Kan.Scipy #1

Welcome to Kan.Scipy #1!

Introducing NLTK

First, some imports:

In [19]:
```
import nltk
import MeCab
from nltk import Tree
from nltk.corpus import brown, gutenberg, treebank
from nltk.tokenize.api import TokenizerI
```

Corpora

NLTK has several built-in corpora and resources

In [6]:
```
treebank.sents()
```

```
---------------------------------------------------------------------------
LookupError                               Traceback (most recent call last)
/home/joseph/Projects/kanscipy/<ipython-input-6-508d6b7c8d5f> in <module>()
----> 1 treebank.sents()

/usr/lib/python2.7/dist-packages/nltk/corpus/util.pyc in __getattr__(self, attr)
     66
     67     def __getattr__(self, attr):
---> 68         self.__load()
     69         # This looks circular, but its not, since __load() changes our

     70         # __class__ to something new:


/usr/lib/python2.7/dist-packages/nltk/corpus/util.pyc in __load(self)
     54             except LookupError, e:
     55                 try: root = nltk.data.find('corpora/%s' % zip_name)
---> 56                 except LookupError: raise e
     57
     58         # Load the corpus.


LookupError:
**********************************************************************
  Resource 'corpora/treebank/combined' not found.  Please use the
  NLTK Downloader to obtain the resource: >>> nltk.download().
  Searched in:
    - '/home/joseph/nltk_data'
    - '/usr/share/nltk_data'
    - '/usr/local/share/nltk_data'
    - '/usr/lib/nltk_data'
    - '/usr/local/lib/nltk_data'
**********************************************************************
```

In [7]:
```
nltk.download('treebank')
```

```
[nltk_data] Downloading package 'treebank' to
[nltk_data]     /home/joseph/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
```

Out[7]: True

In [10]:
```
print treebank.parsed_sents()[0]
```

```
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

NLTK's CorpusReader classes manage files:

In [31]:
```
print brown.abspaths()[:5]
```

```
[FileSystemPathPointer('/home/joseph/nltk_data/corpora/brown/ca01'),
FileSystemPathPointer('/home/joseph/nltk_data/corpora/brown/ca02'),
FileSystemPathPointer('/home/joseph/nltk_data/corpora/brown/ca03'),
FileSystemPathPointer('/home/joseph/nltk_data/corpora/brown/ca04'),
FileSystemPathPointer('/home/joseph/nltk_data/corpora/brown/ca05')]
```

Tokenization

NLTK has a built-in tokenizer for English:

In [11]:
```
text = 'The quick brown fox jumped over the lazy dog.'
nltk.word_tokenize(text)
```

Out[11]: ['The', 'quick', 'brown', 'fox', 'jumped', 'over', 'the', 'lazy', 'dog', '.']

For Japanese:
You can call MeCab from Python:

In [17]:
```
jtext = u'すばしっこい茶色の狐が怠け者の犬の上を飛んでいったとさ。'
mecab = MeCab.Tagger()
print mecab.parse(jtext.encode('euc-jp')).decode('euc-jp')
```

```
す	接頭辞,名詞接頭辞,*,*,す,す,*
ば	名詞,普通名詞,*,*,ば,ば,漢字読み:訓 代表表記:場
しっ	動詞,*,サ変動詞,語幹,しっする,しっ,代表表記:失する
こい	形容詞,*,イ形容詞アウオ段,基本形,こい,こい,代表表記:濃い
茶色	名詞,普通名詞,*,*,茶色,ちゃいろ,代表表記:茶色
の	助詞,接続助詞,*,*,の,の,*
狐	名詞,普通名詞,*,*,狐,きつね,代表表記:狐
が	助詞,格助詞,*,*,が,が,*
怠け者	名詞,普通名詞,*,*,怠け者,なまけもの,代表表記:怠け者
の	助詞,接続助詞,*,*,の,の,*
犬	名詞,普通名詞,*,*,犬,いぬ,漢字読み:訓 代表表記:犬
```

```
の	助詞,接続助詞,*,*,の,の,*
上	名詞,副詞的名詞,*,*,上,うえ,*
を	助詞,格助詞,*,*,を,を,*
飛んで	動詞,*,子音動詞バ行,タ系連用テ形,飛ぶ,とんで,代表表記:飛ぶ
いった	接尾辞,動詞性接尾辞,子音動詞カ行促音便形,タ形,いく,いった,*
と	助詞,格助詞,*,*,と,と,*
さ	助詞,終助詞,*,*,さ,さ,*
。	特殊,句点,*,*,。,。,*
EOS
```

Or define a new NLTK tokenizer using MeCab: (code copied from `https://mhagiwara.googlecode.com/svn/trunk/nltk/jpbook/jptokenizer.py`)

In [24]:
```
class JPMeCabTokenizer(TokenizerI):
    def __init__(self):
        import MeCab
        self.mecab = MeCab.Tagger('-Owakati')

    def tokenize(self, text):
        result = self.mecab.parse(text.encode('euc-jp'))
        return result.decode('euc-jp').strip().split(' ')

print JPMeCabTokenizer().tokenize(jtext)
print u' '.join(JPMeCabTokenizer().tokenize(jtext))
```

