MODEL SOLUTIONS

SETTER: Lucia Specia / Mark Hepple

Data Provided: None

DEPARTMENT OF COMPUTER SCIENCE **Autumn Semester 2012-2013**

TEXT PROCESSING **2 hours**

Answer THREE questions.

All questions carry equal weight. Figures in square brackets indicate the percentage of available marks allocated to each part of a question.

1. In the context of Information Retrieval, given the following two documents:

   Document 1: Sea shell, buy my sea shell!

   Document 2: You can buy lovely SEA SHELL at the sea produce market.

   and the query:

   Query 1: buy lovely sea shell

   a) Explain three types of manipulations (except term weighting) that can be done on document terms before indexing them. What are the advantages and disadvantages (if any) of each of these manipulations? [20%]

ANSWER:

Term manipulations can include any of the following (answer only need 3): capitalisation, stemming, stop-word removal, indexing multi-terms, normalisation. Three exemplar definitions:

A *stoplist* is a list of words ('stop-words') that are ignored when documents are indexed, and likewise discounted from queries. These are words that are so widespread in the document collection that they are of v.little use for discriminating between documents that are/are not relevant to a query. Their exclusion eliminates a large number of term occurrences that would need to be recorded, thereby reducing the size of indexes and saving computational effort during both indexing and retrieval.

Stemming refers to the process of reducing words that are morphological variants to their common root or stem, e.g. so that variants *computer, computes, computed, computing*, etc. are reduced to a stem such as *compute*. For IR, stemming is applied to documents before indexing and to queries. The effect of this is that when a query contains a term such as *computing*, retrieval can potentially return documents on the basis of their containing any of the morphological variants of the same root. This is the intended key benefit of using stemming, although its usefulness is debated. Stemming will also produce some reduction in the size of document indexes.

Capitalisation refers to the process of normalising the case of words so that a single case is used, for example, all words are lowercased (e.g. SHELL = shell). This procedure makes indexing and retrieval more efficient by decreasing the number of terms that have to be represented. It also makes the term weighting more reliable, as higher frequency counts will be observed when putting together different variants of the term. One possible disadvantage is that intended capitalisation in terms for disambiguation purposes will be disregarded, e.g. Turkey (country) vs turkey (bird).

b) Applying stop word removal and capitalisation, show how Document 1 and Document 2 would be represented using an *inverted index*. Provide the stoplist used. [10%]

ANSWER:

Based on *lowercasing* and *stopword removal* assuming a stoplist that includes the following terms: {`at, can, my, the, you, !, ,, .`}, we get the following inverted index:

term-id	word	docs
1	buy	d1:1, d2:1
2	lovely	d2:1
3	market	d2:1
4	produce	d2:1
5	sea	d1:2, d2:2
6	shell	d1:2, d2:1

c) Assuming *term frequency* (TF) is used to weight terms, compute the similarity between each of the two documents (Document 1 and Document 2) and Query 1. Compute this similarity using two alternative metrics (e.g. cosine, Euclidean, dot product). Determine the ranking of the two documents according to each of these metrics and discuss any differences in the results. [25%]

ANSWER:

Using the term order taken from the inverted index above, we can represent the two documents and the query as vectors as follows:

d1: $\langle 1, 0, 0, 0, 2, 2 \rangle$
d2: $\langle 1, 1, 1, 1, 2, 1 \rangle$
q: $\langle 1, 1, 0, 0, 1, 1 \rangle$

Ranking using cosine similarity: The cosine between two vectors $\vec{x}$ and $\vec{y}$ is computed as:

$$cos(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{|\vec{x}||\vec{y}|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}}$$

The vector magnitudes and cosine values are then:

$$|\mathrm{d1}| = \sqrt{1^2 + 0^2 + 0^2 + 0^2 + 2^2 + 2^2} = \sqrt{9} = 3$$

$$|\mathrm{d2}| = \sqrt{1^2 + 1^2 + 1^2 + 1^2 + 2^2 + 1^2} = \sqrt{9} = 3$$

$$|\mathrm{q}| = \sqrt{1^2 + 1^2 + 0^2 + 0^2 + 1^2 + 1^2} = \sqrt{4} = 2$$

$$cos(\mathrm{d1}, \mathrm{q}) = \frac{1.1 + 0.1 + 0.0 + 0.0 + 2.1 + 2.1}{3.2} = 5/6$$

$$cos(\mathrm{d2}, \mathrm{q}) = \frac{1.1 + 1.1 + 1.0 + 1.0 + 2.1 + 1.1}{3.2} = 5/6$$

Thus, the cosine values computed for the two documents are the same, and so the two are equally rated as relevant to the query.

