1 Short Answers 1 (10 pts)

Indicate T(rue) or F(alse).

1. Stemming increases the size of the dictionary in an inverted index	
2. Using positional postings lists, we can determine the width of the smallest window in a document containing all query terms in time linear in the lengths of the postings lists of the query terms	
3. Skip pointers speed up a disjunctive X OR Y query	
4. Using SVMs with nonlinear kernels dramatically improves performance for IR tasks	
5. When sufficient data is available, SVMs generally perform as well or better than other common classifiers	
6. K-means clustering (initialized with random centroids) is deterministic	
7. K-means requires more memory than C-HAC	
8. One can choose the number of clusters after running C-HAC	
9. C-HAC has higher running time than one iteration of K-means	
10. Recall of documents is more important for a legal domain IR system than for a modern web search engine	
11. Precision at 1 is more important for a legal domain IR system than for a modern web search engine	
12. Specialized queries with /k and wildcards are more important for a legal domain IR system than for a modern web search engine	
13. Compared to just a term's tf, multiplying by idf to give a tf-idf score always gives a bigger number	
14. For multi-term queries, the WAND algorithm will always provide a savings in retrieval cost over standard postings traversal	
15. Link analysis provides scoring that is spam-proof	
16. SVMs are only usable when the classes are linearly separable in the feature space	
17. Adding training data always results in a monotonic increase in the accuracy of a Naive Bayes classifier	
18. The main reason to take the logs of probabilities in Naive Bayes is to prevent underflow	
19. Using a variable byte code gives better compression than using gamma encoding	
20. The centroid of a set of (length normalized) unit vectors is a unit vector	

Solution

1. Stemming increases the size of the dictionary in an inverted index. FALSE
2. Using positional postings lists, we can determine the width of the smallest window in a document containing all query terms in time linear in the lengths of the postings lists of the query terms. TRUE
3. Skip pointers speed up a disjunctive X OR Y query
   FALSE
4. Using SVMs with nonlinear kernels dramatically improves performance for IR tasks.
   FALSE
5. When sufficient data is available, SVMs generally perform as well or better than other common classifiers
   TRUE
6. K-means clustering (initialized with random centroids) is deterministic.
   FALSE
7. K-means requires more memory than C-HAC.
   FALSE
8. One can choose the number of clusters after running C-HAC.
   TRUE
9. C-HAC has higher running time than one iteration of K-means.
   TRUE
10. Recall of documents is more important for an IR system for the legal domain than in a modern web search engine.
    TRUE
11. Precision at 1 is more important for an IR system for the legal domain than in a modern web search engine.
    FALSE
12. Specialized queries with /k and wildcard is more important for an IR system for the legal domain than in a modern web search engine.
    TRUE
13. Compared to just a term's tf, multiplying in an idf to give a tf-idf score always gives a bigger number.
    FALSE
14. For multi-term queries, the WAND algorithm will always provide a savings in retrieval cost over standard postings traversal
    FALSE
15. Link analysis provides scoring that is spam-proof
    FALSE
16. SVMs are only usable when the classes are linearly separable in the feature space
    FALSE
17. Adding training data always results in a monotonic increase in the accuracy of a Naive Bayes classifier
    FALSE

18. The main reason to take the logs of probabilities in Naive Bayes is to prevent underflow
TRUE
19. Using a variable byte code give better compression than using gamma encoding
FALSE
20. The centroid of a set of (length normalized) unit vectors is a unit vector
FALSE

2 Short Answers 2 (10 pts)

1. [2pts] Consider a collection with one billion tokens (i.e., with 10^9 tokens). Suppose the first 1,000 of these tokens results in a vocabulary size of 1,000 terms, and the first 100,000 tokens results in a vocabulary size of 10,000 terms. Use *Heap's law* to estimate the vocabulary size of the whole collection.

2. [2pts] For a conjunctive query, is processing postings lists in increasing order of their lengths guaranteed to be optimal execution order? Explain why it is, or give an example where it isn't.

