MODEL SOLUTIONS

SETTER: Lucia Specia / Mark Hepple

Data Provided: None

DEPARTMENT OF COMPUTER SCIENCE **Autumn Semester 2013-2014**

TEXT PROCESSING **2 hours and 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) In the context of Information Retrieval, explain the difference between algorithms that perform boolean search and algorithms that perform a ranked search. What type of algorithm would be better for a regular user (such as an undergraduate student in the Humanities area) who is using a search query with multiple terms, which he/she expects to appear in many documents? Explain the reasons behind your choice of algorithm type. [30%]

ANSWER:

[P1] Algorithms that perform boolean search match the words in the query exactly with the words in the document, requiring boolean operators such as AND, OR, BUT, when multiple terms are entered as a query. The search is a binary decision: either the document contains the terms or it does not contain them. The list of documents retrieved is not ranked by any criteria. With algorithms that perform a ranked search, terms are expressed freely, without boolean operators, and the documents containing such terms are retrieved and ranked using models of frequency of the terms in the query in those documents. Not all keywords need to be present, but documents containing more of the keywords searched and in higher frequency will be ranked first. [P2] A ranked search would be better because [P3] a regular user will not be familiar with boolean operators to connect multiple terms. In addition, since the terms are frequent, it would be interesting to retrieve documents containing more of these terms first.

b) Compression techniques are important due to the growth in volume of the data that must be stored and transmitted.

(i) Explain the difference between **lossy** and **lossless** forms of compression. Discuss the suitability of these alternative forms of compression for different media types (e.g. for text vs. image data). [10%]

ANSWER:

Lossless data compression refers to the class of data compression algorithms that allow the original data to be perfectly reconstructed from the compressed data. By contrast, lossy data compression methods achieve data reduction by discarding (losing) information.
For an example of the latter, we can reduce the data volume of an image that has a high pixel-density by reducing the pixel density. The image that results is typically still interpretable as an image, even if it is of lower quality/fidelity. Such approaches are commonly applied to image, video and audio data (especially for use in streaming contexts). For such media, a substantial amount of data can be discarded before the result is sufficiently degraded for this to be noticed by the user.
Text compression applications require *lossless* compression methods — the idea of a version of text from which N% of the information has been discarded doesn't make sense. In general, we expect decompression to return a text that is identical to the original in both content and form.

(ii) Explain the difference between **static**, **semi-static** and **adaptive** techniques for text compression, noting their key advantages and disadvantages. [10%]

ANSWER:

Compression techniques can also be distinguished by whether they are

- **Static** – use a fixed model or fixed dictionary derived in advance of any text to be compressed
  - adv: model does not need to be transmitted
  - disadv: model may not be well suited to text currently being compressed
- **Semi-static** – use current text to build a model or dictionary during one pass, then apply it in second pass
  - adv: model should be well suited to current text
  - disadv: model must also be transmitted, reducing effectiveness of compression

- **Adaptive** – build model or dictionary adaptively during one pass
  - adv: model does not need to be transmitted
  - disadv: decoder determines model used at each stage from data decoded so far, so cannot do random access into data

c) The two main model components in Statistical Machine Translation are the *Translation Model* and the *Language Model*. Explain the role of each of these components. Describe the type of data that is necessary to build each of them. Mention one way in which these components can be combined to build a translation system. [30%]

ANSWER:

[P1]Suppose we want to translate a text from French (f) to English (e). A translation model estimates the probability of translating the source language into the target language, $P(e|f)$ (this is if the linear model is the underlying framework; if the noisy channel is the underlying framework, then a translation model estimates the probability of translating the target language into the source language $P(f|e)$ – both answers correct). It ensures that e only have words that generally translate words in f, i.e., it focuses on the preserving the content of the source text. The language model estimates the probability $P(e)$ that a text in the target language is a common (and thus fluent) text. It ensures that the translation is fluent English. [P2] The translation model is built from parallel data, i.e., examples of f and e sentences that are translations of each other. The language model is built from target language data e only, as many examples of good English sentences as possible. [P3] These two models can be combined using (both answers are correct, only one needed): (1) a generative framework (e.g. the Noisy Channel model), where the final translation score is given by the product of both models: $P(e|f) = P(f|e) * P(e)$, as they are considered both equally relevant; or (2) a linear model of the type $\sum_{i=1}^{m} \lambda_i h_i$, where λ are learned weights of each these two models h_is (and possibly additional models), and the weighted model scores are summed over to produce the final translation score.

