Lecture 8: N-gram Resources, Corpus Linguistics

Ling 1330/2330 Computational Linguistics
Na-Rae Han, 9/21/2023

Objectives

- N-gram resources
  - Norvig/Google 1T data
  - COCA n-grams
- Processing a corpus with NLTK
- Introduction to corpora

General, LARGER n-gram stats

- The Bible and Austen bigram stats reflect their unique topical content and linguistic traits.
- Can we find n-gram stats that are extracted from…
  - more GENERAL-domain text?
  - LARGER amounts of text?

- Norvig/Google
- COCA

Data Resources:

- NLTK Corpora Index [page]
- Natural Language Corpus Data by Peter Norvig [link]
- Google Books Ngram Viewer Data [link] *(Slow? Try FireFox.)*
- COCA *n*-gram lists at BYU [link]

Excerpted & manageable: Norvig

- Natural Language Corpus Data: Beautiful Data
  - by Peter Norvig
  - https://norvig.com/ngrams/
  - Has lists of large-scale English *n*-gram data: character (1- & 2-grams) and word level (1, 2, 3 grams)
  - Data derived/excerpted from Google Web 1T 5-Gram corpus
    - ← ¼ million most frequent bigrams
      - ← Google's original data is 315 mil

Extremely large: Google Web 1T

- "All our N-gram are Belong to You"
  - https://ai.googleblog.com/2006/08/all-our-n-gram-are-belong-to-you.html
  - **Google Web 1T 5-Gram**, released in August 2006 through LDC (Linguistic Data Consortium)
  - 1-5 grams
  - Compiled from **1-trillion words** of running web text
  - 24 GB of compressed text

    ← Source of Norvig's 1- and 2-gram frequency lists

- Publication of this data triggered huge advances in NLP technologies and applications.

Norvig/Google data: 1- & 2-grams

- count_1w.txt

```
the     23135851162
of      13151942776
and     12997637966
to      12136980858
a       9081174698
in      8469404971
for     5933321709
is      4705743816
on      3750423199
that    3400031103
by      3350048871
this    3228469771
with    3183110675
i       3086225277
you     2996181025
it      2813163874
not     2633487141
or      2590739907
be      2398724162
are     2393614870
from    2275595356
at      2272272772
as      2247431740
your    2062066547
```

Different sorting order

Total # of entries:
← 333k
vs.
¼ mil →

- count_2w.txt

```
you graduate    117698
you grant       103633
you great       450637
you grep        120367
you grew        102321
you grow        398329
you guess       186565
you guessed     295086
you guys        5968988
you had 7305583
you hand        120379
you handle      336799
you hang        144949
you happen      627632
you happy       603963
you has 198447
you hate        637001
you have        135266690
you havent      134438
you having      344344
you he  199259
you head        205910
you hear        2963179
you heard       1267423
```

Unigram frequency: two data objects

```
>>> goog1w_rank[:5]
    [('the', 23135851162), ('of', 13151942776), ('and', 12997637966),
    ('to', 12136980858), ('a', 9081174698)]
>>> goog1w_rank[0]
    ('the', 23135851162)
>>> goog1w_rank[-1]
    ('golgw', 12711)


>>> goog1w_fd['platypus']
    565585
>>> goog1w_fd.most_common(5)
    [('the', 23135851162), ('of', 13151942776), ('and', 12997637966),
    ('to', 12136980858), ('a', 9081174698)]
>>> type(goog1w_fd)
    <class 'nltk.probability.FreqDist'>
```

(1) **a list** where each item is **(word, count)** tuple.

We will keep the original **order**, which reflects the **frequency rank**.

(2) **a frequency distribution**
nltk.FreqDist where each word is mapped to its count

But let's think about this one…

Norvig/Google ngrams as FreqDist & CFD: caution

```
>>> goog1w_fd['platypus']
    565585
>>> goog1w_fd.most_common(5)
    [('the', 23135851162), ('of', 13151942776), ('and', 12997637966),
    ('to', 12136980858), ('a', 9081174698)]

>>> goog2w_fd[('of', 'the')]
    2766332391

>>> goog2w_cfd['so']['beautiful']
    612472
```

`.freq()` method will not be valid with these. WHY?

Relative frequency should be measured against the total corpus size of 1 trillion, but Norvig's files are missing a WHOLE mass of small unigram/bigram counts.

