Text Mining Project/Lab

Behrang QasemiZadeh
behrangatoffice@gmail.com

First thing first!

- Install Python and other third party software you want to use.
- Let’s play safe:
  - NLTK requires Python versions 2.6-2.7.

Installing NLTK

For the most recent instructions always look at http://www.nltk.org/install.html.

Install NLTK on Linux and Mac

- Install Setuptools: http://pypi.python.org/pypi/setuptools
  - `wget https://bootstrap.pypa.io/ez_setup.py -O - | sudo python`
  - `curl https://bootstrap.pypa.io/ez_setup.py -o - | python`
- Install Pip: run **`sudo easy_install pip`**
- Install Numpy (optional): run **`sudo pip install -U numpy`**
- Install NLTK: run **`sudo pip install -U nltk`**
- Test installation: run python then type **`import nltk`**

Install NLTK on Windows

- Currently works for Python 3.4.1 Win32 (avoid 64 bit)
- Install Numpy (optional): download http://sourceforge.net/projects/numpy/files/NumPy/1.8.1/numpy-1.8.1-win32-superpack-python3.4.exe
  - You need to download and install a series of dependencies
    - Read errors on the console
    - Look into Christoph Gohlke's http://www.lfd.uci.edu/~gohlke/pythonlibs/ for the required packages
- Download and Install NLTK from http://pypi.python.org/pypi/nltk
- Test installation: run python then type `import nltk`

NLTK's Resources

- CODE:
  - tokenizers
  - stemmers
  - taggers
  - parsers
  - …
- Data
  - Brown Corpus
  - Project Gutenberg Selections
  - Universal Declaration of Human Rights Corpus
  - Stopwords
  - WordNet
  - Names
  - …

Installing NLTK Data

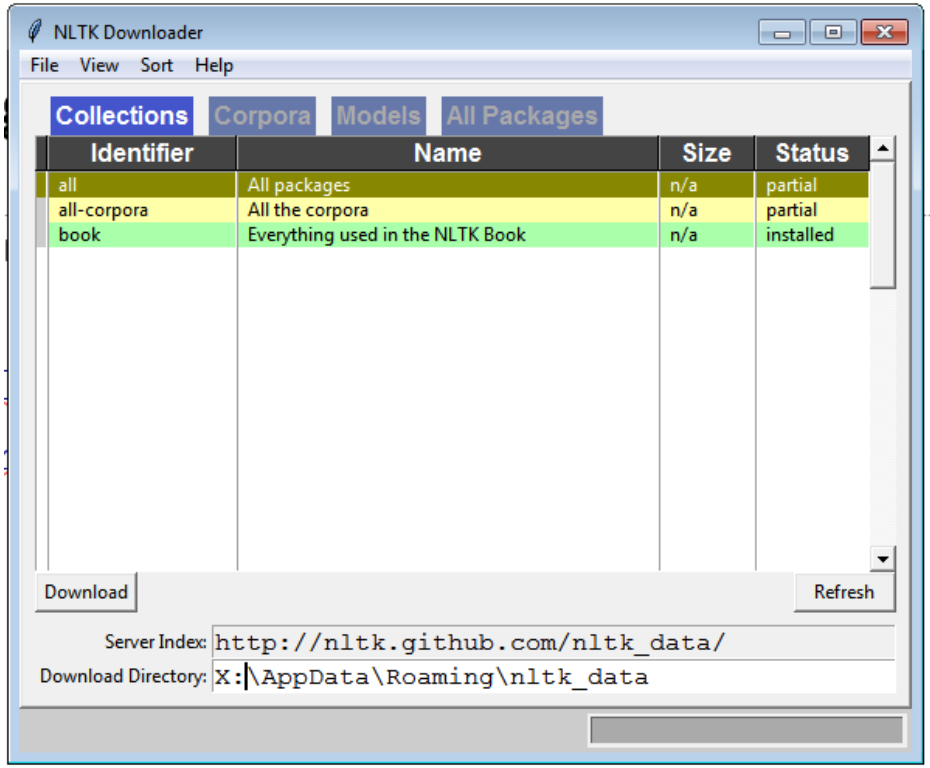
- Using the interactive tool run

```
import nltk
nltk.download()
```

Installing NLTK Data

- Using the interactive tool run

```
import nltk
nltk.download()
```



Installing NLTK Data

- Also, you can use the command line:

```
python -m nltk.downloader all
```

- Use the command –d to specify data location:

```
sudo python -m nltk.downloader -d /usr/share/nltk_data all
```

Installing NLTK Data: test your installation

```
>>> from nltk.corpus import brown
>>> brown.words()
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
>>>
```

Getting Started with NLTK

```
>>> from nltk.book import *
```

Getting Started with NLTK

```
>>> from nltk.book import *
```

from NLTK's book module, load all items.