d) Assume we have a small set of seed words with positive and negative opinions, e.g.: *positive* = {*good, fast, cheap*} and *negative* = {*slow, boring, fragile*}. Explain the **two** most common (semi-)automated approaches to expand these sets with more opinion words or phrases to create lexica for Sentiment Analysis, providing examples whenever possible. Give **one** advantage and **one** disadvantage of each approach. [20%]

ANSWER:

[P1] The two most common approaches to create lexica for Sentiment Analysis are 1) lookup in dictionaries or lexical databases like WordNet for synonyms in both sets of seed words to expand those sets, for antonyms in the negative set (generating positive words) and in the positive set (generating negative words). This can be repeated once the sets become larger. 2) corpus-based approaches where patterns are built from the seed words, such as "good and ", "fragile but ", "very tasty and ", and searched on corpora (such as the Web). In this case, all examples of patterns should result in more positive words or phrases (such as "good and reliable"). [P2] The advantages of the first approach are (only one is necessary) simplicity, as searches are straightforward, and accuracy, as lexical databases are normally built by humans and tend to be accurate. The disadvantages (only one is necessary) are that lexical databases are only available for a limited number of languages, and that they usually contain only words, not phrases. The advantages (only one is necessary) of the corpus-based approach is that it is more flexible (patterns can include phrases), and that corpora of this type are available for most languages. The disadvantages (only one is necessary) are that the results are not always accurate, and that patterns must be cleverly elaborated to avoid noisy results.

SECTION B

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

Document 1: Sea shell, buy my sea shell!

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

Document 3: Product marketing in the Shelly sea is an expensive market.

and the query:

Query 1: sea shell produce market

a) Apply the following term manipulations on document terms: *stoplist removal*, *capitalisation* and *stemming*, showing the transformed documents. Explain each of these manipulations. Provide the stoplist used, making sure it includes punctuation, but no content words. [20%]

ANSWER:

[P1]

Document 1: sea shell buy sea shell

Document 2: can buy love sea shell sea produc market

Document 3: produc market shell sea expens market

[P2] A *stoplist* is a list of words ('stop-words') that are ignored when documents are indexed. These are words that are so widespread in the document collection that they are of very little use for discriminating between documents that are/are not relevant to a query. *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) disregarded, e.g. Turkey (country) vs turkey (bird). *Stemming* is 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*.

[P3] Stoplist: {`may, at, in, is, my, the, you, !, ,, .`}

b) Show how Document 1, Document 2 and Document 3 would be represented using an *inverted index* which includes term frequency information. [10%]

ANSWER:

term-id	word	docs
1	buy	d1:1, d2:1
2	expens	d3:1
3	love	d2:1
4	market	d2:1, d3:2
5	produc	d2:1, d3:1
6	sea	d1:2, d2:2, d3:1
7	shell	d1:2, d2:1, d3:1

c) Using *term frequency* (TF) to weight terms, represent the documents and query as vectors. Produce rankings of Document 1, Document 2 and Document 3 according to their relevance to Query 1 using two metrics: Cosine Similarity and Euclidean Distance. Show which document is ranked first according to each of these metrics. [30%]

ANSWER:

[P1] Using the term order taken from the inverted index above, we can represent the three documents and the query as vectors as follows:

d1: $\langle 1,0,0,0,0,2,2 \rangle$
d2: $\langle 1,0,1,1,1,2,1 \rangle$
d3: $\langle 0,1,0,2,1,1,1 \rangle$
q: $\langle 0,0,0,1,1,1,1 \rangle$

