IPython Notebook - N-gram tutorial

First I'll see how far I can get with N-grams without outside resources

We have a text file for Pride and Prejudice from Project Gutenberg (https://www.gutenberg.org/ebooks/1342) stored as `pg1342.txt` in the same folder as our notebook. Let's load the text to a string since it's only 701KB, which will fit in memory nowadays.

> *Note* : If we wanted to be more memory efficient we should parse the text file and store per word, etc.

In [188]:
```
with open('pg1342.txt', 'r') as f:
    txt = f.read()

# See the number of characters and the first 50 characters to confirm it is there
print len(txt), ',', txt[:50] , '...'
```

```
704149 , The Project Gutenberg EBook of Pride and Prejud ...
```

Great, now lets split into words into a big list, splitting on anything non-alphanumeric [A-Za-z0-9] (as well as punctuation) and forcing everything lowercase

In [189]:
```
words = re.split('[^A-Za-z]+', txt.lower())
words = filter(None, words) # Remove empty strings

# Print length of list
print len(words)
```

```
125897
```

Sets

From this we can now generate N-grams, lets start with a 1-gram, basically the set of all the words

> *Note* : One could use a dictionary instead of a set and keeping count of the occurances gives word frequency

In [190]:
```
import sets

# Create set of all unique words, this throws away any information about frequency however
gram1 = set(words)

print len(gram1)

# Instead of printing all the elements in the set, create an iterator and print 20 elements only
gram1_iter = iter(gram1)
print [gram1_iter.next() for i in xrange(20)]
```

```
6528
['foul', 'four', 'woods', 'hanging', 'woody', 'looking', 'eligible', 'scold', 'lord', 'meadows', 'sinking', 'leisurely', 'bringing', 'disturb', 'recollections', 'wednesday', 'piling', 'persisted', 'succession', 'tired']
```

Lets try and get the 2-gram now, which is pairs of words. Let's have a quick look to see the last 10 and how they look.

In [191]:
```
# See the last 10 pairs
for i in xrange(len(words)-10, len(words)-1):
    print words[i], words[i+1]
```

```
subscribe to
to our
our email
email newsletter
newsletter to
to hear
hear about
about new
new ebooks
```

Okay, seems good, lets get all word pairs, and then generate a set of unique pairs from it

In [192]:
```
word_pairs = [(words[i], words[i+1]) for i in xrange(len(words)-1)]
print len(word_pairs)

gram2 = set(word_pairs)
print len(gram2)

# Print 20 elements from gram2
gram2_iter = iter(gram2)
print [gram2_iter.next() for i in xrange(20)]
```

```
125896

55636
[('her', 'taste'), ('every', 'kind'), ('five', 'shillings'), ('soothe
d', 'but'), ('seemed', 'most'), ('fortune', 'it'), ('of', 'thanking'),
('near', 'she'), ('understand', 'from'), ('it', 'looks'), ('have', 'mad
e'), ('lucas', 'he'), ('fail', 'him'), ('new', 'to'), ('nothing', 'bu
t'), ('fearful', 'on'), ('to', 'wander'), ('write', 'rather'), ('of',
'studying'), ('interruption', 'from')]
```

Frequency

Okay, that was fun, but this isn't enough, we need frequency if we want to have any sense of probabilities, which is what N-grams are about. Instead of using sets, lets create a dictionary with counts

In [193]:
```
gram1 = dict()

# Populate 1-gram dictionary
for word in words:
    if gram1.has_key(word):
        gram1[word] += 1
    else:
        gram1[word] = 1 # Start a new entry with 1 count since saw it fo
r the first time.

# Turn into a list of (word, count) sorted by count from most to least
gram1 = sorted(gram1.items(), key=lambda (word, count): -count)

# Print top 20 most frequent words
print gram1[:20]
```

```
[('the', 4507), ('to', 4243), ('of', 3730), ('and', 3658), ('her', 222
5), ('i', 2070), ('a', 2012), ('in', 1937), ('was', 1847), ('she', 171
0), ('that', 1594), ('it', 1550), ('not', 1450), ('you', 1428), ('he',
1339), ('his', 1271), ('be', 1260), ('as', 1192), ('had', 1177), ('wit
h', 1100)]
```

For Pride and Prejudice, the words 'the', 'to', 'of', and 'and' were the top four most common words. Sounds about right, not too interesting yet, lets see what happens with 2-grams.

