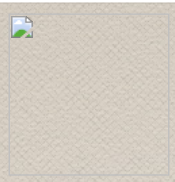


CS 440 Programming Assignment 3:
Hidden Markov Models and Natural Language Processing

Boston University
Instructor: Prof. Margrit Betke
Departments of Computer Science and Mathematics
Wei Wei
Teammates: Yehui Huang, Zhou Xiao, Tianqi Xu
4/5/2016

Problem Definition

In this assignment, we build a basic English sentence recognizer based on hidden Markov models ("HMMs"), which is expected to recognize and parse sentences that use the certain vocabulary.

Method and Implementation

1. Pattern Recognition: Report the observation probability of each input sequence
    1. Retrieve information from the input files (sentence.hmm, example[x].obs) (note that [x] in example[x] is a number
    2. Computes the probabilities for each observations from 'example[x].obs' by forward algorithm and print these probabilities
2. State-Path Determination: Determine the optimal state path for each observation set and report its probability. Viterbi algorithm is used in this part.
    1. Retrieve information from the input files (sentence.hmm, example[x].obs) (note that [x] in example[x] is a number
    2. function viterbi():

        This function uses Viterbi algorithm to find the optimal path and its probability.
3. Model Optimization: Optimize the HMM and report the probabilities before and after optimization with the help of Baum-Welch algorithm
    1. function sentence_info():

        Read info from designated input file ('sentence.hmm')
    2. function example_info():

        Read info from designated input file ('example[x].obs')
    3. function forward():

        helper function that performs the specialized forward operations
    4. function backward():

helper function that performs the backward operations

5. function update_Pi():

   helper function that computes Pi (Initial State Distribution)

6. function lamda():

   helper function that calculates lambda

7. function gamma():

   helper function that calculates gamma

8. function update_A():

   helper function that update A (Transition Matrix)

9. function update_B():

   helper function that updates B (Emission Matrix)

10. function re_estimation():

    Performs the re-estimation process

11. function prob():

    helper function that calculates the probability for a specific observations

12. function write_to():

    Write outputs to a designated file

Experiments

Experiment number: 6

Variables/inputs:

#1: sentence.hmm (filename, used for HMM training)

#2: example[x].obs (filename, used for computation of probability or updates)

#3: [filename].hmm (filename, used for storing the updated variables of HMM)

The results of experiments match with the provided sample outputs given in the homework requirements

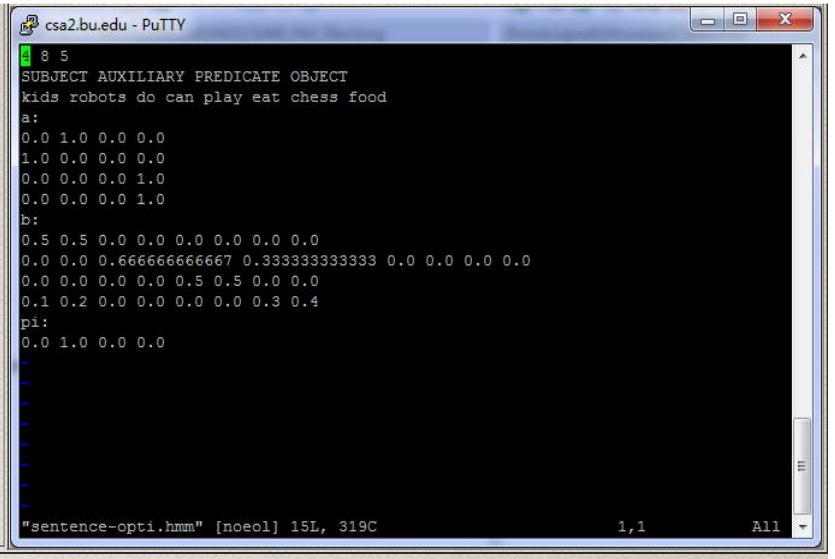
---

Results

In all experiments, where the variables were correctly entered, this program runs without break

Results Table		
Test runs	Inputs & Results/Outputs	Written file
Test 1 for recognize.py		N/A

