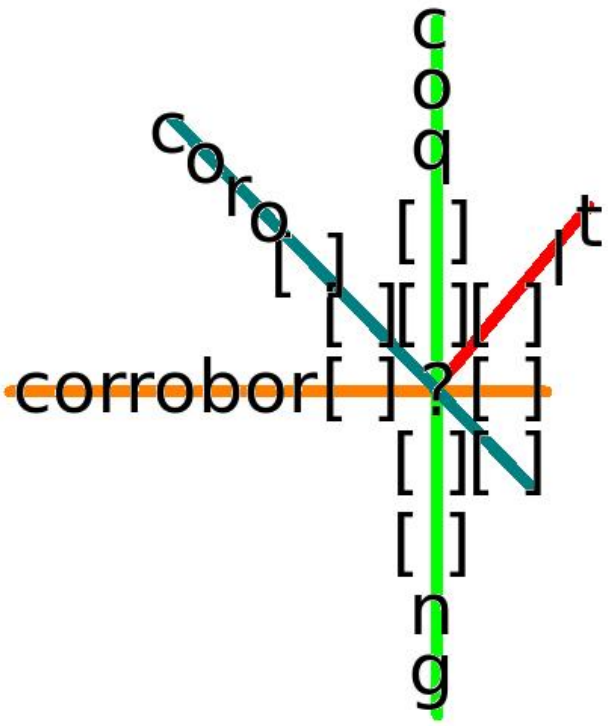
Project 03: Word Cross (100 pts. Due Thur Nov 20 11:59PM)



What's the most possible letter at "?"

What to submit:

You will implement question1_solver.py to question5_solver.py during the assignment. You should submit these five files along with a report. Your code should be well-documented. All the project submissions must be made through Blackboard.

Report (10 points) This should include the description and running time of your algorithm. Specifically, eeport the output of

python wordCross.py -q1 -q2 -q3 -q4 -q5

Demo with TA (10 points) On Friday Nov. 21th, one member in each group will be randomly selected to schedule a 5 minutes' demo meeting with the TA. In the meeting you need to run the project and explain your algorithm to the TA.

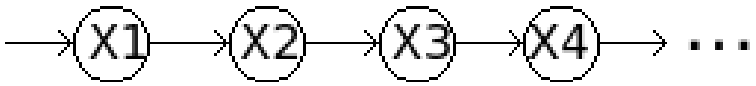
Evaluation Your code will be graded for technical correctness. Please *do not* change the names of any provided functions or classes within the code. If necessary, we will review and grade assignments individually to ensure that you receive due credit for your work.

Academic Dishonesty: We will be checking your code against other submissions in the class for logical redundancy. We trust you all to submit your own work only - own means you and your team member - ; If we find that you copied you will get a F for the course.

Getting Help: Feel free to use the Blackboard discussion board to discuss or get clarifications on homework-related issues. If you find yourself stuck on something, go to office hours or email the TA for help.

Introduction

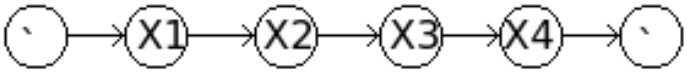
In this project, you are going to model the joint distribution of english letters that form a word. We assume the letters form a graphical model as shown below:



Pr(X) = Pr(X1) Pr(X2|X1) Pr(X3|X2) ...

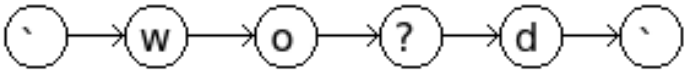
Where X1, X2, X3, ..., represent random variables that can be "a", "b", "c", ..., or "z". To make things easier, also assume the distribution Pr(Xi|X(i-1)=x) = Pr(Xj|X(j-1)=x) for every i, j, x. In other words, the Conditional Probability Table (CPT) is the same for every edge between X(i-1) and Xi. For example, Pr(X=abcd) = Pr(X1=a) Pr(X2=b|X1=a) Pr(X3=c|X2=b) Pr(X4=d|X3=c) = Pr(a) Pr(b|a) Pr(c|b) Pr(d|c); This graphical model is called the **"Markov Chain"**.