If you're curious: with unigram FD, you can manually assign _N to be 1 trillion, after which .freq() will be valid.

COCA: Still pretty large

- COCA *n*-gram lists
  - https://www.ngrams.info/download_coca.asp
  - Word 2-5 grams, each containing top ~1 million entries
  - Based on COCA (The Corpus of Contemporary American English) (https://corpus.byu.edu/coca/), 1.0 billion words (1990-2019)

- COCA's full unigram list is not free.
- COCA's top 5000 words/lemmas
  - https://www.wordfrequency.info/samples.asp
  - Contains lemma and POS of top 5,000 words

COCA n-gram lists

- w2_.txt (bigrams)

66	a	ba
41	a	babble
28	a	babbling
159	a	babe
83	a	baboon
9744	a	baby
31	a	baby-faced
122	a	baby-sitter
237	a	babysitter
23	a	babysitting
95	a	baccalaureate
71	a	bach
1342	a	bachelor
27	a	bachelorette
53	a	bachelors
1924	a	back
38	a	back-and-forth
24	a	back-door
29	a	back-to-basics
27	a	back-to-school
100	a	back-up

- w3_.txt (3-grams)

33	a	ba	in
35	a	babble	of
33	a	babe	in
316	a	baby	and
25	a	baby	as
73	a	baby	at
32	a	baby	before
53	a	baby	bird
57	a	baby	boomer
34	a	baby	born
146	a	baby	boy
36	a	baby	brother
29	a	baby	by
34	a	baby	can
45	a	baby	carriage
45	a	baby	crying
39	a	baby	doll
47	a	baby	for
41	a	baby	from
224	a	baby	girl
35	a	baby	grand

Anything you noticed?

2-grams: Norvig vs. COCA

- count_2w.txt

you get	25183570
you getting	430987
you give	3512233
you go	8889243
you going	2100506
you gone	210111
you gonna	416217
you good	441878
you got	4699128
you gotta	668275
you graduate	117698
you grant	103633
you great	450637
you grep	120367
you grew	102321
you grow	398329
you guess	186565
you guessed	295086
you guys	5968988
you had	7305583
you hand	120379

Compiled from:
← 1 trillion words
vs.
500 million words →

- w2_.txt

39509	you	get
30	you	gets
31	you	gettin
861	you	getting
263	you	girls
24	you	git
5690	you	give
138	you	given
169	you	giving
182	you	glad
46	you	glance
23594	you	go
70	you	god
54	you	goddamn
115	you	goin
9911	you	going
1530	you	gon
262	you	gone
444	you	good
25	you	google
19843	you	got

2-grams: Norvig vs. COCA

- count_2w.txt

```
you get          25183570
you getting      430987
you give         3512233
you go           8889243
you going        2100506
you gone         210111
you gonna        416217
you good         441878
you got          4699128
you gotta        668275
you graduate     117698
you grant        103633
you great        450637
you grep         120367
you grew         102321
you grow         398329
you guess        186565
you guessed      295086
you guys         5968988
you had          7305583
you hand         120379
```

Total # of entries:
← ¼ million*
vs.
1 million →

*NOT google's fault! Norvig only took top 0.1% of 315 million.

Usefulness?

- w2_.txt

```
39509    you    get
30       you    gets
31       you    gettin
861      you    getting
263      you    girls
24       you    git
5690     you    give
138      you    given
169      you    giving
182      you    glad
46       you    glance
23594    you    go
70       you    god
54       you    goddamn
115      you    goin
9911     you    going
1530     you    gon
262      you    gone
444      you    good
25       you    google
19843    you    got
```

NLTK's corpus methods

- NLTK provides methods for reading in a corpus as a "corpus object".
  - https://www.nltk.org/book/ch02.html#loading-your-own-corpus
- Many different methods available for different types of corpora (plain text? POS-tagged? XML format?)
  - `PlaintextCorpusReader` suits our corpora just fine.
- Once a corpus is "loaded" and a corpus object created, we can use pre-defined methods on the corpus.

