MODEL SOLUTIONS

SETTER: Lucia Specia / Mark Hepple

Data Provided: None

DEPARTMENT OF COMPUTER SCIENCE   Autumn Semester 2015-2016

TEXT PROCESSING   **2 hours 30 minutes**

Answer the question in Section A, and THREE questions from Section B.

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

SECTION A

1. a) Explain what is meant by the *bag of words* model in text processing. Discuss the relevance and limitations of this model to *information retrieval*.   [20%]

   ANSWER:

   [P1] (10%) The bag of words model assumes that a text can be represented simply in terms of the words it contains and, possibly, their frequencies in the text. Information about the relations between these words and their position in the text is ignored.

   [P2](5%) Standard IR models treat documents and queries as bags-of-words. The relevance for IR: IR indexing methods provide efficient encodings of bag-of-words type information on documents, discarding information about the order of words within them.

   [P3](5%) The limitation of the model is that all the information that word order conveys is lost. As such, additional treatment is necessary to deal with multi-word queries (phrases, etc.), for example, by indexing sequences of words in addition to individual words.

   b) Explain the intuition of the noisy channel model in the context of Statistical Machine Translation (SMT). Discuss whether and how this model is used in modern SMT approaches (i.e., phrase-based and syntax-based SMT).   [20%]

   ANSWER:

[P1](10%) The noisy channel model was proposed as the first approach to empirically model the machine translation process. Suppose we want to translate a text from French (f) to English (e). The noisy channel model assumes that the true text is English, but that when it was transmitted to us, it went through a noisy communication channel and came out as French. In order to translate we need to recover the original English, i.e., to decode the French to get the English, i.e. to find $P(e|f)$, which is decomposed into two parts: $P(f|e) * P(e)$, the language model ($P(e)$) and the translation model ($P(f|e)$), the inverted translation probability.

[P2](10%) In recent approaches to SMT the noisy channel is not used anymore. Instead of only the inverse translation probability, $P(f|e)$, and the language model probability, $P(e)$, current approaches use the direct translation probability $P(e|f)$, and combine these models linearly using a discrimative approach.

---

c) Give two reasons why it is important to evaluate the performance of text processing systems. Define the notion of a *gold-standard* dataset (also called a *reference dataset*) and explain how it differs from application to application (e.g. in Information Retrieval and Machine Translation). [20%]

ANSWER:

[P1](10%) Evaluation of text processing systems is important in order to compare two or more systems and to compare different settings of the same system, say to measure performance over time.

[P1](10%) A gold-standard or reference dataset is the a set of examples of correct answers for a given benchmark. For certain applications, the correct answer is a simple deterministic yes/no label (e.g. relevant or irrelevant documents for a query), while for others there are multiple correct answers and it might be too expensive and even impossible to generate all possible correct answers (e.g. correct translations for an input text in machine translation).

---

d) For *each* of the following three text processing applications, give a commonly used *automatic* metric for *intrinsic* evaluation: Information Retrieval, Machine Translation and Sentiment Analysis. Discuss any similarities between these metrics. [40%]

ANSWER:

[P1](10%) The most commonly used evaluation metrics for Information Retrieval are (only one necessary) precision and recall (or f-measure, which includes both precision and recall). Assuming that: RET is the set of all documents the system has retrieved for a specific query; REL is the set of relevant documents for a specific query; $RETREL$ is the set of the retrieved relevant documents, i.e.,

$RETREL = RET \cap REL$. Precision is defined as $|RETREL|/|RET|$ and recall is defined as $|RETREL|/|REL|$. F-measure is the harmonic mean between the two metrics, where a weight α can be used to prioritise one or the other: $F_\alpha = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$.

[P2](10%) BLUE is the most commonly used automatic evaluation measure for MT. It compares the system output against reference documents which provide examples of good translations. The comparison is carried out by counting overlapping n-grams (with n normally from 1 to 4 for MT) in the system output and references. Other valid metrics (only one necessary) include TER, back-translation, etc.

[P3](10%) In sentiment analysis, the metrics used are very similar to those in IR (only one necessary). One tends to look at general accuracy (number of correctly classified / number of cases) or precision and recall for either the positive or negative class individually, e.g. Precision Positive = number of segments correctly classiffied as positive / number of segments classiffied as positive. One can also look at f-measure of the positive or negative class.

