CSE 537 - Artificial Intelligence

Report: Project 3

Khan Mostafa **Abhijit Betigeri**
109365509 109229784

{khan.mostafa, abhijit.betigeri}@stonybrook.edu
Department of Computer Science
Stony Brook University

Word Cross: Finding the missing letter in an incomplete word

Q1. Returning the most possible missing letter

Methodology Used: Using the CPT table given. Visualizing the query set as a Markov Chain of the form a -> B -> c. B being blank position under consideration to be found out. Probability for each assignment to B is computed by considering each of the ascii lowercase characters including (`) and then returning most likeliest character with highest conditional probability calculated with a,c & set of ascii characters.

Probability Evaluation Function: P (b, a) * P (c, b)

Execution Details

Command 1:

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

***Accuracy*: 1.000000**
***Time used:* 0.0010000 secs**

Command 2:

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

Accuracy: 0.311102
Time used: 0.196000 secs.

Q2. Return the most possible missing two consecutive letters

Methodology Used: Using the CPT table given. Visualizing the query set as a Markov Chain of the form a -> B -> C -> d. B & C being blank position under consideration to be found out. Probability for each assignment to B & C is computed by considering each of the ascii lowercase characters including (`) and then returning most likeliest characters with highest conditional probability calculated with a,d & set of ascii characters.

Probability Evaluation Function: P (b, a) * P (c, b) * P (d, c)

Execution Details
Command 1:

```
python wordCross.py -q2 --test
```

Accuracy: 1.000000 (two correct letters)
Accuracy: 1.000000 (at least one correct letter)
Time used: 0.004000 secs.

Command 2:

```
python wordCross.py -q2
```

Accuracy: 0.089656 (two correct letters)
Accuracy: 0.360026 (at least one correct letter)
Time used: 6.077000 secs.

Q3. Return the most possible missing letter

Methodology Used: We Visualize the query as a Markov Chain of a->? -> C ->? ->e. C represents the letter we are looking for at (_). a & e are known/given letters in the query.

Query gu--_--n can be set up as following chain: u->??->C->??->n. The level of indirection between a='u' and C is 2, we represent it by h, similarly t for C and e.

Let's break down the problem this way: The probability of having letter C=c given a=a followed by two hidden variables a -> H1 -> c is given by Sum (P (H1|a) * P (c|H1) for all h=H. All the one and two level of hidden values are pre-computed this way and is accessible using the defined getConditionalProbability function (This function helps in getting the CPT based on the number of hidden variables.)

Then we compute the probability of each assignments of C from the set of all lowercase characters including ` and returning most likeliest character with highest conditional probability using probability evaluation function as defined below.

Probability Evaluation Function: P (h, c, a) * P (t, e, c)

Execution Details
Command 1:

```
python wordCross.py -q3 --test
```

Accuracy: 1.000000
Time used: 0.025000 secs

Command 2:

```
python wordCross.py -q3
```

Accuracy: 0.149693
Time used: 0.278000 secs

Q4. Return the most possible missing letter – Given Graphical Model

Methodology Used: We visualize the queries as finding the likely letter in the blank space of intersection of four words, having the knowledge of neighboring letters. Representation of the form:

a1->? ->C->? ->e1
a2->? ->C->? ->e2
a3->? ->C->? ->e3
a4->? ->C->? ->e4

where C represents letter in the blank intersection and a1, e1, a2, e2, a3, e3, a4, e4 are known letters of four words.
Similar to Q3, the probability of having the letter C=c, given a=a followed by two hidden letters a->H1->c is given by Sum (P (H1|a) * P (c|H1), for all h=H). All the one and two level of hidden values are pre-computed this way and is accessible using the defined getConditionalProbability function (This function helps in getting the CPT based on the number of hidden variables.).

Here we take the intersection of four letters, the probability of C=c is product of P (c|ai) * P (ei|c), for i= [1, 4], P is the probability of the hidden intermediate variables.

Then we compute the probability of each assignments of C from the set of all lowercase characters including ` and returning most likeliest character with highest conditional probability using probability evaluation function as defined below.

Probability Evaluation Function: P (h1, c, a1) * P (t1, e1, c) * P (h2, c, a2) * P (t2, e2, c) * P (h3, c, a3) * P (t3, e3, c) * P (h4, c, a4) * P (t4, e4, c)

Execution Details
Command 1:

```
python wordCross.py -q4 --test
```

Accuracy: 1.000000
Time used: 0.025000 secs

Command 2:

```
python wordCross.py -q4
```

Accuracy: 0.426571
Time used: 1.006000 secs

Q5. Based on Second Order Markov Chain- Return the most possible missing letter

Methodology Used:

We visualize the query as 2nd Order Markov Chain of the following form:

a->b->C->d->e

where C represent the letter to be found out. a,b,d,e are known letter of the word.

Eg: ques_ion => e->s->C->i->o

Probability for each assignment to C is computed by considering each of the ascii lowercase characters including (`) and then returning most likeliest character with highest conditional probability calculated with a,b,d,e & set of ascii characters.

Probability Evaluation Function: P (c, a, b) * P (d, b, c) * P (e, c, d)

Execution Details
Command 1:

```
python wordCross.py –q5 --test
```

Accuracy: 1.000000
Time used: 0.002000 secs

Command 2:

```
python wordCross.py –q5
```

Accuracy: 0.465159
Time used: 0.351000 secs

Appendix:

```
>python wordCross.py -q1 -q2 -q3 -q4 -q5 --test
Question 1 accuracy: 1.000000
Q1 time used: 0.001000 secs.

Question 2 accuracy: 1.000000 (two correct letters)
Question 2 accuracy: 1.000000 (at least one correct letter)
Q2 time used: 0.004000 secs.

Question 3 accuracy: 1.000000
Q3 time used: 0.025000 secs.

Question 4 accuracy: 1.000000
Q4 time used: 0.025000 secs.

Question 5 accuracy: 1.000000
Q5 time used: 0.002000 secs.


>python wordCross.py -q1 -q2 -q3 -q4 -q5
Question 1 accuracy: 0.311102
Q1 time used: 0.196000 secs.

Question 2 accuracy: 0.089656 (two correct letters)
Question 2 accuracy: 0.360026 (at least one correct letter)
Q2 time used: 6.077000 secs.

Question 3 accuracy: 0.149693
Q3 time used: 0.278000 secs.

Question 4 accuracy: 0.426571
Q4 time used: 1.006000 secs.

Question 5 accuracy: 0.465159
Q5 time used: 0.351000 secs.
```