Python Natural Language Tool Kit

Chapter 3 – Creating Custom Corpora

Setting up a custom corpus (08/07)

The following shows how to set up a directory for future work. By NLTK convention, the corpora gets stored in /Users/share/nltk\_data.

>>> import os, os.path

>>> path = os.path.expanduser('~/nltk\_data') #sets path to: '/Users/zurich/nltk\_data'

>>> if not os.path.exists(path): #create the directory

os.mkdir(path)

>>> os.path.exists(path) #test the beginning of the if statement again.

True

>>> import nltk.data

>>> path in nltk.data.path

True

>>> path

'/Users/zurich/nltk\_data'

>>> nltk.data.path #there is a list of pre-defined paths in the NLTK

['/Users/zurich/nltk\_data', '/usr/share/nltk\_data', '/usr/local/share/nltk\_data', '/usr/lib/nltk\_data', '/usr/local/lib/nltk\_data']

The folder created above now has sub folders created within it: ~/nltk\_data/corpora/cookbook/.

Creating a word list corpus (09.07)

The following shows a WordListCorpusReader. It produces a list of words from the file. It takes two arguments: a directory path and a list of filenames. If you open the Python console in the same directory as the files, then ‘.’ Is used as the directory path.

>>> from nltk.corpus.reader import WordListCorpusReader

>>> reader = WordListCorpusReader('.', ['wordlist.txt'])

>>> reader.words() [

'nltk', 'corpus', 'corpora', 'wordnet']

>>> reader.fileids() ['wordlist']

The wordlist.txt file contains a word per line.

*Names Corpus*

A corpus that comes with NLTK is the names corpus. It contains male and female names.

>>> from nltk.corpus import names

>>> names.fileids()

['female.txt', 'male.txt']

>>> len(names.words('female.txt'))

5001

>>> len(names.words('male.txt'))

2943

*English Words*

NLTK also comes with a large list of English words. There is one file with 850 basic words and another list with over 200,000 known English words.

>>> from nltk.corpus import words

>>> words.fileids()

['en', 'en-basic']

>>> len(words.words('en-basic'))

850

>>> len(words.words('en'))

234936

Creating a part-of-speech tagged word corpus (09.07)

Most of the time, a tagger must be first trained on a training corpus. But first we must know how to create and use a training corpus of part of speech tagged words.

The/at-tl expense/nn and/cc time/nn involved/vbn are/ber astronomical/jj ./.

The line above was used and put into a file called ‘brown.pos’. This file included all the back slash stuff above. In the code below, the r'.\*\.pos' is a regular expression that matches all files with extension .pos.

>>> from nltk.corpus.reader import TaggedCorpusReader

>>> reader = TaggedCorpusReader('/Users/zurich/nltk\_data/corpora/cookbook', r'.\*\.pos')

>>> reader.words()

['The', 'expense', 'and', 'time', 'involved', 'are', ...]

>>> reader.tagged\_words()

[('The', None), ('expense', None), ('and', None), ...]

The following shows a similar idea except the lists are lists of sentences or lists of paragraphs:

>>> reader.sents()

[['The', 'expense', 'and', 'time', 'involved', 'are', 'astronomical', '.']]

>>> reader.tagged\_sents()

[[('The', 'AT-TL'), ('expense', 'NN'), ('and', 'CC'), ('time', 'NN'), ('involved', 'VBN'), ('are', 'BER'), ('astronomical', 'JJ'), ('.', '.')]]

>>> reader.paras()

[[['The', 'expense', 'and', 'time', 'involved', 'are', 'astronomical', '.']]]

>>> reader.tagged\_paras()

[[[('The', 'AT-TL'), ('expense', 'NN'), ('and', 'CC'), ('time', 'NN'), ('involved', 'VBN'), ('are', 'BER'), ('astronomical', 'JJ'), ('.', '.')]]]

***Changing Default Tokenizers (Word Tokenizer)***

The default tokenizeer is an instance of nltk.tokenizer.WhiteSpaceTokenizer, if you want to use a different tokenizer, then you can pass this in as word\_tokenizer.

>>> from nltk.tokenize import SpaceTokenizer

>>> path = '/Users/zurich/nltk\_data/corpora/cookbook'

>>> reader = TaggedCorpusReader(path, r'.\*\.pos', word\_tokenizer=SpaceTokenizer())

>>> reader.words()

['The', 'expense', 'and', 'time', 'involved', 'are', ...]

***Changing Default Tokenizers (Sentence Tokenizer)***

The default sentence tokenizer is an instance of nltk.tokenize.RegexpTokenizer with ‘\n’ to identify the gaps. It assumes tha that each sentence is on a line all by itself. An individual sentence is on a single line. To customizer this, you can pass is your own tokenizer as sent\_tokenizer.