[P2] **Ranking using cosine similarity**: The cosine between two vectors $\vec{d}$ and $\vec{q}$ is computed as:

$$cos(\vec{d},\vec{q}) = \frac{\vec{d}\cdot\vec{q}}{|\vec{d}||\vec{q}|} = \frac{\sum_{i=1}^{n} d_i q_i}{\sqrt{\sum_{i=1}^{n} d_i^2}\sqrt{\sum_{i=1}^{n} q_i^2}}$$

The vector magnitudes and cosine values are then:

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

$$|\mathrm{d2}| = \sqrt{1^2+0^2+1^2+1^2+1^2+2^2+1^2} = \sqrt{9} = 3$$

$$|\mathrm{d3}| = \sqrt{0^2+1^2+0^2+2^2+1^2+1^2+1^2} = \sqrt{8} = 2.83$$

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

$$cos(\mathrm{d1},\mathrm{q}) = \frac{1*0+0*0+0*0+0*1+0*1+2*1+2*1}{3*2} = 4/6 = 0.667$$

$$cos(\mathrm{d2}, \mathrm{q}) = \frac{1*0+0*0+1*0+1*1+1*1+2*1+1*1}{3*2} = 5/6 = 0.833$$

$$cos(\mathrm{d3}, \mathrm{q}) = \frac{0*0+1*0+0*0+2*1+1*1+1*1+1*1}{2.83*2} = 5/6.66 = 0.88$$

The cosine values computed for the three documents indicate that document 3 is more relevant to the query.

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

$$dis(\mathrm{d1}, \mathrm{q}) = \sqrt{(1-0)^2+(0-0)^2+(0-0)^2+(0-1)^2+(0-1)^2+(2-1)^2+(2-1)^2} = \sqrt{5} = 2.24$$

$$dis(\mathrm{d2}, \mathrm{q}) = \sqrt{(1-0)^2+(0-0)^2+(1-0)^2+(1-1)^2+(1-1)^2+(2-1)^2+(1-1)^2} = \sqrt{3} = 1.73$$

$$dis(\mathrm{d3}, \mathrm{q}) = \sqrt{(0-0)^2+(1-0)^2+(0-0)^2+(2-1)^2+(1-1)^2+(1-1)^2+(1-1)^2} = \sqrt{2} = 1.41$$

Document 3 is again ranked first (this is a distance metric).

d) Explain the intuition behind using TF.IDF (*term frequency inverse document frequency*) to weight terms in documents. Include the formula (or formulae) for computing TF.IDF values as part of your answer. For the ranking in the previous question using cosine similarity, discuss whether and how using TF.IDF to weight terms instead of TF only would change the results. [20%]

ANSWER:

[P1] 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] In the previous question, using TF.IDF would set the weight of two of the query terms to 0: *sea, shell*, since they appear in all (three) documents in the collection and thus $log\frac{|D|}{df_w} = log\frac{3}{3} = log(1) = 0$. The two other query terms *market, produc* would receive a weight different from 0. Document 3 has more occurrences of these terms, and therefore this document would be ranked first again, making no difference in the overall ranking.

e) Explain the metrics Precision, Recall and F-measure in the context of evaluation in Information Retrieval against a gold-standard set, assuming a boolean retrieval model. Discuss why it is not feasible to compute recall in the context of searches performed on very large collections of documents, such as the Web. [20%]

ANSWER:

[P1] 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 (P) is defined as $|RETREL|/|RET|$ and Recall (R) is defined as $|RETREL|/|REL|$. F-measure (F) is the harmonic mean between the two metrics, where a weight α can be used to prioritise one or the other: $F = \frac{1}{\alpha\frac{1}{P}+(1-\alpha)\frac{1}{R}}$. [P2] Recall is not feasible to compute in very large collections of documents because it is virtually impossible to know what are all the relevant documents. Therefore we can only judge whether all the retrieved documents are relevant (precision), but not whether all the relevant documents were retrieved.

