1 Finite-State Technology (10 + 10 + 10 + 30 = 60 Points)

(a) Draw the finite-state automaton (FSA) that corresponds to the regular expression \((a|b^*)^+\).

(b) For each of the following strings, say whether it is part of the language recognized by this automation or not: (i) aaa, (ii) bab, (iii) abab, (iv) aabaab.

(c) Recall the equivalence relation holding between FSAs with output symbols associated to transitions, compared to ones with output symbols associated to states. Which of the two points of view did you choose in part (a) above? Draw the equivalent automaton, this time using the alternate point of view.

(d) Assume an electronic corpus of running text. In a few sentences, summarize the tasks of tokenization and sentence segmentation. What is the relation between these two tasks, if any? Give at least two examples of common challenges (i.e. potential sources of errors) encountered in these tasks. Assuming a finite-state approach, sketch regular expressions that could be employed in identifying token and sentence boundaries.

2 Language Modelling (10 + 20 + 20 + 30 = 80 Points)

(a) How exactly does an \(n\)-gram language model estimate the probability \(P(s)\) for a string \(s = w_1^n\), assuming a first-order \(n\)-gram model. Discuss the central assumption made in this modelling approach; what is the common name of said assumption?

(b) What is the formula for estimating \(n\)-gram probabilities from a training corpus of running text; show the calculations for first- and second-order \(n\)-grams.

(c) 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.

(d) What is the definition of perplexity in the context of \(n\)-gram language models? How can perplexity help compare and evaluate language models?

3 Statistical Parsing (20 + 50 + 50 + 50 = 170 Points)

Assume the following tree taken from Section 23 of the Penn Treebank (PTB):

```
S
  |   VP
  |   NP-SBJ
  |   |   NP
  |   |   NNP
  |   |   Hooker
  |   |   POS
  |   |   philosophy
  \   |   |   QUOTE
  \   |   \   s
  \   \   |
  \   \   VP
  \   |   S-PRD
  \   |   VBD
  \   \   was
  \   |   \   TO
  \   |   |   VP
  \   |   |   VB
  \   |   |   CC
  \   |   |   VB
  \   |   |   build
  \   |   |   and
  \   |   |   sell
```
(a) Which of the elements of the PTB annotation will be ignored in typical statistical parsers, e.g. those of Charniak (1997) or Charniak (2000)?

(b) Assuming this tree was our complete training corpus, extract part of the context-free grammar (CFG) implicitly defined by the training data and estimate rule probabilities for a standard PCFG. For the purpose of this exercise, focus on rules deriving constituents of categories ‘S’ and ‘VP’.

(c) In no more than two sentences, discuss the concept of head lexicalization and how Collins (2003) (which is the condensed write-up for publication in Computational Linguistics of his 1999 dissertation) would apply it to our example tree. For each constituent in the tree, annotate the branch corresponding to its head daughter. In a few sentences, summarize the notion of Markovization applied to PCFG rules in the Collins parser. Pick one of the PCFG example rules from part (a) above that has at least three daughters. Sketch the generative process and associated probability estimates that Collins would use in his parser instead of the original PCFG rule.

(d) In a few sentences, discuss the concept of (grand-)parenting and show how it changes the form of CFG grammar rules and probability estimates for one or two examples from the grammar extracted in part (a) above. What is the general motivation for the use of (grand-)parenting; can you think of examples where the additional distinctions made might be beneficial to the parser?

4 Dependency Grammar (50 + 50 + 50 = 150 Points)

(a) Convert the PTB tree from our example above into an unlabelled dependency graph. In a sentence or two, discuss the general conversion procedure, and what one would have to do further to derive dependency labels.

(b) Recall the wellformedness conditions on dependency graphs discussed by Nivre et al. (2006), specifically the single head and the projectivity conditions. Give a one-sentence, informal definition of each condition. Does the dependency graph that you constructed in part (a) above obey either of the two conditions?

(c) Recall the deterministic parsing algorithm proposed by Nivre et al. (2007). In terms of its core actions—viz. SHIFT, REDUCE, LEFT-ARC(r), and RIGHT-ARC(r)—sketch the first ten parser actions that result in the derivation of the dependency graph above.

5 More High-Level (70 + 70 = 140 Points)

(a) In what sense are the parsers of, for example, Charniak (1997) and Collins (2003) commonly characterized as generative probabilistic models? Abstractly, what is the probability distribution assumed in these models, and what kind of training data is required to estimate their parameters?

(b) Contrasted with your discussion of part (a) above, which part of the parsing model of Charniak & Johnson (2006) can be characterized as discriminative? What is a general property of discriminative probabilistic models, and how can they be trained? Contrasted with earlier parsers, what is the main advantage in discriminative models? Which of the two approaches would be better suited as a language model, i.e. to replace an n-gram language model in, for example, speech recognition?