[P4](10%) BLEU and other metrics used of MT evaluation are a more complex than the metrics used for IR/SA, as the correct answer is not a simple label. They have to account for the fact that there might he partially correct answers (by looking at sub-sequences of the answers) and that there might be multiple correct answers (by dealing with multiple references). However, they can also be interpreted in terms of precision or recall (or f-measure). BLEU, for example, is a precision-oriented metric, since in essence the metric computes the proportion of overlapping n-grams in the MT and reference over the total of n-grams *in the MT*.

SECTION B

2. 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 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:

[P1](6%) 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.

[P1](8%) *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. Stemming will also produce some reduction in the size of document indexes.

[P1](6%) *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.

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:

[P1](3%) Based on *lowercasing* and *stopword removal* we assume a stoplist that includes the following terms: {at, can, my, the, you, !, ,, .}.

[P2](7%) As a result, 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 metrics: Euclidean and cosine. Determine the ranking of the two documents according to each of these metrics and discuss any differences in the results. [40%]

ANSWER:

[P1](10%) 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: $\langle 1, 1, 0, 0, 1, 1 \rangle$

[P2](10%) **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:

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

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

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

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

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

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

[P3](10%) **Ranking using euclidean distance**: Euclidean distance is computed as $\sqrt{\sum_{i=1}^{n}(q_i - d_i)^2}$. In this case:

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

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

[P4](10%) **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) Explain TF.IDF. Include the formula (or formulae) for computing TF.IDF values as part of your answer. 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 than TF in this example?

[30%]

ANSWER:

[P1](15%) 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:

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

[P2](15%)

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.

3. a)   Explain the differences between *rule-based* and *empirical* approaches to Machine
        Translation. Give the main advantage and disadvantage of each of these approaches.
        [20%]

ANSWER:

[P1](10%) Rule-based approaches are based on handcrafted rules designed by linguists,
to translate from a language into a different language. Empirical approaches use
examples of translations previously created (in general by human translators) and
generate the translation of a new source segment by combining partial translations of
words or sequences in the new segment to translate. The combination can be done in
different ways, such as based on a model of translation probabilities.

[P2](5%) Advantage of rule-based approaches: rules can be very precise and gener-
ate good quality translation. Disadvantage (any of the three): 1) labour and time-
intensive: there are often too many grammar rules to design, 2) it is hard to avoid
contradiction and to maintain the grammar consistent over time. 3) Maintenance in
general is an issue as changing one rule could have a knock on effect on many other
rules.

[P3](5%) Advantage of empirical approaches: no expert knowledge is necessary: mod-
els can be generated given data only (abundant for many language pairs and domains)
and can be easily re-generated as new examples are collected. Disadvantage: the very
little or inexistent linguistic information makes models far from precise generating
garbled translations in many cases.

b)   Describe the two main models of a standard *phrase-based* approach to statistical
     machine translation. Explain how these models are combined and how they are applied
     to generate a translation for a new segment.   [30%]

ANSWER:

The two main models of a PBSMT system are $P(F|E)$ (translation model) and $P(E)$
(language model).

[P1](7%) $P(F|E)$: faithfulness model - will ensure that a translation E will have
words that generally translate to words in F.

[P2](7%) $P(E)$: fluency model - will ensure that E reads well and is grammatical in
the target language.

[P3](8%) These as well as the other models 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_i is the weight
of each model.

[P4](8%) To translate a new source segment, a *decoder* applies this combined model to find the translation that covers all source words and maximises the weighted joint translation probability $\sum_{i=1}^{m} \lambda_i h_i$ for that source segment.

c) Explain the intuition behind the IBM Model 1 in the context of Statistical Machine Translation (SMT). Give the most important outcome of this model for an SMT system. Give one direction in which this model can be improved.   [30%]

ANSWER:

[P1](15%) IBM Model 1 is a model for word-alignment between bilingual segments, i.e. to identify likely correspondences between two languages at word level. The model is trained in an unsupervised way based on a (large) training parallel corpus where sentences have been aligned, i.e., a corpus of sentence pairs which are translation of each other. The algorithm is a direct application of the Expectation Maximization (EM) algorithm:

1) Initially consider all alternative word alignments as equally likely

2) Observe across sentences that (e.g.) source word 'x' often links to target word 'y'

3) Increase probability of this word pair aligning

4) Iteratively redistribute probabilities, until probabilities stop changing (convergence)

[P2](5%) Once the corpus is aligned, a dictionary of source-target word pairs with translation probabilities can be extracted by simply taking the output of the final iteration. This dictionary can then be used as the most basic component in SMT.

