

Assignment 2

CMSC 473/673 — Introduction to Natural Language Processing

Item	Summary
Assigned	Friday September 17th
Due	Wednesday September 29th, 11:59 PM Baltimore time
Topic	Probabilities and Count-based Estimation/Generation
Points	80 (+20 Extra Credit)

In this assignment you will gain additional experience with probabilities, implement more counting techniques, and use them to explore a corpus.

You are to *complete* this assignment on your own: that is, the code and writeup you submit must be entirely your own. However, you may discuss the assignment at a high level with other students or on the discussion board. Note at the top of your assignment who you discussed this with or what resources you used (beyond course staff, any course materials, or public Piazza discussions).

The following table gives the overall point breakdown for this assignment.

Question	1	2	3	4
Points	10	20	25	25 (+20 EC)

How you structure your code is up to you. Your code does not have to be pristine. However, we should be able to follow what you write. Ensure your code is appropriately commented and described in your writeup. Try to make the coding easy on yourself.

What To Turn In Turn in a writeup in PDF format that answers the questions; turn in all requested code necessary to replicate your results. Be sure to include specific instructions on how to build (compile) your code. Answers to the following questions should be long-form. Provide any necessary analyses and discussion of your results.

How To Submit Submit the assignment on the submission site:

<https://www.csee.umbc.edu/courses/undergraduate/473/f21/submit>

Be sure to select “Assignment 2.”

Full Questions

1. (10 points) Let's return to the dice rolling problem from A1Q2: let X_i be the random variable representing the i th roll of a six-sided die and let x_i be the rolled (observed) value of the i th roll. Assume that we roll the die N times and that each X_i is an i.i.d. sample.

- (A) Because each X_i is i.i.d., we can work with a single distribution $p(X = k)$. Define c_k as the number of times that we observed a k among our N rolls:

$$c_k = \sum_{i=1}^N (1 \text{ if } X_i = k \text{ otherwise } 0). \quad (1)$$

We define $p(X = k) = \frac{c_k}{Z}$, where Z is a constant that ensures p sums to one:

$$\sum_k p(X = k) = \sum_k \frac{c_k}{Z} = 1. \quad (2)$$

Compute (write a mathematical expression for) Z .

- (B) Describe, in a few sentences, the significance of this problem to NLP (e.g., what is the NLP analogy here?). In constructing your answer, think about what X_i represents, what c_k represents, and the implication for *computing* $p(X = k)$.
2. (20 points) We're going to continue with the die rolling problem. Let's say that *before* we rolled our die, we first flipped a fair coin (2 outcomes, where probability of heads equals probability of tails). Call this random variable Z_i . If $Z_i = H$ (heads), then we roll a uniformly weighted, six-sided die. If $Z_i = T$ (tails), then we roll a six-sided die where the probabilities of rolling each outcome is given by the probability vector $\pi = \left(\frac{1}{9} \quad \frac{2}{9} \quad \frac{1}{9} \quad \frac{2}{9} \quad \frac{1}{9} \quad \frac{2}{9} \right)$.

- (a) Write the conditional distributions $p(X | Z)$. You can think of this as filling out the table

$p(X Z)$	$X = 1$	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$
$Z = H$	$p(X = 1 Z = H) = ???$					
$Z = T$	$p(X = 1 Z = T) = ???$					

Hint: Think about how the conditional distributions $p(X|Z = H)$ and $p(X|Z = T)$ relate to the problem setup of flipping a coin Z and then based on that outcome, rolling a *particular* die.

- (b) Now write the joint distribution $p(X, Z)$. You can think of this as filling out the table

$p(X, Z)$	$X = 1$	$X = 2$	$X = 3$	$X = 4$	$X = 5$	$X = 6$
$Z = H$						
$Z = T$						

(Hint: Think about the number of different distributions you need to write out.)

- (c) One way we can define the marginal distribution of X_i is as

$$p(X_i = k) = \sum_j p(X_i = k | Z_i = j) p(Z_i = j). \quad (3)$$

(Though there may be equivalent ways of computing it!)

Call a flip of the coin and a subsequent roll of the appropriate die a *trial*. Unfortunately, while you recorded the outcome of the die, you forgot to record the outcome of the coin flip. Let's say I tell you that after four independent trials the die came up as 3, 1, 6, and 5. How likely are those trial outcomes? (That is, what are the marginal probability values $p(X_1 = 3)$, $p(X_2 = 1)$, $p(X_3 = 6)$, and $p(X_4 = 5)$?)

