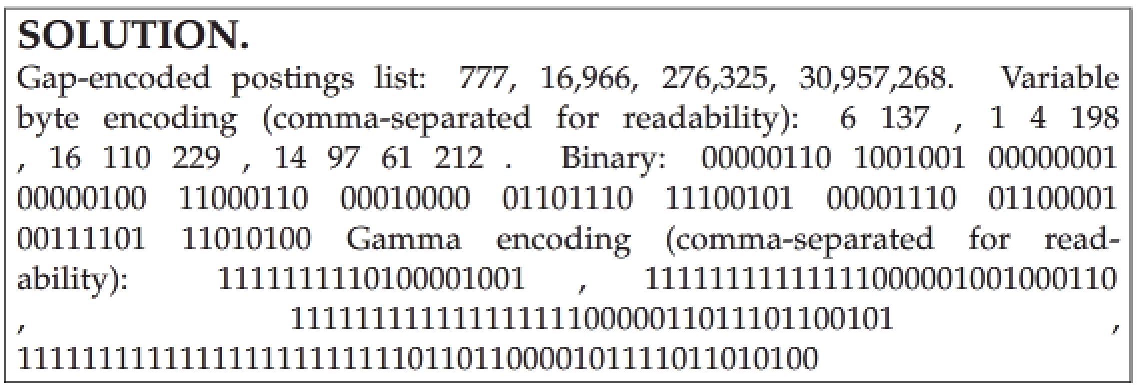
CS726 – Information Retrieval Techniques
SOLUTION

Question NO. 01:

Compute variable byte and γ codes for the postings list (777, 17743, 294068, 31251336). Use gaps instead of docIDs where possible. Write binary codes in 8-bit blocks.



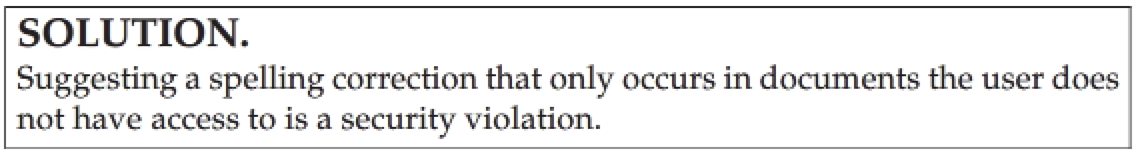
SOLUTION.
Gap-encoded postings list: 777, 16,966, 276,325, 30,957,268. Variable byte encoding (comma-separated for readability): 6 137 , 1 4 198 , 16 110 229 , 14 97 61 212 . Binary: 00000110 1001001 00000001 00000100 11000110 00010000 01101110 11100101 00001110 01100001 00111101 11010100 Gamma encoding (comma-separated for readability): 1111111110100001001 , 11111111111111000001001000110 , 1111111111111111110000011011101100101 , 11111111111111111111111110110110000101111011010100

Question NO. 02:

Can spelling correction compromises document level security? Consider a case where scheme is based on document in which user does not have excess to?

You can control access to data within an index by adding field and document level security permissions to a role. Field level security permissions restrict access to particular fields within a document. Document level security permissions restrict access to particular documents within an index.

A role can define both field and document level permissions on a per-index basis. A role that doesn't specify field level permissions grants access to ALL fields. Similarly, a role that doesn't specify document level permissions grants access to ALL documents in the index.



SOLUTION.
Suggesting a spelling correction that only occurs in documents the user does not have access to is a security violation.

Question NO. 03:

Calculate total memory usage for terms where idf, tf, DocID is given and total no of terms are 500000?

NO Solution – See 1st assignment Solution

Question NO. 04:

Wildcard word sh*pi* is given and we have to show all steps to find the word?
Answer

Consider the wildcard query sh*pi*. We are seeking documents containing any first term that begins with sh and second term with pi. Accordingly, we run the Boolean query $sh AND pi$. This is looked up in the 3-gram index and yields a list of matching terms such as relive, shopping and shipping. Each of these matching terms is then looked up in the standard inverted index to yield documents matching the query

Question NO. 05:

If you wanted to search for s*ng in a permuterm wildcard index, what key(s) would one do the lookup on?

SOLUTION. ng$s*

Question NO. 06:

From the following sequence of γ-coded gaps, reconstruct first the gap sequence and then the postings sequence: 1110001110101011111101101111011.

SOLUTION.
Gaps sequence = 9, 6, 3, 49, 7 Postings sequence = 9, 15, 18, 67, 73

Question NO. 07:

Consider the postings list (4, 10, 11, 12, 15, 62, 63, 265, 268, 270, 400) with a corresponding list of gaps h4, 6, 1, 1, 3, 47, 1, 202, 3, 2, 130i. Assume that the length of the postings list is stored separately, so the system knows when a postings list is complete. Using variable byte encoding:

(i) What is the largest gap you can encode in 1 byte?
(ii) What is the largest gap you can encode in 2 bytes?
(iii) How many bytes will the above postings list require under this encoding?

