CS4740/CS5740: Introduction to NLP
PA 1: Language Modeling

Submit via CMS by 11:59PM, Monday, February 23, 2015

1 Background

The goal of this programming assignment is to help you familiarize yourself with n-gram models and their application in text categorization.

1. **The Corpus** Training, validation, and test set excerpted from the ENRON Email corpus are provided to you via CMS.

   Each data file contains a set of emails in the following format (except for the test set which lacks the UPSPEAK/DOWNSPEAK label):

   ```
   UPSPEAK/DOWNSPEAK

   **START**

   (email content)

   **EOM**
   ```

   The labels UPSPEAK and DOWNSPEAK are explained in more detail in Section 2.2.6.

2. **Working in Groups** You must work in groups of 2–4 students, and the project will be graded without regard to the size of the group. Also, everyone in the group will receive the same grade; however, you are to specify the contribution of each group member at the end of the report. Please form groups in CMS prior to submission. (Piazza may be helpful for finding the partners.)

3. **A Note on the Kaggle Competition** The number of submissions a group can make is limited to 2 submissions per day, just like a typical Kaggle competition. This means that you will benefit from getting started early!

4. **Submission Package** Submit a .zip or .tar file — one submission per group, containing the following:

   - The source code (but not the libraries) and a “readme.txt” file explaining how to compile and execute the code for each section of the assignment. (You may also include a makefile if you wish.)

Also, it's ok if there's no interface for selecting a function to run. For example, it's fine to have instructions like, "comment out lines x-y and uncomment lines i-j in the main function to run the random sentence generator."

- A 5-page(roughly) report in pdf format, consisting of answers to the questions in each section. (Look for "[**Report**]")

2 The Assignment [100 pts]

Note: You may use any programming language of your choice.

2.1 Preprocessing [10 pts]

Using a standard corpus often requires you to preprocess the data to fit the task at hand. In this section, you will process the datasets provided to you to create 5 sets of data: UP_TRAIN, UP_VALIDATION, DOWN_TRAIN, DOWN_VALIDATION, UPDOWN_TEST.

1. UP_TRAIN: First, collect the "(email content)" from emails labeled "UPSPEAK" in the train set. (The collected text consists only of the "(email content)" of the emails, and thus the distinction between the emails is lost.) Then, split the text into tokenized sentences[1] and convert the "(email content)" into the following format:

   ```
   <s> a tokenized sentence (possibly a non-sentence)</s>
   <s> a tokenized sentence (possibly a non-sentence)</s>
   <s> a tokenized sentence (possibly a non-sentence)</s>
   ...
   ```

   You've just created a training corpus for "UPSPEAK".

2. DOWN_TRAIN: Repeat the process with "DOWNSPEAK" emails from the train set

3. UP_VALIDATION: Repeat the process with "UPSPEAK" emails from the validation set

[1]You may use a sentence segmenter/tokenizer of your choice, such as NLTK Tokenizer in Python (http://www.nltk.org/api/nltk.tokenize.html) and Stanford Tokenizer in Java (http://nlp.stanford.edu/software/tokenizer.shtml).

4. DOWN_VALIDATION: Repeat the process with "DOWNSPEAK" emails from the validation set

5. UPDOWN_TEST: Repeat the process with all emails from the test set. This time, though, keep each email apart from the rest and retain the email numbers. For the competition, you'll be classifying each email individually.

You may not use any libraries or toolkits beyond this point.

[Report]

- Give an overview of what you did, explaining any decisions you've made. (E.g. Did you throw away any part of the text? What considered a word token? Why?)

2.2 Unsmoothed n-grams [30 pts]

Train unigram, bigram and trigram models on UP_TRAIN and DOWN_TRAIN(6 models total). To do this, simply count and store the number of occurrences of each n-gram. Then, write a function to compute the (conditional) probability by looking up the counts of relevant n-grams and performing a division. Note that "<s>" and "</s>" should be treated as word types. Also, n-grams traditionally don't go across sentence boundaries, meaning "</s>" will always be the last word of an n-gram if it is part of one.

