LING/COMP 445, LING 645
Problem Set 4

Name: Jan Tiegges , **McGill ID**: 261180937
Collaborators:

Due before 4:35 PM on Wednesday, November 8, 2023

Please enter your name and McGill ID above. There are several types of questions below.

- For questions involving answers in English or mathematics or a combination of the two, put your answers to the question in an answer box like in the example below. You can find more information about LaTeX here `https://www.latex-project.org/`.
- For programming questions, please put your answers into a file called `ps4-lastname-firstname.clj`. Be careful to follow the instructions exactly and be sure that all of your function definitions use the precise names, number of inputs and input types, and output types as requested in each question.

  For the code portion of the assignment, **it is crucial to submit a standalone file that runs**. Before you submit `ps4-lastname-firstname.clj`, make sure that your code executes correctly without any errors when run at the command line by typing `clojure ps4-lastname-firtname.clj` at a terminal prompt. We cannot grade any code that does not run correctly as a standalone file, and if the preceding command produces an error, the code portion of the assignment will receive a 0.

  To do the computational problems, we recommend that you install Clojure on your local machine and write and debug the answers to each problem in a local copy of `ps4-lastname-firstname.clj`. You can find information about installing and using Clojure here `https://clojure.org/`.

Once you have entered your answers, please compile your copy of this LaTeX into a PDF and submit

(i) the compiled PDF renamed to `ps4-lastname-firstname.pdf`
(ii) the raw LaTeX file renamed to `ps4-lastname-firstname.tex` and
(iii) your `ps4-lastname-firstname.clj`

to the Problem Set 4 folder under 'Assignments' on MyCourses.

Example Problem: This is an example question using some fake math like this $L = \sum_0^\infty \mathcal{G}\delta_x$.

Example Answer: Put your answer in the box provided, like this:

Example answer is $L = \sum_0^\infty \mathcal{G}\delta_x$.

Problem 1: In this problem set, we are going to be considering a variant of the hierarchical bag-of-words model that we looked at in class. In class, we used a Dirichlet distribution to define a prior distribution over θ, the parameter vector of the bag of words model. The Dirichlet distribution is a continuous distribution on the simplex—it assigns probability density to all the uncountably many points on the simplex.

For this problem set, we will be looking at a considerably simpler prior distribution over the parameters θ. Our distribution will be *discrete*, and in particular will only assign positive probability to a finite number of values of θ. Further, for the purposes of this problem set where we will only be scoring strings rather than generating them, we will ignore the probability of the `'stop` symbol.

The probability distribution is defined in the code below:

```
(def vocabulary '(call me ishmael))

(def theta1 (list (/ 1 2 ) (/ 1 4 ) (/ 1 4 )))
(def theta2 (list (/ 1 4 ) (/ 1 2 ) (/ 1 4 )))

(def thetas (list theta1 theta2))

(def theta-prior (list (/ 1 2) (/ 1 2)))
```

Our vocabulary in this case consists of three words. Each value of θ therefore defines a bag of words distribution over sentences containing these three words. The first value of θ (`theta1`) assigns $\frac{1}{2}$ probability to the word `'call`, $\frac{1}{4}$ to `'me`, and $\frac{1}{4}$ to `'ishmael`. The second value of θ (`theta2`) assigns $\frac{1}{2}$ probability to `'me`, and $\frac{1}{4}$ to each of the other two words. The two values of θ each have prior probability of $\frac{1}{2}$. **Assume throughout the problem set that the vocabulary and possible values of θ are fixed to these values above.**

In addition to the code above we will be using some helper functions defined in class:

```
(defn score-categorical [outcome outcomes params]
  (if (empty? params)
    (throw "no matching outcome")
    (if (= outcome (first outcomes))
      (first params)
      (score-categorical outcome (rest outcomes) (rest params)))))

(defn list-foldr [f base lst]
  (if (empty? lst)
    base
    (f (first lst)
       (list-foldr f base (rest lst)))))

(defn log2 [n]
  (/ (Math/log n) (Math/log 2)))

(defn score-BOW-sentence [sen probabilities]
  (list-foldr
   (fn [word rest-score]
     (+ (log2 (score-categorical word vocabulary probabilities))
        rest-score))
   0
   sen))

(defn score-corpus [corpus probabilities]
  (list-foldr
   (fn [sen rst]
```

```
    (+ (score-BOW-sentence sen probabilities) rst))
  0
  corpus))

(defn logsumexp [log-vals]
  (let [mx (apply max log-vals)]
    (+ mx
       (log2
        (apply +
               (map (fn [z] (Math/pow 2 z))
                    (map (fn [x] (- x mx)) log-vals)))))))
```

Recall that the function `score-corpus` is used to compute the log probability of a corpus given a particular value of the parameters θ. Also recall (from the Discrete Random Variables module) the purpose of the function `logsumexp`, which is used to compute the sum of log probabilities; you should return to the lecture notes if you don't remember what this function is doing. (Note that the version of `logsumexp` here differs slightly from the lecture notes, as it does not use the `&` notation, so it takes one argument, `log-vals`, a list of log probabilities.)