(Count only space for encoding the sequence of numbers.)

SOLUTION. (i) $127 = 2^7 - 1$ (ii) $16383 = (2^{1}4 - 1)$ (iii) 13

Question NO. 08:

A little trick is to notice that a gap cannot be of length 0 and that the stuff left to encode after shifting cannot be 0. Based on these observations:

(i) Suggest a modification to variable byte encoding that allows you to encode slightly larger gaps in the same amount of space.
(ii) What is the largest gap you can encode in 1 byte?
(iii) What is the largest gap you can encode in 2 bytes?
(iv) How many bytes will the postings list in Exercise 5.6 require under this encoding? (Count only space for encoding the sequence of numbers.)

SOLUTION. (i) Before each byte encoding decision, subtract 1 from the number still to be encoded. (The most common mistake here was to say to subtract 1 from the whole number rather than for each by byte) (ii) $128 (= 2^7)$ (iii) $16512 (= 2^{1}4 + 2^7)$ (iv) Still 13

Question NO. 09:

Estimate the space usage of the Reuters-RCV1 dictionary with blocks of size $k = 8$ and $k = 16$ in blocked dictionary storage.

SOLUTION. Space usage of the Reuters Dictionary: For k=8, reduction in space = (400000/8) *13=0.65 MB Space used=10.8-0.65=10.15 MB For k=16, reduction in space = (400000/16) *29=0.725 MB Space used=10.8-0.725=10.1 MB

Question NO. 10:

Unary code is not a universal code in the sense defined above. However, there exists a distribution over gaps for which unary code is optimal. Which distribution is this?

SOLUTION. Unary code is optimal for $P(n) = 2^{-n}$ over $\mathbb{N}$.

Question NO. 11:

Give some examples of terms that violate the assumption that gaps all have the same size (which we made when estimating the space requirements of a γ-encoded index). What are general characteristics of these terms?

SOLUTION.
For a news collection like Reuters RCV-1, examples of such terms could be: tennis, football, music etc. Since the news collection contains news collected over a period of time, words like tennis and football would occur a lot in news documents collected during their sports seasons and very rarely in documents collected at other times. Although general characteristics of these terms depend on the type of collection we are dealing with and so its difficult to formulate a common property, these terms could be the ones which are rare in the collection

Question NO. 12:

Consider a term whose postings list has size n, say, n = 10,000. Compare the size of the γ-compressed gap-encoded postings list if the distribution of the term is uniform (i.e., all gaps have the same size) versus its size when the distribution is not uniform. Which compressed postings list is smaller?

SOLUTION.
encoding the uniform list 13 13 we need 14 bits
encoding the non-uniform list 2 24 we need 12 bits
in this case non-uniform is shorter because small gaps are encoded efficiently
encoding the uniform list 1000 1000 we need ?? bits
encoding the non-uniform list 200 1800 we need ?? bits
(should be longer)

Question NO. 13:

(i) Show that the size of the vocabulary is finite according to Zipf's law and infinite according to Heaps' law.
(ii) Can we derive Heaps' law from Zipf's law?

SOLUTION.
(i) According to Heaps Law, the vocabulary size M keeps growing with the number of tokens and hence it is infinite. Vocabulary size is a fixed constant in Zipf's law. The law assumes that the vocabulary is fixed and finite.
(ii) No, Zipf's law states that the vocabulary is finite. Thus, it would follow that Heaps' law does not hold.
Note that our definition of Zipf's law states that the exponent is -1. Sometimes the definition is relaxed to include exponents $x < -1$. In that case, the vocabulary is infinite.

Question NO. 14:

If we were to stem jealous and jealousy to a common stem before setting up the vector space, detail how the definitions of tf and idf should be modified.

SOLUTION. If jealousy and jealous are stemmed to a common term t, then their tf's and their df's would be added together, in computing the tf-idf weights.

Question NO. 15:

Compute the vector space similarity between the query "digital cameras" and the document "digital cameras and video cameras" by filling out the empty columns in Table. Assume *N* = 10,000,000, logarithmic term weighting (wf columns) for query and document, idf weighting for the query only and cosine normalization for the document only. Treat and as a stop word. Enter term counts in the tf columns. What is the final similarity score?

SOLUTION.

	query					document			
word	tf	wf	df	idf	q_i = wf-idf	tf	wf	d_i = normalized wf	$q_i \cdot d_i$
digital	1	1	10,000	3	3	1	1	0.52	1.56
video	0	0	100,000	2	0	1	1	0.52	0
cameras	1	1	50,000	2.3	2.3	2	1.3	0.68	1.56

Similarity score: $1.56 + 1.56 = 3.12$.
Normalized similarity score is also correct: 3.12/length(query) = 3.12/3.78 = 0.825

Question NO. 16:

Assume that machines in MapReduce have 100 GB of disk space each. Assume further that the postings list of the term the has a size of 200 GB. Then the MapReduce algorithm as described cannot be run to construct the index. How would you modify MapReduce so that it can handle this case?