In [194]:
```
gram2 = dict()

# Populate 2-gram dictionary
for i in xrange(len(words)-1):
    key = (words[i], words[i+1])
    if gram2.has_key(key):
        gram2[key] += 1
    else:
        gram2[key] = 1

# Turn into a list of (word, count) sorted by count from most to least
gram2 = sorted(gram2.items(), key=lambda (_, count): -count)

# Print top 20 most frequent words
print gram2[:20]
```

```
[(('of', 'the'), 491), (('to', 'be'), 445), (('in', 'the'), 397),
(('i', 'am'), 303), (('mr', 'darcy'), 273), (('to', 'the'), 268), (('o
f', 'her'), 261), (('it', 'was'), 251), (('of', 'his'), 235), (('she',
'was'), 212), (('she', 'had'), 205), (('had', 'been'), 200), (('it', 'i
s'), 194), (('i', 'have'), 188), (('to', 'her'), 179), (('that', 'he'),
177), (('could', 'not'), 167), (('he', 'had'), 166), (('and', 'the'), 1
65), (('for', 'the'), 163)]
```

It looks like "`of the`" and "`to be`" are the top two most common 2-grams, sounds good.

Next word prediction

What can we do with this? Well lets see what happens if we take a random word from all the words, and build a sentence by just choosing the most common pair that has that word as it's start.

In [195]:
```
start_word = words[len(words)/4]
print start_word
```

```
enough
```

I just went ahead and chose the word that appears 1/4 of the way into words, random enough.

Now in a loop, iterate through the frequency list (most frequent first) and see if it matches the first word in a pair, if so, the next word is the second element in the word pair, and continue with that word. Stop after N words or the list does not contain that word.

```
*Note* : gram2 is a list that contains (key,value) where key is a word pair
(first, second),
         so you need element[0][0] for first word and element [0][1] for seco
nd word
```

In [196]:
```
def get2GramSentence(word, n = 50):
    for i in xrange(n):
        print word,
        # Find Next word
        word = next((element[0][1] for element in gram2 if element[0][0]
== word), None)
        if not word:
            break

word = start_word
print "Start word: %s" % word

print "2-gram sentence: \"",
get2GramSentence(word, 20)
print "\""
```

```
Start word: enough
2-gram sentence: " enough to be so much as to be so much as to be so mu
ch as to be so much "
```

It gets stuck in a loop pretty much straight away. Not very interesting, try out other words and see what happens.

In [197]:
```
for word in ['and', 'he', 'she', 'when', 'john', 'never', 'i', 'how']:
    print "Start word: %s" % word

    print "2-gram sentence: \"",
    get2GramSentence(word, 20)
    print "\""
```

```
Start word: and
2-gram sentence: " and the whole of the whole of the whole of the who
le of the whole of the whole of the "
Start word: he

2-gram sentence: " he had been so much as to be so much as to be so m
uch as to be so much "
Start word: she
2-gram sentence: " she was not be so much as to be so much as to be s
o much as to be so "
Start word: when
2-gram sentence: " when she was not be so much as to be so much as to
be so much as to be "
Start word: john
2-gram sentence: " john with the whole of the whole of the whole of t
he whole of the whole of the whole of "
Start word: never
2-gram sentence: " never be so much as to be so much as to be so much
as to be so much as "
Start word: i
2-gram sentence: " i am sure i am sure i am sure i am sure i am sure
i am sure i am "
Start word: how
2-gram sentence: " how much as to be so much as to be so much as to b
e so much as to be "
```

Weighted random choice based on frequency

Same thing. Okay, lets randomly choose from the subset of all 2grams that matches the first word, using a weighted-probability based on counts.

