11-711 Algorithms for NLP
Sample Final Exam

Fall 2012

- Before you go on, **write your name at the space provided on the bottom of this page and every page of the exam**.
- There are 15 pages in this exam (including this page). Verify that you have a complete copy.
- Write up your answers following each question in the exam. Adequate space has been provided.
- If you really feel that you you need more space you may continue on the reverse side, but you must clearly mark the page so that we know your answer continues on the reverse side.
- The exam is open book and open notes and is worth a total of 100 points.
- It should require three hours to complete. Budget your time accordingly.
- Keep answers short and to-the-point. Concise, direct answers will receive more credit than longer, essay-like answers.
- If you find a question ambiguous, state your assumptions precisely, and proceed.

Good Luck!

Student Name: _____

1 Formal Language Theory (30 minutes/18 points)

Let us consider the following restricted class of context-free grammars LG, where every rule in the grammar has one of the following two forms:

$$\begin{aligned} A &\rightarrow aB \\ A &\rightarrow a \end{aligned}$$

where $a \in T$ (a is a terminal symbol), and $A, B \in V$ (A and B are non-terminals).

1. Prove that the class of LG grammars can only recognize **Regular** languages, by showing a general construction that given an LG grammar G, constructs an NDFSA A, such that $L(A) = L(G)$. Argue informally why the construction is correct, and **explicitly** state how you would formally prove that it is correct. You do NOT need to write the details of the formal proof.
   (8 points)

2. Consider the following grammar G that describes simple arithmetic expressions over the numbers $\{0, 1\}$.

$$
\begin{aligned}
S &\rightarrow + A\ S && (1)\\
S &\rightarrow * A\ S && (2)\\
S &\rightarrow 0 && (3)\\
S &\rightarrow 1 && (4)\\
A &\rightarrow 0 && (5)\\
A &\rightarrow 1 && (6)
\end{aligned}
$$

Note, for example, that the strings $+1*10$ and $*1+1+01$ are in $L(G)$, but $0+1*1$ is not.

Convert the grammar G into an equivalent LG grammar G', such that $L(G') = L(G)$. List the rules of the resulting new grammar G'.
(5 points)

3. Consider the class of languages that can be recognized by the restricted class of context-free grammars LG. For each of the following statements, answer whether the statement is **true** or **false**, and give a brief explanation for your answer.
(5 points, 1 point each)

   (a) If L is a language that can be generated by an LG grammar, then L is a context-free language.

   (b) If L is a language that can be generated by an LG grammar, then L must be a finite language.

   (c) If L is a regular language, then there exists some LG grammar G such that $L = L(G)$.

   (d) There exists a context-free language that cannot be generated by an LG grammar.

   (e) The set of languages that can be recognized by an LG grammar is a proper subset of the set of context-free languages.

2 Earley Parsing (30 minutes/18 points)

Several parsers in recent years have found it useful to use the following extended version of context-free grammars, which allows specifying limited forms of regular expressions on the right-hand side (RHS) of context-free production rules. Formally, an extended CFG G is defined as $G = (T, V, P, S)$, where T, V and S are, respectively, a set of terminal symbols, a set of non-terminal symbols, and a starting non-terminal. This is the same as a standard CFG. Productions P, however, consist of a left-hand side (LHS) non-terminal $A \in V$, and a RHS, that is a finite string of zero or more elements $E_1 E_2 \cdots E_k$, where each E_i is one of the following:

1. a terminal symbol $a \in T$
2. a non-terminal $B \in V$
3. $(B|C)$, denoting a disjunction of B or C, where B and C are nonterminals from V
4. $[B]$, denoting an optional $B \in V$
5. B^*, denoting zero or more occurrences of $B \in V$

For example, the rule $A \rightarrow (B|C)\ D^*$ denotes that the RHS starts with either B or C, followed by zero or more Ds.