3. [2pts] In one sentence, why are pairwise ranking approaches more successful for learning to rank search results than classification or regression approaches?

4. [2pts] For learning search engine rankings, what are two problems with adopting a straightforward approach of minimizing pairwise ranking errors, which have been studied and corrected in subsequent research?

   a)

   b)

5. [2pts] In the Binary Independency Model, the retrieval status value formula for each query term i contains the fraction $(1 - r_i)/r_i$ where $r_i = (n_i - s_i)/(N - S)$. Here, n_i is the total number of documents containing the term i, of which s_i documents are relevant to the query.

N is the total number of documents in the collection, of which S documents are relevant.

It was suggested that this quantity $\log(1 - r_i)/r_i$ can be well approximated by the IDF for term $i = \log N/n_i$. What are the two assumptions necessary to get this result?

a)

b)

Solution

1. According to Heaps' law, $n = kT^b$. So, $1000 = k1000^b$ and $10000 = k100000^b$. Solving the two eqs, $\log k$ is 1.5 and b is 0.5. The final answer is 10^6.

2. Not guaranteed to be optimal. Counterexample
   a := 5, 6
   b := 5,6,15
   c := 7,8,9,10

3. The scale of goodness of a search result to a query is not an absolute scale; it it a decision of goodness relative to other documents in the collection (or just information available in the world).

4. a) Correct ranking of highly relevant documents is more important than correct ranking of slightly relevant documents versus other slightly relevant or nonrelevant documents
   b) If some queries have very many results, the pairwise rankings between them tend to dominate the training of the classifier. You want a learning regime more like each query contributes equally in training the classifier.

5. Since $\log(1 - r_i)/r_i = \log(N - n_i - S - s_i)/(n_i - s_i)$, the two assumptions:

   (a) Non-relevant documents with and without the term i are approximated by all documents in the collection with and without the term respectively.

   ($N - n_i - S - s_i \approx N - n_i$ and $n_i - s_i \approx n_i$)

   (b) The total number of documents in the collection is much larger than the number of documents containing a term.

   ($N \gg n_i \Rightarrow \log \frac{N-n_i}{n_i} \approx \log \frac{N}{n_i}$)

3 Short Answers 3 (10 pts)

1. [1pt] What is the largest number that can be stored in 4 bytes using unary encoding?

2. [2pts] Many systems, e.g., Lucene, provide a separate function to index a set of documents in bulk, instead of just one document? What do you think is an advantage of such a function?

3. [3pts] Assume that postings lists are *gap encoded* using γ codes. Using this encoding, suppose that the postings list for the term **information** is the bit sequence:

   1111 1111 1011 1100 1101 0011 1110 0000 0

   and the postings list for the term **retrieval** is the bit sequence:

   1111 1111 1100 0000 0011 1011 1101 111

   What docids match the following query:

   information AND NOT **retrieval**

4. [4pts] Suppose the vocabulary for your inverted index consists of the following 6 terms:

   elite
   elope
   ellipse
   eloquent
   eligible
   elongate

   Assume that the dictionary data structure used for this index stores the actual terms using *dictionary-as-a-string* storage with *front coding* and a *block size* of 3. Show the resulting storage of the above vocabulary of 6 terms. Use the special symbols * and $\diamondsuit$ as used in the discussion on front coding in Chapter 5.

Solution

1. 31 (11111111 11111111 11111111 11111110)
2. Bulk indexing can be much faster than indexing individual documents because less number of merges (and hence disk seeks) may be required to construct the final index.