>>> from nltk.tokenize import LineTokenizer

>>> path = '/Users/zurich/nltk\_data/corpora/cookbook'

>>> reader = TaggedCorpusReader(path, r'.\*\.pos', sent\_tokenizer=LineTokenizer())

>>> reader.sents()

[['The', 'expense', 'and', 'time', 'involved', 'are', 'astronomical', '.']]

***Changing Paragraph Block Reader***

Paragraphs are assumed to be split by blank lines. There are a number of other block reader functions whose purpose is to read blocks of text from a stream. There usage will be covered in “Creating a custom corpus view”

***Simplifying Tags***

There are a number of tag simplification functions defined in nltk.tag.simplify. These can be useful for reducing the number of different part of speech tags:

>>> from nltk.tag import simplify

>>> path = '/Users/zurich/nltk\_data/corpora/cookbook'

>>> reader = TaggedCorpusReader(path, r'.\*\.pos')

>>> reader = TaggedCorpusReader(path, r'.\*\.pos', tag\_mapping\_function=simplify.simplify\_brown\_tag)

>>> reader.tagged\_words(simplify\_tags=True) [('The', 'DET'), ('expense', 'N'), ('and', 'CNJ'), ...]

Creating a chunked phrase corpus (09.07)

A **chunk** is a short phrase in a sentence. Chunks are sub-trees within a sentence tree. They will be covered in more detail in Chapter 5 ‘Extracting Chunks’. The following shows a chunk:

**[Earlier/JJR staff-reduction/NN moves/NNS]** have/VBP trimmed/VBN about/IN **[300/CD jobs/NNS]** ,/, **[the/DT spokesman/NN]** said/VBD ./.

The parts in bold (in brackets) are chunks. Words that are not brackets are part of the sentence tree but are not part of any noun phrase. This file was saved as *‘treebank.chunk’.*

>>> path = '/Users/zurich/nltk\_data/corpora/cookbook'

>>> reader = ChunkedCorpusReader(path, r'.\*\.chunk')

>>> reader.chunked\_words()

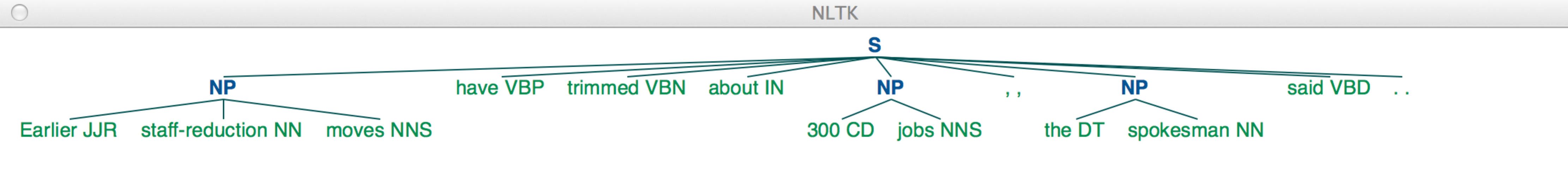
[Tree('NP', [('Earlier', 'JJR'), ('staff-reduction', 'NN'), ('moves', 'NNS')]), ('have', 'VBP'), ...]

In chunked\_words(), you get a list of noun phrase trees alongside tagged tokens of words that were not in a chunk.

***Using the draw method***

The following code produces:

>>> reader.chunked\_sents()[0].draw()



***Not Included***

There were examples on how to use chunks with IOB tags. There were also statementst that the NLTK includes some 5% samples from the Penn Treebank Project and some samples from the Conference on Computational Natural Language Learning (CoNLL).

Creating a categorized text corpus (09.07)

To set this up, two files were created called: movie\_pos.txt & movie\_neg.txt. And these files had some example text. Now, the following code was used, which really just examined the file names:

>>> from nltk.corpus.reader import CategorizedPlaintextCorpusReader

>>> path = '/Users/zurich/nltk\_data/corpora/cookbook'

>>> reader = CategorizedPlaintextCorpusReader(path, r'movie\_.\*\.txt', cat\_pattern=r'movie\_(\w+)\.txt')

>>> reader.categories()

['neg', 'pos']

>>> reader.fileids(categories=['neg'])

['movie\_neg.txt']

>>> reader.fileids(categories=['pos'])

['movie\_pos.txt']

Ther code cat\_pattern=r'movie\_(\w+)\.txt') does most of the work. **The category must be surrounded by grouping brackets.**  The movie\_reviews corpus only has two categories but the the reuters corpus has more than 90 categories.

Creating a categorized chunk corpus reader

To