Joshua Kyle K. Entrata 4CSC

Analyzing a Sentence

Lets start small. What can we learn about a sentence?

In [33]:
```
import nltk
# nltk.download('all')
# nltk.download('wordnet')
# nltk.download('stopwords')
```

In [34]:
```
#
# Preamble
#

# Import our core libraries
import re

# Read in the documents we will use
from nltk.corpus import reuters
coffee = reuters.open('test/19570').read()
gold = reuters.open('test/16589').read()

text = gold
```

Segmentation and Tokenization

The first step for us to do is to actually get a sentence from the document we loaded. The general process of extracting *meaningful units* of text from a larger text is called **segmentation**.

When applied to finding smaller word-like units from text, the process is often referred to as **tokenization** (and you may find these terms used interchangeably).

A *meaningful unit* is anything that is useful for your application, these can be *sections* of a document, *paragraphs*, *words* or in our case *sentences*.

Sentence Segmentation

In [35]:
```
# Simple sentence segmentation

#
# Using string.split
#
sentences = text.split('.')
sentences[0:10]
```

```
# Q1: Are there any issues with this approach?
```

```
Out[35]: ["HOMESTAKE &lt;HM> MULLS BUYING ORE RESERVES\n  Homestake Mining Co is considerin
g\n  acquiring more gold ore reserves in addition to the company's\n  exploration
efforts, chief executive Harry Conger told Reuters\n  in an interview",
 '\n      "We are looking at more options to acquire more reserves\n  rather than
just exploration," Conger said adding, "the move to\n  consider acquisitions repre
sents a change in the company\'s\n  acquisitions policy',
 '"\n      Conger said all of Homestake\'s current cash position of 120\n  mln dlr
s would be available to acquire reserves',
 ' In addition,\n  Homestake has two lines of credit totaling 150 mln dlrs which\n
have not been drawn on today and could be used to finance an\n  acquisition, he sa
id',
 '\n      Conger said he anticipates 1987 exploration budget will be\n  about the
same as 1986 spending of 27',
 '3 mln dlrs',
 " Conger said\n  exploration for precious metals may be slightly higher than\n  l
ast year's spending of 17",
 "7 mln dlrs while oil and gas\n  exploration spending will be slightly less than
last year's 9",
 '6\n  pct',
 "\n      Conger said he sees Homestake's 1987 gold production about\n  the same a
s 1986 gold production of 669,594 ounces"]
```

```
In [36]: #
         # Using re.split
         #
         sentences = re.split(r'[\.\?!]', text)
         sentences[0:10]

         # Q2: How about now? What other corner cases and caveats might
         #    we want to stay aware of?
```

```
Out[36]: ["HOMESTAKE &lt;HM> MULLS BUYING ORE RESERVES\n  Homestake Mining Co is considerin
g\n  acquiring more gold ore reserves in addition to the company's\n  exploration
efforts, chief executive Harry Conger told Reuters\n  in an interview",
 '\n      "We are looking at more options to acquire more reserves\n  rather than
just exploration," Conger said adding, "the move to\n  consider acquisitions repre
sents a change in the company\'s\n  acquisitions policy',
 '"\n      Conger said all of Homestake\'s current cash position of 120\n  mln dlr
s would be available to acquire reserves',
 ' In addition,\n  Homestake has two lines of credit totaling 150 mln dlrs which\n
have not been drawn on today and could be used to finance an\n  acquisition, he sa
id',
 '\n      Conger said he anticipates 1987 exploration budget will be\n  about the
same as 1986 spending of 27',
 '3 mln dlrs',
 " Conger said\n  exploration for precious metals may be slightly higher than\n  l
ast year's spending of 17",
 "7 mln dlrs while oil and gas\n  exploration spending will be slightly less than
last year's 9",
 '6\n  pct',
 "\n      Conger said he sees Homestake's 1987 gold production about\n  the same a
s 1986 gold production of 669,594 ounces"]
```

```
In [37]: #
         # Using NLTK.
         #

         sentences = nltk.sent_tokenize(text)
         sentences[0:10]

         # Notice that this retains the punctuation in the sentences.
```

```
Out[37]: ["HOMESTAKE &lt;HM> MULLS BUYING ORE RESERVES\n  Homestake Mining Co is considerin
         g\n  acquiring more gold ore reserves in addition to the company's\n  exploration
         efforts, chief executive Harry Conger told Reuters\n  in an interview.",
          '"We are looking at more options to acquire more reserves\n  rather than just exp
         loration," Conger said adding, "the move to\n  consider acquisitions represents a
         change in the company\'s\n  acquisitions policy."',
          "Conger said all of Homestake's current cash position of 120\n  mln dlrs would be
         available to acquire reserves.",
          'In addition,\n  Homestake has two lines of credit totaling 150 mln dlrs which\n
         have not been drawn on today and could be used to finance an\n  acquisition, he sa
         id.',
          'Conger said he anticipates 1987 exploration budget will be\n  about the same as
         1986 spending of 27.3 mln dlrs.',
          "Conger said\n  exploration for precious metals may be slightly higher than\n  la
         st year's spending of 17.7 mln dlrs while oil and gas\n  exploration spending will
         be slightly less than last year's 9.6\n  pct.",
          "Conger said he sees Homestake's 1987 gold production about\n  the same as 1986 g
         old production of 669,594 ounces.",
          "However, 1987 first quarter production from its McLaughlin\n  reserve will be ab
         out 10 pct lower than last year's 45,400\n  ounces due to start-up production prob
         lems.",
          'He said he believes gold prices will hold above the 400\n  U.S. dlr an ounce lev
         el for the rest of 1987.',
          'IN 1986, company earnings were based an average market\n  price for gold of 368
         dlrs an ounce.']
```

```
In [38]: # NLTK will also handle cases like Mr. Jones correctly.
         with_salutations = """The quick brown fox jumped over Mr. Jones.
                               Much to the disapproval of the dogs."""
         nltk.sent_tokenize(with_salutations)
```

```
Out[38]: ['The quick brown fox jumped over Mr. Jones.',
          'Much to the disapproval of the dogs.']
```