3. Gaps for "information" are
   998, 2, 32
   So. Docids for "information" are
   998, 1000, 1032

   Gaps for "retrieval" are
   1031, 1, 31
   So docids for "retrieval" are
   1031, 1032, 1063

   So information AND NOT retrieval has docids
   998, 1000

   (Remember that gamma code 0 means the value 1)

4. 8el*igible3 ⋄ ite5 ⋄ lipse
   8elo*ngate2 ⋄ pe5 ⋄ quent

4 Clustering (10 pts)

1. [4pts] Calculate purity and Rand Index of the following two clusterings. D_i's are documents and C_i's are classes. (Purity of a clustering is an average of purity of individual clusters.)
   The true labels of the documents are:
   $\{(D_1 : C_1), (D_2 : C_2), (D_3 : C_1), (D_4 : C_1), (D_5 : C_2)\}$

   Clustering 1: Purity:
   Cluster 1: D_1
   Cluster 2: D_2 Rand Index:
   Cluster 3: D_3
   Cluster 4: D_4
   Cluster 5: D_5

   Clustering 2: Purity:
   Cluster 1: D_1, D_2, D_3, D_4
   Cluster 2: D_5 Rand Index:

2. [2pts] Is purity a good evaluation measure by itself? In 1–2 sentences write why or why not.

3. [4pts] Each iteration of K-means can be run using the Map-Reduce framework. Write down in 1-2 sentences what would be the (key, value) pairs in the map and the reduce step.

   Map step:

   Reduce step:

Solution

1. Purity Clustering1 is 1 and Clustering2 is 7/8
   Purity of individual clusters in Clustering1 is 1, so the average purity is 1. For Clustering2, purity of Cluster 1 is 3/4 and of Cluster 2 is 1, giving an average of 7/8.

   RI of Clustering1 is 5/10 and Clustering2 is 6/10
   RI is defined as (number of pairs of elements that are in the same set and have same class + number of pairs of elements that are in different sets and have different class) divided by total number of possible pairs of elements, which is $\binom{5}{2}$.

2. Purity is not a good measure of evaluation itself. As we can see in the above question, purity of Clustering1 is more than that of Clustering2; purity is maximum when each data point is in a different cluster.

3. Map step: (keys are docids with their document vectors, the values are clusterids with the centroid vectors)

   Reduce step: (keys are clusterids, values are documents (docids and their vectors) assigned to the clusterid)

   In the map step, the distance between documents and centroids are computed, and each document is assigned to a cluster. In the reduce step, new centriods are computed.

5 Link Analysis (10 pts)

Given a collection of text, consider a Markov Chain M built as follows. We have one state for each term in the collection. The transition probability from any state t_1 to another t_2 is the fraction (in the text collection) of times that term t_1 is immediately followed by term t_2. We only have states

for dictionary terms; ignore start/end-of-string markers/symbols.

Consider the text: *a rose is a rose is a rose*

1. [2pts] Draw the corresponding Markov Chain M.

2. [1pt] Is M ergodic?

3. [1pt] We now augment M with a teleportation operation, to create a new Markov Chain M_1. At each step of M_1, we follow M with probability 87%, while with probability 13% we jump to a random state. Is M_1 ergodic?

4. [3pts] Compute the stationary distribution of M_1.

5. [1pt] Is the vector of stationary probabilities of M_1 the same as the vector of fractions of occurrences for each term in the text of part (1)?

6. [2pts] Show where the addition of a single word in the text of part (1) would change the answer in part (5) to the question: are the two vectors the same?

Solution:

1. The Markov chain is a triangle with arrows in one direction only. The three states are "a", "rose" and "is" with arrows from each to the next cyclically in this order.

2. No. It is periodic.

3. Yes.

4. By symmetry the distribution is uniform, hence (1/3, 1/3, 1/3).

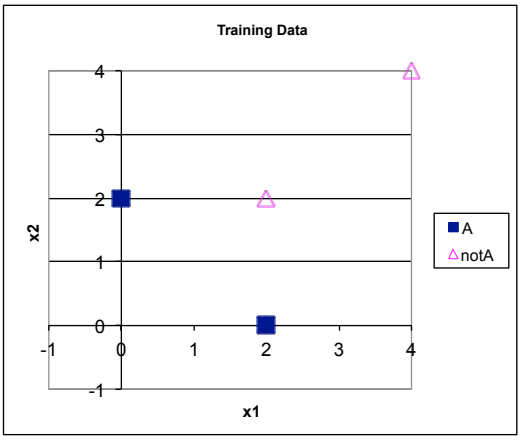
5. No; the occurrence frequencies are (3/8, 3/8, 1/4).

