Week 8

The Natural Language Toolkit (NLTK)

List methods

- Getting information about a list
  - list.index(item)
  - list.count(item)
- These **modify the list in-place**, unlike str operations
  - list.append(item)
  - list.insert(index, item)
  - list.remove(item)
  - list.extend(list2)
    - same as list += list2
  - list.sort()
  - list.reverse()

List exercise

- Write a script to print the most frequent token in a text file.

And now for something completely different

Programming tasks?

- So far, we've studied programming syntax and techniques
- What about tasks for programming?
  - Homework
  - Mathematics, statistics (Sage)
  - Biology (Biopython)
  - Animation (Blender)
  - Website development (Django)
  - Game development (PyGame)
  - **Natural language processing** **(NLTK)**

Natural Language Processing (NLP)

- How can we make a computer understand language?
  - Can a human write/talk to the computer?
    - Or can the computer guess/predict the input?
  - Can the computer talk back?
  - Based on language rules, patterns, or statistics
    - For now, statistics are more accurate and popular

Some areas of NLP

- shallow processing – the surface level
  - **tokenization**
  - **part-of-speech tagging**
  - forms of words
- deep processing – the underlying structures of language
  - **word order (syntax)**
  - meaning
  - translation
- **natural language generation**

The NLTK

- A collection of:
  - Python functions and objects for accomplishing NLP tasks
  - sample texts (corpora)
- Available at: http://nltk.sourceforge.net
  - Requires Python 2.4 or higher
  - Click 'Download' and follow instructions for your OS

Tokenization

- Say we want to know the words in Marty's vocabulary
  - *"You know what I hate? Anybody who drives an S.U.V. I'd really like to find Mr. It-Costs-Me-100-Dollars-To-Gas-Up and kick him square in the teeth. Booyah. Be like, I'm Marty Stepp, the best ever. Booyah!"*
- *How do we split his speech into tokens?*

Tokenization (cont.)

- How do we split his speech into tokens?

```
>>> martysSpeech.split()
['You', 'know', 'what', 'I', 'hate?', 'Anybody',
'who', 'drives', 'an', 'S.U.V.', "I'd", 'really',
'like', 'to', 'find', 'Mr.', 'It-Costs-Me-100-
Dollars-To-Gas-Up', 'and', 'kick', 'him', 'square',
'in', 'the', 'teeth.', 'Booyah.', 'Be', 'like,',
"I'm", 'Marty', 'Stepp,', 'the', 'best', 'ever.',
'Booyah!']
```

- Now, how often does he use the word "booyah"?

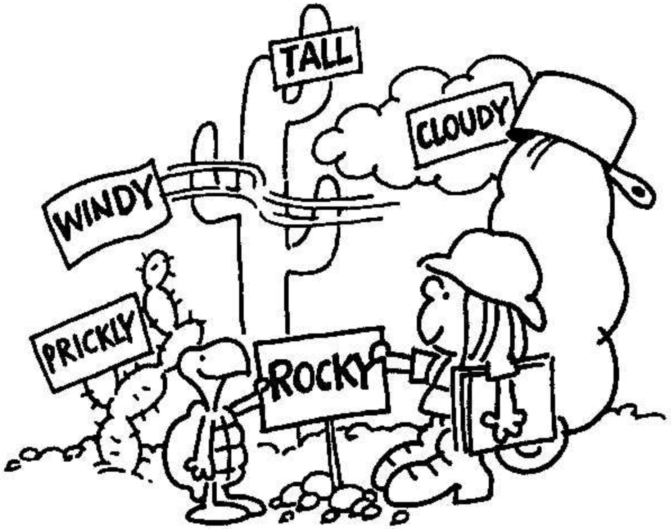
```
>>> martysSpeech.split().count("booyah")
0
>>> # What the!
```

Tokenization (cont.)

- We could lowercase the speech
- We could write our own method to split on "." split on ",", split on "-", etc.
- The NLTK already has several tokenizer options
- Try:
    - nltk.tokenize.WordPunctTokenizer
        - tokenizes on all punctuation
    - nltk.tokenize.**Punkt**WordTokenizer
        - **trained** algorithm to statistically split on words

Part-of-speech (POS) tagging

- If you know a token's POS you know:
  - is it the subject?
  - is it the verb?
  - is it introducing a grammatical structure?
  - is it a proper name?