Ranking using euclidean distance: Euclidean distance is computed as $\sqrt{\sum_{i=1}^{n}(q_i - d_i)^2}$. In this case:

$$dis(\mathrm{d1}, \mathrm{q}) = \sqrt{(1-1)^2 + (0-1)^2 + (0-0)^2 + (0-0)^2 + (2-1)^2 + (2-1)^2} = \sqrt{3}$$

$$dis(\mathrm{d2}, \mathrm{q}) = \sqrt{(1-1)^2 + (1-1)^2 + (1-0)^2 + (1-0)^2 + (2-1)^2 + (1-1)^2} = \sqrt{3}$$

The difference between the two metrics: while in this example the two metrics behave in the same way and they are not able to distinguish the two documents, in general cosine is a better metric for IR because it normalises the differences between the query and document vectors by the length of such vectors. This is important because documents can vary in size and a document should not be considered more relevant simply because it is closer in length to the query. Essentially the cosine metric computes the angle between two vectors, and therefore the length of such vectors is less relevant.

d) Discuss the expected effect of using TF.IDF to weight the terms in Document 1 and Document 2: would this be a better term weighting scheme in this example? Include the formula (or formulae) for computing TF.IDF values as part of your answer.

[25%]

ANSWER:

TF.IDF assigns weights to terms by taking into account two elements: the frequency of that term in a particular document (TF) and the proportion of documents in the corpus in which it occurs. The IDF for a term w is computed as follows, where D is the set of documents in the document collection and df_w is the document frequency of w (the count of documents in which w appears):

$$idf_{w,D} = log\frac{|D|}{df_w}$$

The TF.IDF value for a given term (w) in a given document (d) is:

$$tf.idf_{w,d,D} = tf_{w,d} \cdot idf_{w,D}$$

In this example, using TF.IDF would set the weight of three of the query terms to 0: *buy, sea, shell*, since they appear in all (two) documents in the collection and thus $log\frac{|D|}{df_w} = log\frac{2}{2} = log(1) = 0$. The only query term that receives a weight different from 0 is *lovely*, which only happens in Document 1, and therefore this document is ranked first. TF.IDF has a positive effect in this example and in general, since it disregards terms that are frequent across the whole collection of indexed documents.

e) Explain the intuition behind the PageRank algorithm. Discuss how it can distinguish two or more documents that are ranked equally "relevant" according to the similarity score given by the vector space model. [20%]

ANSWER:

The PageRank algorithm includes an additional component in the ranking of documents to the score of a document *d* with respect to a query *q*: the popularity of the query. It has thus two components: DocumentScore (q, d) = f(IRScore(q, d), PageRankScore (d)):

- **IR** score: a measure of how well the content of the query matches the content of the document (vector space model)
- **PageRank** score: a measure of how *popular* the page is. A link from page P_i to page P_j confers authority on P_j. The popularity of a page P_j is a recursive concept given by the popularity of other pages representing incoming and outgoing links to page P_j.

2. a) Explain what is meant by an *interlingual* approach to Machine Translation. What is the key advantage of using an interlingual approach for translation amongst multiple languages, as compared to alternative rule-based approaches? Identify one problem that arises in the creation of a genuinely interlingual representation. [20%]

ANSWER:

Interlingual MT approaches use an intermediate representation in the translation process that is an *interlingua*, which is a meaning representation language whose characteristics are independent of both the source and target languages of translation. A source language sentence goes through syntactic and semantic analysis to be translated into the interlingua. This representation is then used to generate a sentence in the target language.

This feature allows full translation coverage amongst N different languages to be achieved by providing only N analysis/generation components, i.e. one component to analyse sentences of each language into the interlingua and one component to generate sentences of the language from the interlingua. Then we can translate from any one of the N languages to any other by chaining the appropriate analysis/generation components, mapping content via the interlingua.

Other MT approaches require a specific translation system for each language pair *and direction*, giving rise to the $O(N^2)$ problem, i.e. that to achieve translation amongst N languages, we need $N^2 - N$ translation systems.