6. Append the word "is" to either the beginning or the end of the given text.

6 Classification (10 pts)

We are training an SVM classifier on this small training data set of 4 points:

(2,0)	Class A	(2, 2)	Class not-A
(0,2)	Class A	(4, 4)	Class not-A



Let the class A correspond to $y_i = +1$.

1. [1pt] Is this data linearly separable?

2. [1pt] What is the (geometric) margin for a hard-margin SVM on this data?

3. [2pts] Write an equation for the optimal separating hyperplane for a hard-margin SVM trained on this data. Work out the answer geometrically, based on the data points above.

4. [1pt] What is the SVM decision function (giving the class assignment)?

5. [2pts] For the algebraic formulation, we impose the standard constraint that $\mathbf{w}^T\mathbf{x}_i + b \geq 1$ if $y_i = 1$ and $\mathbf{w}^T\mathbf{x}_i + b \leq -1$ if $y_i = -1$, and then proceed to minimize $\|w\|$. What weight vector $\mathbf{w}$ and corresponding b achieves this?

6. [1pt] What is the name of the quantity calculated by $2/\|\mathbf{w}\|$ and what is its value here?

7. [2pts] Consider now a soft-margin SVM, where the value of C is not too high, so that one but only one point will be moved with a slack variable, as far as needed to maximize the margin (concretely, $C = 0.1$ will achieve this effect given this data, but you don't need to use this value to answer the question). What will the SVM decision function be now, and what is the margin?

Solution

1. Yes
2. $\sqrt{2}$

   The sides of the square are length 2, so the diagonal between the blue points is length $\sqrt{8} = 2\sqrt{2}$, and so the margin is half this.

3. $-x_1 - x_2 + 3 = 0$

   The separating hyperplane will have slope -1 and go through $(1.5, 1.5)$. The normal vector will have slope 1, so coefficients $w = (a, a)$. So the decision boundary will be $x_1 + x_2 - 3 = 0$. SVM linear classifier will be $y = -x_1 - x_2 + 3 = 0$ (or $y = x_1 + x_2 - 3 = 0$)

4. $y = \text{sign}(3 - x_1 - x_2)$

   Or any scaling of this by a constant. Note that it's the bottom left side that is class $+1$.

5. $w = (-1, -1)$, with $\|w\| = \sqrt{2}$, and $b = 3$.

   $\|w\|$ will be minimized when the $\geq$ and $\leq$ relations shown in the question are replaced by equalities for the support vectors, such as $(2, 2)$. So we have:
   $a.2 + a.2 + b = -1$
   $a.0 + a.2 + b = +1$
   Therefore $a = -1$ and $b = 3$. So the weight vector is $w = (-1, -1)$, with $\|w\| = \sqrt{2}$, and $b = 3$.

6. The (functional) margin for the dataset. Again, $\sqrt{2}$.

7. $y = \text{sign}(-x_1 - x_2 + 5)$. The margin is now $3\sqrt{2}$.

7 Evaluation (12 pts)

1. [2pts] In the class, we discussed several ways to evaluate ranking systems, for e.g., precision, recall, NDCG etc. However, for all these metrics, we need the relevance values for the results. Two possible methods to collect relevance values, as mentioned in class, are: a) click feedback from users, b) expert judgements (like what we did for the search ratings task for PA3). Write one advantage and one disadvantage of each of these methods.

   a) Click feedback:

   b) Expert judgments:

2. [1pt] An alternative method to evaluate ranking systems is to do a pairwise comparison of two ranking systems instead of evaluating an individual system, i.e. if you have two ranking systems A and B, instead of evaluating them independently, you can perform the same query on both systems and compare the results. Write why this might give better results than

evaluating individual systems independently.