[P2](10%) Possible directions (only one necessary): 1) add a 'fertility' component, i.e., words in the source language can generate (align too) multiple words in the target language 2) Add component to model absolute position of the words in the two sentences, i.e., words in the certain positions in the source segment tend to align to words in certain positions in the target segment, for example, to learn that the first word in the source segment may be more likely to align to the first or one of the first words in the target language as opposed to one of the last words in the target language 3) Add a component to model word order dependencies, i.e., the fact the alignment of one source word to a given target word may be dependent on the neighboring words in one of both languages

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 a good solution in any of these scenarios?

[20%]

ANSWER:

[P1](10%) 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.

[P2](10%) 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.

4. a) Given sentences like the following:

- *My new phone works well and it is much faster than the old one.*
- *My new phone has 32GB of memory and can play videos.*

What is the first step to detect the sentiment in these two sentences? Should both these sentences be addressed in the same way by sentiment analysis approaches? [20%]

ANSWER:

[P1](10%) The first step is to run a subjectivity analysis. This has to do with detecting whether the text (word/phrase, sentence, document) contains opinions, emotions, sentiment, or simply facts. Only subjective sentences should then be put forward for sentiment analysis, which has to do with the actual polarity of the text (word/phrase, sentence, document): positive, negative, or more fine-grained distinctions.

[P2](10%) In the example, only the first sentence is subjective and therefore has a sentiment that can be analysed.

b) Explain two approaches for Subjectivity Analysis.   [20%]

ANSWER:

[P1](10%) A simple rule-based subjectivity classifier can be built: a sentence/document is subjective if it has at least n (say 2) words that belong to an emotion words lexicon; a sentence/document is objective otherwise.

[P2](10%) A Naive Bayes classifier can be built with features that count emotion words from a lexicon, as well as other features, such as adjectives, negation words, etc. Training data with instances with binary labels (subjective vs objetive) would be required.

c) Explain the weighted lexical-based approach for Sentiment Analysis. Given the following sentences and opinion lexicon, apply this approach to classify *each* sentence in S1-S4 as **positive**, **negative** or **objective**. Show the final emotion score for each sentence and also how this score was generated. Give any general rules that you used to calculate this score as part of your answer. Explain these rules when they are applied.   [30%]

	Lexicon:	
	awesome	5
	boring	-3
	brilliant	2
	funny	3
	happy	4
	horrible	-5

(S1) He is brilliant and funny.
(S2) I am not happy with this outcome.
(S3) I am feeling AWESOME today, despite the horrible comments from my supervisor.
(S4) He is extremely brilliant but boring, boring, very boring.

ANSWER:

[P1](10%) The general weighted lexical-based approach counts positive (Cpos) and negative (Cneg) words in the text and weights them using the weights in the lexicon given:
If Cpos > Cneg then positive
If Cpos < Cneg then negative
If Cpos = Cneg = 1 then objective

[P2](5%) (S1) Emotion(brilliant) = 2; Emotion(funny) = 3. Therefore Cpos = 2+3 and Cneg = 0, so Cpos > Cneg = positive

[P3](5%) (S2) Emotion(happy)= 4; Rule = "not" is detected in neighbourhood, so emotional valence of term is decreased by 1 and sign is inverted. Therefore Emotion(happy)=-3, and Cneg=-3, so Cneg > Cpos = negative

[P4](5%) (S3) Emotion(horrible) = -5, Emotion(awesome) = 5. Rule = "awesome" is intensified because it is in capital letters, and in this case it's intensified by 1 (because it's a positive word). Therefore, Cneg = -5, Cpos = 6, so Cpos > Cneg = positive

[P5](5%) (S4) Emotion(brilliant) = 2; Emotion(boring) = -3. Rules = "boring" happens 3x, so Emotion(boring) = -9. "extremely" is a (positive) intensifier in this case, with +2 added, so Emotion(brilliant) = 4. "very" is a (negative) intensifier in this case, with -2 subtracted from total. Cpos = 4 and Cneg = -11, so Cneg > Cpos = negative

d) Give Bing Liu's model for an **opinion**. Explain each of the elements in the model and exemplify them with respect to the following text. Identify the features present in the text, and for each indicate its sentiment value as either *positive* or *negative*. Discuss two language processing challenges in automating the identification of such elements.
[30%]

"I have just bought the new iPhone 5. It is a bit heavier than the iPhone 4, but it is much faster. The camera lenses are also much better, taking higher resolution pictures. The only big disadvantage is the cost: it is the most expensive phone in the market. Lucia Specia, 12/10/2015."

