

6.864, Fall 2005: Problem Set 1

Total points: 90 regular points, 10 bonus points

Due date: 5pm, 29th September 2005

Late policy: 5 points off for every day late, 0 points if handed in after 1pm on October 4th 2005

Question 1 (20 points)

A probabilistic context-free grammar $G = (N, \Sigma, R, S, P)$ in Chomsky Normal Form is defined as follows:

- N is a set of non-terminal symbols (e.g., NP, VP, S etc.)
 - Σ is a set of terminal symbols (e.g., cat, dog, the, etc.)
 - R is a set of rules which take one of two forms:
 - $X \rightarrow Y_1Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
 - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
 - $S \in N$ is a distinguished start symbol
 - P is a function that maps every rule in R to a probability, which satisfies the following conditions:
 - $\forall r \in R, P(r) \geq 0$
 - $\forall X \in N, \sum_{X \rightarrow \alpha \in R} P(X \rightarrow \alpha) = 1$

Now assume we have a probabilistic CFG G' , which has a set of rules R which take one of the two following forms:

- $X \rightarrow Y_1 Y_2 \dots Y_n$ for $X \in N$, $n \geq 2$, and $\forall i$, $Y_i \in N$
 - $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$

Note that this is a more permissive definition than Chomsky normal form, as some rules in the grammar may have more than 2 non-terminals on the right-hand side. An example of a grammar that satisfies this more permissive definition is as follows:

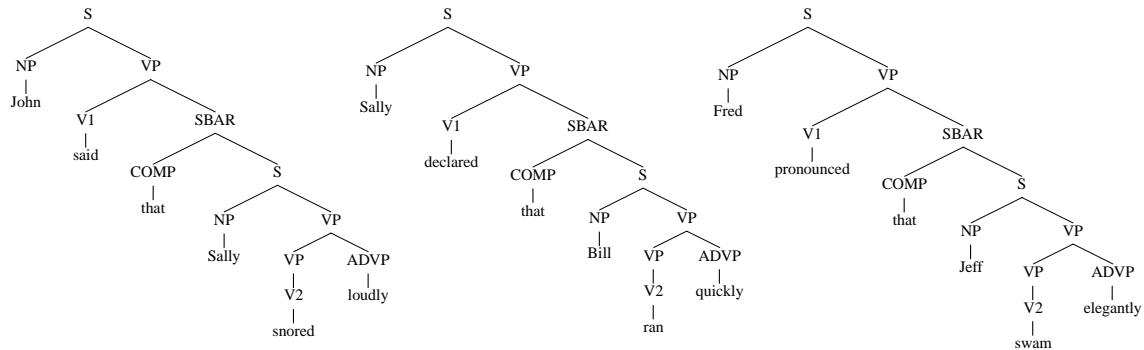
S → NP VP	0.7	Vt → saw	1.0
S → NP NP VP	0.3	NN → man	0.7
VP → Vt NP	0.8	NN → woman	0.2
VP → Vt NP PP	0.2	NN → telescope	0.1
NP → DT NN NN	0.3	DT → the	1.0
NP → NP PP	0.7	IN → with	0.5
PP → P NP	1.0	IN → in	0.5

Question 1(a): Describe how to transform a PCFG G' , in this more permissive form, into an “equivalent” PCFG G in Chomsky normal form. By equivalent, we mean that there is a one-to-one function f between derivations in G' and derivations in G , such that for any derivation T' under G' which has probability p , $f(T')$ also has probability p . (Note: one major motivation for this transformation is that we can then apply the dynamic programming parsing algorithm, described in lecture, to the transformed grammar.) Hint: think about adding new rules with new non-terminals to the grammar.

Question 1(b): Show the resulting grammar G after applying your transformation to the example PCFG shown above.

Question 2 (20 points)

Nathan L. Pedant decides to build a treebank. He finally produces a corpus which contains the following three parse trees:



Clarissa Lexica then purchases the treebank, and decides to build a PCFG, and a parser, using Nathan’s data.

Question 2(a): Show the PCFG that Clarissa would derive from this treebank.

Question 2(b): Show two parse trees for the string “Jeff pronounced that Fred snored loudly”, and calculate their probabilities under the PCFG.

