The University Of Sheffield.

Ancillary Material: Computer Answer Sheets

PLEASE LEAVE THIS EXAM PAPER ON YOUR DESK.
DO NOT REMOVE IT FROM THE HALL.

DEPARTMENT OF COMPUTER SCIENCE **Autumn Semester 2017-18**

TEXT PROCESSING

2 hours

Answer ALL questions.

All questions carry equal weight. Figures in square brackets indicate the percentage of available marks allocated to each part of a question. Questions 1-3 should be answered in the answer book provided. Question 4 should be answered on the Computer Answer Sheet.

Registration number from U-Card (9 digits) — to be completed by student

1. a) Consider the following collection of three short documents:

Document 1: tiger, tiger, burning bright
Document 2: tiger sat on the mat
Document 3: the brightest burning mat

(i) Show how Documents 1, 2 and 3 would be stored in an inverted index which includes information about the frequency with which terms appear in documents but not about the position of their occurrences. Assume that the documents are pre-processed by removing punctuation, stemming terms and using the following list of stopwords: {`on`, `the`} [20%]

(ii) Show how the documents would be represented as vectors in which term weights correspond to term frequencies (TF). [10%]

(iii) Compute the similarity between the three documents and the query "brightest burning tiger" using the cosine metric and determine the ranking of the three documents for this query. Assume that: (1) term frequencies (TF) are used to weight the terms, (2) the cosine metric is used to compute similarity and (3) the query is pre-processed in the same way as the documents (see **1(a)(i)**). [30%]

(iv) Compute the tf.idf (term frequency, inverse document frequency) value for the term "tiger" in Document 1. [10%]

b) Consider the following ranking of ten documents produced by an Information Retrieval system. The symbol ✓ indicates that a retrieved document is relevant and × that it is not.

Document	Ranking	Relevant
d6	1	✓
d1	2	×
d2	3	✓
d10	4	×
d9	5	✓
d3	6	✓
d5	7	×
d4	8	×
d7	9	✓
d8	10	×

(i) Compute the (uninterpolated) average precision for this ranked set of documents. Explain your working. [10%]

(ii) Compute the interpolated average precision for this ranked set of documents assuming that there are a total of five relevant documents in the test collection. Explain your working. [20%]

2. a) Outline 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%]

b) We want to compress a large corpus of text of the (fictitious) language *Fontele*. The writing script of Fontele employs only the six letters found in the language name (`f`,`o`,`n`,`t`,`e`,`l`) and the symbol ␣, used as a 'space' between words. Corpus analysis shows that the probabilities of these seven characters are as follows:

Symbol	Probability
`e`	0.3
`f`	0.04
`l`	0.26
`n`	0.2
`t`	0.04
`o`	0.1
␣	0.06

(i) Show how to construct a Huffman code tree for Fontele, given the above probabilities for its characters. Use your code tree to assign a binary code for each character. [30%]

(ii) Given the code you have generated in 2(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 Fontele text? How does this compare to a minimal fixed length binary encoding of this character set? [20%]

(iii) Use your code to encode the message "`telefone`␣`noel`" and show the resulting binary representation. Compare the average bits-per-character rate achieved for this message to the expected rate that you computed in 2(b)(ii), and suggest an explanation for any difference observed between the two values. [20%]

3. a) Explain the graded lexicon-based approach for Sentiment Analysis. Given the following sentences and opinion lexicon, apply this approach to classify *each* sentence in S1-S3 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. [40%]

Lexicon:

action-packed	4
boring	-3
beautiful	3
compelling	2
dull	-2
great	3
horrid	-4
mostly	1
tedious	-3
very	1

(S1) Tedious movie, mostly boring characters, DULL, DULL, DULL.
(S2) While not action-packed, the plot is very compelling and cinematography beautiful!
(S3) Not great but not horrid either.

b) A second approach to Sentiment Analysis is the corpus-based supervised learning approach.