3. a) List and explain the three paradigms of Machine Translation. What is the dominant (most common) paradigm for open-domain systems nowadays and why is this paradigm more appealing than others, especially in scenarios such as online Machine Translation systems? [20%]

ANSWER:

[P1] The three paradigms are Rule-based MT, Example-based MT and Statistical MT. RBMT uses hand-crafted rules with different levels of linguistic information to transform a text from the source language into the target language. This can be done word-for-word (direct approach), at the syntactic/semantic level (transfer approach) or at more abstract levels (interlingua approach). EBMT is based on the idea of translation by analogy: if a segment (part of a sentence) has been translated before, at translation time, fetch this segment, copy its translation to the target. So the translation process consists in segmenting the source sentence into segments that have been seen before, retrieving their translations, and putting these together to form a target language sentence. SMT is also based on the translation by analogy idea, but in this case instead of a simple match to existing source texts at runtime, examples of translations are used offline to build models of translation, e.g. at word or phrase levels. These models combine a number of information sources, like word/phrase translation probabilities and fluency models to generate the most likely translation for a new source sentence. [P2] SMT is the dominant paradigm for open-domain systems (like Google Translate). It is more appealing because it systems can be quickly built from a few thousands examples of translations using reasonably simple, language-independent statistical models.

b) Lexical ambiguity is known to be one of the most challenging problems in any approach for Machine Translation. Explain how this problem is addressed in Phrase-based Statistical Machine Translation approaches. [20%]

ANSWER:

Lexical ambiguity requires the use of context of source and/or target languages to

help select the best translation. Phrase-based SMT represents context in two forms: neighbour words in the phrases (for both source and target languages) and n-gram language models of the target. Both are poor presentations of context in practice, as phrases tend to be short (up to 7 words) and language models only look at a short window of 3-5 words.

c) List and explain two metrics that can be used for evaluating Machine Translation systems (either manually or automatically). Discuss the advantages of automatic evaluation metrics over manual evaluation metrics. [20%]

ANSWER:

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

1) BLEU: this metric compares n-grams in the system output to n-grams in one or more human translations. N-grams normally go from 1 to 4 words. The higher the number of overlapping n-grams, the closer the system output is to a human output, and therefore, the better the translation is considered. n-grams are weighed proportionally to their length: matches of longer n-grams count more towards the final score.

2) Fluency/Adequacy: Human judges check system outputs compared to either source sentences or manually-created reference translations and attempt to assess their adequacy (how much of the meaning of the source/reference is preserved) and/or their fluency (how well the translations read), using scales of 1(worst) to 5(best), for example.

3) Task-based: 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.

3) Extrinsic: MT systems 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.

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, source text.

[P2] Automatic metrics are much cheaper: once a gold-standard set is collected, they can be repeated for any MT system, they are fast (a matter of seconds to score a dataset), and they are deterministic: different rounds on the same text will always produce the same figures, which is not the case with human judges (disagreements exist both between different judges and the same judge at different points in time).

d) Given the two scenarios:

Scenario 1: English-Arabic language pair, 50,000 examples of translations of very short sentences, on very repetitive material (technical documentation of a product).

Scenario 2: English-French, 500,000 examples of translations for open-domain and creative texts, like novels from many different writers.

In which of these scenarios would Statistical Machine Translation work better? Why would it work better than in the other scenario? [10%]

ANSWER:

SMT would work better for scenario 1. Despite the languages being very different, the content is repetitive so the SMT can easily memorise the translations. Scenario 2 poses more problems for SMT, despite the larger number of examples and proximity between the languages. This is because translations of novels tend to be very far from literal, one-to-one translations, and it is therefore very hard to generalise examples into a statistical model.

e) Explain the main advantage of Hierarchical Phrase-based Machine Translation models over standard Phrase-based Statistical Machine Translation models. What does the phrase table of Hierarchical Phrase-based Machine Translation models look like? Given the following sentence pair and the existing phrases (in the phrase-table), which additional phrases could be generated with a Hierarchical Phrase-based Machine Translation model?

Source: shall be passing on to you some comments
Target: werde Ihnen die entsprechenden Anmerkungen aushäandigen

Existing phrases:

Source	**Target**
shall be	werde
passing on	aushäandigen
to you	Ihnen
some comments	die entsprechenden Anmerkungen
to you some comments	Ihnen die entsprechenden Anmerkungen

[30%]

ANSWER:

[P1] Hierarchical SMT allows to introduce structure into phrase-based SMT models.