```
[u'\u3059', u'\u3070', u'\u3057\u3063', u'\u3053\u3044', u'\u8336\u8272', u'\u306e',
u'\u72d0', u'\u304c', u'\u6020\u3051\u8005', u'\u306e', u'\u72ac', u'\u306e',
u'\u4e0a', u'\u3092', u'\u98db\u3093\u3067', u'\u3044\u3063\u305f', u'\u3068',
u'\u3055', u'\u3002']
す ば しっ こい 茶色 の 狐 が 怠け者 の 犬 の 上 を 飛んで いった と さ 。
```

Ngram Language Models

NLTK provides functionality to build n-gram language models.

A language model is a probabilistic model of language that allows us to measure how likely a given sequence of words is.

An n-gram is a sequence of n words; we count n-grams in a text and calculate a conditional probability distribution like:

$$P(X_i|X_{i-1},..,X_{i-n+1})$$

In [37]:
```
from nltk.model.ngram import NgramModel
from nltk.probability import WittenBellProbDist, LidstoneProbDist

train_words = brown.words()[:-500]
test_words = brown.words()[-500:]
lm = NgramModel(2, train_words, lambda fd, b: LidstoneProbDist(fd, 0.2))
```

In [38]:
```
lm.entropy(test_words)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
/home/joseph/Projects/kanscipy/<ipython-input-38-a41416bdebe8> in <module>()
----> 1 lm.entropy(test_words)

/usr/lib/python2.7/dist-packages/nltk/model/ngram.pyc in entropy(self, text)
    131             context = tuple(text[i - self._n + 1 : i - 1])
    132             token = text[i]
--> 133             e += self.logprob(token, context)
```

```
    134         return e
    135
/usr/lib/python2.7/dist-packages/nltk/model/ngram.pyc in logprob(self, word, context)
     96         """
     97
---> 98         return -log(self.prob(word, context), 2)
     99
    100     def choose_random_word(self, context):

/usr/lib/python2.7/dist-packages/nltk/model/ngram.pyc in prob(self, word, context)
     77             return self[context].prob(word)
     78         elif self._n > 1:
---> 79             return self._alpha(context) * self._backoff.prob(word,
context[1:])
     80         else:
     81             raise RuntimeError("No probability mass assigned to word %s in "
+

/usr/lib/python2.7/dist-packages/nltk/model/ngram.pyc in prob(self, word, context)
     80         else:
     81             raise RuntimeError("No probability mass assigned to word %s in "
+
---> 82                                "context %s" % (word, ' '.join(context)))
     83
     84     def _alpha(self, tokens):

TypeError: not all arguments converted during string formatting
```

Counting

For example, how many words in a corpus are not in WordNet?

In [41]:
```
from nltk.corpus import wordnet
from nltk.probability import ConditionalFreqDist

cfd = ConditionalFreqDist(
      (pos, len(wordnet.synsets(word)) > 0) for word,pos in treebank.tagged_words()
)

cfd.tabulate()
```

```
       False True
     #   16    0
     $  724    0
    ''  694    0
     , 4885    1
 -LRB-  120    0
-NONE- 5493 1099
 -RRB-  126    0
     . 3874    0
     :  563    0
    CC 1651  614
    CD 1527 2019
    DT 5230 2935
    EX    0   88
    FW    1    3
    IN 5354 4503
    JJ  676 5158
```

```
  JJR    1  380
  JJS    1  181
   LS    0   13
   MD  409  518
   NN  785 12381
  NNP 2770 6640
 NNPS    7  237
  NNS   93 5954
  PDT    0   27
  POS  824    0
  PRP  686 1030
 PRP$  423  343
   RB  343 2479
  RBR    0  136
  RBS    0   35
   RP    1  215
  SYM    1    0
   TO 2179    0
   UH    0    3
   VB    4 2550
  VBD    1 3042
  VBG    2 1458
  VBN    5 2129
  VBP   59 1262
  VBZ  104 2021
  WDT  445    0
   WP   74  167
  WP$   14    0
  WRB  164   14
   ``  712    0
```

Missing functionality

Head word identification
NLTK has no functionality to identify the head words of phrases. In this noun phrase, 'man' is the head word, but it is not straightforward to identify it.

In [58]:
```
np = Tree('(NP (D The) (N man) (PP (P with) (NP (D a) (N gun))))')
np.draw()
```

Last words:

A tip

Did you know you can add arbitrary attributes to an object instance?

In [49]:
```
class MyClass: pass

mc = MyClass()

mc.foo = 'bar'

print mc.foo
```

```
bar
```

This is useful for dynamic programming, but how do you test for presence/abscence?

In [54]:
```
print mc.baz is None
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/home/joseph/Projects/kanscipy/<ipython-input-54-63eaf6b88ae4> in <module>()
----> 1 print mc.baz is None

AttributeError: MyClass instance has no attribute 'baz'
```

In [57]:
```
print hasattr(mc, 'baz')
```

```
False
```

But `hasattr` is controversial and may disappear

In [55]:
```
try:
    print mc.baz is None
except AttributeError:
    pass
```

In [56]:
```
def tryattr(obj, attr, default=None):
    try:
        return getattr(obj, attr)
    except AttributeError:
        return default

print tryattr(mc, 'baz')
print tryattr(mc, 'foo')
```

```
None
bar
```