Assignment 6, Part 4: Model Merging (Non-Computational Credit)
Assigned Thursday, March 11th
Due Monday, March 29th, 11:59pm

Model merging is a technique for building complex structured models of some process from data. The idea behind model merging is to begin with a completely unstructured model and then to incrementally modify it to reflect regularities in the data. In doing this, there is a tradeoff between the simplicity of the model and how well it accounts for the data.

In this problem you will apply a simplified version of the model merging algorithm to the task of learning right-regular grammars.

1 Right-regular grammars

- A *grammar* is a tuple $G = (Nonterminals, Terminals, Rules, StartSymbol)$, where $Nonterminals$ or $Terminals$ are sets of symbols used to write the rules in $Rules$, and $StartSymbol$ is a unique start symbol from $Nonterminals$. Each *nonterminal* symbol can be expanded according to any rule in the grammar that has that symbol on the left side. *Terminal symbols* cannot be expanded and appear only on the right side of rules in the grammar.
- A *right-regular* grammar is one in which every rule looks either like

  $$A \rightarrow x_1\, x_2\, \ldots\, x_n\, B$$

  or like

  $$A \rightarrow x_1\, x_2\, \ldots\, x_n$$

  where the x_i are terminal symbols, and A and B are nonterminal symbols. That is, each rule has one nonterminal symbol on the left side; its expansion on the right side contains any number of terminal symbols followed by at most one nonterminal symbol.

An example grammar G_0 has $Nonterminals$ = {S,Y}, $Terminals$ = {the, dog, ran, away, down, the, street}, $StartSymbol$ = S, and $Rules$ consisting of the following:

S	→	the dog ran Y
Y	→	away
Y	→	down the street

G_0 produces exactly two distinct sentences.

2 The model merging algorithm

A simplified greedy version of the model merging algorithm presented in class can be applied to learn a right-regular grammar G that produces a given set of sentences, $Data$. The algorithm has two main procedures:

- **Data incorporation**: Given a sentence from $Data$, add a set of rules to the grammar that generates this sentence from the start symbol. For example, the sentence 'the boy eats carrots' would be incorporated by adding the rule

  S → the boy eats carrots

- **Merging**: Find and perform a merge of grammar rules that improves the grammar by decreasing its *cost*; stop when no such merge can be found. The next two sections describe how merging works (Section 3) and how the cost of a grammar is calculated (Section 4).

The algorithm proceeds by alternating between incorporation and merging as follows:

1. Initialize the grammar to be empty (i.e., contain no rules).
2. Incorporate the next sentence from $Data$ by adding the necessary rules to the grammar.
3. If the next possible merge decreases the cost of the grammar on the (seen) data, perform it.
4. If there are still possible merges left, go to step 3.
5. If there are sentences in $Data$ that haven't yet been incorporated, go to step 2.

3 Merging operations

We will incrementally improve the grammar by using two types of merge operations:

- A *prefix merge* operates on a set of rules with the same left side, and whose right sides share a common prefix. For example, the rules

$$\begin{array}{lcl} N & \rightarrow & a\ b\ c\ d \\ N & \rightarrow & a\ b\ c\ e\ X \\ N & \rightarrow & a\ b\ c\ f\ g \end{array}$$

  would be replaced, in a prefix merge, by:

$$\begin{array}{lcl} N & \rightarrow & a\ b\ c\ N' \\ N' & \rightarrow & d \\ N' & \rightarrow & e\ X \\ N' & \rightarrow & f\ g \end{array}$$

- A *suffix merge* operates on a set of rules whose right sides share a common suffix. For example, the rules

$$\begin{array}{lcl} N_1 & \rightarrow & a\ b\ c\ d\ e \\ N_2 & \rightarrow & f\ c\ d\ e \end{array}$$

  would be replaced, in a suffix merge, by:

$$\begin{array}{lcl} N_1 & \rightarrow & a\ b\ N' \\ N_2 & \rightarrow & f\ N' \\ N' & \rightarrow & c\ d\ e \end{array}$$

4 Calculating the effect of merging

The full-fledged Bayesian version of the model merging algorithm is based on *maximizing* the *posterior probability* of the model given the data. In this assignment, however, we use a simpler *cost* function, which we attempt to *minimize*. Since we take a greedy approach to model merging, any merge that decreases the cost is accepted.