3. [3pts] One such method, which uses click feedback to compare two ranking systems, is balanced interleaving. Given a query q, it combines the results, $A = (a_1, a_2, \ldots)$ of the first system and $B = (b_1, b_2, \ldots)$ of the second system, into one consolidated ranking, I as shown in Algorithm 1. Note that this is a simple merge algorithm which breaks ties using a random coin toss (done once at the beginning).

Algorithm 1 Balanced Interleaving

Input: Rankings $A = (a1; a2; \ldots)$ and $B = (b1; b2; \ldots)$

$I := ()$; //output ranking list, initialized to empty list
$k_a := 1; k_b := 1$; //counters for the lists A and B, respectively
$AFirst :=$ randomly select 0 or 1 //decide which ranking gets priority
while ($k_a <= |A|$ **and** $k_b <= |B|$) **do** //if not at end of A or B
    if ($k_a < k_b$ **or** ($k_a == k_b$ **and** $AFirst == 1$)) **then**
        if $A[k_a] \notin I$ **then** I.append($A[k_a]$) //append a result from A
        $k_a := k_a + 1$
    else
        if $B[k_b] \notin I$ **then** I.append($B[k_b]$) //append a result from B
        $k_b := k_b + 1$
    end if
end while
Output: Interleaved ranking $I = (i_1; i_2; \ldots)$

If the two input ranking lists are:
A: (a; b; c; d; g; h)
B: (b; e; a; f; g; h)
what is the output ranking I (if A wins the toss, i.e., $AFirst = 1$)?

4. [3pts] When the user enters a query q in the search engine, the back end merges the results of the two systems A and B using Algorithm 1, and the user is presented with the output list I. The user then clicks on one or more of the results, and these clicks are used to figure out which ranking, A or B, is preferred by the user.

Formally, let $(c_1; c_2; \ldots)$ be the ranks of the clicks w.r.t. I and c_{max} be the rank of the lowest clicked link, (with i_{cmax} being the corresponding result in list I), i.e., if the user clicks on the 1^{st}, 3^{rd} and 7^{th} link, $c_{max} = 7$. Let, $k = \min\{j : (i_{cmax} = a_j) \vee (i_{cmax} = b_j)\}$; we find how many clicked results occur in the top k results for each list.

For example, if the lowest link clicked by the user on I corresponds to the 5^{th} result in A and 7^{th} result in B, then k = 5 and we'll look at the number of clicked results present in the top 5 results of A and B. To derive a preference between A and B, we simply compare the number of clicks in the top k results of A and B, i.e., if more results from A are clicked, then A is preferred.

In the example lists given in part 3 above, which system, A or B, is preferred if the user clicks on the 1^{st} and 4^{th} link in the output I? Explain your answer.

5. [3pts] Now, consider a bot which clicks on the results presented by the search engine uniformly at random, i.e., each link is equally likely to be clicked by the bot. Write an example of input lists A, B so that such a random clicker generates a strict preference for one of the lists in expectation. Is this a problem? Why or why not?

Solution

1.

Method	Advantage	Disadvantage
Click feedback	easily available at scale	very noisy
Expert judgement	much less noisy	hard/expensive to obtain at scale

2. The argument here is same as that for pairwise vs pointwise learning, i.e., the judgements are better if two things are compared to each other.

3. $I : (a; b; e; c; d; f; g; h)$

4. A is preferred as 2 links a, c are clicked in A whereas only one link a is clicked in B.

5. Consider A:(a;b;c;d) and B:(b;c;d;a). If A wins the toss, I:(a;b;c;d), else if B wins the toss I:(b;c;d;a). In either case, clicking on a generates a preference for A while clicking on anything else generates a preference for B. Therefore, if a bot clicks unformly at random on the links, we will get a strict preference for B, which is undesirable.

