INF4820: Algorithms for AI and NLP (Fall 2009) — Final Exam —

General Instructions

- Please read through the complete exam once before starting to answer questions. About thirty minutes into the exam, the instructor will come around to answer questions of clarification.
- As discussed in class, the exam is given in English, but you are free to answer in any of English, *Bokmål*, or *Nynorsk*.

1 Language Modeling and Markov Models (35 points)

(a) How exactly does an n-gram language model compute the probability $P(s)$ for a string $s = w_1^m$, assuming a first-order n-gram model (i.e. a *bigram* model)? State the central assumption made in this modeling approach.

(b) What is the formula for estimating n-gram probabilities from a training corpus of running text (assuming a straightforward maximum likelihood estimate)? Show the calculations for uni- and bi-grams.

(c) Data sparseness often leads to problems when using maximum likelihood estimates. Discuss these problems in relation to language modeling and n-gram probabilities. What is the general idea common to various smoothing schemes for language modeling?

(d) In a few sentences, explain how Hidden Markov Models extends on simple n-gram models.

(e) Given a sequence of words w_1^m, an HMM-tagger seeks to find the most probable sequence of tags t_1^m:

$$\hat{t}_1^m = \arg\max_{t_1^m} P(t_1^m | w_1^m)$$

Rather than maximizing this expression directly, however, we make some simplifying assumptions and apply Bayes' rule in order to arrive at a more manageable maximization problem. Show the formula we actually seek to maximize in an HMM-model. Include the rewriting steps and state the simplifying assumptions made along the way.

2 Automatic Categorization and Vector Space Models (35 points)

(a) Classification and clustering represent two different approaches to the task of automatic categorization. Explain how they differ.

(b) When working in the context of vector space models, it is common to normalize the feature vectors so that their Euclidean length is 1. In a few sentences, discuss the benefits of using normalized vectors.

(c) Discuss the main differences between k Nearest Neighbor classifiers and Rocchio classifiers.

(d) In hierarchical agglomerative clustering, we need to compare the similarity between clusters in order to decide which clusters to merge at each iteration. Briefly describe the different criterions that are commonly used for measuring cluster similarity.

3 Common Lisp (30 points)

(a) Consider the following Lisp expressions.

```
(defun foo (x y)
  (setf (rest y) (cons (first y) (rest y)))
  (setf (first y) x)
  y)

(defun bar (x)
  (setf (first x) (second x))
  (setf (rest x) (rest (rest x)))
  x)

(setq baz '(a b a))

(foo 'a baz)

(bar baz)
```

Explain the behavior of the functions `bar()` and `foo()` as defined above. What is the value of `baz` after the first call to `foo()`? What is the value of `baz` after the call to `bar()` that follows? Show the effect of the function calls by using "box notation" to keep track of `cons()` cells.

(b) Superficially, the behavior of our functions `foo()` and `bar()` is similar to that of a pair of built-in Common-Lisp macros (assuming that the list arguments are non-nil). Which macros? Describe how the behavior of our functions differs from these.

(c) Write a function `memoize()` that takes a function as an argument. It should return a new anonymous function that is a memoized version of the given function. (If you feel you have to make any assumptions about the particular functions that can be memoized, please state them.)

(d) In this problem you will write a function that operates on binary trees. A tree will be represented as a nested list, like the one bound to `foo` in the example below.

```
(setq foo
  '(A (C (D (E (B () C)
              F)
           (F C C))
        F)
      (A (F D ())
        (B C A))))
```

The first element of each nested list is the root of a (sub-)tree, while the two following elements are its children. As you can see, each child can be one of three things: (i) An atomic value (a symbol), representing a leaf node, (ii) an empty list, representing a missing child (a unary branch), or (iii) a new list representing a subtree. (You might find it helpful to make a drawing of the tree represented by `foo`.)

Write a (non-destructive) recursive function `trim()` that takes a symbol and a binary tree as arguments, and returns a new tree where all subtrees having the given symbol as its root have been excluded. For example:

```
? (trim 'f foo)
→ (A (C (D (E (B NIL C) NIL) NIL) NIL) (A NIL (B C A)))
```

If you find it too difficult to define the function in Lisp, try to write the function using pseudo-code instead.