In [198]:
```
import random
def weighted_choice(choices):
   total = sum(w for c, w in choices)
   r = random.uniform(0, total)
   upto = 0
   for c, w in choices:
      if upto + w > r:
         return c
      upto += w

def get2GramSentenceRandom(word, n = 50):
    for i in xrange(n):

        print word,
        # Get all possible elements ((first word, second word), frequenc
y)
        choices = [element for element in gram2 if element[0][0] == wor
d]
        if not choices:
            break

        # Choose a pair with weighted probability from the choice list
        word = weighted_choice(choices)[1]
```

In [199]:
```
for word in ['and', 'he', 'she', 'when', 'john', 'never', 'i', 'how']:
    print "Start word: %s" % word

    print "2-gram sentence: \"",
    get2GramSentenceRandom(word, 20)
    print "\""
```

```
Start word: and
2-gram sentence: " and laughed the party we do not care of sense of y
ou so far differ from the younger boys were "
Start word: he
2-gram sentence: " he will probably been her relations was not by let
ting her fine i do you do not write again caroline "
Start word: she
2-gram sentence: " she could not fetched them purposely kept from the
tone is fifty after a servant who do it was indeed "
Start word: when
2-gram sentence: " when they descended on the rest of poor regiment w
here he takes part of hunsford every glance from making two "
Start word: john
2-gram sentence: " john told me was as you would not but whatever mig
ht not hope she had passed it so i have "
Start word: never
2-gram sentence: " never exert himself the only be acceptable to my p
retensions whatever manner that she had been hurt by miss bingley "
Start word: i
2-gram sentence: " i had ever be considered her former way said lydia
had she yes ma am only shows some part of "
Start word: how

2-gram sentence: " how to your success and fearful of sense of import
ance in their income as his favour of it was not "
```

Now that's way more interesting! Those are starting to look like sentences!

Note It's pretty interesting to see that for the sentence " when he believe d him from the amiable but mrs hurst s being ill of being the discussion of c ourse of ", we have hurst s, which we can tell came from Hurst's, an artifact of our stripping away all punctuation but keeping the s.

Let's try a longer sentence

In [200]:
```
word = 'it'
print "Start word: %s" % word
print "2-gram sentence: \"",
get2GramSentenceRandom(word, 50)
print "\""
```

```
Start word: it
2-gram sentence: " it impossible not well inquire replied jane one comm
unication of them only hear that she quitted the hope that he started a
nd all what can never look at least if you he had ended only consider t
hat it aloud if a comfort of her to be in paragraph e or "
```

Pretty cool, lets see what happens when we go to N-grams above 2.

Tri-grams and more

Okay, let's create a Ngram generator that can let us make ngrams of arbitrary sizes

In [201]:
```
def generateNgram(n=1):
    gram = dict()

    # Some helpers to keep us crashing the PC for now
    assert n > 0 and n < 20

    # Populate N-gram dictionary
    for i in xrange(len(words)-(n-1)):
        key = tuple(words[i:i+n])
        if gram.has_key(key):
            gram[key] += 1
        else:
            gram[key] = 1

    # Turn into a list of (word, count) sorted by count from most to lea
st
    gram = sorted(gram.items(), key=lambda (_, count): -count)
    return gram

trigram = generateNgram(3)
# Print top 20 most frequent ngrams
print trigram[:20]
```

```
[(('i', 'do', 'not'), 62), (('i', 'am', 'sure'), 62), (('project', 'gut
enberg', 'tm'), 57), (('as', 'soon', 'as'), 55), (('she', 'could', 'no
t'), 50), (('that', 'he', 'had'), 37), (('in', 'the', 'world'), 34),
(('it', 'would', 'be'), 34), (('i', 'am', 'not'), 32), (('i', 'dare',
'say'), 31), (('the', 'project', 'gutenberg'), 31), (('could', 'not',
'be'), 30), (('it', 'was', 'not'), 30), (('that', 'he', 'was'), 29),
(('mr', 'darcy', 's'), 29), (('that', 'it', 'was'), 28), (('on', 'the',
'subject'), 28), (('as', 'well', 'as'), 27), (('would', 'have', 'bee
n'), 27), (('of', 'mr', 'darcy'), 27)]
```

