Computational Linguistics

CSE/LIN 467/567 – Spring 2018

Homework 2

Due date: March 3rd

Instructions

Note: **Answers not submitted as specified below will not be graded**.

Upload to UBLearns a single zip file containing your answers to the two questions below. The zip file must be named '`hw2UBitNAME.zip`, where UBitNAME is your UB email's name. For example, if your email were `nazgul@buffalo.edu` then your homework file must be named '`hw2nazgul.zip`'. Don't forget to click the button after uploading the file, or it will not be submitted.

Students can work with others but must ultimately create their own individual answers. Any homework assignments that are sufficiently similar may be considered instances of plagiarism, and will be penalized and reported. See syllabus for more information about UB's official policy on ethical behavior.

Exercises

1. Your goal is to construct a Prolog bigram language model from the small `DA.txt` corpus at UBLearns. In order to achieve this, you must normalize the text by (at a minimum): (1) removing capitalization, (2) replacing sentence boundary punctuation with '@', (3) removing any other punctuation, and (4) normalizing contractions (*I'd, she'd, he'd, I'll*, etc.). As a simplification, assume that spaces separate each token (in other words, no need to run a tokenizer over the text). Once you are done normalizing, you must create the bigram model and make it Prolog-readable. The result should be two files, each well-formed Prolog (data shown below is fictitious):

   Bigram file:

```
bigram(1874,of,the).
bigram(1138,in,the).
bigram(730,@,the).
bigram(432,from,the).
... and so on
```

   Unigram file:

```
unigram(14074,the).
unigram(11000,be).
unigram(689,to).
unigram(567,of).
... and so on
```

The answer to this exercise should consist of three distinct text files:

- One file, named '`unix_UBitNAME.txt`', must contain all of the **Linux commands**[1] used to create the bigram and unigram files.
- The second and third files should be named '`bigram_UBitNAME.pl`' and '`unigram_UBitNAME.pl`', which as their names indicate, contain the Prolog bigrams and unigrams (see above).

NOTE: the person grading your homework should be able to run your commands to DA.txt and obtain the exact same model, and to load it successfully into Prolog. Please comment your code as illustrated below.[2] Feel free to chain commands via '|'.

```
# Add 'hobbit(' to the beginning of every line in the file
sed -e 's/^/hobbit(/' file1.txt > file2.txt

# Add '!!!' to the end of every line in the file
sed -e 's/$/!!!/' bigrams2.txt > bigrams3.txt
```

The `LexicalProcessing2.pdf` slides contain various examples of the `tr`, `sed`, and `egrep` commands, and `Ngram1.pdf` describes some of the steps needed for creating a bigram model. Moreover, sections 2.1.1 through 2.1.6 of the J&M09 book have many examples of regular expression operators, and their syntax. Additional examples can be found on the web.

[3 points]

2. The answer to this exercise consists of a file '`lm_UBitNAME.pl`' which computes the probability of any word sequence (even sequences containing words not in the model), of any size, via a predicate called `calc_prob`/2. Work in Log space and apply Laplace smoothing on-the-fly. Your file should expand of the following fragment:[3]

```
:- ['bigram_UBitNAME.pl'].
:- ['unigram_UBitNAME.pl'].

calc_prob(ListOfWords,SmoothedLog10Probability):-
      calc_prob(ListOfWords,0,SmoothedLog10Probability).

calc_prob([],N,N).
calc_prob([W1,W2|L], ... , ... ) :-
     ...,
     calc_prob( ..., ..., ...).
```

The `calc_prob` predicate should be able to determine that well-formed strings like *The book that he wanted fell on my feet* is more probable than word salad like *Book the that he wanted fell my on feet.* An example execution is show below (again, figures are fictitious).

```
?- calc_prob([@,the,book,fell,@],X)
X = -5.45
```

[3 points]

[1] For Linux and Mac users, you can simply use your terminal to create the model. For Windows users, you can either use your UBunix account (see here) or download and use MobaXterm Home Edition v10.5 (Portable edition) (use the latter at your own risk). During class, I've connected to UBunix via `Putty` on several occasions (from any computer on campus, you can use the SFTP option to upload and download files to your area, and SSH to run Linux commands).

[2] Using `sed -e` is recommended, as the '-e' allows a wider variety of regular expression operations than standard `sed`.

[3] Some arithmetical operators are discussed in `prolog2.pdf`, in the BBS08 Prolog textbook, and the on-line SWI manual (simply enter text in the 'search documentation' box).