Question 2(c): Clarissa is shocked and dismayed, (see 2(b)), that “Jeff pronounced that Fred snored loudly” has two possible parses, and that one of them—that Jeff is doing the pronouncing loudly—has relatively high probability, in spite of it having the ADVP *loudly* modifying the “higher” verb, *pronounced*. This type of high attachment is never seen in the corpus, so the PCFG is clearly missing something. Clarissa decides to fix the treebank, by altering some non-terminal labels in the corpus. Show one such transformation that results in a PCFG that gives zero probability to parse trees with “high” attachments. (Your solution should systematically refine some non-terminals in the treebank, in a way that slightly increases the number of non-terminals in the grammar, but allows the grammar to capture the distinction between high and low attachment to VPs.)

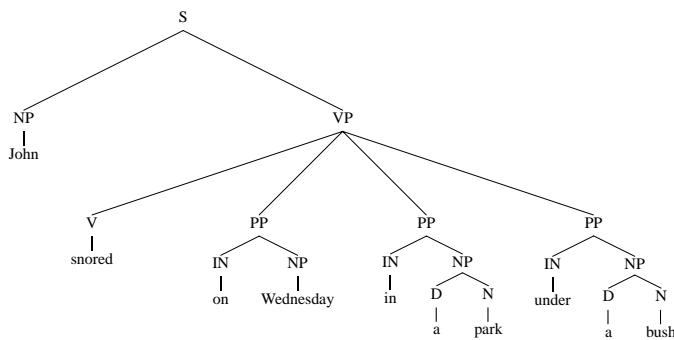
Question 3 (20 points)

Clarissa now decides to build a treebank. A particular issue she runs into is the case where multiple PPs (preposition phrases) modify a verb. For example, in

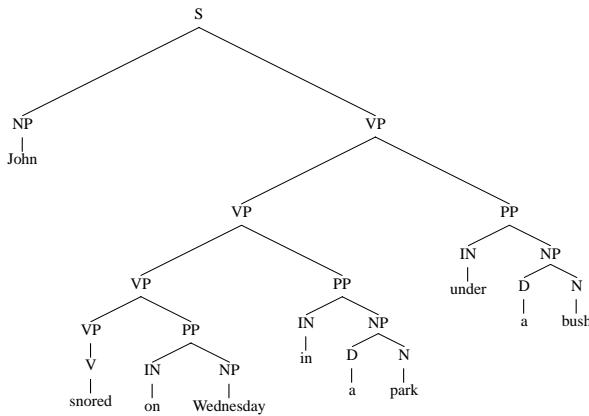
John snored on Wednesday in a park under a bush

the verb *snored* is modified by three PPs (*on Wednesday*, *in a park*, and *under a bush*).

Clarissa opts for a “flat” style of annotation, where a single rule $VP \rightarrow V PP\ PP \dots$ PP captures multiple PPs modifying a verb. For example, Clarissa’s treebank has the following tree



Nathan sees Clarissa’s trees, and insists she has made a big mistake in making this choice. He suggests an alternative treatment of multiple PPs, which is often referred to as “Chomsky adjunction”. In his annotation style, we would have the following tree:



Notice that we now have several VP levels, and that the rule $VP \rightarrow VP\ PP$ is used to introduce each PP modifier.

Question 3(a): Describe the pros and cons of each annotation scheme. For each scheme you should list one pro and one con. Your answer should be 500 words at most in length.

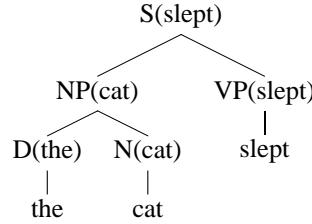
Question 4 (20 points)

We will refer to a “lexicalized PCFG” in Chomsky normal form, as a PCFG $G = (N, \Sigma, R, S, P)$ similar to that in question 1, where each of the rules in R takes one of the following three forms:

- $X(h) \rightarrow Y_1(h) Y_2(w)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$.
e.g., $\text{NP}(\text{man}) \rightarrow \text{NP}(\text{man}) \text{ PP}(\text{with})$.
- $X(h) \rightarrow Y_1(w) Y_2(h)$ for $X \in N$, and $Y_1, Y_2 \in N$, and $h, w \in \Sigma$.
e.g., $\text{S}(\text{snores}) \rightarrow \text{NP}(\text{man}) \text{ VP}(\text{snores})$.
- $X(h) \rightarrow h$ for $X \in N$, and $h \in \Sigma$
e.g., $\text{NP}(\text{man}) \rightarrow \text{man}$.

Here the symbols in the grammar rules are of the form $X(h)$, where X is a symbol such as NP, VP, etc., and h is a lexical item such as *man*, *snores*, etc.