8 Vector Space (8 pts)

Consider the following retrieval scheme:
Eg. for a query ***rising interest rates***

- *Run the query as a phrase query*
- *If* $< K$ *docs contain the phrase* **rising interest rates**, *run two phrase queries* **rising interest** *and* **interest rates**
- *If we still have* $< K$ *docs, run the vector space query* **rising interest rates**. *Rank matching docs by vector space scoring*

Consider the following set of documents:

D1 : Do you like green eggs and ham

D2 : I do not like them Sam I am

D3 : I do not like green eggs and ham

D4 : I do not like eggs

D5 : Why are they green

Let the query be ***green eggs*** and suppose we eventually want to show the user 4 documents. That is, $K = 4$.

1. [2pts] Run the phrase query ***green eggs***. What are the documents that get retrieved in this step ?

2. [2pts] What are the scores of the documents you retrieved in part (1) for the phrase query using cosine similarity on a bigram vector space ?

3. [4pts] If you did not have K documents in the first step, continue scoring according to the technique. What are the final 4 documents that you return and their scores) ?

Solution

1. D1 and D3
2. Score(D1) = 1 / $\sqrt{6}$
   Score(D3) = 1 / $\sqrt{7}$
3. Score(D4) = $1/\sqrt{2}$ * $1/\sqrt{8}$ + $1/\sqrt{2}$ * $1/\sqrt{8}$ = 0.5
   Score(D5) = $1/\sqrt{2}$ * $1/\sqrt{4}$ + $1/\sqrt{2}$ * $1/\sqrt{4}$ = $1/\sqrt{2}$

9 Ranking (8 pts)

BM25 is a measure of how relevant a document is for a query. Below is the formula for Okapi BM25:

$$RSV^{BM25} = \sum_{i \in q} (\log \frac{N}{df_i}) \frac{(k_1+1)tf_i}{k_1((1-b)+b\frac{dl}{avdl})+tf_i}$$

where q is the query.

1. In this part of the problem, we will consider how BM25 performs on the query "employee benefits". Below are some statistics for our corpus.

Number of webpages	1000
Number of webpages containing "employee"	10
Number of webpages containing "benefits"	100
Average document length	78

There are only two documents relevant to "employee benefits". Below are their statistics.

	benefits.pdf	welcome.pdf
Document Length	26	39
Frequency of "employee"	3	4
Frequency of "benefits"	3	2

Let $k_1 = 1$ and $b = 0.6$. Calculate the RSV^{BM25} for each document for the query "employee benefits".

(a) [1pt] $RSV_{benefits.pdf}$:

(b) [1pt] $RSV_{welcome.pdf}$:

2. Suppose our query has only one term, briefly answer in 1 or 2 sentences for each of the following questions on the effects of varying different parameters to the RSV^{BM25}. Assuming all other parameters are fixed:

(a) [2pts] What is the effect on RSV^{BM25} of varying b from 0 to 1?

(b) [2pts] What is the effect on RSV^{BM25} when tf_i goes to infinity?

(c) [2pts] How would you manipulate the parameters k_1 and b so that the RSV^{BM25} approximates TF-IDF? Justify your answer.

Solution

(a) 1. benefits.pdf 2. welcome.pdf

Using the BM25 formula, we add the BM25 scores for "employee" and "benefits".

benefits.pdf:

$$
\begin{aligned}
&RSV^{BM25} \\
&= \log \frac{1000}{10} \times \frac{(1+1)\times 3}{1\times(1-0.6+0.6\times\frac{26}{78})+3} + \log \frac{1000}{100} \times \frac{(1+1)\times 3}{1\times(1-0.6+0.6\times\frac{26}{78})+3} \\
&= 5
\end{aligned}
$$

welcome.pdf:

$$
\begin{aligned}
&RSV^{BM25} \\
&= \log \frac{1000}{10} \times \frac{(1+1)\times 4}{1\times(1-0.6+0.6\times\frac{39}{78})+4} + \log \frac{1000}{100} \times \frac{(1+1)\times 2}{1\times(1-0.6+0.6\times\frac{39}{78})+2} \\
&= 4.886
\end{aligned}
$$

Ranking:

1. benefits.pdf
2. welcome.pdf

(b) When b=0, the document length is not taken into account, and as b increases the relative document length matters more.

