# INF4820: Algorithms for AI and NLP (Fall 2016)
# — Final Exam —

## General Instructions

- Please read through the complete exam once before starting to answer questions. About thirty minutes into the exam, one of the instructors will come around to answer any questions of clarification (including English terminology).
- Please follow the instructions closely. Most of the questions ask for short answers. When a maximum length is given (e.g. 'in no more than two sentences'), please try to stick to those limits.
- 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 (3 + 4 + 5 + 6 = 18 Points)

(a) For each of the following lists, make the underlying structure explicit by either showing the list in 'box notation' or (ii) providing a Common-Lisp s-expression to construct the list, using exclusively the `cons` function, the atoms that are elements of the list, and `nil`.

(i) `(1 2 3)`

(ii) `((1) (2 3))`

(b) Write a two-place function `ditch` that takes an atom as its first and a list as its second argument; `ditch` removes *all* occurrences of the atom in the (top-level) 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)
```

Provide a brief informal summary of the basic principles of your solution, e.g. in terms of base case(s) vs. recursive cases. Note that we are not primarily concerned with specific details of Lisp syntax here. If you found 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.

(c) Develop a revision of `ditch` that does not generate new `cons` cells, i.e. avoids all calls to functions like `cons`, `append`, or `list`.

(d) At various points in the class we talked about data structures to index and efficiently retrieve information, for example the 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 observation (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 for English 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 token) out of the set of possible combinations will typically be used.

## 2 Vector Space Models (8 + 6 + 8 = 22 Points)

(a) Semantic spaces are based on a spatial metaphor which allows us to represent words as vectors in $n$-dimensional spaces. In a few sentences explain:

(i) What is the main hypothesis we make to model semantics in vector space models?

(ii) How can we measure semantic similarity in this spatial metaphor? Specify two measures and provide the formulas that define them. Make sure to highlight the shortcomings of these two measures, if any, and explain a way to overcome them.

(b) Figure 1a and Figure 1b show two distributions of objects in two-dimensional space. The solid circles represent members of class $X$ and the stars represent members of class $Y$.

(a) (b)

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

Given the two classification methods we learned in this course, $k$NN and Rocchio:

(i) Which classification method is better fit to classify the objects in Figure 1a

(ii) Which classification method is better fit to classify the objects in Figure 1b

Motivate your answers in a couple of sentences for each. Make sure to comment on the properties of the classification method(s) you choose.

(c) In a couple of sentences explain how we can evaluate a classifier. Reflect on how the distribution of objects, in Figure 1a and Figure 1b, might influence your choice of the evaluation measure(s)?

## 3 Linear Structures (5 + 6 + 5 + 7 + 7 = 30 Points)

We want to train a language model on the following 'corpus' which consists of imperative sentences, taken from George Orwell's essay *Politics and the English Language*:

> *Never use a long word where a short one will do. Never use the passive where you can use the active. Break any of these rules sooner than say anything outright barbarous.*

We assume a tokenization approach that splits on white-space and punctuations, and a pre-processing step that converts all words in sentence-initial position to lowercase.

(a) The first step in training a language model is learning the model's parameters: What is the formula for estimating $n$-gram probabilities (assuming a straightforward maximum likelihood estimates)? What is the unigram probability $P(never)$ and the bi-gram probability $P(never|use)$? What is the practical motivation for introducing the special start- and end-of-sentence symbols `<s>` and `</s>`?

(b) Explain in a couple of sentences (and at least one equation) how an $n$-gram model estimates the probability $P(s)$ for a string $s = w_1^n$? Explain why it is difficult to obtain reliable probability estimates for long $n$-grams. What assumption is made when using $n$-gram language models to estimate the probability of sentences? Name one practical problem that can be solved (to a certain degree) using language models.

(c) Assuming a bi-gram language model trained on the corpus above, explain why the following sentence is impossible according to our language model:

*I never use the passive.*

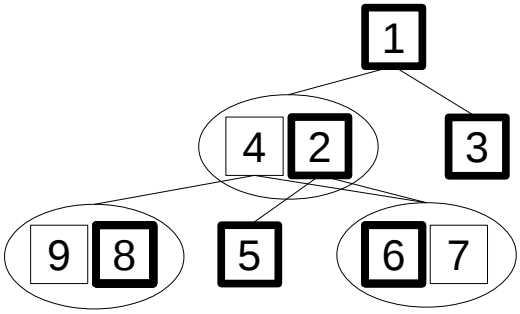
*Smoothing* can be used to remedy our model: Explain what is the general idea behind smoothing. What will be the probability of the above sentence (*I never use the passive.*) after applying Laplace smoothing? Please feel free to leave the final probability in the form of a fraction.

(d) How does an HMM compare to a language model? In a few sentences, sketch the differences between the two approaches and also comment on commonalities, if any. Is it possible to use HMMs to estimate the probability $P(s)$ for a string $s = w_1^n$; if so, how is this usage of HMMs different from the task of part of speech tagging?

(e) In the course, we defined the Viterbi algorithm as a dynamic programming algorithm. How exactly does the Viterbi algorithm achieve dynamic programming? Assuming a bi-gram HMM ($n = 2$), a state set of only $\{C, H\}$, and an observation sequence $s = w_1^k$, what is computed by the Viterbi algorithm? Using the general mathematical notation for the different types of probabilities (i.e. without specific numeric values), sketch the computation of the first two columns of the trellis used in the Viterbi algorithm.

# 4 Hierarchical Structures (5 + 5 + 4 + 9 + 7 = 30 Points)

Consider the following parse forest (which may look familiar):




(a) The parse forest is a compact representation of a set of complete trees. Please draw all complete trees represented by edges 2 and 4. How many complete trees are encoded in this forest?

(b) In parsing natural languages, we have introduced the key notion of *local ambiguity*. In one sentence, please give a definition of local ambiguity. Is there anything in the hypothetical parse forest above indicating local ambiguity? How do the CKY and generalized chart parsers take advantage of local ambiguity? What is the overall effect of this technique on parsing complexity in this algorithms?

(c) We have criticized the CKY algorithm for its restriction to context-free grammars in so-called Chomsky Normal Form (CNF). Compared to general (i.e. unrestricted) context-free grammars, how is the the sub-set of CNF grammars characterized? Can you suggest one example sentence in English that might be not linguistically straightforward to analyze in a CNF grammar?

(d) Assume the following grammar for a sub-set of English, whose start symbol shall be NP:

NP → Det N
N → N N

Det → *the*
N → *kitchen* | *gold* | *towel* | *rack*

Please draw the complete CKY parse table, indexed by sub-string start and end positions (as is customary), for the 'sentence':

The kitchen gold towel rack

(e) As we moved beyond the CKY parser, we introduced two different types of *edges* that are recorded in the parse tables of the generalized chart parser. Very briefly, name the two edge types and explain how they differ. Which of the two types of edges in generalized chart parsing corresponds more closely to the entries of the CKY parse table? In no more than three sentences, what is the purpose of the other edge type, and which benefits does it bring to the generalized chart parser, when compared to the CKY algorithm?