Massachvsetts Institvte of Technology
6.S077, Fall 2018
Department of Electrical Engineering and Computer Science
Reading and Response #2: Colorless green ideas sleep furiously

Released: Wednesday, November 8
Due: Wednesday/Friday November 15 or 17, for in-class discussion (depending on which class you go to), with your annotated hardcopy turned in at end of class on that day

Assignment Goals:

1. Learn what *n-grams* can and cannot model about human language
2. Understand part of the statistical vs. grammatical 'debate' in computational linguistics

Readings (first two available on course wiki):

(1) N. Chomsky, extract on grammaticality from *The Logical Structure of Linguistic Theory*, 1955.

(2) S. Abney, extract from "Statistics and Linguistics," in R. Bods, *Statistical Linguistics,* section 4.2.

(3) Wikipedia entry on *n*-grams, here: https://en.wikipedia.org/wiki/N-gram

Please go through the three readings above before tackling the assignment. Your write-up should consist of two pages: Page 1 will be on Part 1, as per below; and Page 2 will be your consultant's report, Part 2. As usual, please bring the hardcopy to class for the discussion.

Part 1 - Programming *N*-Grams

We want you to answer two very simple questions after running an *n-gram* program in NLTK that we have provided (for both python2 and python3). Make sure to bring your answers to class and write them down on your hardcopy because we'll discuss them first.

Preparation. You can learn what an *n-gram* is by reading the Wikipedia entry, as per above. The programming task that follows should take no more than a few minutes, but feel free to play with the program all you want. We have written a Python program, with helper functions that reads in a training corpus, creates an *n*-gram model from it, and generates sentences from that model. The code is in a directory on the assignment 2 wiki page, **`ngram.zip`**, that you can download and unpack. Once it is unpacked, you can change to the directory **`ngram`**, start python, and then do the following (for python2). We have assumed that you've already downloaded the nltk system, as per RR1; if not, go to the page http://www.nltk.org/install.html and follow the instructions there for your machine. Finally, you need to download some corpora that nltk provides. Go to http://www.nltk.org/data.html and follow the instructions from within python that they provide there to load to your computer the **`treebank`** and **`gutenberg`** corpora. The first corpus is a 10% corpus of the Penn Treebank, about 4000 sentences. The second contains lots of different corpora (a bit more on that later). Now we're ready. For python2:

```
>>cd ngram
>> python
Python 2.7.13 |Anaconda 4.3.0 (x86_64)| (default, Dec 20 2016, 23:05:08)
>>> import nltk
>>> from generator import *
>>> from nltk.corpus import treebank
```

If you are using python3, the only change is to import a different generator program:

```
>> python
>>> import nltk
>>> from generator_three_compatible import *
>>> from nltk.corpus import treebank
```

If you get an error when attempting to import the treebank, it means that you probably haven't loaded it in as per the instructions above for loading nltk data corpora – go back and make sure you have done this. OK, assuming now that nltk, the generator program, and the corpus are all loaded, we can now invoke the *n*-gram generator program on the corpus. The main function, **`ngram_generator`** (or its python3 alternative) takes three arguments:

1. The *n* for your *n*-grams (1=unigram; 2=bigram a.k.a. **Markov order** 1; 3=trigram, etc.) **Please note this terminology in parenthesis of "Markov order"–this is also called the "order of approximation."**
2. The number of sentences to generate.
3. The training corpus to use.

If we invoke the program, it should now pause for a bit as it builds up its table of *n*-gram frequencies, and then spit out 3 sentences constructed from bigram statistics, similar to this:

```
>>> ngram_generator(3,2,treebank)
Sentence 1:
While it would pay for $ 2.875 yesterday 's actions , the test scores .
Sentence 2:
Preston G. Kuhns , so strong .
Sentence 3:
Ad Notes ... raised the economic alternatives to obtain additional cash
squeeze , which included free trip from safety problems with Poland 's
stock market in `` only make markets as the Foster Savings & Poor 's plan .
```

You will get different sentences from the ones above, because the generator selection is random. The sentences are generated by calculating the probability of a word given the $n-1$ words before it, normalizing the probabilities of the possible words. The program selects a random word by producing a random number between 0 and 1 and then picking the appropriate word. It stops when it produces some end-of-sentence symbol (a period, question mark, etc.).

