CS114 (Spring 2015) Homework 2
N-Gram Language Model

Due February 13, 2015

The overall goal of this assignment is for you to train the best language model possible given the same limited amount of data. There will be no test cases for you this time. The grade will depend on the correctness of your implementation, the written report, and the perplexity on the blind test set.

You should first go on Latte and download the data sets. The tar file should contain the training set and development set. The dataset is in one sentence per line format, and each token is separated by a single blank space. So you can simply use `split` function to tokenize each sentence.

Assignment

1. Show that $PP_M(W) = 2^{H(W)}$ where $PP_M(W)$ is the perplexity of language model M on the sequence of n words W and $H(W)$ is the cross entropy of M on W. (Include the solution in the report)

2. Show that $PP_M(W) = \exp(-\frac{\log P_M(W)}{n})$ where P_M is the language model. (Include the solution in the report)

3. Implement language models and come up with the best language model possible given the datasets described above.

   Note that if you train a trigram model, you must append two tokens of `<S>` to each sentence before training. For example,

   ```
   P(I like running) = P(I|<S> <S>) P(like|<S> I) P(running|I like)
   ```

   Also, for simplicity of implementation, treat all punctuations as words. Leave them as they are.

4. Write a short report on the language models that you have explored. You should at least describe the following:

    - The models that you have tried (e.g. 4-gram with simple interpolation)
    - Perplexity on the development set for each model
    - Comparison of different models you have tried.

5. Write a command-line program `compute_perplexity.py` that prints out your name and the perplexity of the input file.

   ```
   > python compute_perplexity.py dev-data.txt
   Te Rutherford 3542
   ```

   The input file is in one sentence per line format, and each token is separated by a single blank space. Nikhil and I will run your language model on the blind test set. You will be graded on the perplexity. You must at least beat bigram model with add-one smoothing.

Submission Instruction

Submit on Latte the pdf-formatted report, your implementation of `compute_perplexity.py`, and your language model file as required by `compute_perplexity.py`.

Tips and suggestions

- It is useful to know that

$$\log \prod_{i=1}^{n} P(w_i) = \sum_{i=1}^{n} \log P(w_i)$$

- After you turn the counts into the probabilities, you should store log-probabilities. Remember that $\exp \log P(w) = P(w)$, so it is easy to convert back and forth between probaiblity and log probability.

- You should use this formula for computing perplexity:

$$PP_M(W) = \exp(-\frac{\log P_M(W)}{n})$$

  because it uses log probability directly. Adding is much faster than multiplying, and the product of probabilities will lead to numerical underflow (the number too small to be represented by a Python float value). The sum of log probabilities helps avoid this problem.

- Start small. Start with bigram with add-one smoothing and evaluate it. From there, it is not too hard to expand to trigram or 4-gram.
- Good-Turing or Kneser-Ney models are excellent choices for improving the performance. The models are described in detail in the book.
- It is OK if your model does not perform too too well compared to your classmates at the end as long as we see in the report that you have tried a good number of different models to make the performance better.
- Do not train on the dev set ever. Instead, use it for tuning the interpolation weight, or deciding between 4-gram or 5-gram.