A key obstacle to formulating a genuine interlingua, i.e. one capable of representing meaning for all languages, is that the lexical concepts of different languages carve up conceptual space in different ways. Furthermore, the conceptual inconsistencies between languages suggests the need for complex semantic processing for decomposing/reassembling conceptual units.

b) Describe three main models (or features) of a standard *phrase-based* approach to statistical machine translation. Explain how these are combined. Explain how they are applied to translate a new sentence. [40%]

ANSWER:

The two main models of a PBSMT system are $P(F|E)$ (translation model) and $P(E)$ (language model): $P(F|E)$ will ensure that a good E will have words that generally translate to words in F. $P(E)$ will help save us from bad English and $P(F|E)$ only needs to say whether a bag of English words corresponds to a bag of French words. Other models include a reordering model, which accounts for the fact that the order of words or phrases in the target language might be different from that of the source language, a penalty on the number of words and/or phrases to bias translations that are closer in length to the source and that use fewer (but longer) phrases. These as well as the other components are computed as independent functions h and combined using a log-linear model of the type $\sum_{i=1}^{m} \lambda_i h_i$, where λ are the weights of each models. At translation time, a *decoder* segments the new source sentence in different possible ways and builds a search graph with possible translation options from the phrase dictionary. The decoder selects the translation that covers all source words and maximises $\sum_{i=1}^{m} \lambda_i h_i$ for that sentence.

c) Discuss two alternative approaches to BLEU that may be used for evaluating Machine Translation systems, indicating their advantages and disadvantages. [20%]

ANSWER:

Various approaches are possible for both manual and automatic evaluation - some possible answers (only 2 are necessary).

1) One way is to rely on human judges, who compare system outputs to manually-created reference translations, and attempt to assess the rate of incorrect sentences produced by the system, producing a 'subjective sentence error rate' score.

2) Alternatively, subjects might read the system outputs and then complete a comprehension test, to measure the effectiveness of the translations in communicating the content of the source document.

These methods have the advantage of being more reliable, since they involve humans. However, such human-based approaches are very costly of time/money, and can only occasionally be repeated (if at all), i.e. cannot frequently be redone to aid the system development process.

3) Another possibility is to use the MT systems being evaluated as a component in some larger system, e.g. for cross-language IR or QA. Whatever means is available for evaluating/benchmarking the performance of that larger system can then be applied to determine whether the system works better or worse with one MT system than another, and hence, by implication, which of the MT systems is more effective. This is, however, a rather indirect means of evaluating the how good the MT system is at producing translations, so that its results have unclear status, and the approach will also carry with it whatever costs are associated with evaluation of the larger system.

4) The back-translation or round-trip translation evaluation approach translates the original text, written in L1, into another language L2 and then back into L1. The quality of the translation is evaluated by checking how close the text produced is to the original text. The advantages are that this approach is that it does not require knowledge of another language (L2) or significant resources (such as reference translations). The disadvantages are that two Machine Translation systems are involved (L1 to L2) and (L2 to L1). The approach is cheap but it cannot distinguish between them and cannot tell which MT system introduced any errors. In addition, one MT system may fix errors made by the other but this will not be detected.

d) Given the two scenarios:

Scenario 1: English-Chinese language pair, 300,000 examples of translations.

Scenario 2: Portuguese-Spanish, 50,000 examples of translations.

In which of these scenarios would statistical machine translation work better and why? Would a rule-based transfer approach be better in any of these scenarios?

[20%]

ANSWER:

SMT is best suited to languages which are structurally similar (e.g. English-French, Spanish-Portuguese) and least suited to those which are structurally different (e.g. English-Chinese). Structural differences can make translation difficult for a number of reasons: (1) different notions of what a word is (so the alignment between words is not one to one), (2) different word orders (leading to a huge number of possible alignments which would be impossible to compute and leave room for noisy alignments), (3) different morphologies (making it difficult to learn translation probabilities).

It is more likely that SMT would work better for Scenario 2, even with significantly fewer examples, given that the language pair is structurally very similar, and also uses many similar (sometimes identical) words. The relatively small number of examples in this case would probably lead to untranslated words in the target language, that is, Portuguese words in the Spanish translation. Because of the lexical overlap between these two languages and the large number of cognates (similar words with similar meanings), translations should still be understandable. For Chinese-English, a rule-based approach with manually designed rules allowing for long-distance reorderings could be more precise, however it would still be very costly.