- (d) Now for each possible outcome k of X , compute the conditional distribution $p(Z|X = k)$. (*Hint*: As with the previous two sub-questions, think about the number of different distributions you need to write out.)
 - (e) Show, using the definition of probabilistic independence, whether X and Z are independent.
 - (f) For simplicity, we considered Z as a coin flip—this is equivalent to a two-sided die. We could have considered Z as some M -sided die instead. This generalization would not have fundamentally changed any of the answers to this problem (except make the tables in parts (a) and (b), and the sum in part (c), larger!). Describe, in a few sentences, how this (generalized) conditional die rolling problem can be relevant to an NLP task or problem. (*Hint*: there are multiple correct answers.)
3. **(25 points)** Read through the “Unigram Word Generation” section of this IPython notebook. It uses some very basic pytorch functions. You may use this Unigram Word Generation code, but you do not have to. Likewise, you may use torch for this question, but do not have to (you don’t even have to use Python, but I strongly encourage it).
- (a) Explain, in a few sentences, why we need the special ### symbol. (Do not just say “we use it stop generating,” or some paraphrase of that. Think about what the implications would be if we did not have it.)
 - (b) This example code is defined over four word types (three “normal” words, plus the special ###). The probabilities listed in the notebook are completely made up. If we wanted to generalize to all word types contained in the training set of the UD-EWT file, how many total vocab types would you define?
 - (c) Generalizing to the UD-EWT training file, explain a reasonable method of estimating the probabilities of each word type. Your explanation should be in English. Some math is okay: no code though. (*Hint*: think about how another question on this assignment may be relevant!)
 - (d) Now, implement your method from part 3c.
 - i. In your writeup, describe how you ensured your estimation method results in bona fide probability values. (That is, how did you test your code to ensure it meets the axioms?)
 - ii. In your writeup, list the twenty words that occur most commonly in the training file, their raw count, and your computed probability score for them.
 - (e) Using your probability estimates, generate 100 sentences.
 - i. Save these sentences to a text file and submit them in your artifact archive file.
 - ii. What is the average length (number of word tokens) of your generated sentences? How does this relate to your probability estimate of ###?
 - iii. In your opinion, does this generation method produce “reasonable” generated sentences (for this, you define “reasonable”)? Justify.
 - iv. Briefly compare and contrast some pros and cons of this generation method. Justify your answers. If you believe there are no pros (or, no cons), that’s okay: justify your answer.
4. **(25 points, +20 Extra Credit)** Read through the “Conditional Word Generation” section of the same IPython notebook from the previous question. As before, you may use this code as part of your solution, but you do not have to. Likewise, you may use torch for this question, but do not have to (you don’t even have to use Python, but I strongly encourage it).
- (a) Explain how we’re now using ###, and why we need to broaden its meaning.

- (b) Let the matrix A be the same matrix as `three_word_conditional_probs`: $A = \begin{pmatrix} 0.1 & 0.4 & 0.5 \\ 0.2 & 0.2 & 0.6 \\ 0.3 & 0.5 & 0.2 \end{pmatrix}$.

Let x be a one-hot vector corresponding to the vocab type a . Write out a formula using matrix vector multiplication between x and A that returns the conditional probability of each word type given a (i.e., $p(\cdot|a)$, where the “.” is a placeholder). Explain why this formula is correct.

- (c) Let A and x have the same meaning. Write out a formula using matrix vector multiplication between x and A that returns the conditional probability of a given each possible word type (i.e., $p(a|\cdot)$). Explain why this formula is correct.

- (d) As in question 3, the conditional probabilities are completely made up. Come up with a reasonable method of estimating those conditional probabilities from counts obtained from the sentences of the UD-EWT training file. Explain your method.

To handle this sentence processing, put boundaries at the start and end of each sentence. For example:

```
counts = ### you initialize appropriately
for sentence in ud_corpus:
    boundary_sent = add_boundaries_to(sentence)
    update_counts(counts, boundary_sent)
```

- (e) Now, implement your method from part 4d. **IMPORTANT:** Do this for only the 10,000 (or so) most frequent word types. If you do not limit it in this way, you will almost certainly use a *huge* amount of RAM (close to $\frac{1}{4}TB!$).

- i. How many conditional probability distributions are you computing?
- ii. In your writeup, describe how you ensured your estimation method results in bona fide probability values. (That is, how did you test your code to ensure all your computed distributions meet the axioms?)

- (f) Using your probability estimates, generate 1,000 sentences.

- i. Save these sentences to a text file and submit them in your artifact archive file.
- ii. In your opinion, does this generation method produce “reasonable” generated sentences (for this, you define “reasonable”)? How does it compare to the unigram generation? Justify.
- iii. Explain the root mathematical/computational cause behind needing to limit to the 10,000 most frequent types.
- iv. Briefly compare and contrast some pros and cons of this generation method. Justify your answers. If you believe there are no pros (or, no cons), that’s okay: justify your answer.

- (g) **(+20 Extra Credit)** Generalize the method in this problem to use the previous two words to generate the next one. This will involve generalizing the conditional probability matrix to a conditional probability tensor.

- i. Explain, in words, your generalization and how it works.
- ii. Implement your method. Depending on the machine you’re using, you’ll likely need to limit yourself to between the 500 most frequent word types (approx. 500 MB) and the 1,000 most frequent word types (approx. 4 GB).

- iii. Generate 1,000 sentences:

A. Save these sentences to a text file and submit them in your artifact archive file.

- B. In your opinion, does this generation method produce “reasonable” generated sentences (for this, you define “reasonable”)? How does it compare to the unigram generation? Justify.
- C. Briefly compare and contrast some pros and cons of this generation method. Justify your answers. If you believe there are no pros (or, no cons), that’s okay: justify your answer.