SOLUTION. partition by docid as well as term for very frequent terms

Question NO. 17:

What is the idf of a term that occurs in every document? Compare this with the use of stop word lists.

SOLUTION.
It is 0. For a word that occurs in every document, putting it on the stop list has the same effect as idf weighting: the word is ignored.

Question NO. 18:

Can the tf-idf weight of a term in a document exceed 1? Explain

ANSWER:
YES, the tf-idf weight of a term in a document exceed 1

TF-IDF is a family of measures for scoring a term with respect to a document (relevance). The simplest form of TF (word, document) is the number of times word appears in document. TFIDF can be 1 in the naive case, or to add the IDF effect, just do it log(number of documents/number of documents in which word is present).

Question NO. 19:

For the queries below, can we still run through the intersection in time $O(x+y)$, where x and y are the lengths of the postings lists for Brutus and Caesar? If not, what can we achieve?

a. Brutus AND NOT Caesar

b. Brutus OR NOT Caesar

SOLUTION. a. Time is O(x+y). Instead of collecting documents that occur in both postings lists, collect those that occur in the first one and not in the second. b. Time is O(N) (where N is the total number of documents in the collection) assuming we need to return a complete list of all documents satisfying the query. This is because the length of the results list is only bounded by N, not by the length of the postings lists.

Question NO. 20:

For a conjunctive query, is processing postings lists in order of size guaranteed to be optimal? Explain why it is, or give an example where it isn't.

SOLUTION. The order is not guaranteed to be optimal. Consider three terms with postings list sizes s1 = 100, s2 = 105 and s3 = 110. Suppose the intersection of s1 and s2 has length 100 and the intersection of s1 and s3 length 0. The ordering s1, s2, s3 requires 100+105+100+110=315 steps through the postings lists. The ordering s1,s3,s2 requires 100+110+0+0=210 steps through the postings lists.

Question NO. 21:

The following pairs of words are stemmed to the same form by the Porter stemmer. Which pairs would you argue shouldn't be conflated. Give your reasoning.

a. abandon/abandonment
b. absorbency/absorbent
c. marketing/markets
d. university/universe
e. volume/volumes

SOLUTION. a. conflate: very similar semantics b. conflate: very similar semantics c. don't conflate: marketing (subject taught at business schools) is too different from market (e.g., stock market) d. don't conflate: university (higher learning) too different from universe (the world) e. conflate: same semantics in most contexts (exception: degree of loudness)

Question NO. 22:

How could an IR system combine use of a positional index and use of stop words? What is the potential problem, and how could it be handled?

SOLUTION.
Is the problem referred to in this question is the problem faced in constructing the positional index after removal of stop words as this preprocessing changes the positions of terms in the original text? As far as the first part of the question is concerned, can you give a hint of what kind of use is the question looking for? I am assuming the answer of the question is not the following: Phrasal queries can handled using both of them. For any query, remove the stop-words and merge the positional indexes of the remaining terms looking for exact phrasal match by determining relative positions.

Question NO. 23:

Give an example of a sentence that falsely matches the wildcard query mon*h if the search were to simply use a conjunction of bigrams.

SOLUTION. His personality is *moonish*.

Question NO. 24:

If $|S|$ denotes the length of string S, show that the edit distance between s_1 and s_2 is never more than $\max\{|s_1|, |s_2|\}$.

SOLUTION.
Consider two character strings s1 and s2. Without any loss in generality, let us assume that $l1 = length(s1) < l2 = length(s2)$. For transforming $s1$ into $s2$,we can replace each character of $s1$ by the first $l1$ characters of $s2$ and insert the remaining characters of $s2$ at the end of $s1$.The number of operations incurred is $l2 = \max(l1, l2)$.

Question NO. 25:

Consider the four-term query catched in the rye and suppose that each of the query terms has five alternative terms suggested by isolated-term correction. How many possible corrected phrases must we consider if we do not trim the space of corrected phrases, but instead try all six variants for each of the terms?

SOLUTION. 6*6*6*6 = 1296

Question NO. 26:

For each of the prefixes of the query — catched, catched in and catched in the — we have a number of substitute prefixes arising from each term and its alternatives. Suppose that we were to retain only the top 10 of these substitute prefixes, as measured by its number of occurrences in the collection. We eliminate the rest from consideration for extension to longer prefixes: thus, if batched in is not one of the 10 most common 2-term queries in the collection, we do not consider any extension of batched in as possibly leading to a correction of catched in the rye. How many of the possible substitute prefixes are we eliminating at each phase?

SOLUTION. At *catched in*, we choose 10 out of 36 possible thus eliminating 26. At *catched in the*, out of 60 surviving alternatives, we eliminate 50 of these, and at catched in the rye, we eliminate 50 of the surviving alternatives.