Additionally, to distinguish the first and last letter, we explicitly add "`" to the boundary of the word and set Pr(`) = 1. For example, a word of length 4 is represented by the following graphical model:



Pr(X) = Pr(X1|`) Pr(X2|X1) Pr(X3|X2) Pr(X4|X3) Pr(`|X4)

Your job is to return the most possible missing letter, given an incomplete word:



What's the most possible missing letter? It should be "r" in this case.

The most possible missing letter x is the one that yields the maximum Pr(X)=Pr(w|`) Pr(o|w) Pr(?|o) Pr(d|?) Pr(`|d).

Question 1 (10 points)

To return the most possible missing letter, please implement question1_solver.py, treating a word of length n as a graphical model with n + 2 variables. The CPT is given and can be shown by:

```
python wordCross.py --print_cpt
```

Each row of the table adds up to 100%.

To test your implementation, run

```
python wordCross.py -q1 --test
```

The function solve(self, query) will be called 7 times, because there are 7 queries in question1_test.txt. The rate of correct returns of the function is shown by the program by the "accuracy" term. You should have at most 1 error in these 7 queries.

The following command runs you implementation on question1.txt

python wordCross.py -q1

The accuracy should be above 0.25 in order to get points. Include the accuracy and time usage in your report (same for all the questions).

Question 2 (15 points)

To return the most possible missing two consecutive letters, please implement question2_solver.py. Run the following commands to test your algorithm:

python wordCross.py -q2 --test

python wordCross.py -q2

For the first command, you should have at most 1 error in the test case. For the second command, the accuracy should be above 0.33 (at least one correct letter) in order to get points.

Question 3 (20 points)

To return the most possible missing letter, please implement question3_solver.py. Run the following commands to test your algorithm:

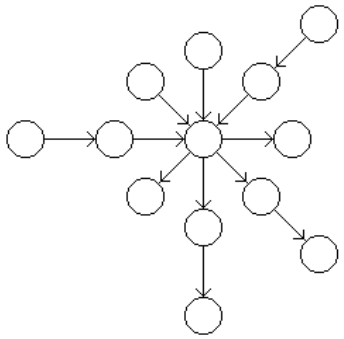
python wordCross.py -q3 --test

python wordCross.py -q3

For the first command, you should have at most 1 error in the test case. For the second command, the accuracy should be above 0.12 in order to get points. Notice that you only need to return the most possible missing letter at "_". You can treat "-" as unkown (hidden) variables (letters). You must do variable elimination in order to get full points. Otherwise your program will be too slow for the large amount of queries.

Question 4 (20 points)

In this question, we assume the following graphical model:



We assume the graphical model above, and

$$Pr(X|P, Q, J, K) = C\ Pr(X|P)\ Pr(X|Q)\ Pr(X|J)\ P(X|K).$$

Where C is a normalizing constant, X is the center variable, P, Q, J, K are the parents of X.

You can ignore C totally because what we need is the relative comparison.

To return the most possible missing letter, please implement question4_solver.py. Run the

following commands to test your algorithm:

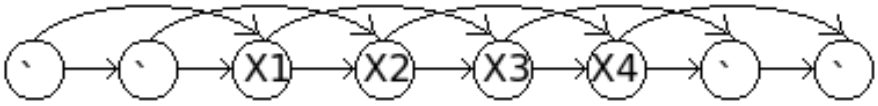
python wordCross.py -q4 --test

python wordCross.py -q4

For the first command, you should have at most 1 error in the test case. For the second command, the accuracy should be above 0.30 in order to get points. Notice that you only need to return the most possible missing letter at "_". You can treat "-" as unkown (hidden) variables (letters). The letter at "_" is shared by four words (please refer to the banner image). You must do variable elimination in order to get full points. Otherwise your program will be too slow for the large amount of queries.

Question 5 (15 points)

In this question, we use the following **"Second Order Markov Chain"**:



We assume the graphical model above.

The second order CPT is given by the program.

To return the most possible missing letter, please implement question5_solver.py. Run the following commands to test your algorithm:

python wordCross.py -q5 --test

python wordCross.py -q5

For the first command, you should have at most 1 error in the test case. For the second command, the accuracy should be above 0.40 in order to get points.