

# CIS 530 Spring 2008

## General information and Assignment 1

(adapted from problem set by Edward Loper)

Due: Tuesday, February 12th, 2008, 4:30pm

### General information

#### How to contact us

The instructor for this class is Mitch Marcus ([mitch@cis.upenn.edu](mailto:mitch@cis.upenn.edu), Levine 503). The TAs for this class are Lucas Champollion ([champoll@ling.upenn.edu](mailto:champoll@ling.upenn.edu), 401/402 Williams Hall) and Qiuye Zhao ([qiuye@seas.upenn.edu](mailto:qiuye@seas.upenn.edu), GRW 571). The course website is <http://www.cis.upenn.edu/~cis530>. Office hours will be announced soon. Feel free to email the TAs for technical support or if you have any questions about the homework. To make sure that your question is answered in time, please send your email at least 24 hours before the due date.

#### Policy on working in groups

This is an introduction to NLP, not a programming course, and we want you to be able to concentrate on NLP issues. So, you are encouraged to help each other debug your programs and tackle the intricacies of Python, NLTK, and any other programming languages you might use. Apart from this specific case, we expect each of you to submit original work. The intellectual effort (designing and writing programs, working on essay questions, etc.) has to be done individually. Submitting homeworks as a group is not possible in this course. If you feel this policy is not optimal or not clear, please come and talk to us.

#### Getting started with Python and NLTK

In this course, we will be using NLTK 0.9.1, available at <http://nltk.sourceforge.net>. Unfortunately, this is not always a straightforward installation, so be sure to start early on in case you run into technical difficulties. Alternatively, you may work remotely on [seas.upenn.edu](http://seas.upenn.edu). If you need a `seas` (i.e. an eniac) account, please email us. IMPORTANT: If you work on `seas`, you need to add the following line to your `.bashrc`:

```
export NLTK_DATA=/home1/c/cis530/data
```

To work with NLTK, you'll want to make extensive use of the API documentation under <http://nltk.org/doc/api>. You may also find the online NLTK textbook on <http://nltk.sourceforge.net/index.php/Book> helpful. It covers many of the topics of the course and often contains coding examples written in NLTK and Python.

Courtesy of Liang Huang, you can find excellent links to textbooks, online tutorials, etc. on the Python language on his CSE399 webpage, <http://www.cis.upenn.edu/~lhuang3/cse399-python/resources.html>.

## How to submit

We ask that you put all of your code for the assignment in a single file. Call it “`hw1_yourpennkey.py`”, and include your test code for each problem at the bottom of the file. That way, you can test what you’ve written so far by evaluating the whole file (with “`ctrl-f5`” from IDLE; with “`ctrl-c ctrl-c`” from Python mode in Emacs; or with “`python hw1_yourpennkey.py`” from the command line).

We have set up an electronic submission system. This will give you immediate feedback on whether your submission was successful. It also greatly simplifies our work. So please submit this and subsequent assignments electronically, rather than by email.

To electronically submit homework, copy the file containing your solution to your SEAS Eniac account using, for example, scp:

```
% scp <filename> eniac.seas.upenn.edu:  
Password: *****
```

Then connect via ssh to `seas.upenn.edu` and use the `turnin` command to submit your file for grading:

```
% turnin -c cis530 -p hw1 <filename>
```

This should print out a confirmation message. Your files must be world-readable in order for them to be submitted.

`turnin` can be run multiple times before the deadline. IMPORTANT: each time you run `turnin`, it overwrites your previous submission for that assignment. To check if the homework submitted OK (only works before the homework deadline):

```
% turnin -c cis530 -v
```

This will show a list of the file(s) you have submitted.