Cool! Okay, let's see a selection of sentences for N-grams with N = 2 to 10 and a few starting words!

In [206]:
```
def getNGramSentenceRandom(gram, word, n = 50):
    for i in xrange(n):
        print word,
        # Get all possible elements ((first word, second word), frequenc
y)
        choices = [element for element in gram if element[0][0] == word]
        if not choices:
            break

        # Choose a pair with weighted probability from the choice list
        word = weighted_choice(choices)[1]
for n in xrange(2,10):
    # Generate ngram list
    print
    print "Generating %d-gram list..." % n,
    ngram = generateNgram(n)
    print "Done"

    # Try out a bunch of sentences
    for word in ['and', 'he', 'she', 'when', 'john', 'never', 'i', 'ho
w']:
        print "  %d-gram: \"" % n,
        getNGramSentenceRandom(ngram, word, 15)
        print "\""
```

```
Generating 2-gram list... Done
  2-gram: " and aunt her mother s laws and agreeable surprise in the
objections i have more "
  2-gram: " he had not be an abominable i am i desire to ride to the
dishes "
  2-gram: " she must feel it would go so many many of a part of jane
would "
  2-gram: " when they were to all praise of appearing highly expedien
t to him good wishes it "
  2-gram: " john with the expectation of jane must be lamented becaus
e she hoped to account of "
  2-gram: " never saw no one before michaelmas but there was neither
such a young lady lucas "
  2-gram: " i anything but a rival to me fresh excuse for that this b
e successful love "
  2-gram: " how to be in without some time since they were consigned
over wickham s estate "

Generating 3-gram list... Done
  3-gram: " and never for the table between darcy s approaching the b
eginning but the object to "
  3-gram: " he replied endeavouring to be in gracechurch street who h
ad not sink under the list "
  3-gram: " she when so much by mr collins no further opportunities o
f advantage the year ago "
  3-gram: " when i shall see her coughs said i do you mean by what i
hope "
  3-gram: " john told me were insufficient to the father she had less
```

```
eager in hertfordshire anxiously "
  3-gram: " never disgrace which officers had promised me for a compl
iment to afford by no look "
  3-gram: " i am very great to him from netherfield and the acknowled
ged had formerly spoken in "
  3-gram: " how is that point upon herself and once for the ladies in
which is sadly "

Generating 4-gram list... Done
  4-gram: " and whenever any time without her sister panting for no r
eason in those beautiful oaks "
  4-gram: " he liked him too soon as i shall be much for his misfortu
nes have been "
  4-gram: " she consulted one sister we will make them directly retre
ating but that he thinks of "
  4-gram: " when they entered the table what they were not be they so
laced their fortune she "
  4-gram: " john told miss bingley both as she was at pemberley house
amidst the excellent understanding "
  4-gram: " never could she did she had given him in such very long d
oes not he "
  4-gram: " i suppose are as lady catherine s being loved them if i r
eally vexed her "
  4-gram: " how he deserves elizabeth eagerly calling back on i wish
sir william was very constantly "

Generating 5-gram list... Done
  5-gram: " and i have been more i felt himself to find it shows that
though his "
  5-gram: " he was a hurry as soon out to imagine till they are jokin
g lizzy said "
  5-gram: " she was the conclusion of such a suitable to your humilit
y mr collins arrived mrs "
  5-gram: " when she read the former residence in such an understandi
ng the subject before any extraordinary "
  5-gram: " john with a family everybody as to be the concern in repl
y to every body "
  5-gram: " never heard nothing of poverty comparative height of spea
king mr wickham repeatedly exclaiming i imagine "
  5-gram: " i will never play and when you had alone what he asked hi
m how lydia "
  5-gram: " how she made no farther it will not afraid i can he was a
nother had "

Generating 6-gram list... Done
  6-gram: " and elizabeth was to think caroline s shop and affording
her daughter mr robinson did "
  6-gram: " he had been mr bennet s sour looks just over hertfordshir
e and such very powerful "
  6-gram: " she drew their view it was therefore instead of her drew
them with incredulous solicitude "
  6-gram: " when she really with us give the happy he is the appearan
ce created a prudential "
  6-gram: " john with her fancy her inquiries after sitting together
```

