INF4820: Algorithms for AI and NLP (Fall 2014) — Final Exam —

General Instructions

- Please read through the complete exam once before starting to answer questions. About thirty minutes into the exam, our colleague Erik Velldal will come around to answer any questions of clarification (including English terminology).
- As discussed in class, the exam is only given in English, but you are free to answer in any of *Bokmål*, English, or *Nynorsk*.
- To give you an idea about the relative weighting of different questions during grading, we have assigned points to each of them; the total sum of points is 100.

1 Common Lisp (5 + 5 + 5 = 15 Points)

(a) How many elements are contained in the list returned by the following expression? Please draw the internal structure of the list, using the 'box and pointer' notation, i.e. with explicit `cons()` cells and `first()` and `rest()` values. What would happen if we used a function like `length()` to count the number of elements?

```
(let ((foo (list 42)))
  (setf (rest foo) foo))
```

(b) At various points in the class we talked about data structures to index and efficiently retrieve information, for example the so-called transition and emission matrices in the context of HMMs. Recall that, for the emission matrix, we typically looked up probabilities for a pair of the current state (encoded as an integer) and the current word (a string). Given this scenario, briefly discuss the relative strengths and weaknesses of lists, arrays, and hash tables for associative retrieval, i.e. the look-up of values associated with keys that (conceptually) pair an integer with a string. Reflect on the number of distinct values (in a typical HMM, say one used for PoS tagging) along both dimensions, and further take into consideration whether you expect the emission matrix to be densely or sparsely populated, i.e. what proportion of combinations (state plus word) out of the set of possible combinations will typically be used.

(c) Write a non-destructive function `ditch()` that takes a symbol as its first and a list of atoms as its second argument; `ditch()` removes *all* occurrences of its first argument in the list, e.g.

```
? (ditch 'c '(a b c d e c))
→ (A B D E)
? (ditch 'f '(a b c d e c))
→ (A B C D E C)
```

What does it mean for a function to be non-destructive? Can you implement a variant of `ditch()` that does not allocate new memory, i.e. avoids creating new `cons()` cells?

Note that we are not primarily concerned with specific details of Lisp syntax here. If you find that easier, feel free to use elements of 'pseudo code' in your function definition, as long as it is clear how exactly everything will work. Give a brief informal summary of the basic principles (e.g. in terms of base case(s) vs. recursive cases) and general approach of your implementation.

2 Classification and Clustering (8 + 7 + 9 = 25 Points)

(a) Describe the different types of contexts in the distributional approach to lexical semantics. Discuss how different context types can impact the similarity between words.

(b) The diagram in Figure 1 shows the distribution of objects in two-dimensional space. The **X** points represent members of class X, and the **O** points represent members of class O. Our task is to classify the example denoted with **?** into one of the two classes.

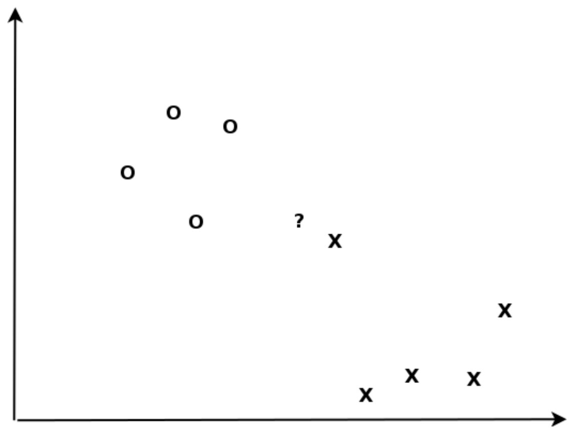

Figure 1: Distribution of objects in two-dimensional space.

(i) What class will the Rocchio classifier assign to the **?** instance?

(ii) What class will 1NN classifier assign to the **?** instance?

Motivate your answers in a couple of sentences for each case.

(c) Describe different techniques for linkage in bottom-up hierarchical clustering.

3 Linear Structures (5 + 5 + 5 + 5 + 5 = 25 Points)

(a) What is the formula for estimating n-gram probabilities in a language model from a training corpus of running text; show the calculations for bi- and tri-grams.