## What to submit

Please turn in an electronic version of your assignment by the beginning of class on the due date. Please put your name, email address, and “Problem Set 1” at the top of your assignment file. Be sure to clearly mark which parts of your solution go with which questions. For each problem, you should turn in:

- The documented source code you wrote for the problem. Documenting your programs is important; it helps us read your code, and helps you write it. You should document all of your code with comments, docstrings, or both. You can also include external documentation, where it seems appropriate.
- Test cases and output from those test cases, showing that your code works (it’s ok to just copy and paste from an interactive session). In most cases, we’ll let you know which test cases we’d like to see the output from, but feel free to include other test cases.
- Written answers to questions (for Problems 5 and 6). Please submit PDF, or text files, or simply add your answer as a comment to the code.
- Any additional comments, questions, or observations.

## 1 Creating a Random Experiment (5 points)

Write a function `roll(n,m)`, which models the experiment of rolling  $n$   $m$ -sided dice, and summing their values. You will probably want to use the function `randint` from Python's `random` module to generate the outcome for each die roll. Remember that the numbers on a die range from 1 to  $m$ , not from 0 to  $m - 1$ .

Test your function with several different values of  $n$  and  $m$ .

## 2 Estimating the Probabilities of Words (15 points)

Create a frequency distribution (named `unigram_dist`) that records the frequency of every word that occurs in Jane Austen's *Emma*. Use the frequency distribution you created to find the count and the frequency for 5 of your favorite words.

*Emma* is included in the NLTK corpora package as a part of the Gutenberg corpus. You can load it like this:

```
from nltk.corpus import gutenberg
corpus = gutenberg.words('austen-emma.txt')
```

## 3 Generating Words (20 points)

Using the class `FreqDist` in the module `nltk.probability`, write a function `generate(fdist)`, that generates a word based on the frequency distribution  $fdist$ . In particular, the probability that `generate(fdist)` generates a word  $w$  should be equal to  $fdist.freq(w)$ . You will probably want to use either the function `random` or the function `randint`, both of which are defined by the `random` module.

Use `generate` to randomly generate 10 words from the frequency distribution that you created for Problem 2 (`unigram_dist`).

## 4 Estimating Conditional Probabilities of Words (15 points)

Using the class `ConditionalFreqDist` in the module `nltk.probability`, create a conditional frequency distribution (named `bigram_dist`) that records the frequency of each word in Austen's *Emma*, given the word that it follows. In particular, the distribution's sample outcomes should be the words in the text, and the context for each sample outcome should be the preceding word.

Use the conditional frequency distribution to find the count and frequency with which your 5 favorite words follow the word "and."

## 5 Generating Sentences (25 points)

Write a function `generate_sequence(cfdist, first_word)` that randomly generates a 10-word sequence of words based on the conditional frequency distribution  $cfdist$ , starting with  $first\_word$ . In particular, the probability that `generate_sequence` picks word  $w_i$  to follow word  $w_{i-1}$  should be equal to

$f_{dist}[w_{i-1}].freq(w_i)$ . You should use the `generate` function that you wrote for Problem 3 (`bigram_dist`) to generate the words.

Use `generate_sequence` to generate five 10-word sequences. Pick a different initial word for each sequence.

Write *brief* answers (about 1 paragraph each) to the following questions:

- How much do the sequences of words generated by `generate_sequence` look like sentences of English? How might you modify the conditional frequency distribution and/or the generation procedure to produce sequences of words that look more like sentences?
- What does your function do if it encounters an unseen context word? What could cause your function to encounter an unseen context word?

## 6 More Context (20 points)

Repeat Problems 4 and 5, but use the previous *two* words for context. Name your conditional frequency distribution `trigram_dist`, and name your function `generate_trigram_sequence`. Note that `generate_trigram_sequence` will need to take two seed words.

How does this additional context affect the quality of the generated sequences? What would happen if you kept adding more and more context?

## 7 Debriefing

1. Approximately how many hours did you spend on this assignment?
2. Would you rate it as easy, moderate, or difficult?
3. How deeply do you feel you understand the material it covers (0%-100%)?
4. Any other comments?

This question is intended to help us calibrate the homework assignments. Your answers will not affect your grade.