In addition, for any symbol of the form $X(h)$, there is a probability $P_S(X(h))$ which is the probability of $X(h)$ being chosen as the root of a parse tree. As one example, the tree



would have probability

$$\begin{aligned}
 & P_S(\text{S}(\text{slept})) \times \\
 & P(\text{S}(\text{slept}) \rightarrow \text{NP}(\text{cat}) \text{ VP}(\text{slept}) | \text{S}(\text{slept})) \times \\
 & P(\text{NP}(\text{cat}) \rightarrow \text{D}(\text{the}) \text{ N}(\text{cat}) | \text{NP}(\text{cat})) \times \\
 & P(\text{D}(\text{the}) \rightarrow \text{the} | \text{D}(\text{the})) \times \\
 & P(\text{N}(\text{cat}) \rightarrow \text{cat} | \text{N}(\text{cat})) \times \\
 & P(\text{VP}(\text{slept}) \rightarrow \text{slept} | \text{VP}(\text{slept}))
 \end{aligned}$$

Question 4(a): Describe a dynamic programming algorithm, similar to the one in lecture, which finds the highest scoring parse tree under a grammar of this form. Your algorithm should make use of a dynamic programming table $\pi[i, j, k, X]$ where

$$\pi[i, j, k, X] = \text{highest probability for any parse tree whose root is the symbol } X(w_k), \text{ and which spans words } i \dots j \text{ inclusive}$$

For example, if the sentence being parsed is $w_1, w_2, \dots, w_6 = \text{the cat sat on the mat}$, then $\pi[4, 6, 4, \text{PP}]$ would store the maximum probability for any parse tree whose root is *PP*(on), and which spans the string *on the mat*.

Note: your algorithm should also allow recovery of the parse tree which achieves the maximum probability.

Question 4(b): What is the running time of your algorithm?

Question 5 (10 regular points, 10 bonus points)

Recall the definition of a PCFG in Chomsky normal form from question 1. Now assume we have a probabilistic CFG G' , which has a set of rules R which take one of the three following forms:

1. $X \rightarrow Y_1 Y_2$ for $X \in N$, and $Y_1, Y_2 \in N$
2. $X \rightarrow Y$ for $X \in N$, and $Y \in \Sigma$
3. $X \rightarrow Y$ for $X \in N$, and $Y \in N$

Note that this is very similar to a Chomsky normal form grammar, but that we are now allowed rules of form (3), such as $S \rightarrow VP$, where there is a single symbol on the right-hand-side of the rule, and this symbol is a non-terminal. We will refer to these new rules as **unary productions**. (Note that productions of the form in (2), such as $N \rightarrow \text{dog}$, will *not* be referred to as unary productions, as their right-hand-side is a terminal symbol.) We will refer to rules captured by cases (1) and (2) as **non-unary productions**.

As one example, the following grammar contains unary productions:

S	\rightarrow	NP	VP	0.7
S	\rightarrow	SBAR	VP	0.3
VP	\rightarrow	Vi		0.4
VP	\rightarrow	Vt	NP	0.4
VP	\rightarrow	V3	SBAR	0.2
NP	\rightarrow	NN		0.3
NP	\rightarrow	DT	NN	0.7
SBAR	\rightarrow	COMP	S	0.6
SBAR	\rightarrow	S		0.4

Vi	\rightarrow	sleeps	1.0
Vt	\rightarrow	saw	1.0
V3	\rightarrow	said	1.0
NN	\rightarrow	man	0.7
NN	\rightarrow	woman	0.2
NN	\rightarrow	telescope	0.1
DT	\rightarrow	the	1.0
COMP	\rightarrow	that	1.0

In this question, we'll attempt to convert a PCFG with unary productions into an “equivalent” PCFG which is in Chomsky normal form (note that we'll have to be careful with what we mean by “equivalent”, we'll come to this shortly).

As a first step, we will use the classic transformation for (non-probabilistic) context-free grammars, that results in a new grammar that accepts the same set of strings as the original grammar, but which has all unary productions removed. Applying this transformation to the grammar above results in the CFG shown in figure 1 (note that the probabilities are missing – we'll fill them in soon).