	[zhouxiao@csa2:~/CS440/PA3]$ python recognize.py sentence.hmm example1.obs 0.027 0.0288 0.0	
Test 2 for recognize.py	[zhouxiao@csa2:~/CS440/PA3]$ python recognize.py sentence.hmm example2.obs 0.000588	N/A
Test 1 for statepath.py	[zhouxiao@csa2:~/CS440/PA3]$ python statepath.py sentence.hmm example1.obs 0.027 SUBJECT PREDICATE OBJECT 0.0288 SUBJECT PREDICATE OBJECT 0.0	N/A
Test 2 for statepath.py	[zhouxiao@csa2:~/CS440/PA3]$ python statepath.py sentence.hmm example2.obs 0.000588 AUXILIARY SUBJECT AUXILIARY SUBJECT AUXILIARY	N/A
Test 1 for optimize.py	[zhouxiao@csa2:~/CS440/PA3]$ python optimize.py sentence.hmm example1.obs sentence-optimized.hmm 0.027 0.027 0.0288 0.0288 0.0 0.0	csa2.bu.edu - PuTTY 8 4 SUBJECT AUXILIARY PREDICATE OBJECT kids robots do can play eat chess food a: 0.0 0.4 0.6 0.0 0.7 0.0 0.3 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 b: 0.5 0.4 0.0 0.0 0.0 0.0 0.05 0.05 0.0 0.0 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.1 0.2 0.0 0.0 0.0 0.0 0.3 0.4 pi: 0.6 0.3 0.1 0.0 "sentence-optimized.hmm" [noeol] 15L, 299C 1,1 All
Test 2 for optimize.py	[zhouxiao@csa2:~/CS440/PA3]$ python optimize.py sentence.hmm example2.obs sentence-optimized.hmm 0.000588 0.037037037037	csa2.bu.edu - PuTTY 8 5 SUBJECT AUXILIARY PREDICATE OBJECT kids robots do can play eat chess food a: 0.0 1.0 0.0 0.0 1.0 0.0 0.0 0.0 0.0 0.0 0.0 1.0 0.0 0.0 0.0 1.0 b: 0.5 0.5 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.666666666667 0.333333333333 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.5 0.5 0.0 0.0 0.1 0.2 0.0 0.0 0.0 0.0 0.3 0.4 pi: 0.0 1.0 0.0 0.0 "sentence-opti.hmm" [noeol] 15L, 319C 1,1 All

Test 3 for optimize.py	[zhouxiao@csa2:~/CS440/PA3]$ python optimize.py sentence.hmm example2.obs sentence-opti.hmm 0.000588 0.037037037037	

Discussion

Discuss your method and results:

1. Question: For the current application, why does this probability seem lower than we expect? What does this probability tell you? Does the current HMM always give a reasonable answer? For instance, what is the output probability for the below sentence?

   "robots do kids play chess"

   "chess eat play kids"

   Answer: Because the transition and emission matrices are not accurate enough. This probability tells us although the HMM can recognize basic grammar patterns, it still can't distinguish between logic orders. The current model doesn't always give a reasonable answer. The outputs are 0.001512 and 0.0

2. Question: What can we tell from the reported optimal path for syntax analysis purpose? Can the HMM always correctly distinguish "statement" from "question" sentence? Why?

   Answer: We can observe the arrangement of the syntax which allows us to analyize and distinguish different states(grammar checking) Since the HMM can correctly performing grammar checking, as long as it discovers that the first state of the observation is auxilliry, then it knows that it's a question sentence, otherwise it's a normal statement.

3. Question: Why should you not try to optimize an HMM with zero observation probability?

   Answer: If we try to optimize an HMM with zero observation probability, since certains states are never reached during the training, then at some time t, forward_t(i) and backward_t(i) for i in range(N) will be equal to 0 at that point, and hence the P(O|lambda) will be 0. Apparently we can't apply a 0 as the denominator to the gamma and xi function, which leads to an output of 0 for that state in transition and emission matrix, that's why P(O|lambda) = 0 HMM can't be optimized.

4. Question: What kinds of changes will you need to make in the above HMM? Please describe your solution with an example of the modified matrices a, b and pi in the submitted web page.

   Answer: we need to add states and observations. for each n states we want to add the the HMM, we need to expand a's rows and colomns both by n, length of pi by n, and b's rows by n. for o observations we add to the HMM, we need to expand b's colomns by o.

   As an example , consider this new sentence.hmm, which is an example where we add "Adverb" to states and "good" and "hardly" to observations. Note that "good" can also be "Subject" or "object" (b[0][8] and b[3][8])

Conclusions

HMMs might be helpful when trying to diagonize the english or some other languages in some cases, on the other hand, trainning with multiple observation sequences is critical if we want to have satisfying results

This assignment challenges our understanding of algorithms realted to HMM and python coding skills. After finished this assignment, we have learn a lot and will try some more methods in the future.

Credits and Bibliography

References:

Notes from Professor Betke's class

A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition by Rabiner [1989]

Discussed with teammates

Contributions:

Yehui Huang: State-Path determination (statepath.py) & Optimization (optimize.py)

Tianqi Xu : Report & Optimization (optimize.py)

Wei Wei : Forward part (recognize.py) & Optimization (optimize.py)

Xiao Zhou : Report & Optimization (optimize.py)

Back