3. a) What is text summarisation? Provide three criteria according to which different types of summary have been classified and explain each of these criteria. [30%]

ANSWER:

In its most general sense, summarisation is the art of abstracting key content from one or more information sources. This has most commonly been done for textual sources (e.g. newpaper articles, journal papers), but this is not necessarily so, for example, a full-length video of a football match might be summarised by a short video of edited highlights (video>video).

Standard ways in which summaries have been distinguished are as follows. **3** of these are required in the answer.

1. abstract vs. extract. Abstract summary is a new document while an extractive summary is a set of sentences from the original document. An abstractive summary system will be required to generate text while the goal of an extractive system is to identify the important sentences in a document.
2. indicative vs. informative vs. evaluative (critical) summaries. Indicative summaries provide an indication of the content of the document and help users to decide whether to read the full document or not. Informative summaries cover all key information in the document and act as a convenient replacement for the full document that conveys the important information quickly. Critical summaries evaluate the content of the document.
3. single vs. multi-document summaries. A single document summarisation system requires only one document as input while a multi-document summariser will make use of multiple documents. Single document summarisation decides which information to include by examining only one document while multi-document may consider whether a piece of information is repeated several times in a set of documents.
4. generic vs. query-based vs. user-profile-based summaries. A generic summary would be generated for any document while an automatic summariser would generate different query-based summaries and user-profile-based summaries. Query-based summaries are tailored to an individual query (such as the snippets returned by Google) while user-profile-based summaries make use of some pre-defined model of the user to tailor the summary to their needs.
5. summary for expert vs. novice users. Summaries for novice users should avoid the use of complex terminology and generate summaries that do not require background knowledge to be understood, although this can be assumed to be known by expert users.

b) Discuss three characteristics of a good textual summary. Provide two examples of *extrinsic* evaluation strategies for summaries. [20%]

ANSWER:

Properties of a good text summary include the following. It should be considerably shorter than the input text, whilst covering the main points of the original (or, for non-generic summaries, cover the main points relevant to the user's needs, etc). It should be *truth-preserving*, i.e. should not carry false implications. It should be a good text in its own right, i.e. read well/be coherent.

Examples of tasks used in extrinsic evaluations include the subject's ability to answer questions about the topic of the original text, or to assign a topic category to the original text, when given the automatic summary, as opposed to using the original text or a human summary.

c) Discuss the differences between *deep* and *shallow* approaches to automatic text summarisation. For each approach, give one example of an application where the approach would work well. [20%]

ANSWER:

Potential discussion points include the following:

Deep approaches:

- are knowledge-rich
- involve a significant extent of 'semantic analysis'
- employ a sizeable knowledge-base of rules that are costly to create/maintain
- are domain dependent
- are difficult/costly to adapt to new domains
- produce abstracts

By contrast, **shallow approaches**:

- use an analysis based primarily on words and other shallow document features
- no significant 'semantic analysis'
- are knowledge-poor (or '-light')
- have little domain dependence and so are easily ported to new domains
- produce extraction-based summaries

One example of application for deep summarisation is creating short versions of news stories that are to be published in the front page of a newspaper. One example of application for shallow summarisation is creating snippets of documents to show as a list of ranked documents in IR.

d) A 'lead-based summary' is one consisting simply of the first N sentences of a document. Discuss whether this is a reasonable approach to summarisation and, if so, whether it is a useful approach either in general or in some specific context. [10%]

ANSWER:

Whether or not this approach is a reasonable one will largely depend on the type of text which is being summarised. The approach is not useful in general but could be useful for types of text written in a style that places the key information at the start of the document, for example newswire stories.

e) Mention and explain two criteria that have been used by *shallow* approaches to automated summarisation for identifying the most significant sentences of a document. Discuss how the indications from several such criteria can be combined together for sentence selection. [20%]

ANSWER:

Criteria used include (2 are necessary):

- **keywords**:
  - assume frequent content words indicate topic of document
  - identify set of keywords by (e.g.) TF.IDF criterion
  - score sentences w.r.t. occurrence of keywords within them
- **title method**:
  - assume title/subheadings indicative of important content
  - use words from them to identify useful sentences
- **position**:
  - use position in document as indicator of importance of text segment
  - notable positions include: (i) being close to start of text, (ii) in certain sections (e.g. intro/conclusion), (iii) first sentence of paragraph, etc.
- **cue method**:
  - certain phrases may signal important content (e.g. "this paper presents ...")
  - may distinguish both positive and negative indicators, as 'bonus' and 'stigma' words/phrases