This grammar transformation works in the following way. We form a new grammar G' from an existing grammar G by first taking all non-unary rules from G . Then, if there is any sequence of n productions in G

$$B_0 \rightarrow B_1 \rightarrow B_2 \dots B_{n-1} \rightarrow \alpha$$

such that $B_i \rightarrow B_{i+1}$ for $i = 0 \dots n - 2$ are unary productions in the grammar, and $B_{n-1} \rightarrow \alpha$ is a non-unary production, then we add the rule

$$B_0 \rightarrow \alpha$$

to G' . For example, in the above example we have the sequence

$$\text{SBAR} \rightarrow S \rightarrow \text{NP VP}$$

S	→	NP	VP
S	→	SBAR	VP
VP	→	sleeps	
VP	→	Vt	NP
VP	→	V3	SBAR
NP	→	man	
NP	→	woman	
NP	→	telescope	
NP	→	DT	NN
SBAR	→	COMP	S
SBAR	→	NP	VP
SBAR	→	SBAR	VP
Vi	→	sleeps	
Vt	→	saw	
V3	→	said	
NN	→	man	
NN	→	woman	
NN	→	telescope	
DT	→	the	
COMP	→	that	

Figure 1: A transformed grammar, G'

so we add the rule $\text{SBAR} \rightarrow \text{NP VP}$ to G' . As another example, we have the sequence

$$\text{VP} \rightarrow \text{Vi} \rightarrow \text{sleeps}$$

so we add the rule $\text{VP} \rightarrow \text{sleeps}$ to G' .

We now come to the definition of equivalence. We will say that the PCFG G' is “equivalent” to a PCFG G if:

- For any string w , if $T(w)$ is the highest probability parse tree for w under the grammar G , and $T'(w)$ is the highest prob. parse under G' , then these two parse trees have the same probability under their respective grammars.
- There is a function f such that $T(w) = f(T'(w))$. i.e., there is a function such that the highest probability parse tree in the original grammar can be recovered from the highest probability parse tree under G' .
- In some cases, we will allow the PCFG G' to be **deficient**. This means that we will relax the requirement on probabilities on rules to satisfy the condition $\forall X \in N, \sum_{X \rightarrow \alpha \in R} P(X \rightarrow \alpha) < 1$ rather than $\forall X \in N, \sum_{X \rightarrow \alpha \in R} P(X \rightarrow \alpha) = 1$, as in the definition in question 1.

Question 5(a): (10 regular points) Add probabilities to the grammar in figure 1, so that the new PCFG G' is equivalent to the old PCFG G . Describe the function f that maps a parse tree in G' to a parse tree in G .

Question 5(b): (10 bonus points) Describe a strategy for creating an equivalent PCFG G' for *any* PCFG G in the form described at the start of this question (i.e., a PCFG that may have unary productions in addition to Chomsky normal form rules). You may assume that for any unary rule, its probability is strictly less than 1. Describe also the function f used to recover the highest probability tree under G from the highest probability tree under G' . Note that your resulting PCFG G' may be deficient in some cases. Illustrate your transformation on the two PCFGs shown in figure 2 (these grammars will help you, in terms of illustrating some “tricky” cases that you’ll run into with unary productions). **Hint: remember throughout this question that the goal of G' is to allow recovery of the maximum probability parse under G .**

Grammar 1:

S	→	NP	VP	0.7
S	→	SBAR	VP	0.3
VP	→	Vi		0.4
VP	→	Vt	NP	0.4
VP	→	V3	SBAR	0.2
NP	→	NN		0.3
NP	→	DT	NN	0.7
SBAR	→	COMP	S	0.4
SBAR	→	S		0.5
SBAR	→	NP	VP	0.1

Vi	→	sleeps	1.0
Vt	→	saw	1.0
V3	→	said	1.0
NN	→	man	0.7
NN	→	woman	0.2
NN	→	telescope	0.1
DT	→	the	1.0
COMP	→	that	1.0

Grammar 2:

S	→	NP	VP	0.7
S	→	SBAR	VP	0.2
S	→	SBAR		0.1
VP	→	Vi		0.4
VP	→	Vt	NP	0.4
VP	→	V3	SBAR	0.2
NP	→	NN		0.3
NP	→	DT	NN	0.7
SBAR	→	COMP	S	0.4
SBAR	→	S		0.5
SBAR	→	X		0.1
X	→	S		0.1
X	→	NP	NP	0.9

Vi	→	sleeps	1.0
Vt	→	saw	1.0
V3	→	said	1.0
NN	→	man	0.7
NN	→	woman	0.2
NN	→	telescope	0.1
DT	→	the	1.0
COMP	→	that	1.0

Figure 2: Two grammars with unary productions