because there was exceedingly troublesome she "
  6-gram: " never have been he did elizabeth felt that her to dance w
ith whom they are "
  6-gram: " i grieve to make no effect of my own want one of mr darcy
but "
  6-gram: " how much better have imagined that the whole scene betwee
n us we first and she "

Generating 7-gram list... Done
  7-gram: " and that would not laugh at longbourn desiring her return
my friend s eyes in "
  7-gram: " he poor family they met he imagined that i wish to throw
any way to "
  7-gram: " she came in so mr collins cried up the door with the same
way to "
  7-gram: " when he might be and not return to colonel and steadfastl
y given him mr bennet "
  7-gram: " john with a great an accidental meeting him or other day
but yet i do "
  7-gram: " never pay his marrying one of most wished for your sash m
y watchfulness has the "
  7-gram: " i have gone down and the sense of superior bingley but wh
atever were consigned over "
  7-gram: " how could colonel of a sensibility when the whole party a
t his head i should "

Generating 8-gram list... Done
  8-gram: " and at all the insolent and within view the skill of our
sentiments you at "
  8-gram: " he was completely puzzled mrs bennet accompanied by kitty
she shall hope in every thing "
  8-gram: " she turned again and perhaps it is a shorter space of rel
ations at length return "
  8-gram: " when the truth the acquaintance long enough do on the net
herfield i am glad you "
  8-gram: " john told the invitation could have the terms again her f
ather however bare of mr "
  8-gram: " never have already written with an upper window watching
her family and danced next morning "
  8-gram: " i feel as such a remarkably well as deficient in and orde
r to make his "
  8-gram: " how much less interesting object but my mind my word of f
elicity really anxious to "

Generating 9-gram list... Done
  9-gram: " and was to my youngest is affection elizabeth went into h
er i you are wanted "
  9-gram: " he replied as to hear of the possibility of her and the o
wner of assisting "
  9-gram: " she seldom appeared kitty stared at longbourn if a few we
eks with great deal to "
  9-gram: " when bingley had before expressed her home these two peop
le on either i shall be "

```
  9-gram: " john told us long to kitty for him but let us of its beau
ty elegance "
  9-gram: " never occurred to stay i will not have been quite so exce
ssively deceived in quest "
  9-gram: " i am sure unless the greater that they had cost him his s
eeing them and "
  9-gram: " how to the compliment of being most active useful and she
been intended to have "
```

You can clearly see the sentences getting better and better with larger n-grams, this correlates to the ngram having more foresight into the sentence structure.

In [209]:
```
# Generate 10gram list
print
print "Generating %d-gram list..." % n,
gram10 = generateNgram(10)
print "Done"
```

```
Generating 9-gram list... Done
```

Let's play with the 10gram and see what sort of sentence comes out.