[P2] Instead of representing only flat phrases with corresponding source and target words/phrases, these models allow the representation, in the phrase table, of phrases with variables which, at decoding time, can be substituted by other phrases/words in the phrase table, until no more variables exist.

[P3]

Source	**Target**
shall be passing on X	werde X aushäandigen
shall be passing on X1 X2	werde X1 X2 aushäandigen

4. a) Differentiate *subjectivity* from *sentiment*. How are the tasks of Subjectivity Classification and Sentiment Analysis related? [10%]

ANSWER:

[P1] Subjectivity has to do with whether the text (word/phrase, sentence, document) contains opinions, emotions, sentiment, or simply facts. Sentiment has to do with the actual polarity of the text (word/phrase, sentence, document): positive, negative, or more fine-grained distinctions. Subjectivity classification is often the first step for sentiment analysis: subjective versus objective texts, e.g.:

- **Objective**: *I bought an iPhone a few days ago.*
- **Subjective**: *It is such a nice phone.*

However:

- Subjective sentences do not always have positive or negative opinions, e.g.: *I think he came yesterday.*
- Objetive sentences can express opinion, e.g.: *My phone broke in the second day.*

b) Explain the steps involved in the lexicon-based approach to Sentiment Analysis of features in a sentence (e.g. features of a product, such as the *battery* of a mobile phone). Discuss the limitations of this approach. [20%]

ANSWER:

[P1] This approach uses a lexicon of opinion words/phrases, like: good, bad, horrible, great, etc., where positive words are assigned a positive weight (e.g., 1), negative words, a negative weight (e.g., -1), and neutral words are assigned 0. Starting with a pair (f, s), where f is a product feature and s is a sentence that contains f, segment sentence by features, e.g. based on BUT words (but, except that, ...). Then work on the segment s_f containing f. Let the set of opinion words in s_f be $w_1, .., w_n$. Sum up their orientations (1, -1, 0), and assign the orientation to (f, s) accordingly: positive if bigger than 0, neutral if 0, negative if smaller than 0. The output is whether the opinion on f in s is positive, negative, or neutral.

[P2] The limitations are that a lexicon needs to be in place, and it disregards context: certain opinion words have context-independent orientations, e.g. "great", while other words have context-dependent orientations, e.g. "small" in "small power consumption" vs "small screen". This cannot be taken into account as the opinion words are used in isolation.

c) Given the following sentences and opinion lexicon (adjectives only), apply the weighted lexical-based approach to classify EACH sentence as positive, negative or objective.

Show the final emotion score for each sentence. In addition to use of the lexicon, make sure you consider any general rules that have an impact in the final decision. Explain these rules when they are applied. [20%]

Lexicon:

boring	-3
brilliant	2
good	3
horrible	-5
happy	5

(S1) He is brilliant but boring.
(S2) I am not good today.
(S3) I am feeling HORRIBLE today, despite being happy with my achievement.
(S4) He is extremely brilliant but boring, boring.

ANSWER:

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

(S1) Emotion(brilliant) = 2; Emotion(boring) = -3. Therefore Cpos = 2 and Cneg = -3, so Cneg > Cpos = negative

(S2) Emotion(good)= 3; "not" is detected in neighbourhood (of 5 words around). Emotional valence of term is decreased by 1 and sign is inverted. Therefore Emotion(good)=-2, and Cneg=-2, so Cneg > Cpos = negative

(S3) Emotion(horrible) = -5, Emotion(happy) = 5, but "horrible" is intensified because it is in capital letters, and in this case it's intensified by -1 (because it's a negative word). Therefore, Cneg = -6, so Cneg > Cpos = negative

(S4) Emotion(brilliant) = 2; Emotion(boring) = -3, but it happens twice, so Emotion(boring) = -6. "extremely" is a (positive) intensifier in this case, with +2 added, so Emotion(brilliant) = 4. Cpos = 4 and Cneg = -6, so Cneg > Cpos = negative

d) Specify the five elements of Bing Liu's model for Sentiment Analysis, 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 am in love with my new Toshiba Portege z830-11j. With its i7 core processors, it is extremely fast. It is the lightest laptop I have ever had, weighting only 1 Kg. The SSD disk makes reading/writing operations very efficient. It is also very silent, the fan is hardly ever used. The only downside is the price: it is more expensive than any Mac. Lucia Specia, 10/04/2012."