Where several such methods are used together, where each assigns a *numeric score*, their scores might be combined as a linear weighted sum, following Edmundson (1969), i.e. using weights $(\alpha, \beta, \ldots)$ to give a combined score, as (e.g.) in:

$$Score(S) = \alpha.Title(S) + \beta.Cue(S) + \gamma.Keyword(S) + \delta.Position(S)$$

These combined score values can be used to rank sentences, with the top-ranked sentences being selected (according to some compression requirement). The weight values in the equation can be determined using training data and a suitable minimization technique.

Other approaches are possible, e.g. Kupiec *et al* (1995) use a modified Naive Bayesian approach to combine multiple criteria, all of which are discrete valued (e.g. a binary sentence length feature (true if sentence longer than a certain threshold value), a key phrase feature (true if any of a list of phrases appears in sentence), etc).

4. a) Sketch the algorithm for Huffman coding, i.e. for generating variable-length codes for a set of symbols, such as the letters of an alphabet. What does it mean to say that the codes produced are *prefix-free*, and why do they have this property? [30%]

ANSWER:

- To begin, we need to know the relative probabilities of all the symbols in the symbol set.
- We start the algorithm by creating a leaf node for each symbol, which is labelled with its associated probability.
- Then, iterate over this collection of nodes, until only one node remains, as follows:
  - with each iteration, select and remove two nodes, which are joined under a new parent node
  - crucially, the nodes selected must have the lowest associated probabilities (or strictly, there must be no other nodes that have lower probabilities)
  - the new parent node is labelled with a probability value that is the sum of probabilities of the two child nodes
  - the new node is added back into the collection of nodes
- The single node that remains represents a binary tree whose root should be labelled with probability 1.0, and whose leaf nodes store the full set of symbols.
- Finally, a code-tree is produced by labelling the left and right side of each branch within the tree with 0 or 1, respectively. Then, the path from the root to the symbol at any leaf node gives a unique sequence of 0s and 1s, which is the code for this symbol.

That codes are *prefix-free* means that no symbol has a code which is a prefix of any other code, e.g. as would hold if there were codes `110` and `1101`. This property follows from the fact that symbols only appear on the leaf nodes of code trees, as a consequence of the algorithm. There could only be a code that was the prefix of another if its symbol was labelled on a non-leaf node, i.e. so that the path from the root to this symbol's node then *continued on* to the node for the second symbol.

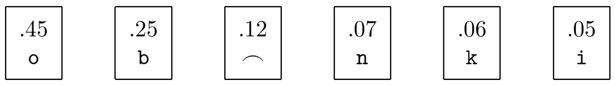
b) We want to compress a large corpus of text of the (fictitious) language *Bonobo*. The writing script of Bonobo uses only the letters {`b, i, k, n, o`} and the symbol $\frown$ (which is used as a 'space' between words). Corpus analysis shows that the probabilities of these six characters are as follows:

Symbol	Probability
`b`	0.25
`i`	0.05
`k`	0.06
`n`	0.07
`o`	0.45
$\frown$	0.12

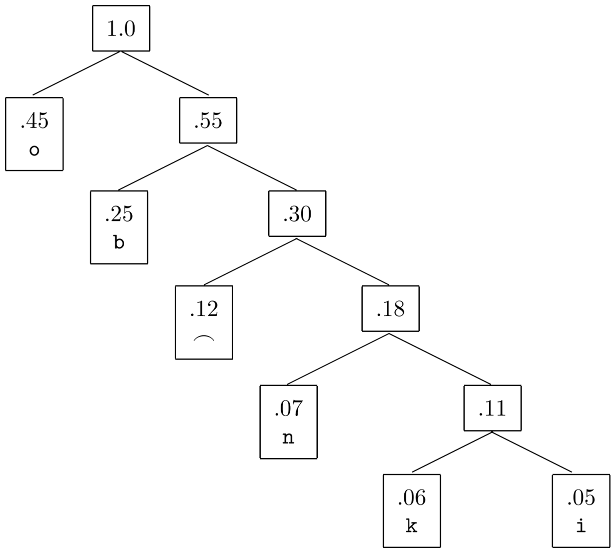
Apply the method you described in part (a) to create a Huffman code for the Bonobo character set. [30%]