Word Tokenization

We can apply the same approach to finding word-like units of text. All the above methods could be used but similar caveats will apply. So lets jump straight into using nltk for the tokenization.

```
In [39]: # We first need to tokenize our sentence into smaller units, in our case words.
         with_salutations = """The quick brown fox jumped over Mr. Jones.
                               Much to the disapproval of the dogs."""
         words = nltk.word_tokenize(with_salutations)
```

```python
# Lets print out all the things in this list and the lengths of the words
# Notice that python is whitespace sensitive with indentation
# indicating block structure.
for word in words:
    word_len = len(word)
    print(word + ' : ' + str(word_len))
```

```
The : 3
quick : 5
brown : 5
fox : 3
jumped : 6
over : 4
Mr. : 3
Jones : 5
. : 1
Much : 4
to : 2
the : 3
disapproval : 11
of : 2
the : 3
dogs : 4
. : 1
```

Your Turn!

Plot the lengths of all the sentences in the **gold** article.

In [40]:
```python
# Exercise 1
# Plot the lengths of all the sentences in the __gold__ article.
# First print them and then see if you can make an ascii bar chart of them.
# It could look something like this...
#
# ****
# ******************
# ***********
#

gold = reuters.open('test/16589').read()
sentences = nltk.sent_tokenize(gold)

# Step one. Find all the lengths and print them to the console/notebook.
sentence_lengths = [len(sentence) for sentence in sentences]
print("Sentence lengths:", sentence_lengths)

# Step two (extra credit). Make an horizontal ascii bar graph of the numbers
print("\nASCII Bar Graph of Sentence Lengths:")
for length in sentence_lengths:
    print('\n' + '*' * length)
```

```
Sentence lengths: [228, 207, 110, 163, 110, 204, 112, 165, 103, 93, 230]

ASCII Bar Graph of Sentence Lengths:

************************************************************************************
************************************************************************************
************************************************************

************************************************************************************
************************************************************************************
***************************************

************************************************************************************
**************************

************************************************************************************
*******************************************************************************

************************************************************************************
**************************

************************************************************************************
************************************************************************************
************************************

************************************************************************************
****************************

************************************************************************************
*********************************************************************************

************************************************************************************
*******************

************************************************************************************
*********

************************************************************************************
************************************************************************************
**************************************************************
```

Parts of Speech

What else can we find out about this sentence. We can look for specific patterns we have already identified, but we can also try and deduce what type of this word this is, e.g. is it a noun or a verb.