1. We would like to modify the Earley parsing algorithm to handle this extended version of context-free grammars with limited forms of regular expressions on the RHS of context-free production rules. For simplicity, assume that the extended grammar formalism allows **only cases 1, 2 and 3: elements on the RHS of a rule are either a terminal b, a non-terminal B, or $(B|C)$, denoting a disjunction of the two non-terminals: B or C.**

   Modifying the algorithm to directly handle extended rules requires revising the operators. Modify the original operators of the Earley algorithm to correctly handle grammar rules of the extended-CFG formalism. For your convenience, the original algorithm for each operator is provided below. Write your modified algorithm in the space below that. Briefly explain why the modified algorithm is correct.
   (12 points)

Earley's three operators:

- PREDICT: If state $[A \rightarrow X_1 ... \bullet C ... X_m, j] \in S_i$ then for every rule of the form $C \rightarrow Y_1 ... Y_k$, add to S_i the state $[C \rightarrow \bullet Y_1 ... Y_k, i]$
  Modified Version:

- COMPLETE: If state $[A \rightarrow X_1 ... X_m \bullet, j] \in S_i$ then for every state in S_j of form $[B \rightarrow X_1 ... \bullet A ... X_k, l]$, add to S_i the state $[B \rightarrow X_1 ... A \bullet ... X_k, l]$
  Modified Version:

- SCAN: If state $[A \rightarrow X_1 ... \bullet a ... X_m, j] \in S_i$ and the next input word is $x_{i+1} = a$, then add to S_{i+1} the state $[A \rightarrow X_1 ... a \bullet ... X_m, j]$
  Modified Version:

- Informally explain why your modified algorithm is correct.

2. Does the revised Earley algorithm still have the property that at most **one** operator can apply to any item? Briefly explain your answer, or demonstrate with an example. **(3 points)**

3. Do the modifications you proposed have any consequences on the $O(n^3)$ worst-case time complexity of the algorithm? Briefly explain why, or why not. **(3 points)**

3 Dependency Parsing with Labels (60 minutes/30 points)

In class, you learned how the problem of finding the maximum-scoring dependency parse for a sentence can be accomplished using dynamic programming (if only projective trees are considered valid parses) or the Chu-Liu-Edmonds (CLE) algorithm (if nonprojective trees are also considered valid). Let the words in the input sentence be denoted $w_1 w_2 \dots w_n$, and let the score of the attachment of word w_j as a child of word w_i be $s_{j,i}$. We let $w_0 = \$$ denote the root symbol, so that $s_{j,0}$ is the score of attaching word w_j to the root. Let $\boldsymbol{s}$ denote the entire collection of attachment scores. If, in a given tree, we let π_j denote the index of w_j's parent, then the score of the tree is:

$$\sum_{j=1}^{n} s_{j,\pi_j} \tag{17}$$

Note that the algorithms we discussed assume that the score of a parse tree is defined by the sum of the scores of directed attachments in the tree.

In class, we only discussed in detail the case of *unlabeled* dependency parsing, though dependencies are usually described linguistically as having types or labels. Of course, the set of dependency labels used by a given parser are a design decision. One kind of label captures syntactic relations like the one held between a verb and its subject or a preposition and its complement. Another kind of label can be used to encode phrase-structure syntax (i.e., what is modeled by context-free grammars).

1. Describe how to transform any phrase-structure tree into a labeled dependency tree *without* losing any information about the original tree. Assume that the phrase-structure tree already has head-marking (i.e., every nonterminal node is annotated with the index of its head child). Do not explain how to recover the parent for each word; we assume you already know how to do this. It may help you to draw an example to work out the problem, but your solution should not make use of an example; it should explain how to construct the labels for the dependency tree, given any head-marked phrase-structure tree. For full credit, your solution should make use of the minimal set of labels when applied to a treebank (i.e., the same labels should be reused to the greatest extent possible in different trees).
   (7 points)

2. Describe how to recover the phrase-structure tree from a labeled dependency tree constructed according to your answer to part 1. In your solution, let π_i denote the index of the parent of w_i, ϕ_i denote the number of children of w_i, $\boldsymbol{\xi}_i = \langle \xi_{i,1}, \ldots, \xi_{i,\phi_i} \rangle$ denote the sequence of w_i's children, and L_i be the label of the attachment of w_i as a child of w_{π_i}. You may not need to use all of these symbols.
(3 points)

3. One solution to labeled dependency parsing is to solve it in two stages. First, find the best unlabeled dependency parse using CLE or dynamic programming with the scores $\boldsymbol{s}$. Then use a separate model designed to label the edges of that tree. This is called a "pipeline." You will define an algorithm that finds the best labeling of a tree. Letting L_i again denote a label for the attachment of w_i to its parent w_{π_i}, let the score of labeling of a (fixed) tree be defined by:

$$\textit{labelingscore}(L_0, L_1, \ldots, L_n) = \sum_{i=1}^{n} \sigma(i, L_i, L_{\pi_i}) \tag{18}$$

That is, we score each word's attachment label, taking into consideration the label that its parent's attachment receives. (For completeness, let the label of the root's "attachment" be $L_0 = \emptyset$, a special null symbol.)

