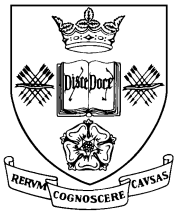

THE UNIVERSITY OF SHEFFIELD

DEPARTMENT OF COMPUTER SCIENCE

Autumn Semester 2005-06 2 hours

TEXT PROCESSING

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. a) Explain what is meant by the “bag of words” model which is used for various text processing tasks. [10%]

   b) Explain how the bag of words model can be used in Information Retrieval and word sense disambiguation. [15%]

   c) Write down Bayes Rule and explain how it is used in the naive Bayesian classifier for word sense disambiguation. [20%]

   d) What are the similarities between part of speech tagging and word sense disambiguation? Why are techniques for part of speech tagging not suitable for word sense disambiguation? [15%]

   e) Write a short Perl program that will read from a file on `STDIN` and that does nothing with the line that is read *until* it encounters a line containing the string `<BEGIN>`. Thereafter, each subsequent line will be printed (to `STDOUT`) *until* it encounters a line containing the string `<END>`. This line and any that follow are not printed. You may assume that the `<BEGIN>` and `<END>` strings will occur only once in a file. [20%]

   f) Assume that, in reading text from an HTML file, we can identify a URL (‘web address’) expression *within* a line of text provided that the string meets the following requirements:

      i. the string should fall between double quotes (")
      ii. it should start with `http://`
      iii. it should end with `.htm` or `.html`
      iv. between this start and end, there may be any sequence of characters except that ", < and > may not appear.

      Write a Perl regular expression that will match against strings that contain a URL expression, with the the URL string found being assigned to the variable $1. [10%]

   g) Specify what will be printed by each of the following pieces of Perl code:

      i. [5%]
```
$_ = 'abcde';
s/(\w)(\w)/$2$1/g;
print "$_\n";
```

      ii. [5%]
```
$_ = 'abcde';
while (s/(a)(\w)/$2$1/) { }
print "$_\n";
```

2. a) Describe Lesk’s algorithm for word sense disambiguation using dictionary definition overlap and explain its limitations. [25%]

   b) Describe two techniques which can be used to annotate text with word sense information automatically without the need for manual annotation. These techniques are used to provide data which can be used to train and/or test a word sense disambiguation system. Describe the disadvantages of each. [15%]

   c) What assumption is made by a naive Bayes classifier when it is used for word sense disambiguation? [10%]

   d) Explain the differences between direct, transfer and interlingua approaches to Machine Translation. [30%]

   e) Describe the approach used by the BLEU system for evaluation of Machine Translation systems. [20%]

3. a) What are stop words, in the context of an Information Retrieval system? Why are they generally not included as index terms? [10%]

b) Consider the following collection of three short documents:

Document 1: Tyger, tyger, burning bright
Document 2: Tyger sat on the mat
Document 3: The bright mat

Show how the documents would be represented in a vector space model for Information Retrieval, as vectors in which term weights correspond to term frequencies. Do not remove stop words, and do not use stemming in creating these representations. [10%]

c) The cosine coefficient can be used by Information retrieval systems to rank the relevance of documents in relation to a query. Compute the similarity that would be produced between Document 1 and the query "burning tyger" using this measure. [15%]

d) Explain what is meant by term weighting in Information Retrieval systems and why it is used. [15%]

e) Explain how tf.idf term weighting is used to assign weights to terms in Information Retrieval. Include the formula for computing tf.idf. [15%]

f) Show the weights which would be generated if tf.idf weighting was applied to Document 1 in the document collection of three documents shown above. [10%]

g) 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	×

Compute the (uninterpolated) average precision and interpolated average precision for this ranked set of documents. Explain your working. [25%]

4. We want to create a Perl program that will identify the tokens appearing in two text files (ignoring stop words), and then use this information to compute a measure of the similarity of the two files, using a metric that is detailed below.

   a) Write a Perl subroutine `tokenise` which takes a single input parameter, which is a string (corresponding to a line of input from a text file), and returns a list of strings for the *tokens* appearing in the input. For example, given the input string `"Today, John found anti-matter.\n"`, the subroutine would return the list `("today", "john", "found", "anti", "matter")`. Assume that tokens are any consecutive sequence of alphabetic characters, with any non-alphabetic characters (including punctuation) serving to separate tokens, as the example illustrates. We are not interested in case distinctions, e.g. `John` vs. `john`, so the tokens should be mapped to lowercase. [25%]

   b) Write a Perl subroutine `readStoplist` that takes as its single parameter the name of a file that contains a list of stopwords, and which reads in and stores the stopwords for subsequent use. As part of your solution, you will need to determine a good data structure for storing the words, i.e. one allowing them to be efficiently accessed. Define also a subroutine `isaStopword` which when given a string for a token, determines whether or not it is a stopword, i.e. so `isaStopword($w)` returns `1` if the value of `$w` is a stopword, and returns `0` otherwise. [25%]

   c) Write a Perl subroutine `countTokens` that takes a file name as its single parameter, and which counts the occurrences of tokens in the file, ignoring stopwords, and returns its results as a hash, i.e. a hash in which each key is a token appearing in the file, for which the associated value is a positive integer indicating how many times the token appears. Your definition should use the subroutines specified in parts (a) and (b) of the question (irrespective of whether you were able to provide working definitions for them). [25%]

   d) Write a Perl *program* which requires three arguments on the command line, of which the first is the name of the stopword file, and the others are the names of two files that are to be compared, i.e. which might be invoked as:

      ```
      perl myprogram.pl stoplist.txt file1.txt file2.txt
      ```

      The program should use the previously specified subroutines to load the stopwords, and to count the tokens in the other two files, so that this information can be used to compute a similarity score for the two files. The measure to be computed is the proportion of terms (i.e. distinct tokens) appearing in *either* file that appear in *both* files. Note that this particular measure makes no use of the specific *counts* that have been made of token occurrences, only whether or not a specific distinct token appears in a given file or not. If A is the *set* of terms (irrespective of counts) appearing in one file, and B the corresponding set for the other file, then this measure can be expressed as:

      $$\frac{|A \cap B|}{|A \cup B|}$$

      The calculation of this measure can be done either in a subroutine or in the main body of the code, as you prefer. Your program should print the similarity value of the two files to `STDOUT` before terminating. [25%]

END OF QUESTION PAPER