Two sample corpora

- MLK.zip
  - A corpus of Martin Luther King Jr. speeches
- sou.zip
  - State-of-the-Union speeches by Bush and Obama
- Download, and unzip

Assignments:

- Exercise #1, #2, #3, #4, #5
- Homework #1, #2

Learning Python:

- Python 3 Notes
- FAQ
- Text samples (for copy-pasting)
- Short text files: mary-short.txt, tale.txt, how-do-i.txt, gettysburg_address.txt, gift-of-magi.txt
- Zipped corpora: MLK.zip (Martin Luther King Jr. speeches), sou.zip (SoU speeches by Bush & Obama)

Loading your own corpus: MLK

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = "C:/Users/narae/Documents/ling1330/MLK"
>>> mlkcor = PlaintextCorpusReader(corpus_root, '.*txt')
>>> type(mlkcor)
    <class 'nltk.corpus.reader.plaintext.PlaintextCorpusReader'>
>>> mlkcor.fileids()
    ['1963-I Have a Dream.txt', '1964-Nobel Peace Prize Acceptance
    Speech.txt', '1967-Beyond Vietnam.txt', "1968-I've been to the
    Mountain Top.txt"]
>>> len(mlkcor.fileids())
    4
>>> mlkcor.fileids()[0]
    '1963-I Have a Dream.txt'
```

.* means any character, in any number

File names without path work as the file ID

.words() is a list of word tokens

```
>>> mlkcor.words()
    ['I', 'Have', 'A', 'Dream', 'by', 'Dr', '.', 'Martin', ...]
>>> len(mlkcor.words())
    15801
>>> mlkcor.words()[:50]
    ['I', 'Have', 'A', 'Dream', 'by', 'Dr', '.', 'Martin', 'Luther',
    'King', 'Jr', '.', 'Delivered', 'on', 'the', 'steps', 'at',
    'the', 'Lincoln', 'Memorial', 'in', 'Washington', 'D', '.', 'C',
    '.', 'on', 'August', '28', ',', '1963', 'I', 'am', 'happy',
    'to', 'join', 'with', 'you', 'today', 'in', 'what', 'will',
    'go', 'down', 'in', 'history', 'as', 'the', 'greatest',
    'demonstration']
>>> mlkcor.words('1967-Beyond Vietnam.txt')
    ['Beyond', 'Vietnam', ':', 'A', 'Time', 'To', 'Break', ...]
>>> len(mlkcor.words('1967-Beyond Vietnam.txt'))
    7555
>>> len(mlkcor.words('1963-I Have a Dream.txt'))
    1778
```

Spill-proof!

All tokenized words in this corpus

Word tokens in individual file/ speech

.sents() is a list of sentence tokens

```
>>> mlkcor.sents()
    [['I', 'Have', 'A', 'Dream'], ['by', 'Dr', '.', 'Martin', 'Luther',
    'King', 'Jr', '.'], ...]
>>> mlkcor.sents()[0]
    ['I', 'Have', 'A', 'Dream']
>>> mlkcor.sents()[6]
    ['It', 'came', 'as', 'a', 'joyous', 'daybreak', 'to', 'end', 'the',
    'long', 'night', 'of', 'captivity', '.']
>>> mlkcor.sents()[-1]
    ['Mine', 'eyes', 'have', 'seen', 'the', 'glory', 'of', 'the',
    'coming', 'of', 'the', 'Lord', '.']
>>> len(mlkcor.sents())
    718
```

Again, spill-proof.

1st sentence.

Last sentence

718 sentences in this corpus

Unlike `nltk.sent_tokenize()` output, each sentence is a list of word tokens!

.sents() is a list of sentence tokens

```
>>> mlkcor.sents('1967-Beyond Vietnam.txt')
    [['Beyond', 'Vietnam', ':', 'A', 'Time', 'To', 'Break',
    'Silence'], ['Address', 'to', 'the', 'Clergy', 'and', 'Laity',
    'Concerned', 'about', 'Vietnam', ',', 'Riverside', 'Church',
    ',', 'New', 'York', 'City', 'April', '4', ',', '1967', 'by',
    'Dr', '.', 'Martin', 'Luther', 'King', 'Jr', '.'], ...]
>>> len(mlkcor.sents('1967-Beyond Vietnam.txt'))
    304
>>> for fname in mlkcor.fileids():
...     print(fname, 'has', len(mlkcor.sents(fname)), 'sentences.')
...
    1963-I Have a Dream.txt has 83 sentences.
    1964-Nobel Peace Prize Acceptance Speech.txt has 49 sentences.
    1967-Beyond Vietnam.txt has 304 sentences.
    1968-I've been to the Mountain Top.txt has 282 sentences.
```

Sentences in this one speech only

For-loop through file IDs, print out sentence count of each speech

.raw() is a raw text string

