MIT 6.806/6.864: Advanced Natural Language Processing **Fall 2021**

Homework 1

Due Date: Mon. September 27, 2021, 11:59 PM EDT (*Canvas submission*)

Classification

Introduction. In this assignment, you'll explore two different ways of doing language modeling as well as one way of doing machine translation. In each case, you will implement, train, and evaluate your model on a real dataset, after which you will prove and discuss some theoretical properties. Finally, you will submit a report describing both experiments and answers to the theoretical and computational questions in this handout. Your report should also describe the effects of different modeling decisions on the performance of your models. We provide scaffolding code for each part of the lab Jupyter notebooks; the code is in `6864_hw1.ipynb`, (this notebook is available in this GitHub repository. We recommend saving a copy of each notebook and using Google Colab to write and execute your code.[1]

General report guidelines. Homework assignments should be submitted in the form of a research report on Canvas. Please upload a single PDF of your report concatenated with print outs of your code (i.e., the Jupyter Notebooks with your implementation). The report section of your submitted PDF should consist of a maximum of four single-spaced pages (6.806) or six single-spaced pages (6.864) typeset with LaTeX.[2] Reports should have one section for each part of the assignment below. Each section should describe the details of your code implementation and include whatever analysis and figures are necessary to answer the corresponding set of questions.

Part 1: Count-based Language Modeling

The first part of your lab report should discuss any important details and design decisions you made when implementing the Count-based Language Model. Additionally, you should answer the following questions in your report:

(a) (*Theoretical*) Recall from lecture the task of language modeling. A popular way to approach the problem is to factorize the probability of a string left-to-right,

$$p(w_1, \ldots, w_T) = \prod_{t=1}^{T} p(w_t \mid w_1, \ldots, w_{t-1}),$$

and estimate the individual factors involved in the product, i.e. $p(w_t \mid w_1, \ldots, w_{t-1})$. Due to data sparsity this quantity is often hard to estimate accurately in practice.

[1] You can open the notebook in Google Colab with this link: `https://colab.research.google.com/github/mit-6864/hw1/blob/master/6864_hw1.ipynb`.

[2] If you'd like, you can use the Association for Computational Linguistics style files, available at: `https://2021.aclweb.org/downloads/acl-ijcnlp2021-templates.zip`

Count-based n-gram language models make an n-th order *Markov* assumption:

$$p(w_t \mid w_1, \dots, w_{t-1}) \approx p(w_t \mid w_{t-n}, \dots, w_{t-1}),$$

and instead estimate $p(w_t \mid w_{t-n}, \dots, w_{t-1})$ by counting and dividing. In lecture we stated without proof that setting the model parameters to be equal to the following count-based estimation,

$$p(w_t \mid w_1, \dots, w_{t-1}) = \frac{c(w_{t-n}, \dots, w_t)}{c(w_{t-n}, \dots, w_{t-1})} = \frac{c(w_{t-n}, \dots, w_t)}{\sum_{w_t'} c(w_{t-n}, \dots, w_t')},$$

is an instance of *maximum likelihood learning.*

Let us assume that we are using a *bigram* language model. The parameters in this bigram language model are $\theta_{u,v}$ for $u, v \in \mathcal{V}$, where $\mathcal{V} = \{1, \dots, V\}$ is the vocabulary (represented as an integer), and $\theta_{u,v} = P(w_t = u \mid w_{t-1} = v)$. (Note that we represent a probability mass function with parameters a lower-case p, and the probability of an event with an upper-case P.) Prove that the maximum likelihood estimate (MLE) for a bigram language is given by

$$\hat{\theta}_{u,v} = \frac{c(u, v)}{\sum_{u \in \mathcal{V}} c(u, v)}.$$

That is, show that $\hat{\theta}_{u,v}$ as defined above is the solution to the following optimization problem

$$\arg\max_{\theta} \; p_\theta(w_1, \dots, w_T).$$

(For the purposes of this problem we can treat the training set as one long string $w_1 w_2 \dots w_T$).

Hint: You may want to start with the unigram case and show $\hat{\theta}_u = \frac{c(u)}{T}$ is the MLE for $P(w_t = u)$. Recall that the likelihood of a corpus under the unigram model is given by $\prod_{t=1}^{T} p_\theta(w_t)$. Try to relate this quantity to $c(u)$, the number of times u occurs in the corpus. Since log is a monotonic function, maximizing the likelihood with respect the model parameters is the same as maximizing the *log* likelihood. How can we find the critical point of the log likelihood of the corpus? How can we be sure that the critical point is a global maximum? To incorporate the constraint that θ_u are probabilities, recall that we can solve constrained optimization problems of the form