OK, what happens if we boost the *n*-gram "window" to 3? (This is called a trigram model since it conditions the third word produced on the preceding two.) We are then increasing the context we make use of. Intuitively, this should make the approximation to English "better." Here is an example:

```
>>> ngram_generator(3,3,treebank)

Sentence 1: But then came Oct. 13 stock market into a referendum on
increasing trade flows .

Sentence 2: `` Compare two candidates for mayor , the Chamber of Commerce ,
the announcer .

Sentence 3: `` I would have fallen 2.1 % .
```

This certainly seems like more natural English. If you want to try out the generator with other corpora besides the treebank, you can do this:

```
>>> from nltk.corpus import gutenberg
>> ngram_generator(3, 3, gutenberg.words('whitman-leaves.txt'))
Sentence 1:  WHISPERS OF HEAVENLY DEATH ] } Proud Music of the universe
does .) Where has fail ' d up , and in ourselves , We welcome what we call
Being .

Sentence 2:  Pioneers !

Sentence 3:  These States , We are executive in ourselves , We have not
learn ' d , the Alb and the master myself , Must I change my triumphant
songs ?
```

***N*-Gram Python Warmup Questions**

1. Try generating a few 4-gram and 5-gram sentences from the treebank corpus. (Please include them in your writeup.) Can you describe how natural and fluent these examples are compared to the 2-gram and 3-gram examples? Are they more or less fluent? Is this effect "real" (i.e., is it the result of boosting the window size the *n*-gram looks at? Please explain what other factor seems to be entering in to the fluency of the sentences that are generated.

2. As *n* increases, the apparent "approximation" to English seems to get better and better. Will it ever match English? Please explain with a concrete example why or why not.

Part 2. GoogleTalk

You have just been hired as a consultant at Google, to assist with their new "GoogleTalk" project. Much like their Google Books project, GoogleTalk aims to collect many, many millions of examples of spoken sentences, initially just in English, transcribed into written text.[1]
You overhear Google employee #25 talking to Google employee #200 about their plans for this data:

"Look," she says, "plainly, the number of sentences one person will *ever* hear or speak in a lifetime is finite – and so therefore is the collection of all the sentences we'll ever put into Google Speech, even if it's trillions and trillions of examples. This collection, a 'corpus,' constitutes the set of 'observables' for natural language. It is this corpus that we have to model. It's just like when we observe some other natural phenomenon, like the motion of the planets. We can use lots and lots of astronomical observations, and then, once we know the position of, say, Saturn at many points in time, we can predict where it will be in the next moment, just by using our collected data. So for example, the probability of where Saturn is at time *t* is just contingent on where it was at some finite measured number of instances in the past. With sentences, we can do the same thing via the method called *n–grams*. An *n–gram* is just a way of predicting what the n^{th} word in a sentence will be, given the *n*–1 preceding words. And that's what we have lots of data about. We can use the probabilities of such sequences to capture what we need to know. For example, if we see the sequence "I'd like to make a collect…" then a very likely next word is *call*, or *phone*, or *international*, but not *the.* It should be a snap. We can do fancier statistics if we need to – I know that sometimes specific, very long sequences won't ever show up in our corpus, so they'll have a frequency of zero, but we now have sophisticated ways of estimating this kind of missing data."

[1]GoogleTalk is fictional… or used to be. When I first wrote this assignment some years back, I thought of the idea of Google having an *n*-gram corpus with so much data as a kind of joke. Little did I realize that Google would take me seriously.