```
>>> mlkcor.raw()[:200]
'I Have A Dream\n\nby Dr. Martin Luther King Jr.\n\nDelivered on
the steps at the Lincoln Memorial\nin Washington D.C. on August
28, 1963\n\nI am happy to join with you today
down in history a'
>>> mlkcor.raw()[-200:]
"tonight, that we, as a people will get to the promised land.
And I'm happy, tonight. I'm not worried about anything. I'm not
fearing any man. Mine eyes have seen the glory of the coming of
the Lord.\n\n"
>>> len(mlkcor.raw())
77183
>>> mlkcor.raw('1967-Beyond Vietnam.txt')[:300]
'Beyond Vietnam:\nA Time To Break Silence\n\nAddress to
Clergy and Laity Concerned about Vietnam,\nRiverside Church, New
York City April 4, 1967\nby Dr. Martin Luther King Jr.\n\nMr.
Chairman, ladies and gentlemen, I need not pause to say how very
delighted I am to be here tonight, and how very delighte'
```

.raw() is not spill-proof, so we must be careful. Use slicing!

Character count of entire corpus

Text of a single speech

Try it out

15 minutes

- Download MLK.zip, then unzip the folder into your script directory
- Then, load the corpus through NLTK:

```
>>> from nltk.corpus import PlaintextCorpusReader
>>> corpus_root = "C:/Users/narae/Documents/ling1330/MLK"
>>> mlkcor = PlaintextCorpusReader(corpus_root, '.*txt')
>>> dir(mlkcor)
    ['CorpusView', '__class__', '__delattr__', '__dict__', ...
    'abspath', 'abspaths', 'citation', 'encoding', 'ensure_loaded',
    'fileids', 'license', 'open', 'paras', 'raw', 'readme', 'root',
    'sents', 'unicode_repr', 'words']
```

- Try:
  - `mlkcor.fileids()`
  - `mlkcor.words()` `mlkcor.words('1967-Beyond Vietnam.txt')`
  - `mlkcor.sents()` `mlkcor.sents('1967-Beyond Vietnam.txt')`
  - `mlkcor.raw()[:200]` `mlkcor.raw('1967-Beyond Vietnam.txt')[:n]`
  - `for x in mlkcor.fileids():`

MLK Corpus Practice Worksheet

1. How big is the MLK corpus? (aka word token count)
2. What is the word type count of the MLK corpus?
3. What is the most frequent content word in this corpus?
4. How many sentences does the "I Have a Dream" speech have?
5. How long is the longest sentence?
6. What is the longest speech? Shortest?
7. How does MLK end his speeches?

MLK Corpus Practice Worksheet

1. How big is the MLK corpus? (aka word token count)
2. What is the word type count of the MLK corpus?
3. What is the most frequent content word in this corpus?
4. How many sentences does the "I Have a Dream" speech have?
5. How long is the longest sentence?
6. What is the longest speech? Shortest?
7. How does MLK end his speeches?

Definition of a corpus

- In principle, any collection of more than one text can be called a corpus
  - Shakespeare corpus
- “Corpus” in modern linguistic theories has more specific connotations; it has four main characteristics:
  1. Sampling and representativeness
  2. Finite size (not always)
  3. Machine-readable form
  4. A standard reference
- "Corpus" in language engineering tends to de-emphasize 1, 2 and 4

The "first corpus"

The very first *modern* corpus: **Brown Corpus** (1967)

- The Brown University Standard Corpus of Present-Day American English
- 1 million words; Consists of 500 samples, distributed across 15 genres. Each sample contains about 2,000 words.
- 15 genres include: press (reportage, editorial, reviews), religion, skill and hobbies, popular lore, fiction (science, adventure, romance), learned, humor, etc.

So, what does a corpus really look like?

1. They can just look like a bunch of ordinary text files
   - Raw corpora

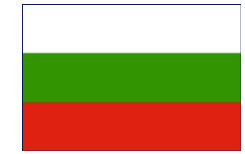
2. Or they can look a bit more complex, with more bits of information embedded…
   - Raw corpora with html/xml tags
   - Annotated corpora (part of speech, syntactic structures, etc.)

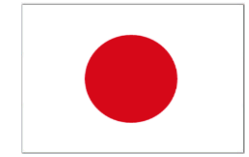
XML format, POS and lemma information

The BNC