(i) Explain how a Naive Bayes classifier can be trained and then used to predict the polarity class (positive or negative) of a subjective text. Give the mathematical formulation of the Naive Bayes classifier as part of your answer. [20%]

(ii) Suppose you are given the following set of labelled examples as training data:

Doc	Words	Class
1	Amazing movie, the perfect way to make a sequel.	Positive
2	Hypnotic, surrealist, and most of all, maybe the most beautiful movie of the year.	Positive
3	Beautiful film. Well-paced; never felt it was overly long.	Positive
4	Visually stunning and amazing. A bit long, perhaps, but never boring.	Positive
5	Great plot but bad acting. Too long, boring in the middle.	Negative
6	Very boring, not entertaining, too artsy, plot holes galore, too long.	Negative
7	Visually beautiful but way too long and the soundtrack was annoying	Negative

Using as features just the adjectives (underlined words in the examples), how would a Naive Bayes sentiment analyser trained on these examples classify the sentiment of the new, unseen text show below?

Doc	Words	Class
9	Beautiful sets but too long and sooo boring.	???

Show how you derived your answer. You may assume standard pre-processing is carried out, i.e. tokenisation, lowercasing and punctuation removal. You do not need to smooth feature counts.

[40%]

4. Please answer each of the following multiple choice questions by filling in the column(s) corresponding to your selected answer on the provided answer sheet. Some questions allow multiple answers – these are clearly indicated. An incorrect answer will **not** attract a negative mark. For questions with multiple answers the number of incorrect answers you give will be subtracted from the number of correct answers you give yielding an integer d: if $d < 0$ then the mark assigned is 0; otherwise the mark is d/t of the marks available where t is the total number of possible correct answers. Each question counts for 8.5% of the total marks for this question. [100%]

a) **hexdump** is a Unix/Linux utility that prints out file contents as rows of 16 hexadecimal numbers, where each two hexadecimal digits represent one byte. By adding the `-C` flag then in addition to printing out file contents as hexadecimal numbers `hexdump` interprets the numbers as ASCII codes and prints out the corresponding characters. Suppose we run `hexdump` on a file and get this output:

```
> hexdump -C foo1
  00000000  52 65 64 20 51 75 65 65  6e 20 61 6e 64 20 57 68  |Red Queen and Wh|
  00000010  69 74 65 20 52 61 62 62  69 74 0a                 |ite Rabbit.|
  0000001b
```

(the numbers at the left of each row are hexadecimal byte offsets into the file).

We now run `hexdump -C` on a file containing just the 11 character string "`Turing Test`". What are the 11 hexadecimal ASCII codes we can expect to see?

(i) 74 75 72 69 6e 67 20 74 65 73 74

(ii) 53 75 72 69 6e 67 20 53 65 72 73

(iii) 54 75 72 69 6e 67 20 54 65 73 74

(iv) 54 75 72 69 6e 66 20 54 65 73 74

(v) 54 75 72 69 6e 67 20 74 65 73 74

(Hint: ASCII codes number sequentially from A-Z and a-z for both upper and lower case characters, though the ranges used for upper case and lower case are not adjacent.)

b) In the Unicode model of text encoding UTF-8, UTF-16 and UTF-32 are all instances of **character encoding forms**. A character encoding form is:

(i) a mapping from an abstract character repertoire to a set of integers (the codespace).

(ii) a mapping from a set of integers to a set of sequences of code units of specified width.

(iii) a mapping from a set of sequences of code units to a serialized sequence of bytes.

(iv) a mapping from a set of sets of glyphs (visually differing representations of the same abstract character) to a set of characters.

(v) a mapping from a set of sets of glyphs (visually differing representations of the same abstract character) to a set of integers (the codespace)

c) The **basic multilingual plane** in Unicode is a codespace region:

(i) of size 127 and with the same character code assignments as ASCII, covering those languages with the same alphabet as English.

(ii) of size 256 and with the same character code assignments as ISO/IEC 8859, covering most the languages of Europe.

(iii) of size $FFFF_{16}$ and covering most of the world's languages, but excluding the scripts of Eastern Asia (e.g. Chinese, Japanese (Kanji) and Korean).

(iv) of size $FFFF_{16}$ and covering almost all modern languages and a large number of symbols.

