Sample Questions for the Final

Anoop Sarkar – anoop@cs.sfu.ca

(1) Describe the relationship between subsumption (denoted by $\sqsubset$) and unification (denoted as $\sqcup$).

(2) Let's consider a collection of documents. Let *tf(w)* be the frequency of a word *w* across all documents, and let *df(w,d)* be the frequency of a word *w* within a particular document *d*. Assuming that this collection is in English, and picking some particular document *d* in this collection, give at least two examples of words *w* such that $\frac{tf(w)}{df(w,d)}$ will have the lowest values.

(3) Write down a sentence with at least 5 different types of ambiguity.

(4) Assume that we wanted to compactly represent subcategorization frames for verbs using the following recursive CFG rules:

$$\begin{aligned} VP &\rightarrow Verb \\ VP &\rightarrow VP\ X \end{aligned}$$

The non-terminal X can be associated with the category of the various arguments for each verb using a feature structure. For example, for a verb which takes an NP arguments, X is associated with the feature structure: $\left[\text{cat: NP}\right]$. What other feature structures need to be added to these two rules to enable the representation of the following subcategorization frames:

1. no arguments
2. NP
3. NP NP
4. NP PP
5. S
6. NP S

(5) Practice various examples of feature structure subsumption and unification.

(6) Consider the sentence *John saw the man with the telescope on the hill*. How many prepositional phrases (PPs) in this sentence. Show how you can use the Catalan numbers to compute the number of parse trees for this sentence using the number of PPs. Note that the use of Catalan numbers implies the use of the following grammar for PPs:

$$\begin{aligned} S &\rightarrow NP\ VP \\ NP &\rightarrow NP\ PP \\ VP &\rightarrow VP\ PP \\ PP &\rightarrow P\ NP \\ P &\rightarrow with \mid on \\ NP &\rightarrow John \mid Det\ N \\ N &\rightarrow man \mid telescope \mid hill \\ Det &\rightarrow the \end{aligned}$$

In particular, the Catalan numbers compute the ambiguity for the rules $VP \rightarrow VPPP$ and $NP \rightarrow NPPP$. Of course, for the sentence *John saw the man with the telescope on the hill*, you can always write a CFG that produces only a single parse, but it won't be linguistically plausible.

(7) Consider the following set of trees which are repeated several times:

- 2× (S (B a a) (C a a))
- 1× (S (C a a a))
- 7× (S (B a))

What is the set of CFG rules that can be extracted from this set of trees. Based on the frequency of the trees shown above, compute the probability of each CFG rule.

Given this probabilistic CFG what is the most likely tree for the input: *aaaa*

(8) Provide transformation-based learning rules to disambiguate the two meanings of *plant* in the following sentence fragments:

```
Z closing_VBG three_CD missile_NN     plant    s_NNS in_IN southern_JJ California_NNP a
_IN the_DT whole_JJ Chernobyl_NNP     plant    _NN in_IN 1991_CD ,_, five_CD years_NNS
IN a_DT hill_NN ,_, gardeners_NNS     plant    _NN begonias_NNS ,_, making_VBG floral_J
$_$ 200_CD million_CD printing_NN     plant    _NN in_IN Brooklyn_NNP ,_, Ohio_NNP ,_,
of_IN incompletely_JJ oxidated_JJ     plant    _NN and_CC animal_NN sediment_NN are_VBP
whenever_WRB you_PRP eat_VBP a_DT     plant    _NN ._. ''_''
n_IN return_NN for_IN a_DT new_JJ     plant    _NN near_IN Tuscaloosa_NNP ._.
T carmaker_NN could_MD finance_VB     plant    _NN construction_NN with_IN the_DT money
```

(9) Provide a context-free grammar for each context-free language below.

a. $L = \{a^i b^i \mid i \geq 1\}$

b. $L = \{a^{2i} b^{3i} \mid i \geq 1\}$

c. $L = \{a^i b^j c^i \mid i, j \geq 1\}$

d. $L = \{a^i b^i c^j \mid i, j \geq 1\}$

e. $L = \{a^i a^i b^j c^j \mid i, j \geq 1\}$

f. $L = \{u a^i w b^i y \mid i, j \geq 1\}$

g. $L = \{w w^R \mid w \in T^*, w^R \text{ is } w \text{ written backwards}\}$

Answers:

a. $S \rightarrow ab \mid aSb$

b. $S \rightarrow aabbb \mid aaSbbb$

c. $S \rightarrow aBc \mid aSc,\ B \rightarrow b \mid bB$

d. $S \rightarrow DC,\ D \rightarrow ab \mid aDb,\ C \rightarrow c \mid cC$

e. $S \rightarrow AC,\ A \rightarrow aAa \mid aa,\ C \rightarrow bc \mid bCc$

f. $S \rightarrow uDy,\ D \rightarrow awb \mid aDb$

g. $S \rightarrow xx \mid xSx \text{ for all } x \in T$

(10) For the CFG G given below:

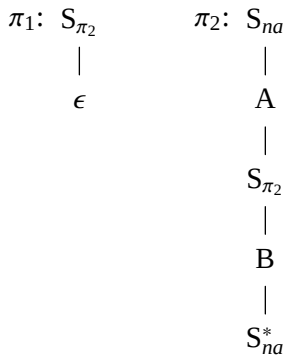
$$
\begin{aligned}
S &\rightarrow A \mid \epsilon \\
A &\rightarrow X \\
X &\rightarrow B \\
B &\rightarrow S
\end{aligned}
$$

a. What is the language $L(G)$?

b. What is the tree set $T(G)$?

c. Consider the path from the root node to the leaf node in each tree in the tree set $T(G)$. Can you write a grammar to describe this set of paths. Is a context-free grammar needed to describe this set? Is a regular expression sufficient?

(11) For the TAG G given below:



a. What is the language $L(G)$?

b. What is the tree set $T(G)$?

c. Consider the path from the root node to the leaf node in each tree in the tree set $T(G)$. Can you write a grammar to describe this set of paths. Is a context-free grammar needed to describe this set? Is a regular expression sufficient?

(12) Based on the implementation of feature structures in Homework 5, write a Perl function `subsume` which takes two dags as arguments and returns 1 if the first dag *subsumes* the second dag, 0 otherwise. Also write Perl function `copyDag` which takes one dag as input and creates a copy and returns the new dag.

(13) Provide the directed acyclic graph (dag) for the following feature structure:

$$\begin{bmatrix} \text{f} : \begin{bmatrix} \text{h} : \boxed{1} \\ \text{i} : \boxed{2} \end{bmatrix} \\ \text{g} : \boxed{1} \\ \text{k} : \boxed{2} \end{bmatrix}$$

(14) You are given a visual display of minimum distance alignments for three examples. From these examples, derive the costs for the operations of insertion, deletion and replacement in the target word. Assume that matching between characters that are unchanged is a zero cost operation.

The 1st line of the visual display shows the *target* word and the 3rd line shows the *source* word. An insertion in the target word is represented as an underscore in the 3rd line aligned with the inserted letter in the 1st line. Deletion in the target word is represented as an underscore '_' in the 1st line aligned with the corresponding inserted character in the source on the 3rd line. Finally, if a letter is unchanged between target and source then a vertical bar (the pipe symbol '|') is printed aligned with the letter in the 2nd line.

```
min edit distance = 1.25
g a r g l i n g
| |     |
g a m b l _ _ e

min edit distance = 0.75
g a m b l e
```

```
|       | |
g o o g l e

min edit distance = 1.45
u n g a r g l _ _ e
    |     | |
_ _ g o o g l i n g
```

Provide the cost of letter insertion, the cost of letter deletion and the cost of letter replacement in the target word.