$$\max_{\theta} \; L(\theta), \quad \text{such that } f(\theta) = 0,$$

by introducing a *Langrangian multiplier* λ and finding the critical points of

$$\mathcal{L}(\theta, \lambda) = L(\theta) - \lambda f(\theta).$$

(b) (**6.864 students only**; *Theoretical*) The generalized form of the add-one smoothing technique you used in part (a) is called **Laplace smoothing** (or **additive smoothing**). In the unigram language modeling case, with Laplace smoothing the probability of a word u is given by

$$\theta_u = \frac{c(u) + \alpha}{T + \alpha|\mathcal{V}|},$$

where $c(u)$ is the number of times u is observed, T is the total number of words in the corpus, $|\mathcal{V}|$ is the vocabulary size, and $\alpha > 0$ is a smoothing parameter. In lecture we remarked that this is one heuristic for avoiding issues with regard to data sparsity. In this problem we will show that this estimate can also be motivated by adopting a Bayesian perspective.

Consider a unigram language model with parameters π_u for $u \in \mathcal{V}$. In Bayesian modeling, we treat the model parameters π as a random variable and specify a *prior distribution* over them. A natural prior for the multinomial distribution is given by the *Dirichlet* distribution. The Dirichlet distribution with parameters $\beta = \{\beta_1, \dots, \beta_V\}$ (with $\beta \geq \mathbf{0}$) defines a distribution over the simplex,

$$\left\{\pi : \pi \in \mathbb{R}^{|\mathcal{V}|},\ \mathbf{0} \leq \pi \leq \mathbf{1},\ \sum_{u \in \mathcal{V}} \pi_u = 1,\right\}$$

where the density is given by

$$p(\pi\,;\,\beta) = \frac{1}{\mathrm{B}(\beta)} \prod_{u \in \mathcal{V}} \pi_u^{\beta_u - 1}.$$

Here $B(\beta)$ is a normalizing constant whose exact expression we don't care about for this problem. Given a (1) prior, (2) likelihood model $p(D \,|\, \pi)$ (which is the same likelihood function as in problem (a)), and (3) observed data $D = (w_1, \dots, w_T)$, we can define the *posterior distribution* over the parameters given the data via Bayes' rule, i.e.,

$$p(\pi \,|\, D\,;\,\beta) = \frac{p(D \,|\, \pi) p(\pi\,;\,\beta)}{p(D)}.$$

Show that letting $\beta_u = \alpha$ for all $u \in \mathcal{V}$ results in a posterior distribution whose mean is given by the Laplace smoothing estimate. That is, show that

$$\mathbb{E}_{p(\pi \,|\, D\,;\,\beta)}[\pi] = \theta.$$

Hint: we don't actually need to calculate $p(D)$, and you can use (without proof) the fact that the mean vector of a Dirichlet distribution is given by $\mathbb{E}[\pi] = \frac{1}{\sum_{v=1}^{V} \beta_v} \beta$.

Part 2: RNN Language Modeling

Part 2 of your lab report should discuss any implementation details that were important to filling out the implementation of your RNN language model. Then, answer the following question:

(a) (*Computational*) Consider the Bounded Parentheses Language (BPL), which consists of strings of "(" and ")" characters where the number of "(" characters equals the number of ")" characters and where every prefix of the string consists of at least as many "(" characters as ")" characters. For example, strings such as "(()(()))" and "()((())())" are in BPL, while strings like "(()()))" and "(()((())" are not—though note that the latter *is* a valid prefix whereas the former is not. Given the RNN hidden state update equation, $h_t = f(Wx_t + Uh_{t-1})$ and provide appropriate values of f, W, U, and h_0 such that at any time t, h_t indicates whether the string $x_1x_2 \ldots x_t$ is in BPL.

Hint: How many dimensions should x_t and h_t be?

Part 3: Seq2Seq Models

In the remaining part of this assignment you'll discuss any implementation details that were important to filling out the implementation of your seq2seq translation model, including (for 6.864 students) your implementation of attention. Then, use your code to answer the following questions:

(a) (*Experimental*) **Analyzing Decoding Output.** Let's explore the outputs of the model that you've trained! Using your EncoderDecoder model, identify two examples of errors that it produces. The two examples you find should be different error types from one another. For each example you should:

1. Write the source sentence in Vietnamese.
2. Write the target English translation.
3. Write your model's English translation.
4. Identify the error in the model's translation.
5. Provide a reason why the model may have made the error (either due to a specific linguistic construct or specific model limitations).
6. Describe one possible way we might alter the system to fix the observed error.

Note: Fluency in Vietnamese is definitely not required! It might help to use Google Translate to see what each "word" or chunk of "words" corresponds to (in Vietnamese, spaces do not always indicate word boundaries).