This activity is known as **Part of Speech Tagging** *(POS Tagging)*, and a number of algorithms have been developed to do this, some are statistical and others are rule based.

Lets look at how we can do this in nltk.

In [41]:
```
# Lets look at the first sentence of the article
sentence = sentences[0]
print(sentence)
```

```
HOMESTAKE &lt;HM> MULLS BUYING ORE RESERVES
  Homestake Mining Co is considering
  acquiring more gold ore reserves in addition to the company's
  exploration efforts, chief executive Harry Conger told Reuters
  in an interview.
```

In [42]:
```
# We first need to tokenize our sentence into smaller units, in our case words.
words = nltk.word_tokenize(sentence)
words
```

Out[42]:
```
['HOMESTAKE',
 '&',
 'lt',
 ';',
 'HM',
 '>',
 'MULLS',
 'BUYING',
 'ORE',
 'RESERVES',
 'Homestake',
 'Mining',
 'Co',
 'is',
 'considering',
 'acquiring',
 'more',
 'gold',
 'ore',
 'reserves',
 'in',
 'addition',
 'to',
 'the',
 'company',
 "'s",
 'exploration',
 'efforts',
 ',',
 'chief',
 'executive',
 'Harry',
 'Conger',
 'told',
 'Reuters',
 'in',
 'an',
 'interview',
 '.']
```

In [43]:
```
# Note that the result includes punctuation as tokens.
```

```
In [44]: # Now that we have the words we can use nltk's POS tagger

         tagged = nltk.pos_tag(words)
         tagged
```

```
Out[44]: [('HOMESTAKE', 'NNP'),
          ('&', 'CC'),
          ('lt', 'NN'),
          (';', ':'),
          ('HM', 'NNP'),
          ('>', 'NNP'),
          ('MULLS', 'NNP'),
          ('BUYING', 'NNP'),
          ('ORE', 'NNP'),
          ('RESERVES', 'NNP'),
          ('Homestake', 'NNP'),
          ('Mining', 'NNP'),
          ('Co', 'NNP'),
          ('is', 'VBZ'),
          ('considering', 'VBG'),
          ('acquiring', 'VBG'),
          ('more', 'JJR'),
          ('gold', 'NN'),
          ('ore', 'NN'),
          ('reserves', 'NNS'),
          ('in', 'IN'),
          ('addition', 'NN'),
          ('to', 'TO'),
          ('the', 'DT'),
          ('company', 'NN'),
          ("'s", 'POS'),
          ('exploration', 'NN'),
          ('efforts', 'NNS'),
          (',', ','),
          ('chief', 'JJ'),
          ('executive', 'NN'),
          ('Harry', 'NNP'),
          ('Conger', 'NNP'),
          ('told', 'VBD'),
          ('Reuters', 'NNP'),
          ('in', 'IN'),
          ('an', 'DT'),
          ('interview', 'NN'),
          ('.', '.')]
```

These tags like NN and NNP come from a standardized set of tags known as the **Penn Treebank POS Tags**

You can see the full list here https://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html

```
In [45]: # Lets extract all the nouns from all the sentences in the document

         # Will return all the words in the words input param that match a given tag.
         def get_tagged(words, tag):
```

```python
    tagged = nltk.pos_tag(words)

    matches = []
    for tup in tagged:
        if tup[1] == tag:
            matches.append(tup)
    return matches

# This is the same function as above, but uses a List Comprehension
def get_tagged(words, tag):
    tagged = nltk.pos_tag(words)
    return [tup for tup in tagged if tup[1] == tag]

# Will word tokenize a list of sentence and return those tokens
def tokenize(sentences):
    return [nltk.word_tokenize(sentence) for sentence in sentences]
```