Our initial corpus will consist of two sentences:

```
(def my-corpus '((call me)
                 (call ishmael)))
```

Write a function `theta-corpus-joint`, which takes three arguments: `theta`, `corpus`, and `theta-probs`. The argument `theta` is a value of the model parameters θ, and the argument `corpus` is a list of sentences. The argument `theta-probs` is a prior probability distribution over the values of θ. The function should return the **log** of the joint probability $\Pr(C = \texttt{corpus}, \theta = \texttt{theta})$.

Use the chain-rule identity discussed in class: $\Pr(C, \theta) = \Pr(C|\theta)\Pr(\theta)$. Assume that the prior distribution $\Pr(\theta)$ is defined by the probabilities in `theta-probs`, which is a list containing the prior probability of each value of θ (that is, a list with two entries, one for the probability of `theta1` and one for `theta2`).

After defining this function, you can call `(theta-corpus-joint theta1 my-corpus theta-prior)`. This will compute log joint probability of the model parameters `theta1` and the corpus `my-corpus`.

Answer 1: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 2: Write a function `compute-marginal`, which takes two arguments: `corpus` and `theta-probs`. The argument `corpus` is a list of sentences, and the argument `theta-probs` is a prior probability distribution on values of θ. The function should return the **log** of the marginal likelihood of the corpus, when the prior distribution on θ is given by theta-probs. That is, the function should return $\log[\sum_{\theta\in\Theta}\Pr(\mathbf{C} = \texttt{corpus}, \Theta = \theta)]$.

Hint: Use the `logsumexp` function defined above.

After defining `compute-marginal`, you can call `(compute-marginal my-corpus theta-prior)`. This will compute the marginal likelihood of `my-corpus` (which was defined above), given the prior distribution `theta-prior`.

Answer 2: Please put the answer in `ps4-lastname-firstname.clj`.

Problem 3: Write a function `compute-conditional-prob`, which takes three arguments: `theta`, `corpus`, and `theta-probs`. The arguments have the same interpretation as in Problems 1 and 2. The function should return the **log** of the conditional probability of the parameter value `theta`, given the corpus. Remember that the conditional probability is defined by the equation:

$$\Pr(\Theta = \theta | \mathbf{C} = \texttt{corpus}) = \frac{\Pr(\mathbf{C} = \texttt{corpus}, \Theta = \theta)}{\sum_{\theta \in \Theta} \Pr(\mathbf{C} = \texttt{corpus}, \Theta = \theta)} \tag{1}$$

[Note: don't forget that your `compute-conditional-prob` should return a **log** probability.]

Answer 3: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 4: Write a function `compute-conditional-dist`, which takes two arguments: `corpus` and `theta-probs`. For every value of θ in `thetas` (i.e., `theta1` and `theta2`), it should return the log conditional probability of θ given the corpus. That is, it should return a two-element list of log conditional probabilities, one for each of the two values of θ.

Answer 4: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 5: Call `(compute-conditional-dist my-corpus theta-prior)`. What do you notice about the conditional distribution over values of θ? You may want to exponentiate the values you get back, so that you can see the regular probabilities, rather than the log probabilities. Explain why the conditional distribution looks the way it does, with reference to the properties of `my-corpus`. In particular, if one value of θ has higher conditional probability than the other, say why.

Answer 5: Please put your answer in the box below.

Given the vocabulary, the model parameterized by *theta1* is twice as probable ($\frac{2}{3}$) compared to the model parameterized by *theta2* ($\frac{1}{3}$) . When analyzing the values of both thetas, one notices that theta1 implies the word *call* has a 50% chance of appearing and the words *me* and *ismael* each have a 25% chance. *Theta2* implies the word *me* has a 50% chance of appearing and the words *call* and *ismael* each have a 25% chance. Looking now at our corpus, we can see that *call* appears 2 times, while *me* and *ismael* each appear once. This explains why the model parameterized by *theta1* is twice as likely to have generated the corpus than the model parameterized by *theta2*.