Provide a dynamic programming algorithm for finding the best labeling of the tree assuming the local scoring function σ is given. Let $\mathrm{t}(L, i)$ be the score of the best possible labeling for the subtree rooted at w_i assuming that w_i's attachment to its parent is labeled L. Below are the base case and the definition of "goal" (the score of the best complete labeling for the tree). You need to state the recurrence and give a description in English of how these equations can be used to find the best labeling.

$$\begin{aligned} \mathrm{t}(L, i) &= 0 \text{ whenever } w_i \text{ is a leaf} \\ \text{goal} &= \max_{L} \sum_{i:\pi_i=0} \mathrm{t}(L, i) + \sigma(i, L, \emptyset) \end{aligned}$$

(Note that in standard dependency parsing, there is only one word that attaches to the root (\$), so the summation for calculating goal will only include one value of i. The equations above are slightly more general, allowing multiple attachments to \$.)
(10 points)

4. We next consider a single-pass algorithm that finds a labeled tree all at once. Here is a sketch of how the algorithm works:

   (a) Construct a directed graph. (This is the part you will describe in more detail below.)

   (b) Run the CLE algorithm find the best scoring directed spanning tree of the graph.

   (c) Apply zero or more operations to the tree given by CLE. (This is the part you will describe in part 5.)

   Assume you are given the scores of all possible *labeled* attachments, denoted $s_{j,i,L}$ (for attaching child w_j to parent w_i with label L), and an implementation of the CLE algorithm. Note that there may be multiple labels possible for any given attachment! Describe how to construct the graph in step 4a.
   (3 points)

5. Describe how the output of CLE can be transformed in step 4c above to produce the desired output.
   (3 points)

6. Describe the advantages and disadvantages of the two methods you defined above (pipeline and single-pass).
   (4 points)

4 CFG with Unification Constraints (15 minutes/8 points)

Consider the following context-free grammar that recognizes simple sentences such as "*the students teach the students*":

Grammar:

```
S --> NP VP
NP --> DET N
NP --> N
VP --> V
VP --> V NP
```

Lexicon:

```
DET --> the
N --> student
N --> students
V --> look
V --> looks
V --> teach
V --> teaches
```

1. While this grammar parses all grammatical sentences for the given lexicon, it also parses many ungrammatical sentences. For example, sentences with subject/verb disagreement, such as "*the student teach*", are parsed. In addition to sentences with subject/verb disagreement, identify two types of ungrammatical sentences that can be parsed with the given grammar and provide an example for each.
   (4 points)

2. We would like to augment the basic grammar with unification constraints such that all grammatical sentences unify, producing valid parses, while ungrammatical sentences fail unification. In a paragraph, describe how the grammar could be augmented with unification constraints to address subject/verb disagreement and the additional types of ungrammatical sentences you identified above. Clearly indicate what features should be added to the lexical entries and what constraints should be added to the grammar rules.
   (4 points)

5 Semantic Role Labeling (15 minutes/6 points)

Thematic Roles	
AGENT	***The waiter*** spilled the soup.
EXPERIENCER	***John*** has a headache.
FORCE	***The wind*** blows leaves into the yard.
THEME	He broke ***the ice***.
RESULT	The government has built ***a new railroad system***.
INSTRUMENT	He opened the door ***with a key***.
BENEFICIARY	He made the hotel reservation ***for his boss***.
SOURCE	She flew in ***from Boston***.
GOAL	She drove ***to Portland***.

Assign the various verb arguments in the following examples to their appropriate thematic roles, using the set of roles shown above. You should underline the words involved in each role and write the name of the role below the words. We have marked the verbs with SMALL CAPS. Arguments in roles not included above should not be marked (for example, TIME).

1. The hurricane STRUCK New York on Tuesday.

   (2 points)

2. Subways and tunnels all over town WERE FILLED with millions of gallons of seawater.

   (2 points)

3. Many residents ARE RETURNING to their homes anyways.
   (2 points)