ANSWER:

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

- o_j is a target object: Toshiba Portege z830-11j
- f_{jk} is a feature of the object o_j: speed, weight, disk-technology, level-of-noise, price
- so_{ijkl} is the sentiment value of the opinion: positive, positive, positive, positive, negative
- of the opinion holder h_i (usually the author of the post): Lucia Specia
- on feature f_{jk} of object o_j at time t_l: 10/04/2012.

[P2] 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" = Toshiba Portege z830-11j in second sentence.
- Need synonym match when words used do not belong to lexicon of opinion words, e.g. silent versus quiet

e) Differentiate *direct* from *comparative* Sentiment Analysis. What are the elements necessary in comparative models of Sentiment Analysis? [20%]

ANSWER:

Direct sentiment analysis focuses on one object or feature. It can thus directly use opinion words with absolute meaning (and corresponding positive/negative weights), like "great" in "the picture quality of this camera is great.". Comparative sentiment analysis is more complex as it expresses similarities or differences between objects, e.g., "car x is cheaper than car y." No positive/negative opinions are extracted.

Given an opinionated document d, the goal is to extract comparative opinions: $(O_1, O_2, F, p0, h, t)$, where O_1 and O_2 are the object sets being compared based on their shared features F, po is the preferred object set of the opinion holder h, and t is the time when the comparative opinion is expressed.

5. a) (i) Explain how the LZ77 compression method works. [30%]

ANSWER:

The **key idea** underlying the LZ77 adaptive dictionary compression method is to replace substrings with a pointer to previous occurrences of the same substrings in same text. The encoder output is a series of triples where

- the first component indicates how far back in decoded output to look for next phrase
- the second indicates the length of that phrase
- the third is next character from input (only necessary when not found in previous text, but included for simplicity)

Issues to be addressed in implementing an adaptive dictionary method such as LZ77 include

- how far back in the text to allow pointers to refer
  - references further back increase chance of longer matching strings, but also increase bits required to store pointer
  - typical value is a few thousand characters
- how large the strings referred to can be
  - the larger the string, the larger the width parameter specifying it
  - typical value $\sim$ 16 characters
- during encoding, how to search window of prior text for longest match with the upcoming phrase
  - linear search very inefficient
  - best to index prior text with a suitable data structure, such as a trie, hash, or binary search tree

A popular high performance implementation of LZ77 is **gzip**

- uses a hash table to locate previous occurrences of strings
  - hash accessed by next 3 characters
  - holds pointers to prior locations of the 3 characters
- pointers and phrase lengths are stored using variable length Huffman codes, computed semi-statically by processing 64K blocks of data at a time
- pointer triples are reduced to pairs, by eliminating 3rd element
  - first transmit phrase length – if 1 treat pointer as raw character; else

treat pointer as genuine pointer

(ii) Assuming the encoding representation presented in class (i.e. in the lectures of the Text Processing module), show what output would be produced by the LZ77 decoder for the following representation. Show how your answer is derived. [15%]

$$\langle 0,0,b\rangle\ \langle 0,0,e\rangle\ \langle 2,2,n\rangle\ \langle 4,4,e\rangle\ \langle 1,3,b\rangle\ \langle 2,1,n\rangle$$

ANSWER:

1.	$\langle 0,0,b\rangle$	: go back 0 ;	copy len 0 ;	copies '' ;	add final b ;	overall: b
2.	$\langle 0,0,e\rangle$	: go back 0 ;	copy len 0 ;	copies '' ;	add final e ;	overall: be
3.	$\langle 2,2,n\rangle$	: go back 2 ;	copy len 2 ;	copies 'be' ;	add final n ;	overall: $beben$
4.	$\langle 4,4,e\rangle$	: go back 4 ;	copy len 4 ;	copies '$eben$' ;	add final e ;	overall: $bebenebene$
5.	$\langle 1,3,b\rangle$	: go back 1 ;	copy len 3 ;	copies 'eee' ;	add final b ;	overall: $bebenebeneeeeb$
6.	$\langle 2,1,n\rangle$	: go back 2 ;	copy len 1 ;	copies 'e' ;	add final n ;	overall: $bebenebeneeeeben$