Getting Started with NLTK

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book *** Loading
text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

Getting Started with NLTK

```
>>> from nltk.book import *
*** Introductory Examples for the NLTK Book *** Loading
text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Gr
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by
>>>
```

Follow the instructions to explore the book model.

Getting Started with NLTK

```
>>> texts()
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
>>>
```

Getting Started with NLTK

```
>>> sents()
sent1: Call me Ishmael .
sent2: The family of Dashwood had long been settled in Sussex .
sent3: In the beginning God created the heaven and the earth .
sent4: Fellow - Citizens of the Senate and of the House of Representatives :
sent5: I have a problem with people PMing me to lol JOIN
sent6: SCENE 1 : [ wind ] [ clop clop clop ] KING ARTHUR : Whoa there !
sent7: Pierre Vinken , 61 years old , will join the board as a nonexecutive director Nov. 29 .
sent8: 25 SEXY MALE , seeks attrac older single lady , for discreet encounters .
sent9: THE suburb of Saffron Park lay on the sunset side of London , as red and ragged as a cloud of sunset .
>>>
```

Getting Started with NLTK

```
>>> text9
<Text: The Man Who Was Thursday by G . K . Chesterton 1908>
>>>

>>> sent1
['Call', 'me', 'Ishmael', '.']
>>>
```

Searching Text – Concordance view

- Examine the context of a text using concordance view.
- **Concordance view** is a major tool for lexicography and building dictionaries.
- **Concordance view** shows us every occurrence of a given word, together with some context.

```
>>> text1.concordance(A WORD)
```

Searching Text – Concordance view

```
>>> text1.concordance("woman")
Displaying 10 of 10 matches:
nclude that , like the dyspeptic old woman , he must have " broken his digester
flections by the sight of a freckled woman with yellow hair and a yellow gown ,
e ' s rheumatic back . Never did any woman better deserve her name , which was
ir limbs . Nor can any son of mortal woman , for the first time , seat himself
" Mr . Har -- yes , Mr . Harry --( a woman ' s pinny hand ,-- the man ' s wife
in this same last or shoe , that old woman of the nursery tale , with the swarm
eginning at the end . It ' s the old woman ' s tricks to be giving cobbling job
men have for tinkers . I know an old woman of sixty - five who ran away with a
transparently pure and soft , with a woman ' s look , and the robust and man -
Starbuck die , if die he must , in a woman ' s fainting fit . Up helm , I say -
```

Searching Text – Concordance view

```
>>> text1.concordance("woman")
Displaying 10 of 10 matches:
nclude that , like the dyspeptic old woman , he must have " broken his digester
flections by the sight of a freckled woman with yellow hair and a yellow gown ,
e ' s rheumatic back . Never did any woman better deserve her name , which was
ir limbs . Nor can any son of mortal woman , for the first time , seat himself
" Mr . Har -- yes , Mr . Harry --( a woman ' s pinny hand ,-- the man ' s wife
in this same last or shoe , that old woman of the nursery tale , with the swarm
eginning at the end . It ' s the old woman ' s tricks to be giving cobbling job
men have for tinkers . I know an old woman of sixty - five who ran away with a
transparently pure and soft , with a woman ' s look , and the robust and man -
Starbuck die , if die he must , in a woman ' s fainting fit . Up helm , I say -
```