ANSWER:

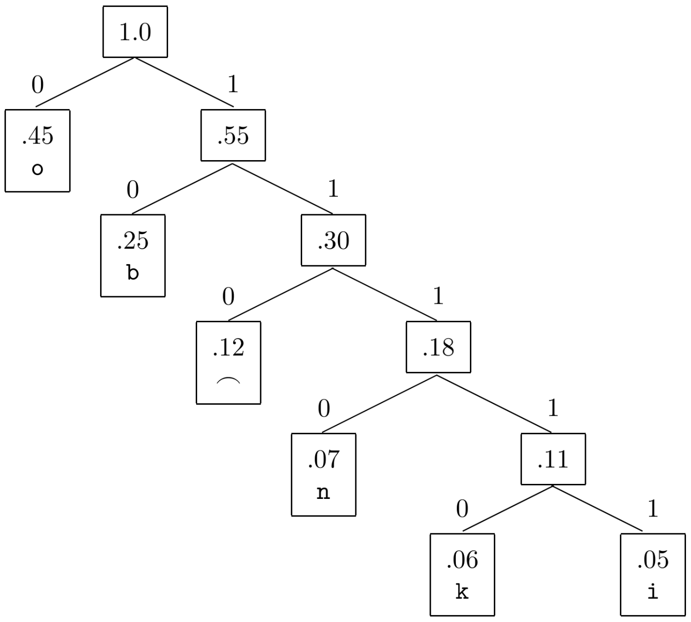
First create initial nodes as follows:



Running the algorithm produces tree:



and assigning 0s and 1s to branches gives:



This gives the following codes for the symbols:

Symbol	Probability	Code	Code length
o	0.45	0	1
b	0.25	10	2
⌢	0.12	110	3
n	0.07	1110	4
k	0.06	11110	5
i	0.05	11111	5

c) Given the code you have generated in part (b), what is the average bits-per-character rate that you could expect to achieve, if the code was used to compress a large corpus of Bonobo text? How does this compare to a minimal fixed length binary encoding of this character set? [20%]

ANSWER:

we assume that the distribution of characters in the large corpus is the same as the one that we have been given from previous corpus analysis. Then, can calculate the average bits-per-character rate summing across symbols for the probability of that symbol times by its code length:

Symbol	Probability	Code	Code length	$p \times len$
o	0.45	0	1	0.45
b	0.25	10	2	0.50
$\frown$	0.12	110	3	0.36
n	0.07	1110	4	0.28
k	0.06	11110	5	0.30
i	0.05	11111	5	0.25
				2.14

giving a cost of 2.14 bits-per-character. A minimal fixed length binary encoding would requre 3 bits per character, i.e. this is sufficient to encode up to $2^3 = 8$ characters, whilst a 2 bit code would allow only $2^2 = 4$ characters, and we have 6 to encode. Using a 3 bit fixed-length code would result in corpus storage requiring about 40% more space than the Huffman codes.

d) Use your code for Bonobo to encode the following two messages, and compute for each message the average bits-per-character rate that results:

bonobo$\frown$okobo

iniko$\frown$nikoni

Discuss why the two bits-per-character rates achieved differ, comparing them also to the expected rate that you computed in part (c). [20%]

ANSWER:

Encoding the first message we get:

```
b  o n    o b  o -   o k     o b  o
10 0 1110 0 10 0 110 0 11110 0 10 0 = 24 bits
```

as the original message has 12 chars, this give $24/12 = 2.0$ bits per character.

For the second message we get:

```
i     n    i     k     o -   n    i     k     o n    i
11111 1110 11111 11110 0 110 1110 11111 11110 0 1110 11111 = 47 bits
```

Again the message has 12 chars, giving $47/12 = 3.92$ bits per character.

The two bpc rates differ considerably. The first is slightly better than the expected rate for a large corpus, whilst the second is considerably worse — worse even than a fixed length encoding. This is due to the differential extents to which the distribution of chars within the two messages diverge from the distribution used when computing the Huffman codes. For the first message the distribution is close to the corpus distribution: high frequency/probability letters predominate, so many short codes are assigned. The distribution in the second message runs contrary to the corpus distribution, with low frequency letters predominating, so that many long codes are assigned.

END OF QUESTION PAPER