(v) of size $10FFFF_{16}$ and covering almost all modern languages and a large number of symbols. as well as most historic writing systems .

d) **Shannon's Source Coding Theorem** can be informally stated as "a string of length N made of symbols from alphabet X cannot be compressed into fewer than $N \times H(X)$ bits without possible loss of information". Here H is the entropy of the probability distribution of X. Suppose we have a string S = *fabdad* and know that $P(a) = 1/2$, $P(b) = 1/4$ and $P(d) = P(f) = 1/8$. Then the shortest lossless encoding of S is:

(i) 4.5 bits

(ii) 6 bits

(iii) 9.5 bits

(iv) 10.5 bits

(v) 18 bits

e) Suppose we are given:

1. a static zero order probabilistic character model that tells us $P(a) = 0.4$, $P(b) = 0.25$ and $P(d) = .2$ and $P(f) = 0.15$; and
2. the bit sequence $B = 111$.

We are told B has been coded from source string S using **arithmetic coding**, where the ranges assigned to each character are ordered with the most probable character assigned the lowest range, the second most probable character assigned the next to lowest range and so on till the least probable character is assigned the highest range (i.e. as done in the examples in the Text Processing lectures). Then the source string S is:

(i) faa

(ii) fab

(iii) fad

(iv) faf

(v) fafa

f) LZ77 is a popular compression method, used in common compression utilities such as *gzip*. The following shows some possible LZ77 encoder output (assuming the encoding representation presented in the lectures of the Text Processing module):

$$\langle 0,0,b\rangle\langle 0,0,a\rangle\langle 0,0,d\rangle\langle 3,3,b\rangle\langle 1,2,a\rangle\langle 1,2,d\rangle\langle 1,2,a\rangle\langle 9,2,a\rangle$$

What output would be produced by decoding the above representation?

(i) `badbbbbaaddaaaa`

(ii) `badbadbaaddaba`

(iii) `badbadbdadddadddddddda`

(iv) `badbadbbaaddabba`

(v) `badbadbbbaaadddabba`

g) Bing Liu's model for an opinion is expressed as a quintuple of the form:

$$(o_j, f_{jk}, X, h_i, Y)$$

Here X and Y are placeholders standing for missing elements. The missing elements are:

(i) X is so_{ijkl} the sentiment value of the opinion of opinion holder h_i towards feature f_{jk} of object o_j at time t_l; Y is the time t_l at which the opinion is held.

(ii) X is a_{jkl} the k-th aspect of object o_j towards which opinion holder h_i holds sentiment or feeling f_{jk} at time t_l; Y is the time t_l at which the opinion is held.

(iii) X is t_m is the time at which opinion holder h_i holds sentiment or feeling f_{jk} towards object o_j; Y is e_{jk} is the evidence opinion holder h_i puts forward for his feeling f_{jk} towards object o_j.

(iv) X is so_{ijkl} the sentiment value of the opinion of opinion holder h_i towards feature f_{jk} of object o_j with evidence e_{jk}; Y is the evidence opinion holder h_i puts forward for his sentiment so_{ijkl} towards object o_j.

(v) X is a_{jkl} the k-th aspect of object o_j towards which opinion holder h_i holds sentiment or feeling f_{jk} with evidence e_{jk}; Y is the evidence opinion holder h_i puts forward for his feeling f_{jk} towards object o_j.

h) Which of the following is **NOT** one of the comparative relation types identified by Bing Liu as occurring in comparative, as opposed to direct, opinions:

(i) gradable.

(ii) equative.

(iii) superlative.

(iv) contrastive.

(v) non-gradable comparitive.

i) Which of the following are **subjective** statements? (there may be multiple correct answers)

(i) Barack Obama has said nachos are his favourite snack food.

(ii) The new Panasonic combination microwave is brilliant for cooking quiche.

(iii) I believe the England will win the next World Cup.

(iv) My pizza arrived covered in jalapenos, which I had not ordered.

(v) The Blade Runner is a fabulous movie, stunning, well crafted with good action and good sci-fi story telling.

j) Consider the sentence:

After taking the tube from Camden to South Kensington John Handey visited the Victoria and Albert Museum.

Which of the following best represents the standard labelling that the BIO approach adopts to labelling named entities for training a named entity recogniser, assuming the three entity types person, organisation and location.

(i) [O After taking the tube from] [B$_{LOC}$ Camden] [O to] [B$_{LOC}$ South Kensington] [[B$_{PER}$ John Handey] [O visited the] [B$_{ORG}$ [I$_{PER}$ Victoria] and [I$_{PER}$ Albert Museum]].