Problem 6: When you call `compute-conditional-dist`, you get back a log probability distribution over values of θ (the conditional distribution over θ given an observed corpus). This is a probability distribution just like any other. In particular, it can be used as the prior distribution over values of θ in a hierarchical bag of words model. Given this new hierarchical BOW model, we can do all of the things that we normally do with such a model. In particular, we can compute the marginal likelihood of a corpus under this model. This marginal likelihood is called a *posterior predictive distribution*.

Below we have defined the skeleton of a function `compute-posterior-predictive`, which you must complete. It takes three arguments: `observed-corpus`, `new-corpus`, and `theta-probs`. The argument `observed-corpus` is a corpus which we have observed, and are using to compute a conditional distribution over values of θ. Given this conditional distribution over θ, we will then compute the marginal likelihood of the corpus `new-corpus`. The function `compute-posterior-predictive` should return the marginal log likelihood of the new corpus given the conditional distribution on θ.

```
(defn compute-posterior-predictive [observed-corpus new-corpus theta-probs]
  (let [conditional-dist ...
    (compute-marginal ...
```

Once you have implemented `compute-posterior-predictive`, call `(compute-posterior-predictive my-corpus my-corpus theta-prior)`. What does this quantity represent? How does its value compare to the marginal likelihood that you computed in Problem 2? Why is this to be expected?

Answer 6: Please put your code in `ps4-lastname-firstname.clj` and write the text part of the answer in the box below.

> The quantity returned by `compute-posterior-predictive` represents the marginal log likelihood given the updated beliefs about θ after observing the corpus `my-corpus`. The value is higher compared to the one computed in Problem 2, indicating that given our updated beliefs about θ, the corpus `my-corpus` is more likely to have been generated by the model than before observing the corpus. This is to be expected, since the corpus `my-corpus` is the same as the observed corpus, and therefore the updated beliefs about θ should be more likely to have generated the corpus than the prior beliefs.

Problem 7: In the previous problems, we have written code that will compute marginal and conditional distributions *exactly*, by enumerating over all possible values of θ. In the next problems, we will develop an alternate approach to computing these distributions. Instead of computing these distributions exactly, we will approximate them using random sampling.

The following functions were defined in class, and will be useful for us going forward:

```
(defn normalize [params]
  (let [sum (apply + params)]
    (map (fn [x] (/ x sum)) params)))

(defn flip [weight]
  (if (< (rand 1) weight)
    true
    false))

(defn sample-categorical [outcomes params]
  (if (flip (first params))
    (first outcomes)
    (sample-categorical (rest outcomes)
                        (normalize (rest params)))))

(defn sample-BOW-sentence [len probabilities]
  (if (= len 0)
    '()
    (cons (sample-categorical vocabulary probabilities)
          (sample-BOW-sentence (- len 1) probabilities))))
```

Recall that the function `sample-BOW-sentence` samples a sentence from the bag of words model of length `len`, given the parameter vector `probabilities`.

Define a function `sample-BOW-corpus`, which takes three arguments: `theta`, `sent-len`, and `corpus-len`. The argument `theta` is a value of the model parameters θ. The arguments `sent-len` and `corpus-len` are positive integers. The function should return a sample corpus from the bag of words model, given the model parameters `theta`. Each sentence should be of length `sent-len` and number of sentences in the corpus should be equal to `corpus-len`. For example, if `sent-len` equals 3 and `corpus-len` equals 2, then this function should return a list of 2 sentences, each consisting of 3 words.

Hint: Use `sample-BOW-sentence`. You may also want to use the built-in function `repeatedly`.

Answer 7: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 8: Below we have defined the skeleton of the function `sample-theta-corpus` which you must complete. This function takes three arguments: `sent-len corpus-len` and `theta-probs`. It returns a list with two elements: a value of θ sampled from the distribution defined by `theta-probs`; and a corpus sampled from the bag of words model given the sampled θ. (The number of sentences in the corpus should equal `corpus-len`, and each sentence should have `sent-len` words in it.)