The *cost* of a grammar G given data D is defined as:

$$c(G) = \alpha s(G) + d(G, D)$$

Here is an explanation of the terms in this function:

- $s(G)$ denotes the *size* of the grammar, and is equal to the number of symbols (i.e., terminals plus nonterminals) occurring on the right side of rules in the grammar, counting repeats. So, for example, the size of the simple grammar G_0 from Section 1 would be 8.
- $d(G, D)$ denotes the *total derivation length* of the data given this grammar, which is a reflection of how well the model accounts for the data. The derivation length of a sentence is defined to be the number of rules that have to be applied to the start symbol S to produce the sentence. The total derivation length of the data is the sum of the derivation lengths of all the sentences.
- α is a constant indicating the importance of the grammar size (relative to the total derivation length) and can be seen as controlling the algorithm's tendency to generalize. (A smaller grammar tends to generalize better to new examples, and a larger α leads to a greater cost being associated with large grammars.)

One of the good things about this cost function is that it allows us to efficiently calculate the effect of different merges without actually applying them and reparsing the data each time.

Problem 1. Cost change for prefix merges

Consider the set of rules

$$\begin{array}{lcl} N & \rightarrow & x_1\, x_2\, \ldots\, x_k\, y_{11}\, y_{12}\, \ldots\, y_{1l_1} \\ N & \rightarrow & x_1\, x_2\, \ldots\, x_k\, y_{21}\, y_{22}\, \ldots\, y_{2l_2} \\ & \ldots & \\ N & \rightarrow & x_1\, x_2\, \ldots\, x_k\, y_{m1}\, y_{m2}\, \ldots\, y_{ml_m} \end{array}$$

(Don't be put off by the notation — it's just a precise way of expressing that these m rules share a common prefix of length k, followed by strings of varying length l_i.)

(a) Using this notation, write down the new rules that would result from applying a prefix merge.

(b) Calculate the change in size resulting from this merge (your answer should depend on k and m but not on the l_i).

(c) Assume now that the i^{th} rule in the original grammar was used r_i times when parsing the data. Calculate the number of times each rule in the modified grammar would be used.

(d) Use the result of part (c) to calculate the change in the total derivation length that would result from this merge.

(e) Show that the total change in cost is $\alpha(k + 1 - mk) + \sum_{i=1}^{m} r_i$.

Problem 2. Cost change for suffix merges

We can do the same thing for suffix merges. Consider the rules

$$\begin{array}{lcl} N_1 & \rightarrow & y_{11}\, y_{12}\, \ldots\, y_{1l_1}\, x_1\, x_2\, \ldots x_k \\ N_2 & \rightarrow & y_{21}\, y_{22}\, \ldots\, y_{2l_2}\, x_1\, x_2\, \ldots x_k \\ & \ldots & \\ N_m & \rightarrow & y_{m1}\, y_{m2}\, \ldots\, y_{ml_m}\, x_1\, x_2 \ldots x_k \end{array}$$

Perform a similar analysis as in the previous problem and show that the change in cost after applying a suffix merge to these rules is $\alpha(m + k - mk) + \sum_{i=1}^{m} r_i$.

Problem 3. Example grammar

Consider the following grammar:

$$\begin{aligned}
S &\rightarrow \text{The cat went Y} && (1)\\
S &\rightarrow \text{The cat slept} && (2)\\
S &\rightarrow \text{The cat caught X} && (3)\\
S &\rightarrow \text{Mary caught X} && (4)\\
S &\rightarrow \text{Mary went Y} && (5)\\
S &\rightarrow \text{The company went Y} && (6)\\
X &\rightarrow \text{a mouse} && (7)\\
X &\rightarrow \text{the ball} && (8)\\
X &\rightarrow \text{a cold} && (9)\\
Y &\rightarrow \text{into the hat} && (10)\\
Y &\rightarrow \text{nuts} && (11)\\
Y &\rightarrow \text{home} && (12)\\
Y &\rightarrow \text{bankrupt} && (13)
\end{aligned}$$

and suppose the data consists of the following sentences:

```
The cat slept
Mary went home
Mary caught a cold
The company went bankrupt
The cat caught a mouse
The cat went into the hat
The cat went nuts
Mary caught the ball
```

(a) Calculate the current cost of this grammar with this data, assuming that $\alpha = 1$.

(b) Assume that $\alpha = 1$. Find, if possible, a prefix merge that reduces the total cost of the grammar. Describe the merge (indicate which rules are affected and show the resulting merged rule) and calculate the change in the cost. If you cannot find one that reduces the cost, do this for the least harmful prefix merge that you can find (i.e., the one that increases the cost the least).

(c) Find, if possible, a suffix merge that reduces the total cost of the grammar. Again, assume $\alpha = 1$, and show the best suffix merge (which rules are involved, resulting merged rule, change in cost).

(d) Repeat parts (a) and (b) with $\alpha = 5$. Comment on why the results differ.