(ii) After$_O$ taking$_O$ the$_O$ tube$_O$ from$_O$ Camden$_{B_{LOC}}$ to$_O$ South$_{B_{LOC}}$ Kensington$_{I_{LOC}}$ John$_{B_{PER}}$ Handey$_{I_{PER}}$ visited$_O$ the$_O$ Victoria$_{B_{ORG}}$ and$_{I_{ORG}}$ Albert$_{I_{ORG}}$ Museum$_{I_{ORG}}$.

(iii) [O After taking the tube from] [B$_{LOC}$ Camden] [O to] [B$_{LOC}$ South Kensington] [[B$_{PER}$ John Handey] [O visited the] [B$_{ORG}$ Victoria and Albert Museum].

(iv) After taking the tube from [B Camden] to [B South Kensington] [I John Handey] visited the [O Victoria and Albert Museum].

(v) After$_O$ taking$_O$ the$_O$ tube$_O$ from$_O$ Camden$_{B_{LOC}}$ to$_O$ South$_{B_{LOC}}$ Kensington$_{I_{LOC}}$ John$_{B_{PER}}$ Handey$_{I_{PER}}$ visited$_O$ the$_O$ Victoria$_{B_{PER}B_{ORG}}$ and$_{I_{ORG}}$ Albert$_{B_{PER}I_{ORG}}$ Museum$_{I_{ORG}}$.

k) The cost and effort of generating labelled training data for supervised approaches to relation extraction has led researchers to explore ways to reduce this cost and effort. One of these approaches is **distant supervision**. Distant supervision involves:

(i) paying relatively low cost remote workers to do the required annotation work via crowdsourcing websites such as Amazon's *Mechanical Turk* or *CrowdFlower*.

(ii) taking annotations produced in one domain and transferring them to the domain of interest, thereby reusing the annotations and saving the cost of annotating again from scratch.

(iii) using some information source (e.g. a manually built database) that contains information about relations holding between entities and using this information to label texts that mention these entities in the same sentence, resulting in a "weakly labelled" corpus, which can be used to train relation detectors and/or classifiers.

(iv) iteratively finding entity pairs in a labelled relation instance in a small labelled corpus, extracting patterns from the surrounding text and then finding matches of these patterns in a pool of unlabelled data, which are then labelled and added to the training data.

(v) finding entity pairs in a labelled relation instance in a small labelled corpus, then finding sentences in a pool of unlabelled data that mention the same entity pair and then, assuming these other sentences express the same relation as the initial sentence, labelling these sentences as relation instances and adding them to the training data.

l) The **DIPRE algorithm** is a classic example of a bootstrapping approach to relation extraction, where we wish to identify a large collection of entities standing in a particular relation (e.g.the author and title of a book) from a text collection. It works as follows:

(i) Start with a small set of seed tuples T and a document set D. Find all sentences S in documents in D that contain the entities from a seed tuple in T and then generate patterns P from S that match the left, centre and left context of the tuple entity mentions in S. Finally, apply the learned patterns P to D to find new tuples T', add them to T, and then repeat the whole process.

(ii) Start with a small set of reliable seed patterns P that match sentences that express the desired relation and a document set D. Find all matches of P in D and from them extract the entity mentions, form a new tuple and add it to the tuple store that is being built. The new tuple is then also be used to retrieve sentences from D that contain the tuple entities and from these new patterns are extracted, as in (i), and then iterates.

(iii) Start with a small set of seed tuples T and a document set D. Find all sentences S in documents in D that contain the entities from a seed tuple in T. Annotate the sentences with the entities and the relation expressed by it and use these annotated sentences to train any standard supervised relation detection system.

(iv) Start with a small set of reliable seed patterns P that match sentences that express the desired relationand a document set D. Find all matches of P in D and annotate the sentences with the entities and the relation expressed by it and use these annotated sentences to train any standard supervised relation detection system.

(v) Like (i) but as the learned patterns P may overgenerate, i.e. recognise incorrect tuples, introduce a measure of confidence on patterns which is calculated by computing the proportion of pattern matches for $p \in P$ that are correct. A confidence threshold is set and only patterns that meet or exceed the threshold are retained.

END OF QUESTIONS