Part-of-speech (POS) tagging

- Exercise: most frequent proper noun in the Penn Treebank?
  - Try:
    - nltk.corpus.treebank
    - Python's **dir()** to list attributes of an object
      - Example:

```
>>> dir("hello world!")
[..., 'capitalize', 'center', 'count',
'decode', 'encode', 'endswith', 'expandtabs',
'find', 'index', 'isalnum', 'isalpha',
'isdigit', 'islower', 'isspace', 'istitle',
'isupper', 'join', 'ljust', 'lower', ...]
```

Tuples

- tagged_words() gives us a list of **tuples**
  - **tuple**: the same thing as a list, but you can't change it
  - in this case, the tuples are a (word, tag) pairs

```
>>> # Get the (word, tag) pair at list index 0
...
>>> pair = nltk.corpus.treebank.tagged_words()[0]
>>> pair
('Pierre', 'NNP')
>>> word = pair[0]
>>> tag = pair[1]
>>> print word, tag
Pierre NNP
>>> word, tag = pair           # or unpack in 1 line!
>>> print word, tag
Pierre NNP
```

POS tagging (cont.)

- How do we tag plain sentences?
  - A NLTK tagger needs a list of tagged sentences to train on
    - We'll use nltk.corpus.treebank.tagged_sents()
  - Then it is ready to tag any input! (but how well?)
  - Try these tagger objects:
    - nltk.UnigramTagger(tagged_sentences)
    - nltk.TrigramTagger(tagged_sentences)
  - Call the tagger's tag(tokens) method

```
>>> tagger = nltk.UnigramTagger(tagged_sentences)
>>> result = tagger.tag(tokens)
>>> result
[('You', 'PRP'), ('know', 'VB'), ('what', 'WP'),
('I', 'PRP'), ('hate', None), ('?', '.'), ...]
```

POS tagging (cont.)

- Exercise: Mad Libs
  - I have a passage I want filled with the right parts of speech
  - Let's use random picks from our own data!
  - This code will print it out:

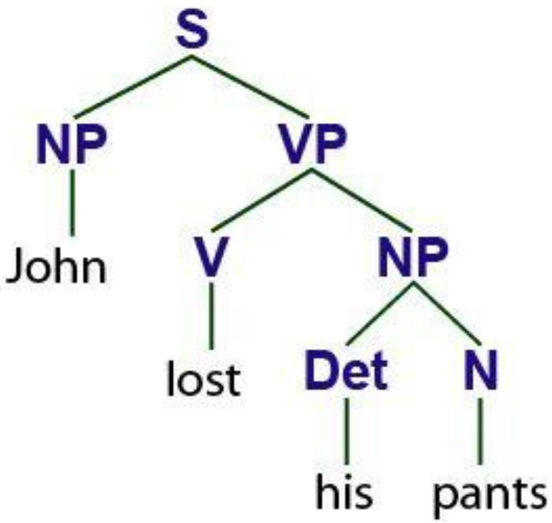
```
print properNoun1, "has always been a", adjective1, \
    singularNoun, "unlike the", adjective2, \
    properNoun2, "who I", pastVerb, "as he was", \
    ingVerb, "yesterday."
```

Eliza (NLG)

- Eliza simulates a Rogerian psychotherapist
- With while loops and tokenization, you can make a chat bot!
  - Try:
    - nltk.chat.eliza.eliza_chat()

Parsing

- Syntax is as important for a compiler as it is for natural language
- Realizing the hidden structure of a sentence is useful for:
  - translation
  - meaning analysis
  - relationship analysis
  - a cool demo!
    - Try:
      - nltk.draw.rdparser.demo()



Conclusion

- NLTK: NLP made easy with Python
  - Functions and objects for:
    - tokenization, tagging, generation, parsing, ...
    - and much more!
  - Even armed with these tools, NLP has a lot of difficult problems!
- Also saw:
  - List methods
  - dir()
  - Tuples

python™