ANSWER:

[P1](20%) An **opinion** is a quintuple $(o_j, f_{jk}, so_{ijkl}, h_i, t_l)$, where:

- o_j is a target object: in the example, iPhone 5
- f_{jk} is a feature of the object o_j: in the example, weight, speed, camera/lenses/pictures, price
- so_{ijkl} is the sentiment value of the opinion: in the example, negative, positive, positive, negative
- of the opinion holder h_i (usually the author of the post): in the example, Lucia Specia
- on feature f_{jk} of object o_j at time t_l: in the example, 12/10/2015.

[P2](10%) Some of the challenges are (only two necessary):

- Need Named Entity Recognition to identify target objects.
- Need Information Extraction to extract features of the target object (properties of objects), as well as time and holder.
- Need co-reference resolution to know that "it" = iPhone.
- Need synonym match when words used do not belong to lexicon of opinion words, e.g. fast versus efficient

5. 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 ⌒ (which is used as a 'space' between words). Corpus analysis shows that the probabilities of these six characters are as follows:

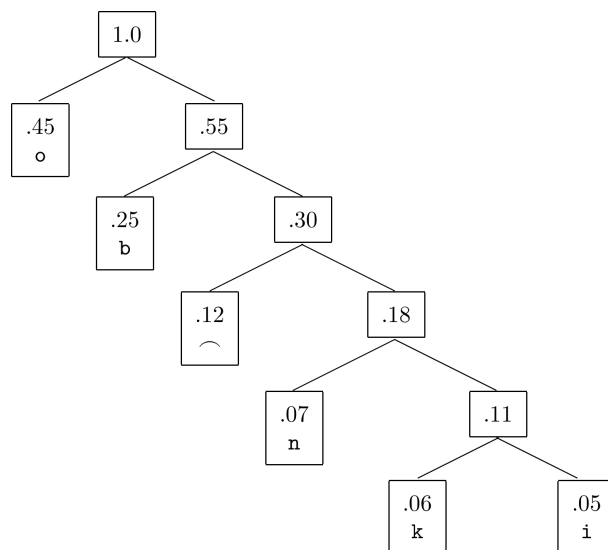| Symbol	Probability
b	0.25
i	0.05
k	0.06
n	0.07
o	0.45
⌒	0.12

Apply the method you described in 5(a) to create a Huffman code for the Bonobo character set.   [30%]

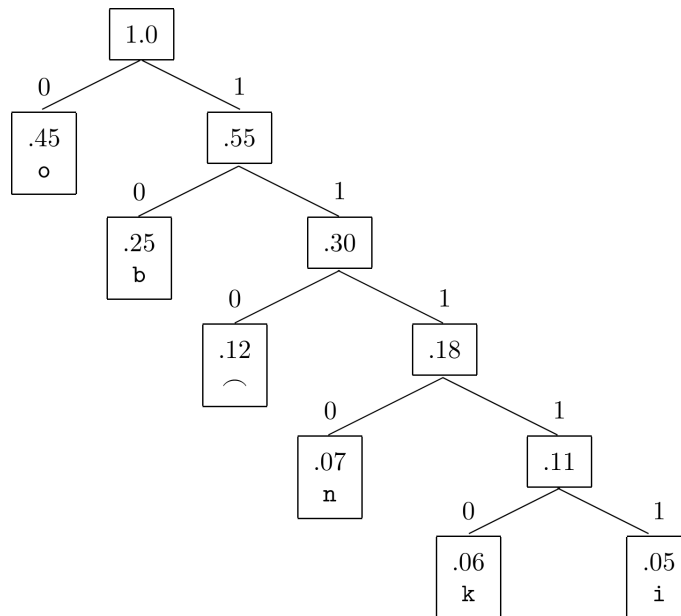ANSWER:

First create initial nodes as follows:

.45	.25	.12	.07	.06	.05
o	b	⌒	n	k	i

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 5(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
⌢	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.

Where the student has answered 5(b) of the question incorrectly (resulting in an incorrect code), they can still gain full marks on this part of the question, provided that their answer demonstrates the correct method for computing the required values, and illustrates it w.r.t. their (incorrect) code from 5(b).

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:

$$\text{bonobo⌢okobo}$$

$$\text{iniko⌢nikoni}$$

Discuss why the two bits-per-character rates achieved differ, comparing them also to the expected rate that you computed in 5(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.

Where the student has answered 5(b) of the question incorrectly (resulting in an incorrect code), they can still gain full marks on this part of the question, provided that their answer demonstrates the correct method for computing the required values, and illustrates it w.r.t. their (incorrect) code from 5(b).

END OF QUESTION PAPER