In [210]:
```
# Try out a bunch of sentences
for word in ['and', 'he', 'she', 'when', 'john', 'never', 'i', 'how']:
    print "  %d-gram: \"" % n,
    getNGramSentenceRandom(ngram, word, 100)
    print "\""
```

```
  9-gram: " and tried to please she is somewhere i am much but he had
felt jane said he means discouraged by many people did she promised t
o see of mrs collins in every savage can you the following each other
side by his felicity to request that we shall hope there is unaccount
able my readiness to leave would be happy than they parted from jane
exclaimed elizabeth was now and unvarying society as placing a word e
specially towards them to for he briefly replied to occur in her frie
nd but he had brought up to invite him proud ill natured and "
  9-gram: " he looked a very ill health imputing his attachment i had
her power of which case here beg you will be learnt for a man ten tho
usand pounds were persuaded of removing from their sisters are very p
romising thing in the ladies drove off we acted is about netherfield
i should at last how well cried her the room was still he first came
and thought and at rosings and answer therefore without any direction
s about three or have led him conditionally only six officers may be
equalled by his respect forbid it would not repeat every doubt they w
ere "
  9-gram: " she thought i should not thought of my head of what she h
ad suffered and proofread public domain print editions will be sure w
e do nothing the two women had heard him to prepare or refrain from a
step forward may compare our best part of the room and each of both s
ides but as possible and be greater than gratified indeed i suppose h
```

```
e had sources of lydia s fancying himself and my name throughout nume
rous but bingley has left london and returned home but of injuring yo
ur rejection of natural degree of a visit till sir william who "
  9-gram: " when we darcy drawn to the usual satisfaction they must n
ot be the next sentence might occasionally a voice was nothing due to
longbourn and let me for the happy spirits which had seen him at each
and had related the particulars but she times his frequent mention su
ch gentleness in it could not that the eye was a proposal accepted th
e pemberley with us if you thought only regretting that he held out t
he honour of accident that in this subject occasioned his returning t
o the impertinent myself what are never distinguished by her justice
he was such a "
  9-gram: " john with such civility mr bennet was trying to lady luca
s to be entitled to expect to send her in the work on saturday night
out of my uncle mr darcy has been revealed for though her and with yo
u all mamma what they had certainly be freely to longbourn and the fo
od of the man of writing and introduce you like him walking several m
inutes said with perfect symmetry in a melancholy to disguise yoursel
f mr darcy it was occasionally laughing at her though she was persuad
ed that i give pleasure i wish to be to be in the "
  9-gram: " never act do not hear it is to have mr darcy as well bred
were surprised was satisfactory a family obstacles of mr wickham shou
ld be as they ought to the room at all the honour which as he would n
ow for my fair as to purchase in the case the ladies in some curious
to be so full of her ladyship was a dozen in nursing your account for
by everyone to discharge of merit with which had been to think of a m
atter it to her friends than what she was exceedingly as for the grea
test patience in "
  9-gram: " i believe her daughter would not to anybody for a renewal
of either of a cause of her own but she longed to find herself the re
st of letting them and shopping and talk and just from it to have bee
n so alarming you wish to her she ventured to music cried elizabeth h
ad for avoiding him as soon as we have been led to fret over this may
consult the pleasure in the fee is indeed i came in bingley saw herse
lf as warmly in which must have been cautious of her mother s speech
she was not judge "
  9-gram: " how strange yes it is insinuating and her reflections by
repeated conversations occurring at that no one admitted a very jane
however temporary its windings but lady catherine with herself and ga
ve birth to return they could be in the word that he meant to any fat
her and excepting even at longbourn of girl to a companion lydia and
agreeable woman but that is to newcastle and consequently after her m
other she wanted you were just returned instantly driven from a quart
er than marriage of that emphatic exclamation cried bingley came to f
ight wickham suspended amongst which had feared it "
```

Looks almost like normal sentences if you squint a little! Well, that was fun. Next up let's see some ways to improve upon this.

Instead of just taking the next word every time, we could take the next k words etc.

To be continue...

In [ ]: