# CMPT 413 - Spring 2014 - Final Prep Questions

(1) **Language Modeling**:

TrueCasing is the process of taking text with missing or unreliable case information, e.g. if the input looks like:

```
as previously reported , target letters were issued last month to michael milken ,
drexel 's chief of junk-bond operations ; mr. milken 's brother lowell ; cary
maultasch ,  a drexel trader ; james dahl , a drexel bond salesman ; and bruce
newberg , a former drexel trader .
```

Then the output of the TrueCasing program should be:

```
As previously reported , target letters were issued last month to Michael Milken ,
Drexel 's chief of junk-bond operations ; Mr. Milken 's  brother Lowell ; Cary
Maultasch ,  a Drexel trader ; James Dahl , a Drexel bond salesman ; and Bruce
Newberg , a former Drexel trader .
```

Assume we are given a *translation probability* $P(w \mid W)$ where $w$ is the lowercase variant of the TrueCase word $W$ (note that the TrueCase word might still be lowercase). A language model $P(W_1, \ldots, W_n)$ is used to provide the probability of a sentence. A *bigram language model* approximates the probability of a sentence as follows:

$$P(W_1, \ldots, W_n) = \prod_{i=1}^{n} P(W_i \mid W_{i-1})$$

We assume that $W_{-1} = w_{-1} = \textit{none}$ is a dummy word that begins each sentence.

a. Describe how you can train the translation probability $P(w \mid W)$. Assume you have access to a sufficient amount of TrueCase text.

*Answer:* For each TrueCase word $W$ convert it to lowercase $w$ and count $f(w, W)$ and $f(W)$. Then,

$$P(w \mid W) = \frac{f(w, W)}{f(W)}$$

b. Complete the following formula to provide a model of the TrueCasing task by using only the translation probability and the bigram language model:

$$\begin{aligned} W_1^*, \ldots, W_n^* &= \operatorname*{argmax}_{W_1, \ldots, W_n} P(W_1, \ldots, W_n \mid w_1, \ldots, w_n) \\ &= \textit{provide this formula} \end{aligned}$$

*Answer:*

$$\begin{aligned} W_1^*, \ldots, W_n^* &= \operatorname*{argmax}_{W_1, \ldots, W_n} P(W_1, \ldots, W_n \mid w_1, \ldots, w_n) \\ &= \frac{P(W_1, \ldots, W_n) \cdot P(w_1, \ldots, w_n \mid W_1, \ldots, W_n)}{P(w_1, \ldots, w_n)} \\ &\approx P(W_1, \ldots, W_n) \cdot P(w_1, \ldots, w_n \mid W_1, \ldots, W_n) \\ &= \prod_{i=1}^{n} \underbrace{P(w_i \mid W_i)}_{\textit{translation probability}} \cdot \underbrace{P(W_i \mid W_{i-1})}_{\textit{bigram language model}} \end{aligned}$$

c. Why was the mapping between TrueCase words and their lowercase variant defined using $P(w \mid W)$ instead of $P(W \mid w)$?

*Answer:* To use the noisy channel model, this has to be the way to model this task. There can be other ways to directly find the most appropriate mapping which could use $P(W \mid w)$.

d. Provide the pseudo code or recurrence relation that will find the score for $W_1^*, \ldots, W_n^*$, which is the most likely TrueCase output according to our model. Provide the time complexity of the algorithm.

*Answer:* We can use the Viterbi algorithm. For input $w_1, \ldots, w_n$ we can efficiently compute the best score upto TrueCase word candidate $W_{i+1}$ recursively as follows:

$$\text{Viterbi}[i+1, W_{i+1}] = \max_{W_i} (\text{Viterbi}[i, W_i] \times P(w_{i+1} \mid W_{i+1}) \times P(W_{i+1} \mid W_i))$$

We initialize the recursion with the value for Viterbi$[1, W_1]$.

If the number of lowercase words is $|w|$ and the number of TrueCase words is $|W|$ then the time complexity is $O(|w| \cdot |W|^2)$.

e. For the probability $P(w \mid W)$ it is possible that we do not have enough data for a robust estimate for all possible $w, W$ pairs. Complete the following formula which attempts to solve this issue using a parameter $\delta$ such that $0 < \delta <= 1$. The new probability $P_s(w \mid W)$ provides non-zero probabilities for every $w, W$ pair:

$$P_s(w \mid W) = \frac{f(w, W) + \delta}{\textit{provide this part of the formula}}$$

*Answer:* denominator = $f(W) + \delta|V|$

(2) **Context-free Grammars and Parsing**:

Consider the CFG below with $S$ as the start symbol:

$$\begin{array}{rcl} S & \rightarrow & NP\ VP \\ VP & \rightarrow & V\ NP \\ VP & \rightarrow & VP\ PP \\ NP & \rightarrow & NP\ PP \\ NP & \rightarrow & Det\ N \\ PP & \rightarrow & P\ NP \end{array} \qquad \begin{array}{rcl} NP & \rightarrow & Calvin \\ V & \rightarrow & saw \\ Det & \rightarrow & the \mid a \\ N & \rightarrow & man \mid telescope \mid hill \\ P & \rightarrow & with \mid on \end{array}$$

a. Draw the parse tree for the input sentence: *Calvin saw a man*

b. Consider the input sentence: *Calvin saw the man on the hill with a telescope*.

i. In an Earley parser, what states can we predict from the state $(S \rightarrow \bullet\ NP\ VP, [0, 0])$.

*Answer:*
$(NP \rightarrow \bullet\ NP\ PP, [0, 0])$
$(NP \rightarrow \bullet\ Calvin, [0, 0])$
$(NP \rightarrow \bullet\ Det\ N, [0, 0])$
$(Det \rightarrow \bullet\ the, [0, 0])$

ii. Subsequently, what states can we complete from the state $(NP \rightarrow Calvin\ \bullet, [0, 1])$.

*Answer:* The span [0,1] has to be correct!
$(S \rightarrow NP\ \bullet\ VP, [0, 1])$
$(NP \rightarrow NP\ \bullet\ PP, [0, 1])$

c. Can the CKY algorithm be used with the above grammar without any modifications. Why?

*Answer:* Yes. G is in CNF.

d. Consider a sub-class of CFGs that is only allowed to have rules of the *type* $A \rightarrow a\ B\ b$ or $A \rightarrow a$, where $A, B$ are any non-terminals, and $a, b$ are any terminals. For example, the following CFG for *ambiguous palindromes* is in this sub-class:

$$\begin{aligned} S &\rightarrow aXa \mid bXb \mid aYa \mid cYc \\ X &\rightarrow aXa \mid bXb \mid a \mid b \\ Y &\rightarrow aYa \mid cYc \mid a \mid c \end{aligned}$$

By analogy to the analysis used for the CKY algorithm, show that this class of CFGs can be parsed in time $O(n^2)$.

*Hint*: Derive a normal form for the above sub-class of CFGs (analogous to Chomsky Normal Form and the relation of CNF to the CKY algorithm).

*Answer:* $A \rightarrow a$ is already in CNF. Rules like $A \rightarrow a\ B\ b$ can be converted into two rules $A \rightarrow a\ X$ and $X \rightarrow B\ b$ where $X$ is a new non-terminal not used anywhere else in the CFG. These rules can be converted into CNF easily: $A \rightarrow A'\ X$ and $X \rightarrow B\ B'$ where $A' \rightarrow a$ and $B' \rightarrow b$. However, unlike standard CKY where the right hand side could have arbitrary spans $(i, k)$ and $(k, j)$ in this case we can only have the following two cases $(i-1, i)$ and $(i, j)$ OR $(i, j-1)$ and $(j-1, j)$ because either $A'$ or $B'$ can only span one terminal in this type of grammar. Therefore, instead of varying all spans $(i, j)$ with all possible $k$ values in between $i$ and $j$ like in regular CKY, in this special restricted case we only need to search through all spans $(i, j)$ and either look at the span in $(i-1, i)$ or $(j, j+1)$ which is a constant time lookup into the chart matrix. Since we only need to loop over all valid $(i, j)$ values the complexity is $O(n^2)$.

(3) **Probabilistic Context-free Grammars**

Consider a treebank which consists of three tree *types*: $T_1, T_2, T_3$. By counting the number of times each tree type was observed, we discover that each tree type occurs with the following probability:

$p_1$ $T_1$ = (S (B a) (C a a))
$p_2$ $T_2$ = (S (B a a))
$p_3$ $T_3$ = (S (C a a a))

a. From the treebank shown above, extract a probabilistic CFG (PCFG) $G$.
*Hint:* note that the rule S → BC appears in $G$ with probability $p_1$.

*Answer:*

| | |
|---|---|
| $p_1$ | S → BC |
| $p_2$ | S → B |
| $p_3$ | S → C |
| $\frac{p_1}{p_1+p_2}$ | B → a |
| $\frac{p_2}{p_1+p_2}$ | B → aa |
| $\frac{p_1}{p_1+p_3}$ | C → aa |
| $\frac{p_3}{p_1+p_3}$ | C → aaa |

b. A PCFG is proper if every rule $A \rightarrow \alpha$ in the CFG has probability $P(A \rightarrow \alpha \mid A)$. What is the condition on $p_1, p_2$ and $p_3$ that will ensure that PCFG G is proper.

*Answer:*

$$\sum_i p_i = 1$$

c. Provide the tree set $\mathcal{T}$ for the CFG $G$.

*Answer:*

$T_4 =$ (S (B a) (C a a a))
$T_5 =$ (S (B a a) (C a a))
$T_6 =$ (S (B a a) (C a a a))
$T_7 =$ (S (B a))
$T_8 =$ (S (C a a))

$$\mathcal{T} = \{T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8\}$$

d. Provide the language $\mathcal{L}$ (the set of strings) for the CFG $G$.

*Answer:*

$$\mathcal{L} = \{aaa, aa, aaaa, aaaaa, a\}$$

e. Let $p_1 = 0.2, p_2 = 0.1, p_3 = 0.7$. Find the parse tree with highest probability according to PCFG $G$ for the input string *aa*

*Answer:*

| | |
|---|---|
| 0.2 | S → BC |
| 0.1 | S → B |
| 0.7 | S → C |
| $\frac{2}{3}$ | B → a |
| $\frac{1}{3}$ | B → aa |
| $\frac{2}{9}$ | C → aa |
| $\frac{7}{9}$ | C → aaa |

$P(T_2) = 0.0333$ and $P(T_8) = 0.1555$, so $T_8$ is the parse tree with highest probability for input string *aa*.

f. Consider a case where for some input string accepted by the PCFG $G$, the parse tree with highest probability is *not* the one that was observed in the treebank used to create the PCFG in the first place. Provide such an input string and briefly (in one sentence) point out the relevant property of PCFGs which explains why such a situation can occur.

*Answer:* Since a PCFG is "context-free" every context into which a PCFG rule is equivalent. So if a rule $r = B \rightarrow \alpha$ is expanded from a rule $r_1 = A \rightarrow BC$ with a different frequency in the treebank as opposed to a rule $r_2 = X \rightarrow BY$, then the PCFG cannot capture the fact that trees in the treebank with $r_1$ and $r$ might be more frequent than a tree with $r_2$ and $r$.

(4) **Language Modeling**

Consider a language model over character sequences that computes the probability of a word based on the characters in that word, so if word $w = c_0, c_1, \ldots, c_n$ then $P(w) = P(c_0, \ldots, c_n)$. Let us assume that the language model is defined as a bigram character model $P(c_i \mid c_{i-1})$ where

$$P(c_0, \ldots, c_n) = \prod_{i=1,2,\ldots,n} P(c_i \mid c_{i-1}) \tag{1}$$

For convenience we assume that we have explicit word boundaries: $c_0 = \texttt{bos}$ and $c_n = \texttt{eos}$ where `bos` stands for *begin sentence marker* and `eos` stands for *end of sentence marker*.

Based on this model, for the English word *booking* the probability would be computed as:

$$P(booking) = P(b \mid \texttt{bos}) \times P(o \mid b) \times P(o \mid o) \times P(k \mid o) \times P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{eos} \mid g)$$

The inflection *ing* is a suffix and is generated after the stem *book* with probability

$$P(ing) = P(i \mid k) \times P(n \mid i) \times P(g \mid n) \times P(\texttt{eos} \mid g)$$

In Semitic languages, like Arabic and Hebrew, the process of inflection works a bit differently. In Arabic, for a word like *kitab* the stem would be *k-t-b* where the place-holders '-' for inflection characters have been added for convenience. We will assume that each word is made up of a sequence of consonant-vowel sequences CVCVCV… and the vowels always form the inflection.

a. Provide the definition of an $n$-gram model that will compute the probability for the word *kitab* and *k-t-b* as follows:

$$P(kitab) = P(k \mid \texttt{bos}) \times P(t \mid k) \times P(b \mid t) \times P(i \mid b) \times P(a \mid i) \times P(\texttt{eos} \mid a)$$

$$P(k\text{-}t\text{-}b) = P(k \mid \texttt{bos}) \times P(t \mid k) \times P(b \mid t) \times P(\text{-} \mid b) \times P(\text{-} \mid \text{-}) \times P(\texttt{eos} \mid \text{-})$$

Write down the equation for this $n$-gram model in the same mathematical notation as equation (1).