[Report]

- Give an overview of what you did, explaining any decisions you've made. (E.g. Did you keep the capitalization? Which punctuation marks did you keep, if any?)
- List 10 most frequent n-grams for each of the 6 models.
- Compare the n-grams. What do they tell you about the respective models?

2.3 Random sentence generation [10 pts]

Write a function to generate random sentences using the 6 models from the previous section. To generate a random sentence, Let the 0th word be "<s>" and repeatedly select a random word conditioned on the N-1 (or fewer if the history is not available in the beginning of the sentence) preceding words in the partially generated sentence, until "</s>" is selected. Here, the chance of a word being selected should be proportional to the (conditional) probability assigned by the given n-gram model. (For unigram models, we assume P(W_i="<s>") = 0 when $i \neq 0$.)

[Report]

- Give an overview of what you did, explaining any decisions you've made.
- Provide 5 example sentences each of the 6 models.
- Compare the sentences. What do they tell you about the respective models?

2.4 Unknown Word Handling and Laplace Smoothing [20 pts]

Incorporate Laplace(add-one) smoothing[2] and an approach for handling **unknown words** in the test data.

[Report]

- Give an overview of what you did, explaining any decisions you've made. (E.g. How are you handling the unknown words and why?)

2.5 Perplexity [10 pts]

Compute the perplexities of your models on the respective validation set as follows: (Use UP_VALIDATION for the models trained on UP_TRAIN, and DOWN_VALIDATION for the models trained on DOWN_TRAIN).

$$PP = \left(\prod_{i=0}^{N} \frac{1}{P(W_i | W_{i-(n-1)}, \ldots, W_{i-1})} \right)^{1/N}$$

where N is the total number of tokens (including "</s>", but not "<s>") in the validation set and n=1,2,3 for unigram, bigram, and trigram, respectively.

[2]This is to make sure everyone can compute the perplexity. You are welcome to, and should, implement a more complicated smoothing method in Section 2.6.

Here, we are disallowing the history to cross the sentence boundaries, as we did in training. Thus if W_i = "</s>", then you will only see "<s>" in the history of W_{i+1}. And again, we are assuming P(W$_0$="<s>") = 1.

[Report]

- Provide the perplexities of the 6 models with respect to the respective validation set.
- Compare the perplexities. Are they consistent with your expectations? Why or why not?

2.6 Contest: Social Power Relationship Prediction [20 pts / Extra Credit by Kaggle Rank: #1 - 10 pts, #2 - 7pts, #3 - 5pts, #4&5 - 3 pts, #6-10 - 1pt]

As language models capture which sequences of words are more likely to appear in a given domain, they are applicable to many natural language processing tasks. In this assignment, we consider a type of text classification problem.

The goal here is to guess whether the sender the given email is of a lower rank than the recipient (UPSPEAK) or vice versa (DOWNSPEAK) based on the content of the email. The creation of a decently performing classifier for this task would confirm the idea that people indeed write differently depending on the relative rank of the recipient.

To do this, first train a language model for UPSPEAK, and another one for DOWNSPEAK. You need to train new models using a smoothing method (or methods) of your choice, other than an additive smoothing, to get the full credit. You may also employ a different preprocessing (Here, you may use a stemmer or lemmatizer), etc. to boost the performance.

Then, given an email from UPDOWN_TEST, compute the probability of the email assigned by the 2 language models. If the UPSPEAK model assigns the higher probability, classisfied the email as UPSPEAK, and DOWNSPEAK, otherwise. (You may take a more complicated approach if you wish!)

Classify all the emails in the test set in this way and submit a prediction file of the following format to the Kaggle competition website (`https://inclass.kaggle.com/c/cs-4740-uptalk-vs-downtalk`):

```
Id,Prediction
1,0
```

```
2,0
3,1
4,1
...
```

where 0 = DOWNSPEAK and 1 = UPSPEAK. The predictions will be evaluated automatically, and the rankings will be updated on Kaggle. Note: You can only submit a new solution twice a day, so you will benefit from starting early.

[Report]

- Explain how you train your the language models. (Which smoothing method? Are you doing any special preprocessing?)
- Explain how you classify each email as UPSPEAK or DOWNSPEAK, given the probabilities computed by using the language models.
- How well does you classifier perform?