Homework 2[1,2]

Due by 11:59pm 3/5/2019

Project Premise

While in class we focused on predicting the next *word* given some history, there are situations where it might be helpful for the model to predict the next *letter(s)*. Some examples include: entering text on mobile phones, using Latin alphabet to enter non-Latin languages (such as Chinese or Japanese characters), or helping users with disabilities to input text more quickly. In this assignment, you will design and compare some letter-based language models for two use case scenarios.

- Predict the next letter for an English IRC corpus
- Predict the most contextually suitable Chinese (graphical) character based on its English-ified pronunciation (known as pinyin) as entered from a QWERTY keyboard. The challenge here is that for most sounds, the same pronunciation can map to many different Chinese characters (e.g., the pinyin “xiong” can be mapped to 27 unique characters). Language model may help to pick out the right one given the context.

In both cases, you may assume you are working with a predefined vocabulary list (you can preprocess the train/dev/test sets to compile the vocabulary).

Project Setup

Prof. David Chiang has given us the permission to reuse his code stub and data files, so please download from his github repository. Feel free to take a look at their accompanying homework description, but **do not** follow their instructions because the instructions for our assignment are somewhat different than his for his class.

Project Requirements

1. Implement (from scratch) a **letter-based** Ngram model. Your program should:
    - Take a positive integer and a training file as inputs and build the corresponding Ngram model (we will only test for N=2..5, but you may wish to experiment with higher values of N -- see Step 2).
        - You may wish to use the supplied skeletal file, unigram.py, as a template for your Ngram models.
        - For this step, you do not need to implement smoothing.
    - Implement an evaluator for testing -- Given an input file (e.g., `english/dev`), have the trained Ngram model predict the next letter given (up to) the N-1

[1] This assignment is also available on Google Drive: https://drive.google.com/drive/folders/1x_tXak_Mu4zbt-GygHNj1vmo_AbR8mpY?usp=sharing

[2] This assignment is adapted from one designed by Prof. David Chiang of Notre Dame. We thank him for sharing his code and data files with us.

previous characters (reset for each sentence); check to see whether the predicted letters matches the expected answers; report a final accuracy.

2. Improve your basic model by implementing at least one smoothing method that **is not** add-one (or add-delta) and try higher-order N's (e.g., 8-gram or 9-gram). Design an experiment and compare the models of Steps 1 and 2 for the English IRC corpus. **NB:** you may use `english/test` only for your final test; any hyper-parameter tuning should be done on `english/dev`.
3. Adapt an RNN (based on one of the tutorials you've read) for the same task. You should clearly indicate in the README what your platform is. Tune all your hyper-parameters on `english/dev`. Test and compare your RNN against the best results of Step 2.
4. Adapt the models to predict the current Chinese character given its Latin pronunciation input (pinyin) and the previous history. For this part, a smaller N value (2 or 3) will probably work well enough. You do not need to know Chinese for this part. Just use the supplied Chinese character map file (`chinese/charmap`) which displays the Chinese characters in the first column and the corresponding pinyin in the second column. So, overall, you will need to do the following:
    - Adapt the language models and train them on `chinese/train.han` (For RNN, you may use pre-trained embeddings -- please include citations.)
    - At application time, you need to read in the input from `chinese/test.pin` (or `chinese/dev.pin`). Each token is space delimited. There are three types:
        - A pronunciation (pinyin) -- e.g., "ni" -- which should map to a Chinese character (e.g., 你).
        - An English letter -- e.g., "q" -- which could be mapped to the letter q; though note that it is possible for a pinyin to be a single letter long too, so you may not assume that an input token that is a single letter long is definitely an English letter.
        - A special token called "<space>" which maps to a space
    - For each input token, generate a list of candidate Chinese characters
    - Use the trained language model and the history of the correct Chinese characters so far to select the most likely Chinese character from the candidate list
    - Compare the model's choice with the correct answer `chinese/test.han` to see how well the model performed.

What to commit

- Your code and data files
    - Please document enough of your program to help the TA grade your work.
- A README file that addresses the following:
    - Describe the computing environment you used, especially if you used some off-the-shelf modules. (Do not use unusual packages. If you're not sure, please ask us.)
    - List any additional resources, references, or web pages you've consulted.

    - List any person with whom you've discussed the assignment and describe the nature of your discussions.
    - Discuss any unresolved issues or problems.
- A REPORT document that discusses the following:
    - How well did your various unsmoothed Ngram models perform on the English task? How high an N did you try? At what point did smoothing become a problem?
    - What kind of smoothing method(s) did you implement? If there are some hyper-parameters for the smoothing methods, how did you choose the values for them? How well did these models perform?
    - What kind of an RNN model did you adapt for this task? Describe your architecture choices and how you tuned the hyper-parameters. How well did the model do?
    - Overall, what seems to be the best working model for the English task? How well do you think your model would do on some other English corpus? Any other observations about this task?
    - Describe the steps you took to adapt the models for the Chinese character prediction task.
    - Compare the models. How well did they do? Were there any surprises in the outcomes? Any other observations about this task?

Grading Guideline

Assignments are graded qualitatively on a non-linear five point scale. Below is a rough guideline:

- 1 (40%): A serious attempt at the assignment. The README clearly describes the problems encountered in detail.
- 2 (60%): Correctly completed Step 1 of the assignment, but encountered significant problems for other steps. Submitted a README documenting the unresolved problems. Submitted a REPORT for Step 1 outcomes.
- 3 (80%): Correctly completed Steps 1 and 2 of the assignment, but encountered problems beyond that. Submitted a README documenting the unresolved problems. Submitted a REPORT discussing the outcomes of Steps 1 and 2.
- 4 (93%): Correctly completed the assignment through Step 3. Submitted a README that appropriately documented the details of the models. Submitted a REPORT that discussed the outcomes of the English letter prediction task.
- 5 (100%): Correctly completed the assignment. Submitted a README that appropriately documented the details of the models. Submitted a REPORT that is well-written and insightful.