Searching Text – Concordance view

```
>>> text1.concordance("man")
Displaying 25 of 527 matches:
 Civitas ) which is but an artificial man ." -- OPENING SENTENCE OF HOBBES ' S
y of that sort that was killed by any man , such is his fierceness and swiftnes
in his deepest reveries -- stand that man on his legs , set his feet a - going
it ? The urbane activity with which a man receives money is really marvellous ,
, and that on no account can a monied man enter heaven . Ah ! how cheerfully we
ure truly , enough to drive a nervous man distracted . Yet was there a sort of
ss needle sojourning in the body of a man , travelled full forty feet , and at
 him ), bustles a little withered old man , who , for their money , dearly sell
ld put up with the half of any decent man ' s blanket . " I thought so . All ri
king as much noise as the rest . This man interested me at once ; and since the
 . I have seldom seen such brawn in a man . His face was deeply brown and burnt
ions had mounted to its height , this man slipped away unobserved , and I saw n
us to the entrance of the seamen . No man prefers to sleep two in a bed . In fa
 an uncomfortable feeling towards the man whom you design for my bedfellow -- a
lord , that harpooneer is a dangerous man ." " He pays reg ' lar ," was the rej
nd a papered fireboard representing a man striking a whale . Of things not prop
 me . I remembered a story of a white man -- a whaleman too -- who , falling am
ter all ! It ' s only his outside ; a man can be honest in any sort of skin . B
eard of a hot sun ' s tanning a white man into a purplish yellow one . However
 you sabbee me , I sabbee -- you this man sleepe you -- you sabbee ?" " Me sabb
ng about , thought I to myself -- the man ' s a human being just as I am : he h
th him . But I don ' t fancy having a man smoking in bed with me . It ' s dange
ter of my breeding . Nevertheless , a man like Queequeg you don ' t see every d
opriety that I ever heard of , is any man required to be private when putting o
e more ' s the pity . So , if any one man , in his own proper person , afford s
>>>
```

Searching Text – similar words

- We can find out by appending the term `similar` to the name of the text in question.

```
>>> text1.similar('man')
whale ship one it boat thing time all that ahab him sea captain way
whales moment matter head world as
>>>
```

Searching Text – common context

- Find context words that are shared by two or more words using the command **`common_contexts:`**
  - Note: input to this function is a list of words therefore we use brackets **`[]`**

```
>>> text1.common_contexts(['man', 'woman'])
old_he old_s old_of a_s
>>>
```

Searching text – other approaches

- You can explore the *location* of a word in the text.
- Use **`dispersion_plot`**

Searching text – other approaches

- You can explore the *location* of a word in the text.
- Use **`dispersion_plot`**

```
text1.dispersion_plot(['man', 'woman'])
```

