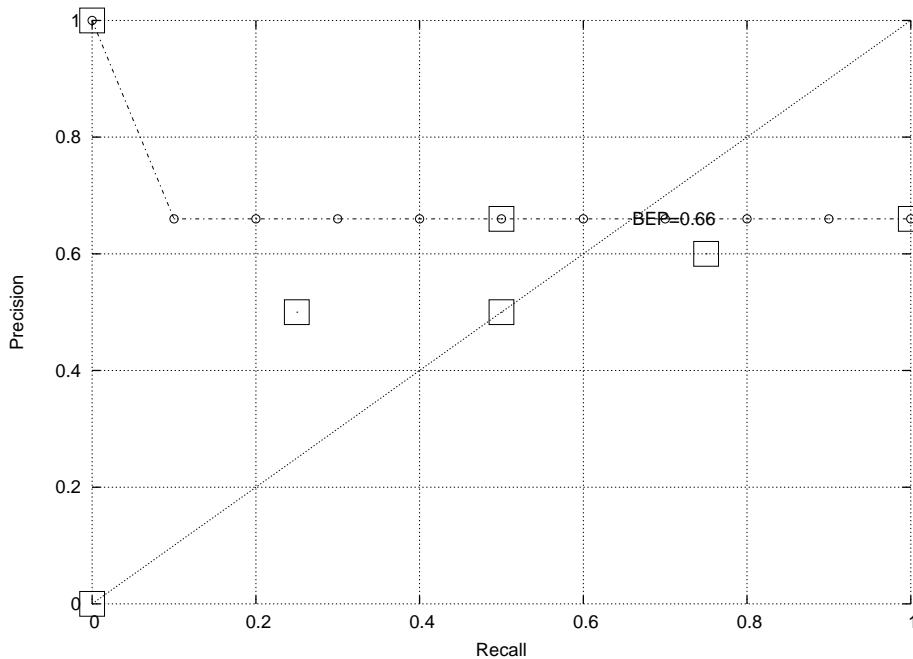
The interpolated precision  $P_{interp}$  at a certain level  $\rho$  of recall is defined as the highest precision found for any recall level  $\rho' \geq \rho$ :

$$P_{interp}(\rho) = \max_{\rho' \geq \rho} P(\rho')$$

Thus the interpolated precision at level  $\rho = 0.5$  of recall is  $P_{interp}(0.5) = 0.66$

**18.2)** The  $F_1$  – measure =  $\frac{2 \times P \times R}{P + R}$  when all the documents are returned is  $F_1 = 0.795$  (by taking the P and R values computed in the last row of the table).

**18.3)** To find the BEP we plot the 11-point interpolated precision curve:



**Exercise 19**

Suppose that a user's initial query is *cheap CDs cheap DVDs extremely cheap CDs*. The user examines two documents,  $d_1$  and  $d_2$ . She judges  $d_1$ , with the content *CDs cheap software cheap CDs* relevant and  $d_2$  with content *cheap thrills DVDs* nonrelevant. Assume that we are using direct term frequency (with no scaling and no document frequency). There is no need to length-normalize vectors. Using Rocchio relevance feedback what would the revised query vector be after relevance feedback? Assume  $\alpha = 1$ ,  $\beta = 0.75$ ,  $\gamma = 0.25$ .

(*"An Introduction to Information Retrieval" preliminary draft, Manning, Raghavan, Schütze, Cambridge University Press 2008*)

**Exercise 20**

Omar has implemented a relevance feedback web search system, where he is going to do relevance feedback based only on words in the title text returned for a page (for efficiency). The user is going to rank 3 results. The first user, Jinxing, queries for:

*banana slug*

and the top three titles returned are:

1. *banana slug Ariolimax columbianus*
2. *Santa Cruz mountains banana slug*
3. *Santa Cruz Campus Mascot*

Jinxing judges the first two documents relevant, and the third nonrelevant. Assume that Omar's search engine uses term frequency but no length normalization nor IDF. Assume that he is using the Rocchio relevance feedback mechanism, with  $\alpha = \beta = \gamma = 1$ . Show the final revised query that would be run. (Please list the vector elements in alphabetical order.)

(*"An Introduction to Information Retrieval" preliminary draft, Manning, Raghavan, Schütze, Cambridge University Press 2008*)