In [46]:
```python
tokenized = tokenize(sentences)
```

In [47]:
```python
nnps = [get_tagged(tokens, 'NNP') for tokens in tokenized]
nnps
```

Out[47]:
```
[[('HOMESTAKE', 'NNP'),
  ('HM', 'NNP'),
  ('>', 'NNP'),
  ('MULLS', 'NNP'),
  ('BUYING', 'NNP'),
  ('ORE', 'NNP'),
  ('RESERVES', 'NNP'),
  ('Homestake', 'NNP'),
  ('Mining', 'NNP'),
  ('Co', 'NNP'),
  ('Harry', 'NNP'),
  ('Conger', 'NNP'),
  ('Reuters', 'NNP')],
 [('Conger', 'NNP')],
 [('Conger', 'NNP'), ('Homestake', 'NNP')],
 [('Homestake', 'NNP')],
 [('Conger', 'NNP')],
 [('Conger', 'NNP')],
 [('Conger', 'NNP'), ('Homestake', 'NNP')],
 [('McLaughlin', 'NNP')],
 [('U.S.', 'NNP')],
 [],
 [('Conger', 'NNP')]]
```

As you can see. POS Tagging takes a little bit of time. It can also yield imperfect results. However it can be an interesting approach to finding out more about a text.

It is also useful to keep in mind that there are actually a number of different algorithms and models that can be used for POS tagging.

See http://www.nltk.org/book/ch05.html and http://www.nltk.org/api/nltk.tag.html for a reference to what is in NLTK.

Your Turn!

Plot the number of verbs (VB, VBD, VBG, VBN, VBP) across all sentences in the gold article. Which sentence has the most number of verbs.

In [48]:
```python
# Exercise 2
# Count the number of verbs (VB) in each sentence of the gold article
# You may want to make some helper functions.

# Helper function to count verbs in a POS-tagged sentence
def count_verbs(tagged_sentence):
    return sum(1 for _, tag in tagged_sentence if tag.startswith('VB'))

gold = reuters.open('test/16589').read()


# Step 1. Tokenize the article to sentences
sentences = nltk.sent_tokenize(gold)

# Step 2. For each sentence tokenize to words and store that list
verb_counts = []
for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    tagged_words = nltk.pos_tag(words)
    verb_count = count_verbs(tagged_words)
    verb_counts.append(verb_count)

# Step 3. POS tag all the tokens in each sentence and filter out non verb tokens.
for i, count in enumerate(verb_counts):
    print(f"Sentence {i + 1}: {count} verbs")

# Step 4. Print out the counts for each sentence. Which sentence has the most verbs
max_verbs = max(verb_counts)
max_index = verb_counts.index(max_verbs)
print(f"\nSentence {max_index + 1} has the most verbs with {max_verbs} verbs.")
```

```
Sentence 1: 4 verbs
Sentence 2: 7 verbs
Sentence 3: 3 verbs
Sentence 4: 9 verbs
Sentence 5: 3 verbs
Sentence 6: 3 verbs
Sentence 7: 2 verbs
Sentence 8: 1 verbs
Sentence 9: 4 verbs
Sentence 10: 2 verbs
Sentence 11: 6 verbs

Sentence 4 has the most verbs with 9 verbs.
```

Feature Selection

One way we can think of what we have been just been doing is finding '*things*' that give some information about our sentences. Features cound be anything countable, whether it is is the amount of money mentioned in a sentence, or the number of characters. A big part of text analysis, particularly the statistical and pattern based approaches we will be looking at is feature selection and extraction.

Q3: What other features could you think to extract from a sentence?

Examples

We can also see how even simple seeming features can be used to good effect in building visualizations.

Listening Post by Mark Hansen and Ben Rubin is an installation that displays sentence gathered from the internet that contain phrases like "I am" or "I love."

We Feel Fine by Jonathan Harris and Sep Kamvar begins by searching for blog posts that containt phrases like "I am feeling", "I feel" ... [Regex]

Stereotropes by Bocoup uses POS tagging to extract adjectives from text to then visualize descriptions of characters in film. [POS Tagging]

```
In [ ]:
```