- Note: **`dispersion_plot`** uses **`NumPy`** and **`Matplotlib`**

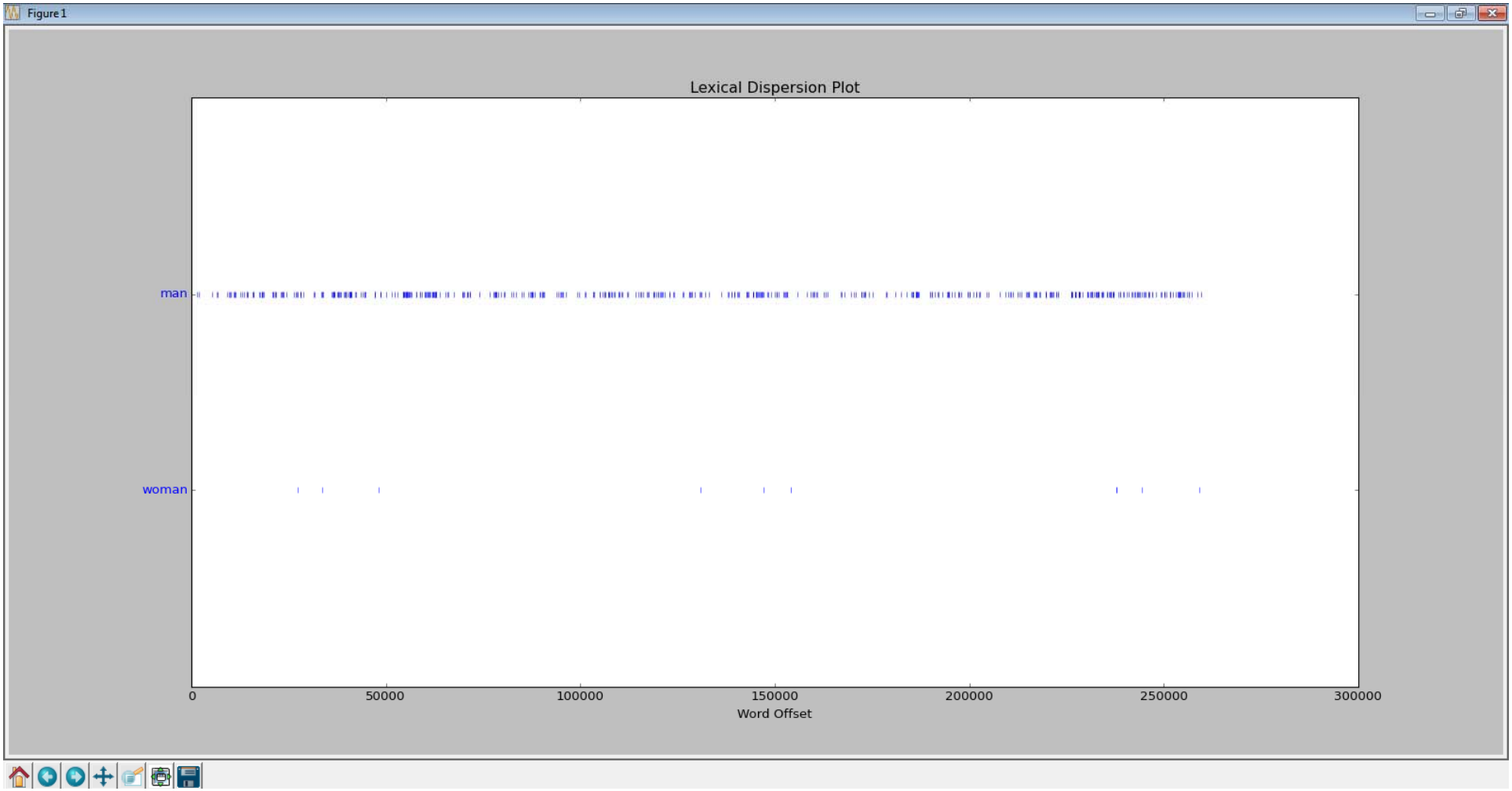


Figure 1
Lexical Dispersion Plot
man
woman
0
50000
100000
150000
200000
250000
300000
Word Offset

Counting Words

- Find the length of a text, in terms of the words and punctuation symbols that appear **(TOKENS).**

```
>>> len(text1)
260819
>>>
```

- Reminder: text1 is a list of tokens!

Token, Type and Vocabulary

- Tokens often (not necessary) show word boundaries.
- We can create a set from tokens. This implies that all duplicates are collapsed together.
- The elements of the obtained set are called “types”.
- The obtained set shows the vocabulary for this text.
- Vocabulary also called “lexicon”.
- Vocabulary ≈ Lexicon ≈ a set of tokens ≈ list of types

Counting Words

- To get number of types, we use:

```
>>> len(set(text1))
19317
>>>
```

Counting Words

- To get number of types, we use:

```
>>> len(set(text1))
19317
>>>
```

- Herman Melville used a vocabulary of size 19317 to write Moby Dick, which has 260819 tokens.

Let’s practice - Lexical Richness/Diversity

- In text5, Chat Corpus, how many times the word lol appear?
- How many times does the word *lol* appear in text5?
- How much is this as a percentage of the total number of words?

- Hint: to make sure Python uses floating-point division, use

```
>>> from __future__ import division
```

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))
```

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))
```

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))
```

Pay attention to auto-indent

Remember in Python we set boundaries using indentation

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
        return len(input_text) / len(set(input_text))
```

`input_text` is our input parameter. It is known as the *argument* for the function

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))
```

How many function do you see here?

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))

>>> lexical_diversity(text1)
```

We **call** a function such as lexical_diversity() by typing its name.

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))

>>> lexical_diversity(text1)
```

We **call** a function such as lexical_diversity() by typing its name.

Ask to do the computation for various text by passing them as the **arguments** of the function.

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))

>>> lexical_diversity(text1)
13.502044830977896
```

Lets write a Lexical Richness Function

- Arrange your calculation of lexical richness into a function **`lexical_diversity`**

```
>>> def lexical_diversity(input_text):
      return len(input_text) / len(set(input_text))

>>> lexical_diversity(text1)
13.502044830977896
>>> lexical_diversity(text5)
7.420046158918563
>>>
```

Frequency Distributions

- **Frequency distributions** tell us the frequency of each vocabulary, or in general any kind of pattern or event we can imagine.
  - A kind of tabular data, the first element is a symbol (vocabulary iterm/event) and the second element is a number.

Frequency Distributions

- **Frequency distributions** tell us the frequency of each vocabulary, or in general any kind of pattern or event we can imagine.
  - A kind of tabular data, the first element is a symbol (vocabulary iterm/event) and the second element is a number.
  - Use **`FreqDist`** to get the distribution of words in a text.

```
>>> fdist = FreqDist(text2)
```

Frequency Distributions

- **Frequency distributions** tell us the frequency of each vocabulary, or in general any kind of pattern or event we can imagine.
  - A kind of tabular data, the first element is a symbol (vocabulary iterm/event) and the second element is a number.
  - Use **`FreqDist`** to get the distribution of words in a text.

```
>>> fdist = FreqDist(text2)
```

What is the type of `fdist`?

