11-411/11-611 Homework Assignment 2

NLP Teaching Staff

Due: October 24, 2024

1 Introduction

In this homework, you will be building your first language models. You will be expected to build an n-gram language model and an recurrent neural network (RNN) language model. You will also implement Laplace Smoothing for the n-gram nodel (a lazy version) to account for unknown words.

2 Learning Objectives

Language modeling is one of the most important concepts in natural language processing. As you learned in class, LMs assign a probability score to a sequence of words (or characters, or sub-words but we will limit ourselves to words!)

Today, language models have become ubiquitous: they are widely used in speech recognition, optical character recognition, machine translation and probably any other NLP task that you can imagine.

Thanks to language models, [insert your favorite voice assistant] can differentiate when you want an “ice cream” and when you just want to scream (“I scream”).

In this assignment, you will be programming your own language models using two different strategies: n-grams and RNNs.

3 Task 1: Programming (60 points)

Refer to the notebook that is provided as part of the handout. You will be downloading the required data files, along with **utils.py** and **main.py**. **Do not edit these files**. After you are done coding, paste the functions and the classes you implemented into the **ngram_lm.py** and **rnn_lm.py** files that are part of the handout and then upload them to the HW2 Programming Submission on Gradescope, without zipping them.

4 Task 2: Written (40 points)

Answer the following questions based on the code you've written. You can use the latex file in the handout to answer the questions. Upload the PDF to the HW2 Written Submission on Gradescope.

4.1 n-gram counts (10 points)

Train the n-gram language model on the data/bbc/business.txt dataset for $n = 2$ and $n = 3$. Then do the same for data/bbc/sports.txt datset

1. How many unique 2-grams are present in the business dataset? (2 points)

2. How many unique 3-grams are present in the business dataset? (2 points)

3. How many unique 2-grams are present in the sports dataset? (2 points)

4. How many unique 3-grams are present in the sports dataset? (2 points)

5. How many possible 2- and 3- grams could there be, given the same vocabulary? How do the empirical counts given above compare to the number of possible 2- and 3- grams? (2 points)

4.2 Song Attribution (8 points)

You are scrolling through the top hits playlist on Spotify when you notice a new unknown song at the top. It's recorded by an anonymous artist but the lyrics sound uncannily similar to some other songs you have heard. You have narrowed it down to three artists but are unable to choose one: *it could be any of them!*

You go along the rest of the day thinking who could it be. You reach Posner Hall to attend an NLP 11-411/611 lecture and Professor Strubell is teaching language models. Wait: language models. It suddenly hits you: language models can help in this task!

Train tri-gram (n=3, smoothing= 0.1) language models on collections of song lyrics from three popular artists ('data/lyrics/') and use the model to score a new unattributed song.

Note: In reality, perplexity should only be used to compare language models when they have the same vocabularies but we will relax that condition for this question.

1. What are the perplexity scores of the test lyrics against each of the language models? (6 points)
   (a) Taylor Swift:
   (b) Green Day:
   (c) Ed Sheeran:
2. Who is most likely to be the lyricist? (2 points)

4.3 Introduction to Decoding and Text Generation (8 points)

It is said, Ed Sheeran will have a new album next year. Whether it is true or not, let us try to predict some lyrics might be shown up. To predict the content of the tracks based on their titles, you plan to use an RNN language model trained on his lyrics (stored in `ed_sheeran.txt`).

You are particularly interested in the following three tracks:

- s1: Yellow
- s2: Fell the fall
- s3: Down Bad

Run the code provided in the notebook and fill in the answers below:

1. For each of these track titles s1 to s3, list the top five word candidates predicted to follow each sequence. Ensure you exclude special tokens added during training to indicate the end-of-sentence and start-of-sentence. You may need to modify or print specific details from the `generate_sentence` function to achieve this. (6 points)

   (a) s1:

   (b) s2:

   (c) s3:

2. Report any one of the generated sentences here. Which generation mode do you think is better and why? (2 points)

4.4 Comparison to a GPT (14 points)

Run the code provided in the notebook and fill in the answers below.

1. What is the perplexity of your LM models (n-gram and RNN)? (2 points)

2. What is the perplexity of the GPT-2 model? (2 points)

3. How might you reason about the differences in perplexity between these three models? Think about the parameters, size of the vocabulary, training data, etc. used to build your language models, compared to that of GPT-2, and how might this impact their respective performance? (6 points)

4. What are some of the trade-offs between using a simpler model like your language models versus a more complex model like GPT-2, and how might these trade-offs affect their performance on different tasks? (4 points)

5 Submission

This assignment will be submitted via Gradescope in two parts. Upload both deliverables to Gradescope.

1. **Programming.** Paste the functions and classes from the notebook into two files called `ngram_lm.py` and `rnn_lm.py`, which are included in the handout. Upload then files without zipping them.

2. **Written.** Submit answers to the questions as a PDF file.

Submissions are due October 24, 2024 at 11:59pm EST via Gradescope.