(The higher the document length is, the more term frequency is discounted.)

(c) As tf goes to ∞, the effect of tf is canceled and the RSV approximates a scaled version of IDF.

(d) Set $b = 0$ and make k_1 very large, ideally infinite. This removes length normalization and the RSV approximates TF-IDF.

$$\begin{aligned}\lim_{k_1 \to \infty} BM25 &= \lim_{k_1 \to \infty} \log \frac{N}{df} \times \frac{(k_1+1)tf}{(k_1+tf)} \\ &= \log \frac{N}{df} \times tf && k_1 + 1 \approx k_1 + tf \\ &= idf \times tf\end{aligned}$$

10 Index Construction (12 pts)

Consider constructing a nonpositional index. Due to hardware constraints and the nature of each document collection, several indexing algorithms have been proposed, e.g., the blocked sort-based indexing (BSBI) algorithm, the single-pass in-memory indexing (SPIMI) algorithm, and dynamic indexing. In this exam question, we will test your understanding of various indexing algorithms.

1. Briefly answer in **one sentence** each of the below questions:

(a) [1pt] What hardware constraint does BSBI address?

(b) [1pt] What is the advantage of using termIDs instead of terms in BSBI?

(c) [1pt] What limitation of BSBI does SPIMI address?

(d) [1pt] When do we need to use dynamic indexing?

2. [3pts] In the text book (IIR Figure 4.4), only part of the SPIMI algorithm, the inversion procedure for each block, is described, not the entire algorithm.

Your task is to complete the SPIMI algorithm (**in no more than 10 lines of code**), similar to the BSBI algorithm as detailed in IIR Figure 4.2 or below.

```
BSBIndexConstruction()
1  n ← 0
2  while (all documents have not been processed)
3  do n ← n + 1
4     block ← ParseNextBlock()
5     BSBI-Invert(block)
6     WriteBlockToDisk(block, f_n)
7  MergeBlocks(f_1, ..., f_n; f_merged)
```

For your convenience, you can:

- Call the function *SPIMI-Invert(token_stream)*
- Call any functions used in the BSBIndexConstruction algorithm above.
- View all documents in a collection as a single *token_stream* given by the function *Get-Collection-Stream()*.
- Query the method *Has_Next(token_stream)* which will return *False* if the end of the collection has been reached.

(continues on next page)

3. [5pts] Now, apply the SPIMI algorithm to the following collection:

d1: bsbi use term id
d2: sort term id doc id
d3: spimi use term
d4: no term id sort

Assume that main memory can only hold two documents at a time, i.e., the SPIMI algorithm will write to disk each time after two documents, a block, have been processed.

Write out the content of each block **just before merging** and the result **after merging** in the following format:

Block 1:
bsbi → 1
...
term → 1, 2

Solution

1. (a) Insufficient main memory. (The sorting step needs to be done externally, i.e. using disk, and we want to have an algorithm that minimizes the number of random disk seeks during sorting).
   (b) For space efficiency. (An integer costs less space than a string).
   (c) For very large collections, the dictionary that maps from terms to termIDs in BSBI does not fit into memory. SPIMI uses terms instead of termIDs and does not hold the vocabulary in memory.
   (d) When documents in the collection are frequently updated, e.g. the Web.

2. n = 0
   token_stream = Get-Collection-Stream()
   while Has-Next(token_stream) **do**
   n = n + 1
   f_n = SPIMI-Invert(token_stream)
   end while
   MergeBlocks(f_1, ..., f_n; f_{merged})

3. *Block 1*:
   bsbi → 1
   doc → 2
   id → 1, 2
   sort → 2
   term → 1, 2
   use → 1

   Block 2:
   id → 4

no → 4
sort → 4
spimi → 3
term → 3, 4
use → 3

Merged Block of 1 and 2:
bsbi → 1
doc → 2
id → 1, 2, 4
no → 4
sort → 2, 4
spimi → 3
term → 1, 2, 3, 4
use → 1, 3