(15) Given the alphabet $\{a, b, c, \ldots, z, \sqcup\}$, where $\sqcup$ is the space character, provide a transducer that implements the traditional rule of spelling correction — "*i before e except after c*". Use it to correct the inputs *'yeild'* and *'reciept'* (provide the state sequence in your transducer, the input word and the output word). Check if your transducer works correctly on the input word *'either'*. You can use Perl character classes to generalize over groups of letters from the alphabet, e.g. `[a-z,␣]` stands for any letter in the alphabet and `[^a]` stands for any letter in the alphabet except `a`.

(16) Let the alphabet $\Sigma = \{a, b\}$ and the language be Σ^* so $L = \{\epsilon, a, b, aa, bb, ab, bb \ldots\}$. Consider the unigram model where $P(a) = P(b) = 0.25$ and $P(\texttt{stop}) = 0.5$. Provide a proof for the following statement:

$$\sum_{w \in L} P(w) = 1$$

(17) Let c_{i+k}^{i} represent a sequence of characters $c_i, c_{i+1}, \ldots, c_{i+k}$. Assume that you're given a 4-gram character model: $P(c_i \mid c_{i-1}^{i-3})$. Note that $\sum_{c_i} P(c_i \mid c_{i-1}^{i-3}) = 1$. Assume that all the characters have been observed at least once in the training data such that $P(c_i)$ is never zero in unseen data.

a. (5pts) Consider a backoff smoothing model $\hat{P}$ which deals with events that have been observed zero times in the training data:

$$\hat{P}(c_i \mid c_{i-1}^{i-3}) = \begin{cases} P(c_i \mid c_{i-1}^{i-3}) & \textit{if } f(c_i^{i-3}) > 0 \\ P(c_i \mid c_{i-1}^{i-2}) & \textit{if } f(c_i^{i-3}) = 0 \textit{ and } f(c_i^{i-2}) > 0 \\ P(c_i \mid c_{i-1}) & \textit{if } f(c_i^{i-2}) = 0 \textit{ and } f(c_i^{i-1}) > 0 \\ P(c_i) & \textit{otherwise} \end{cases}$$

where $f(c_{i+k}^{i})$ is the number of times the n-gram c_{i+k}^{i} was observed in the training data. What condition that holds in the original model is violated by $\hat{P}$?

b. (5pts) Consider a Jelinek-Mercer style interpolation smoothing model $\hat{P}$:

$$\hat{P}(c_i \mid c_{i-1}^{i-3}) = \lambda_1 \cdot P(c_i \mid c_{i-1}^{i-3}) + \lambda_2 \cdot P(c_i \mid c_{i-1}^{i-2}) + \lambda_3 \cdot P(c_i \mid c_{i-1}) + \lambda_4 \cdot P(c_i)$$

State the condition on values assigned to $\lambda_1, \ldots, \lambda_4$ for $\hat{P}$ to be a well-defined probability model.

c. (10pts) Assume you are given some additional training data (separate from your original training data). Let's say this data is your *held-out* data called T. T will contain ngrams that were unseen in our original training data, and we can exploit this fact to compute values for λ_i in an interpolation smoothing model.

Let W_T be the length of T in number of character tokens. For each *token* t_i in T, where $1 \leq i \leq W_T$, let $g_1(t_i) = 1$ iff the 4-gram probability $P(t_i \mid t_{i-1}^{i-3})$ had a non-zero value (that is, whenever $f(t_i^{i-3}) > 0$). Similarly, let $g_2(t_i)$, $g_3(t_i)$ and $g_4(t_i)$ equal 1 when the trigram, bigram and unigram probability respectively had a non-zero value for each token t_i in T.

Show how you can compute the values for $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ for the equation in Question 17b by using g_1, g_2, g_3, g_4. State why the condition stated in your answer to Question 17b is satisfied by your answer.

(18) Consider a part of speech tagged sentence represented as a list of strings where each element in the list is a word and associated part of speech tag:

```
@sentence1 = ("Show/VB", "me/PRP", "the/DT", "flights/NNS", "from/IN",
"San/NNP", "Diego/NNP", "to/TO", "Toronto/NNP");

@sentence2 = ("From/IN", "Washington/NNP", "D/NNP", "C/NNP", "to/TO",
"Philadelphia/NNP", "on/IN", "Wednesdays/NNPS");
```

In these sentences, there are some meaningful strings that span multiple words. For example, *"San Diego"*, *"Toronto"* and *"Washington D C"* are names of cities. If we wanted to find names, places, or days of the week in part of speech tagged text then a useful heuristic to apply is to search for the sequences of words that are tagged as *proper nouns*.

Assume that you are using the Penn Treebank tagset: `NNP` is the tag for singular proper noun, e.g. `Toronto/NNP`; and `NNPS` is the tag for plural proper noun, e.g. `Wednesdays/NNPS`.

Write a function (subroutine) in Perl called `chunkProperNouns` that takes a part of speech tagged sentence represented as a list of word/tag strings and merges sequences of proper nouns into a single string and returns a list of all such merged strings.

For example,

`chunkProperNouns(@sentence1)` returns `("San/NNP Diego/NNP", "Toronto/NNP")`

`chunkProperNouns(@sentence2)` returns `("Washington/NNP D/NNP C/NNP", "Philadelphia/NNP", "Wednesdays/NNPS")`

Provide your Perl code or precisely defined pseudo-code for `chunkProperNouns`.

(19) Consider the following definition for the trigram probability over part of speech tags: $P(t_i \mid t_{i-2}, t_{i-1})$ and the emit probability of a word given a part of speech tag: $P(w_i \mid t_i)$. The part of speech tag definitions are as follows: `bos` (*begin sentence marker*), N (*noun*), V (*verb*), D (*determiner*), P (*preposition*), `eos` (*end of sentence marker*).

$P(t_i \mid t_{i-2}, t_{i-1})$	t_{i-2}	t_{i-1}	t_i
1	D	N	`eos`
1	`bos`	`bos`	N
1	P	D	N
$\frac{1}{2}$	`bos`	N	N
$\frac{1}{2}$	`bos`	N	V
1	V	D	N
1	V	V	D
$\frac{1}{3}$	N	V	D
$\frac{1}{3}$	N	V	V
$\frac{1}{3}$	N	V	P
$\frac{1}{2}$	N	N	V
$\frac{1}{2}$	N	N	P
1	N	P	D
1	V	P	D

$P(w_i \mid t_i)$	t_i	w_i
1	D	an
$\frac{2}{5}$	N	time
$\frac{2}{5}$	N	arrow
$\frac{1}{5}$	N	flies
1	P	like
$\frac{1}{2}$	V	like
$\frac{1}{2}$	V	flies
1	`eos`	`eos`
1	`bos`	`bos`

a. (5pts) The probability of a part of speech tagged sentence

$$s = \texttt{bos/bos}, \texttt{bos/bos}, w_0/t_0, w_1/t_1, \ldots, w_{n-1}/t_{n-1}, \texttt{eos/eos}$$

is given by:

$$P(s) = \prod_{i=0}^{n} P(t_i \mid t_{i-2}, t_{i-1}) \times P(w_i \mid t_i)$$

Using the probability tables given above, compute the probability of the two sentences given below. Which of these two sentences gets the higher probability?

1. bos/bos, bos/bos, time/N, flies/V, like/P, an/D, arrow/N, eos/eos
2. bos/bos, bos/bos, time/N, flies/N, like/V, an/D, arrow/N, eos/eos

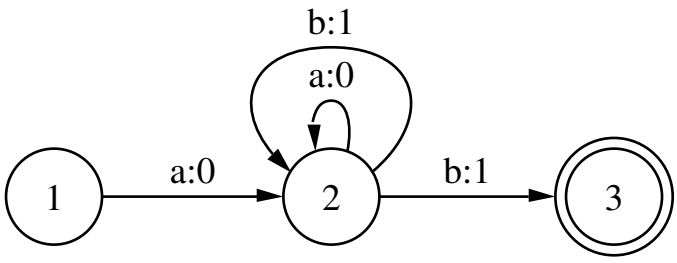
b. (15pts) A hidden markov model is defined as a set of states $\mathbf{s} = (s_0, \ldots, s_m)$ for some fixed, finite value of m where each state s_i can emit an output symbol w_i with probability $P(w_i \mid s_i)$ and each state s_{i+1} can be reached from a state s_i with probability $P(s_{i+1} \mid s_i)$. Provide a hidden markov model (*hmm*) that maps the trigram probabilities $P(t_i \mid t_{i-2}, t_{i-1})$ shown in the table above to transition probabilities $P(s_{i+1} \mid s_i)$ in your hmm. First define a mapping between the trigrams and the set of states in your hmm. Then define the transition probabilities. You don't need to provide the emit probabilities $P(w_i \mid s_i)$.

You can either draw the hmm graphically as a state machine graph with the transition probabilities on the arcs *or* you can provide a tabular representation of the transition probability $P(s_{i+1} \mid s_i)$. You do not need to show transitions with zero probability or any states that are not useful in representing the trigam probabilities.

(20) Consider the following representation in Perl of a finite state transducer:

```
$startstate = 1;
$finalstate[3] = 'true';
$edges[1] = [2];
$edges[2] = [2,3];
$input[1][2] = ['a'];     $output[1][2] = ['0'];
$input[2][2] = ['a','b']; $output[2][2] = ['0','1'];
$input[2][3] = ['b'];     $output[2][3] = ['1'];
```

It is a slight extension of the representation you have used for finite state automata. The only addition is the `@output` array which is analogous to the `@input` array. The above Perl data represents the finite state transducer shown below:



Just as finite state automata (FSA) recognize strings, a finite state transducer (FST) can be viewed as a recognizer over *pairs* of strings. For example, the pair of strings (*abab*, 0101) is accepted by the above FST. This means we can extend the algorithm `NDRecognize` for FSA to an algorithm called `fstRecognize` which can be used to decide whether a *string pair* is accepted by a given FST.

The process of accepting a pair of strings will be represented in Perl with the use of two ‘tapes’. An input tape which will be a reference to a list of elements from the input alphabet of the FST, and an output tape which is a reference to a list of elements from the output alphabet of the FST. A Perl implementation of the algorithm `fstRecognize` uses the definition of the FST as Perl data provided above. It takes the input tape and the output tape as parameters and returns `1` if the pair of strings is accepted by the FST and `0` otherwise.

For example,

```
$inputTape = ['a', 'b', 'a', 'b'];
$outputTape = ['0', '1', '0', '1'];
fstRecognize($inputTape, $outputTape) returns 1
```

On the following page, you are given the Perl code for `fstRecognize` which uses two additional functions: `acceptState` and `generateNewStates`. You are also given the definition of `acceptState`. You have to provide either Perl code *or* precisely defined pseudo-code for the function `generateNewStates`.

```
sub fstRecognize($$) {
  my ($inpTape, $outTape) = @_;
  my @agenda = ( $startstate, -1, -1 );
  while (1) {
    my $state = shift(@agenda);
    my $inpIndex = shift(@agenda);
    my $outIndex = shift(@agenda);
    if (acceptState($state, $inpIndex, $outIndex, $#$inpTape, $#$outTape)) {
      return 1;
    } else {
      my @newItems = generateNewStates($state, $inpIndex, $outIndex, $inpTape, $outTape);
      push(@agenda, @newItems);
      if ($#agenda == -1) { return 0; }
    }
  }
}

sub acceptState($$$$$) {
  my ($state, $inpIndex, $outIndex, $endInp, $endOut) = @_;
  if (($finalstate[$state] eq 'true') and ($inpIndex==$endInp) and ($outIndex==$endOut)) {
    return 1;
  }
  return 0;
}
```