Thus, the decoded string is: $bebenebeneeeeben$

b) The writing script of the (fictitious) language *Sinbada* employs only the letters (`s`,`i`,`n`,`b`,`a`,`d`) and the symbol $\sim$, used as a 'space' between words. Corpus analysis shows that the probabilities of these seven characters are as follows:

Symbol	Probability
`s`	0.04
`i`	0.1
`n`	0.2
`b`	0.04
`a`	0.3
`d`	0.26
$\sim$	0.06

(i) Sketch the algorithm for Huffman coding. Illustrate your answer by constructing a code tree for Sinbada, based on the above probabilities for its characters. [30%]

ANSWER:

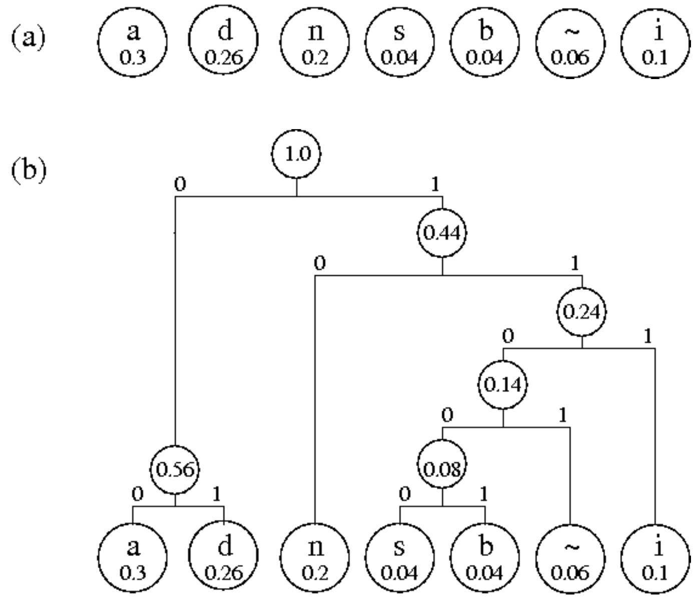
To begin, we need to know the relative probabilities of all the symbols in the symbol set.

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.

Applying this method to the case of Sinbada, we get an initial set of nodes as in (a) below. The algorithm then generates a binary tree as in (b), which has also been labelled with 0s and 1s.



(ii) Given the code you have generated in 5(b)(i), 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 Sinbada text? [10%]

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	$p \times len$
s	0.04	11000	5	0.20
i	0.1	111	3	0.30
n	0.2	10	2	0.40
b	0.04	11001	5	0.20
a	0.3	00	2	0.60
d	0.26	01	2	0.52
$\sim$	0.06	1101	4	0.24
				2.46

giving a cost of 2.46 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 7 to encode. Using a 3 bit fixed-length code would result in corpus storage requiring about 22% more space than the Huffman codes.

Full marks can be achieved if the student does an appropriate calculation of average bit-rate based on their code tree from part (b)(i), even if there is some error in that code tree.

(iii) Use your code tree to encode the message "`niad`$\sim$`badasina`" and show the resulting binary encoding. How does the bits-per-character rate achieved on this message compare to the rate that you calculated in 5(b)(ii)? [15%]

ANSWER:

Encoding for "`niad`$\sim$`badasina`" will be

```
n  i   a  d  ~    b     a  d  a  s     i   n  a
10 111 00 01 1101 11001 00 01 00 11000 111 10 00
```

The original message has 13 characters, and the codetree encoding has 36 bits, giving a bits-per-character rate of 2.77, which is better than the 3 bits per character of a minimal fixed length binary encoding, but not as good as the 2.54 bits-per-character rate calculated above, for a large corpus whose character distribution closely reflected the probability model used.

Again, full marks can be achieved if an appropriate answer is given for the code tree provided in (b)(i), even if there is some error in that code tree.

END OF QUESTION PAPER