```
        <w pos="ADJ" hw="scottish" c5="AJ0">Scottish </w>
        <w pos="SUBST" hw="city" c5="NN2">cities</w>
        <c c5="PUN">.</c>
    </s>
- <s n="136">
        <w pos="SUBST" hw="church" c5="NN2">Churches </w>
        <w pos="PREP" hw="in" c5="PRP">in </w>
        <w pos="ADJ" hw="these" c5="DT0">these </w>
        <w pos="SUBST" hw="area" c5="NN2">areas </w>
        <w pos="ADV" hw="particularly" c5="AV0">particularly </w>
        <w pos="VERB" hw="need" c5="VVB">need </w>
        <w pos="PREP" hw="to" c5="TO0">to </w>
        <w pos="VERB" hw="be" c5="VBI">be </w>
        <w pos="VERB" hw="inform" c5="VVN">informed</w>
        <c c5="PUN">, </c>
        <w pos="ADJ" hw="involved" c5="AJ0">involved </w>
        <w pos="PREP" hw="in" c5="PRP">in </w>
        <w pos="SUBST" hw="community" c5="NN1">community </w>
        <w pos="SUBST" hw="care" c5="NN1">care </w>
        <w pos="CONJ" hw="and" c5="CJC">and </w>
        <w pos="ADJ" hw="supporting" c5="AJ0-VVG">supporting </w>
        <w pos="ADJ" hw="christian" c5="AJ0">Christian </w>
        <w pos="SUBST" hw="worker" c5="NN2">workers </w>
        <w pos="VERB" hw="seek" c5="VVG">seeking </w>
        <w pos="PREP" hw="to" c5="TO0">to </w>
        <w pos="VERB" hw="prevent" c5="VVI">prevent </w>
        <w pos="ADJ" hw="new" c5="AJ0">new </w>
        <w pos="SUBST" hw="hiv" c5="NP0">HIV </w>
        <w pos="SUBST" hw="infection" c5="NN1">infection </w>
        <w pos="PREP" hw="in" c5="PRP">in </w>
        <w pos="SUBST" hw="school" c5="NN2">schools</w>
        <c c5="PUN">.</c>
        <c c5="PUQ">’</c>
    </s>
</p>
```

HW 3: Two EFL Corpora





Bulgarian Students

It is time, that our society is dominated by industrialization. The prosperity of a country is based on its enormous industrial corporations that are gradually replacing men with machines. Science is highly developed and controls the economy. From the beginning of school life students are expected to master a huge amount of scientific data. Technology is part of our everyday life.

Children nowadays prefer to play with computers rather than with our parents' wooden toys. But I think that in our modern world which worships science and technology there is still a place for dreams and imagination.

There has always been a place for them in man's life. Even in the darkness of the …

Japanese Students

I agree greatly this topic mainly because I think that English becomes an official language in the not too distant. Now, many people can speak English or study it all over the world, and so more people will be able to speak English. Before the Japanese fall behind other people, we should be able to speak English, therefore, we must study English not only junior high school students or over but also pupils. Japanese education system is changing such a program. In this way, Japan tries to internationalize rapidly. However, I think this way won't suffice for becoming international humans. To becoming international humans, we should study English not only school but also daily life. If we can do it, we are able to master English conversation. It is important for us to master English honorific words. …

Assessing writing quality

- Measurable indicators of writing quality

1. **Syntactic complexity**
   - Long, complex sentences vs. short. simple sentences

   ← Average sentence length, types of syntactic clauses used
2. **Lexical diversity**
   - Diverse vocabulary used vs. small set of words repeatedly used

   ← Type-token ratio (with caveat!) or other measures
3. **Vocabulary level**
   - Common, everyday words vs. sophisticated & technical words

   ← Average word length (common words tend to be shorter)

   ←% of word tokens in top 1K, 2K, 3K most common English words (Google Web 1T n-grams!)

Wrap-up

- Homework 3 is out
  - Larger at 60 points
  - WEEK-LONG, which means it's due next THU
  - Start NOW!

- Next class (Tue):
  - Corpus linguistics