*Answer:*

$$P(c_0, \ldots, c_n) = \begin{cases} \prod_{i=1}^{n} P(c_i \mid c_{i-1}) & \text{if } n \leq 3 \\ \Big(P(c_1 \mid c_0) \times \prod_{i=3,5,\ldots}^{\ell} P(c_i \mid c_{i-2})\Big) \times & \\ \Big(P(c_2 \mid c_{\ell_o}) \times \prod_{i=4,6,\ldots}^{\ell} P(c_i \mid c_{i-2}) \times P(c_n \mid c_{\ell_e})\Big) & \text{if } n > 3 \end{cases}$$

Define $\ell = n - (n \bmod 2)$ and $\ell_o$ is the last odd number less than $\ell$ and $\ell_e$ is the last even number less than $\ell$. As long as the boundary cases are right for the bigrams, we don't penalize off by one in the length, and we don't penalize for $n \leq 3$.

b. Using your $n$-gram model show how $P(kitab) = P(ktb) \times P(ia)$.

*Answer:*

$$\begin{aligned} P(kitab) &= P(c_0 = \texttt{bos}, c_1 = k, c_2 = i, c_3 = t, c_4 = a, c_5 = b, c_6 = \texttt{eos}) \\ &= P(ktb) \times P(ia, \texttt{eos}) \\ P(ktb) &= P(c_1 = k \mid c_0 = \texttt{bos}) \times P(c_3 = t \mid c_1 = k) \times P(c_5 = b \mid c_3 = t) \\ &\quad \text{this term corresponds to the first bracket in the eqn above} \\ P(ia) &= P(c_2 = i \mid c_{\ell_o} = c_5 = b) \times P(c_4 = a \mid c_2 = i) \times P(c_n = c_6 = \texttt{eos} \mid c_{\ell_e} = c_4 = a) \\ &\quad \text{corresponds to the second bracket in the eqn above} \end{aligned}$$

(5) **Tagging and Chunking**

Tagging is the process of providing a sequence of tags $T_1, \ldots, T_n$ to a sequence of observations $O_1, \ldots, O_n$. We can describe the tagging task using the equation:

$$T_1^*, \ldots, T_n^* = \operatorname*{argmax}_{T_1,\ldots,T_n} P(T_1, \ldots, T_n \mid O_1, \ldots, O_n)$$

a. Define a noisy channel model for tagging by applying Bayes rule to $P(T_1, \ldots, T_n \mid O_1, \ldots, O_n)$ and providing the resulting equation.

*Answer:*

$$\begin{aligned} T_1^*, \ldots, T_n^* &= \operatorname*{argmax}_{T_1,\ldots,T_n} P(T_1, \ldots, T_n \mid O_1, \ldots, O_n) \\ &= \frac{P(T_1, \ldots, T_n) \cdot P(O_1, \ldots, O_n \mid T_1, \ldots, T_n)}{P(O_1, \ldots, O_n)} \\ &\approx P(T_1, \ldots, T_n) \cdot P(O_1, \ldots, O_n \mid T_1, \ldots, T_n) \end{aligned}$$

b. Define a separate unigram, bigram, and trigram model *over tags* by modifying the equation derived in Q (5a) – it should continue to include the probability of emission of observations. Assume suitable padding of extra observations and tags in order to simplify the form of the equation.

*Answer:*

$$P(T_1, \ldots, T_n) \cdot P(O_1, \ldots, O_n \mid T_1, \ldots, T_n) =$$
$$\prod_{i=1}^{n} P(T_i) \cdot \prod_{i=1}^{n} P(O_i \mid T_i) \text{ (unigram)}$$
$$\prod_{i=1}^{n} P(T_i \mid T_{i-1}) \cdot \prod_{i=1}^{n} P(O_i \mid T_i) \text{ (bigram)}$$
$$\prod_{i=1}^{n} P(T_i \mid T_{i-2}, T_{i-1}) \cdot \prod_{i=1}^{n} P(O_i \mid T_i) \text{ (trigram)}$$

c. Combine the unigram, bigram and trigram models you provide in Q (5b) into a single model for tagging using interpolation. Provide the equation for the interpolated model.

*Answer:*

$$P(T_1, \ldots, T_n) \cdot P(O_1, \ldots, O_n \mid T_1, \ldots, T_n) =$$
$$\prod_{i=1}^{n} P(O_i \mid T_i) \cdot \left( \lambda_1 \prod_{i=1}^{n} P(T_i) + \lambda_2 \prod_{i=1}^{n} P(T_i \mid T_{i-1}) + \lambda_3 \prod_{i=1}^{n} P(T_i \mid T_{i-2}, T_{i-1}) \right)$$
$$= \prod_{i=1}^{n} P(O_i \mid T_i) \cdot \prod_{i=1}^{n} (\lambda_1 P(T_i) + \lambda_2 P(T_i \mid T_{i-1}) + \lambda_3 P(T_i \mid T_{i-2}, T_{i-1}))$$

where, $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.

d. If the observations $O_i$ are words from a vocabulary $\mathcal{V}$ and the tags $T_i$ are taken from a set of part of speech tags $\mathcal{T} = \{Noun, Verb, Prep, Det, \ldots\}$ then we can use the tagging model shown above in order to solve the *part of speech tagging* task.

For each of the following tasks show how you can define the set of observations $\mathcal{V}$ and the tag set $\mathcal{T}$ (provide the full set of tags to be used) in order to solve that task using the above model of tagging.

1. Noun phrase chunking.

   *Answer:* $\mathcal{V}$ are words and $\mathcal{T} = \{B, I, O\}$ for Begin, Inside, or Outside a noun phrase chunk.

2. Word segmentation (for languages like Chinese which typically do not use spaces to separate words).

   *Answer:* $\mathcal{V}$ are characters in Chinese and $\mathcal{T} = \{B, I\}$ for Begin, or Inside a word.

3. Named entity recognition (find person, organization or location names in text).

   *Answer:* $\mathcal{V}$ are words and $\mathcal{T}$ = {B-PER, I-PER, B-ORG, I-ORG, B-LOC, I-LOC, O} for Begin or Inside (PERson, ORGanization, or LOCation) or Outside any NER.

4. FAQ segmentation (for a given document of sentences, label each sentence as a question or answer sentence assuming the typical structure of a Frequently Asked Questions document).

   *Answer:* $\mathcal{V}$ are sentences and $\mathcal{T} = \{Q, A\}$ for Question, or Answer sentence.

(6) **Context-free Grammars**:
For the CFG $G$ given below:

$$\begin{aligned} S &\rightarrow A \mid c \\ A &\rightarrow B\ a \\ B &\rightarrow b\ S \end{aligned}$$

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

*Answer:* $b^n c a^n : n \geq 0$

b. What is the tree set $T(G)$? You can use any convenient short-hand notation to represent an infinite set of trees.

*Answer:*

```
T(G) = {
(S (A (B b
         (S (A (B b
                  (S ... c))
                a))
      a)))
}
```

or:

```
T(G) =
{ (S c) ,
  (S (A (B b (S c)) a)),
  (S (A (B b (S (A (B b (S c)) a)) a))),
  ...
}
```

c. Assign probabilities to each rule in the CFG above so that for each string $w \in L(G)$:

$$P(w) = exp\left(\frac{|w| - 1}{2} * ln(0.3) + ln(0.7)\right)$$

where, $|w|$ is the length of string $w$, *exp* is exponentiation, and *ln* is *log* base $e$.

*Answer:*

```
0.3 S -> A
0.7 S -> c
1.0 A -> B a
1.0 B -> b S
```

d. Convert the PCFG from the answer to Q (6c) into Chomsky Normal Form (CNF). The CNF grammar must also have the right probabilities.

*Answer:*

```
0.7 S -> c
0.3 S -> B A'
1.0 B -> B' S
1.0 B' -> b
1.0 A' -> a
```

e. Provide a leftmost derivation using your CNF grammar for the input string *bbcaa*. The derivation should include probabilities for each step – you can keep them as multiples of individual probabilities if you wish.

*Answer:*

```
S => BA' (0.3)
=> B'SA' (0.3 * 1.0)
=> bBA'A' (0.3)
=> bB'SA'A' (0.3 * 0.3)
=> bbSA'A' (0.09)
=> bbcA'A' (0.09 * 0.7)
=> bbcaA' (0.027)
=> bbcaa (0.027)
```

f. Briefly explain why conversion into CNF implies that we can parse any CFG in time $O(G^2n^3)$ where $G$ is the number of non-terminals in the CFG, and $n$ is the length of the input string. (A short 1–2 line answer will suffice).

*Answer:* The rules of type $A \rightarrow a$ can be used to fill in each span of length one, and to find a start symbol spanning the entire string we use rules of type $A \rightarrow BC$ to recursively find a span $i, j$ if we have previously found a span from $i, k$ for $B$ and $k, j$ for $C$. Since $B$ and $C$ span over all non-terminals we need $G^2$ time, and since we need to find all spans $i, j$ with every $k$ in between we need $n^3$ time.