We will call the return value of this function a `theta-corpus` pair.

```
(defn sample-theta-corpus [sent-len corpus-len theta-probs]
  (let [theta ...
    (list theta ...
```

Answer 8: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 9: Below we have defined some useful functions for us. The function `get-theta` takes a `theta-corpus` pair, and returns the value of `theta` in it. The function `get-corpus` takes a `theta-corpus` pair, and returns its corpus value. The function `sample-thetas-corpora` samples multiple theta-corpus pairs, and returns a list of them. In particular, the number of samples it returns equals sample-size. The function `get-count` counts the number of times an outcome appears in a list, and will be useful in this problem as well as Problem 11.

```
(defn get-theta [theta-corpus]
  (first theta-corpus))

(defn get-corpus [theta-corpus]
  (first (rest theta-corpus)))

(defn sample-thetas-corpora [sample-size sent-len corpus-len theta-probs]
  (repeatedly sample-size (fn [] (sample-theta-corpus sent-len corpus-len theta-probs))))

(defn get-count [outcome lst]
  (let [filtered-lst
        (filter (fn [x] (= outcome x)) lst)]
  (count filtered-lst)))
```

We are now going to estimate the marginal likelihood of a corpus by using random sampling. Here is the general approach that we are going to use. We are going to sample some number (for example 1000) of `theta-corpus` pairs. These are 1000 samples from the joint distribution defined by the hierarchical bag of words model. We are then going to throw away the values of `theta` that we sampled; this will leave us with 1000 corpora sampled from our model.

We are going to use these 1000 sampled corpora to estimate the probability of a specific target corpus. The process here is simple. We just count the number of times that our target corpus appears in the 1000 sampled corpora. The ratio of the occurrences of the target corpus to the number of total corpora gives us an estimate of the target's probability.

More formally, let us suppose that we are given a target corpus $\mathbf{t}$. We will define the indicator function $\mathbb{1}_{\mathbf{t}}$ by:

$$\mathbb{1}_{\mathbf{t}}(c) = \begin{cases} 1, & \text{if } t = c \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We will sample n corpora $c_1, \ldots, c_n$ from the hierarchical bag of words model. We will estimate the marginal likelihood of the target corpus $\mathbf{t}$ by the following formula:

$$\sum_{\theta \in \Theta} \Pr(\mathbf{C} = \mathbf{t}, \Theta = \theta) \approx \frac{1}{n} \sum_{i}^{n} \mathbb{1}_{\mathbf{t}}(c_i) \tag{3}$$

Define a procedure `estimate-corpus-marginal`, which takes five arguments: `corpus`, `sample-size`, `sent-len`, `corpus-len`, and `theta-probs`. The argument `corpus` is the target corpus whose marginal likelihood we want to estimate. `sample-size` is the number of corpora that we are going to sample from the hierarchical model (its value was 1000 in the discussion above). The arguments `corpus-len` and `sent-len` characterize the number of sentences in the corpus and the number of words in each sentence, respectively. The argument `theta-probs` is the prior probability distribution over θ for our hierarchical model.

The procedure should return an estimate of the marginal **(not log)** likelihood of the target corpus, using the formula defined in Equation 3.

Hint: Use `sample-thetas-corpor`a to get a list of samples of `theta-corpus pairs`, and then use `get-corpus` to extract the corpus values from these pairs (and ignore the `theta` values).

Answer 9: Please put your answer in `ps4-lastname-firstname.clj`.

Problem 10: Call `(estimate-corpus-marginal my-corpus 50 2 2 theta-prior)` a number of times. What do you notice? Now call `(estimate-corpus-marginal my-corpus 10000 2 2 theta-prior)` a number of times. How do these results compare to the previous ones? How do these results compare to the exact marginal likelihood that you computed in Problem 2?

Answer 10: Please put your answer in the box below.

Mean and Variance of the estimated marginal likelihood of the target corpus `my-corpus` (over 100 runs):
Exact: 0.01171875
Sample Size 50: $0.012 \pm 2.56e-4$
Sample Size 10000: $0.01187 \pm 8.6e-7$

We can see that the means of the estimated marginal likelihood of the target corpus `my-corpus` are generally close to the exact marginal likelihood. When computing both estimates 100 times, we can see that the variance of the estimated marginal likelihood of the target corpus `my-corpus` is much smaller when using a sample size of 10000 compared to a sample size of 50. The mean of the estimated marginal likelihood of the target corpus `my-corpus` is also closer to the exact marginal likelihood when using a sample size of 10000 compared to a sample size of 50. This is to be expected, since the larger the sample size, the more accurate the estimate of the marginal likelihood of the target corpus `my-corpus` will be.