Employee #200 replies, "Wait a minute. Are you sure that's the right thing to study? Isn't the set of sentences that even one person can potentially produce countably infinite? How do you determine what goes on your list, and what does not? And I'm a bit troubled by your physics analogy. I don't think Newton would have appreciated it. Sure, Copernicus and Kepler collected lots and lots of data sequences of planetary motion, but what underlies them, the Copernican heliocentric model, isn't just a statistical approximation – it's an absolute principle. A *theory.* What you want to model – the true 'observables' – isn't what's in the 'outside world', the sequences of words or sentences, but rather the principles of the 'inside world' – the 'cognitive machinery' that produces or perceives this or that collection of sentences. While it's true that one could use the earth-centered Ptolemaic model and epicycles to get an even more exact prediction of planetary motion, the problem is that epicycles don't describe just planetary motion, they can predict *any* periodic motion, real or not. In contrast, the heliocentric Copernican model can't describe *any* periodic motion, *just* the periodic motions of actual planetary objects. In this sense, the heliocentric Copernican model 'reveals complete nature' as Feynman once put it. Similarly, while *n*–grams might be terrific at predicting word sequences, they don't reveal 'complete nature' (i.e., reality) in this sense." (Patrick mentioned this point about epicycle power in lecture last week during an aside regarding the extraneous power of deep nets. It's worth pondering.)

Your assignment: take the first employee's side and argue for her viewpoint. Then, switch sides to the second employee and argue **against** the first employee's viewpoint. **Note:** you will be called on in class to argue both sides, so be prepared!

A good consultant should be able to do a good job arguing both the positive and negative sides of a thesis. And it's quite common for one or both of the positions to be poorly formulated –people are only human. A good consultant should aim at a sympathetic interpretation no matter what the position and attempt to sharpen the proposals, since the goal isn't to score debating points with cheap, shallow, and easy shots.

Before you write your report, you do a bit of research on your own about *n*-grams and the like from **Part 1** of this assignment and answering the questions there in order to focus what you say. You also read through the readings given at the beginning of the assignment:

• You read section 4.2 from Abney's paper, and focus on the second page (the number at the bottom of the page says '20') where he attempts to define 'grammaticality' in terms of probability and Markov order of approximation as follows.

For any sentence *s*, let $P_n(s)$ be the probability of *s* according to the best n^{th} order approximation (i.e., *n*+1-gram model) to English. Then a sentence is grammatical iff the limit as *n* goes to infinity of $P_n(s)$ is greater than 0, i.e.,

$$\text{grammatical}(s) \leftrightarrow \lim_{n\to\infty} P_n(s) > 0$$

In your answer, you attempt to address what Abney's statement means. Is it coherent? Does it even make sense? What does it really say about human language sentences? Does it 'work'? Is it too broad? Too narrow? What's right about it? What's wrong about it? And so on!

• You read the extract starting at section 36.1 from Chomsky's 1954-55 book on one difference between "probable" and "grammatical" and how statistics and linguistics might connect. This is the first appearance in print of the famous sentence "Colorless green ideas sleep furiously," but unfortunately it's not often read in its full context. Pay attention to the empirical evidence that Chomsky uses to determine how we know that the example "colorless green ideas sleep furiously" is actually just as grammatical as revolutionary new ideas appear infrequently," even though their strict probability of occurrence might

differ wildly. Incidentally, Chomsky also mentions Markov models when he uses the (then conventional) term 'n^{th} order of approximation.' In your answer, compare Chomsky's explanation to Abney's and the *n-gram* approach. What is Abney's notion of a "model of human knowledge of language"? What is Chomsky's? How are they the same or different? Note: this is *not* a straightforward debate–people are still arguing about it, and it speaks to the very essence of what it means to construct a scientific theory of human language.

With all that behind you, you should now write up your 1-page analyst's report on GoogleTalk.

Bonus to add to your commentary: Please meditate on the following quote attributed to the late Fred Jelinek, a pioneer of statistical machine translation, relating it to your report: "*Anytime a linguist leaves the group our machine translation accuracy goes up.*" (Specifically, compare and contrast the following analogical aphorism in relationship to Jelinek's remark: "*Anytime a particle moving near the speed of light leaves the group, our classical mechanics prediction accuracy goes up."*)