(b) In a few sentences, discuss the problem of data sparseness and its consequences for a naïve application of an n-gram language model. What is the general idea common to various smoothing schemes. Explain the particular method of add-one (or Laplace) smoothing and show how the estimation of bi-gram probabilities changes.

(c) Assume the following part-of-speech tagged training 'corpus' of just one sentence:

still	,	time	s	move	is	being	received	well	,	once	again	.
RB	,	NNP	POS	NN	VBZ	VBG	VBN	RB	,	RB	RB	.

Ignoring smoothing and making the standard simplifying assumptions for a naïve bi-gram HMM (including the assumption that the training corpus provides the full inventory of distinct tags and complete vocabulary), calculate the following:

(i) For each tag t, the probability of t following the tag RB, i.e. $P(t|\text{RB})$

(ii) The emission probabilities $P(\textit{move}|\text{NNP})$, $P(\textit{move}|\text{NN})$, and $P(\textit{well}|\text{RB})$.

(d) Assume further that for each tag t, $P(t|\texttt{<s>}) = P(t)$; in one sentence, what does it mean to make this assumption. Construct part of the Viterbi trellis for tagging the utterance *once again , time .* Rather than calculating all values, indicate the total size of the trellis and the computations for filling in the first two columns.

(e) In a few sentences, summarize the key points of the Viterbi algorithm. What is the interpretation of each cell in the trellis? What is the complexity of the algorithm, i.e. the number of computations performed in relation to (i) the length of the input sequence and (b) the size of the tag set? Very briefly, sketch an alternative, naïve method for computing the most probable tag sequence t_1^n, given an input string w_1^n; state how the Viterbi algorithm improves over this approach.

4 Hierarchical Structures (5 + 5 + 5 + 5 + 5 = 25 Points)

Consider the language defined by the following grammar (Mirror English):

S → VP NP	NP → *kim*
VP → PP VP	NP → *oslo*
NP → PP NP	NP → *snow*
VP → NP V	V → *adores*
PP → NP P	P → *in*

(a) For each of the following strings, say whether they are part of the language generated by the grammar or not.

(i) *oslo in snow adores kim.*

(ii) *kim adores snow in oslo.*

(iii) *snow adores in oslo kim.*

(b) Semi-formally, what does it mean for a string w_1^n to be part of the language generated by a grammar G; given one specific string w_1^n, how is the set of wellformed parse trees from G characterized?

(c) Is there anything in the grammar of Mirror English that would lead to computational problems for the naïve top-down parser that we discussed briefly (the recursive-descent parser)? Is the grammar of Mirror English suitable for use with the CKY parser? If so, why? If not, why not?

(d) In a few sentences, discuss the concept of *local ambiguity* that is at the core of the CKY and generalized chart parsers. Do any of the examples (i) to (iii) from part (a) above exhibit local ambiguity in Mirror English? If so, where exactly, what constituent category is involved, and what would happen in chart parsing?

(e) Recall very briefly the role of the chart in the generalized chart parser. What units of information are associated with each edge in the chart? How are active edges different from passive ones (feel free to use the 'dating' metaphor, if you find it useful), and what is the general form of the *fundamental rule* in chart parsing?

5 Structured Probabilistic Modelling (5 + 5 = 10 Points)

(a) Following is a probabilistic variant of the grammar of Mirror English:

S $\rightarrow$ VP NP [1.0]	NP $\rightarrow$ *kim* [0.2]
VP $\rightarrow$ PP VP [0.2]	NP $\rightarrow$ *oslo* [0.2]
NP $\rightarrow$ PP NP [0.2]	NP $\rightarrow$ *snow* [0.4]
VP $\rightarrow$ NP V [0.9]	V $\rightarrow$ *adores* [0.2]
PP $\rightarrow$ NP P [1.0]	P $\rightarrow$ *in* [0.2]

Assuming 'S' as the start symbol and sets of non-terminal and terminal symbols as implicitly given by the rules above, is this grammar a valid PCFG? If so, in what sense? If not, why not?

(b) Whether or not this grammar is a valid PCFG, determine the probability of the sentence *oslo in snow adores kim* according to the rules above. In what sense are the probabilities associated to the productions of a PCFG conditional, i.e. how can we alternatively write P(VP $\rightarrow$ PP VP), and why does this make sense in PCFG parsing?