Problem 11: In Problem 9, we introduced a way of approximating the marginal likelihood of a corpus by using random sampling. We can similarly approximate a conditional probability distribution by using random sampling.

Suppose that we have observed a corpus $\mathbf{c}$, and we want to compute the conditional probability of a particular θ. We can approximate this conditional probability as follows. We first sample n theta-corpus pairs. We then remove all of the pairs in which the corpus does not match our observed corpus $\mathbf{c}$. We finally count the number of times that θ occurs in the remaining theta-corpus pairs, and divide by the total number of remaining pairs. This process is an example of *rejection sampling*.

Define a function `rejection-sampler` which has the following form:

```
(defn rejection-sampler
  [theta observed-corpus sample-size sent-len corpus-len theta-probs]
  ...
)
```

This function should use the rejection sampling method (as described above) to estimate the conditional probability of `theta`, given that we have observed the corpus `observed-corpus`. The function must estimate this conditional probability by taking `sample-size` samples (you may assume this argument is a positive integer) from the joint distribution on `theta-corpus` pairs. The procedure should filter out any `theta-corpus` pairs in which the corpus does not equal the observed corpus. If there are no remaining pairs after filtering, then the function should return `nil`. Otherwise, it should then count the number of times that `theta` occurs in the remaining pairs, and divide by the total number of those pairs.

Hint: Use `get-count` to count the number of occurrences of `theta`.

Answer 11: Please put your answer to the coding problem in `ps4-lastname-firstname.clj`.

Problem 12: Call `(rejection-sampler theta1 my-corpus 100 2 2 theta-prior)` a number of times. What do you notice? Try with larger sample sizes (such as $200, 500, 1000 \ldots$). How large does `sample-size` need to be until you get a stable estimate of the conditional probability of `theta1`? Why does it take so many samples to get a stable estimate?

Answer 12: Please answer the questions in the box below.

Mean and Variance of the estimated conditional probability of `theta1` (over 100 runs):
Exact: $\frac{2}{3}$
Sample Size 100: 0.5484 ± 0.2018
Sample Size 200: 0.6621 ± 0.1095
Sample Size 500: 0.6838 ± 0.0510
Sample Size 1000: 0.6708 ± 0.0161

We can observe that with increasing sample size, the mean of the estimated conditional probability of `theta1` gets closer to the exact conditional probability of `theta1`. The variance of the estimated conditional probability of `theta1` is also much smaller when using a sample size of 1000 compared to a sample size of 100. This is to be expected, since the larger the sample size, the more accurate the estimate of the conditional probability of `theta1` will be. It takes so many samples to get a stable estimate because the probability of sampling a `theta-corpus` pair that matches the observed corpus is quite low. Therefore, we need to sample a large number of `theta-corpus` pairs to get a stable estimate of the conditional probability of `theta1`.

LONG FORM READING QUESTION:

(This section is optional for students in LING/COMP 445, but must be completed if taking LING 645.)

You must answer this question on your own.

Based on Warstadt & Bowman 2022 and our discussion in class, choose two differences between human learners' and language models' learning environment that you think are the most critical for language learning. Discuss both (1) why you think they are the most important and (2) how models may be modified to account for these differences – this second point can be hypothetical if no models have addressed them yet.

Answer: Please put your answer in the box below.

Warstadt & Bowman (2022) discuss differences between the learning environment of human learners and the learning environment of language models in order to ultimately point out the hurdles to the transferability of findings from language models to human language learners. In my opinion, the most important underestimates are the quantity of available data and facets of the learning environment.

The quantity and quality of data available to language models far exceeds that available to human learners. Language models are trained with very large amounts of textual data from various sources, while human learners interact with their environment and learn language through social interactions. Studies on the number of words children are exposed to in the course of their education show that this is far less than the amount of data available to language models. This difference is important because it affects the type of linguistic knowledge that can be acquired. Human learners are exposed to a wide range of language structures and contexts, whereas language models may be biased towards specific types of language structures because of the data on which they have been trained. However, simply reducing the number of words used for training significantly reduces performance, so it may be important what subset is actually used for training.

Secondly, the learning environment of human learners is based on real-life experiences and interactions with other agents, whereas language models learn in an unstructured environment. This difference is important because it affects the way language is learned and used. Human learners are able to base their linguistic knowledge on real-life experiences, whereas language models may struggle with this. In the future, models could be modified to be trained in learning environments such as interactive simulations or virtual reality environments. This would allow the models to learn language in an environment that is more similar to the human learning environment.