Frequency Distributions

- **Frequency distributions** tell us the frequency of each vocabulary, or in general any kind of pattern or event we can imagine.
  - A kind of tabular data, the first element is a symbol (vocabulary iterm/event) and the second element is a number.
  - Use **`FreqDist`** to get the distribution of words in a text.

```
>>> fdist = FreqDist(text2)
>>> len(fdist)
6833
```

What is the output?
What does it say?

Frequency Distributions

- **Frequency distributions** tell us the frequency of each vocabulary, or in general any kind of pattern or event we can imagine.
  - A kind of tabular data, the first element is a symbol (vocabulary iterm/event) and the second element is a number.
  - Use **`FreqDist`** to get the distribution of words in a text.

```
>>> fdist = FreqDist(text2)
>>> len(fdist)
6833
>>> fdist['the']
3861
>>>
```

Assignment from the Last Session

Write a Python program that:

1. Reads a large text file (i.e. a corpus), e.g. a book or any kind of text; tokenize it using Python String built-in functions;
2. Make a dictionary of tokens and their frequencies;
3. And, write the dictionary into another text file (each line of the output file is a token followed by the frequency of that token in tab separated format).

Practice and Discussion (1)

Re-implement your code using NLTK

Practice and Discussion (2)

Alter the code so that the dictionary only contains words that:

1. **are longer than 5 characters;**
2. **and, start with the letter ‘b’.**

Bigrams, Trigrams, … and Collocations

- **Bi-grams**:
  a sequence of 2 words.
- **Tri-grams**:
  a sequence of 3 words.
- **…**

Bigrams, Trigrams, … and Collocations

- **Bi-grams**:
  a sequence of 2 words.
- **Tri-grams**:
  a sequence of 3 words.
- **…**
- **Collocations:**
  a sequence of words that occur together more than often.
- Thus, not all the bi/tri/4/…-grams are collocations.

Generate Bigrams using NLTK

- Generate bigram using **`bigrams`**:

```
>>> for pair in nltk.bigrams(['more', 'is', 'said', 'than', 'done']):
        print(pair)

('more', 'is')
('is', 'said')
('said', 'than')
('than', 'done')
>>>
```

Generate Bigrams using NLTK

- Generate bigram using **`bigrams`**:

```
>>> for pair in nltk.bigrams(['more', 'is', 'said', 'than', 'done']):
        print(pair)

('more', 'is')
('is', 'said')
('said', 'than')
('than', 'done')
>>>
```

Practice

- Extract bigrams from text1 (Moby Dick!).
- Extract collocations:
  - How to find them?
  - Let’s start with frequent bigrams!

Generate Collocations Using NLTK

- In NLTK, you can use **`collocations`**.

```
>>> text1.collocations(5)
Sperm Whale; Moby Dick; White Whale; old man;
Captain Ahab
>>>
```

Summary: what do we expect to know?

- In Python, a text is represented using lists: ['Monty', 'Python'] and we can use:
  - Indexing, slicing, and the len() function on lists;
  - We operate on each item of a text using **[f(x) for x in text]**.

Summary: what do we expect to know?

- Remember: a "**token**" is a particular appearance of a given word in a text, i.e., the number of words in text is **len(text)**.
- Remember "**type**" is the unique form of the word as a particular sequence of letters, i.e., **len(set(text)) is the vocabulary size!**

Summary: what do we expect to know?

- Remember: a "**token**" is a particular appearance of a given word in a text, i.e., the number of words in text is **len(text)**.
- Remember "**type**" is the unique form of the word as a particular sequence of letters, i.e., **len(set(text)) is the vocabulary size!**
- A **frequency distribution** is a collection of items along with their frequency counts (e.g., the words of a text and their frequency).

NLP and Language Resources

- In Natural Language Processing, we often uses language resources.
- Language resource:
  - Large bodies of linguistic data, i.e. **corpora**;
  - List of words (and their annotation), i.e. **Lexicons/Vocabularies**.
  - **Ontologies and thesaurus**, e.g. **WordNet**
    - Words are grouped by and related to each other in a kind of structure.
  - And so on…

NLP and Language Resources

- Where to find language resources?
- NLTK comes with a set of language resources:
  - Project Gutenberg electronic text archive
  - Brown corpus
  - WordNet
  - …
- Language Resources are often developed with a specific application in mind, e.g. Brown Corpus vs. Project Gutenburg

Accessing Corpora in NLTK

- Make sure you have downloaded language resources
- Let’s look into “How to’s” of NLTK!