Introduction to NLTK: Worksheet 2

Trevor Cohn and Yves Peirsman

Euromasters Summer School 2005

Refer to the running instructions from the previous worksheet.

Parsing

Please run the following parsing demonstrations:

```
>>> from nltk.draw.srparser import *
>>> demo()  # the shift-reduce parser

>>> from nltk.draw.rdparser import *
>>> demo()  # the recursive-descent parser

>>> from nltk.draw.chart import *
>>> demo()  # the chart parser
```

Chunking

Please run the chunking demonstration:

```
>>> from nltk.parser.chunk import *
>>> demo()
```

Develop a noun phrase chunker for the chunking corpus (`nltk.corpus.chunking`) using the regular-expression chunk parser `REChunkParser`. Use any combination of rules (`ChunkRule`, `ChinkRule`, `UnChunkRule`, `MergeRule`, `SplitRule`, and `REChunkParserRule`).

Load a sample of the CoNNL data, as shown below. This loads the first twenty or so sentences from the corpus. It then supplies the word tokens to the chunker, which adds a `TREE` property to the sentence. These chunked sentences are then compared to the originals, using the `ChunkScore` class. Your chunker should use more advanced rules than the one given here, which simply chunks any sequence of determiners, adjectives and nouns.

```
from nltk.parser.chunk import *
from nltk.corpus import chunking
from nltk.tokenreader import ConllTokenReader

# load just the first few sentences [feel free to remove this limitation]
text = chunking.raw_read('train.txt')[:8830]
reader = ConllTokenReader(('NP',), SUBTOKENS='WORDS')
sentences = reader.read_token(text)

# create a chunk parser
rule1 = ChunkRule(r'<DT|JJ|N.*>+', 'Chunk NPs')
chunkparser = RegexpChunkParser([rule1],
        chunk_node='NP', top_node='S', trace=1, SUBTOKENS='WORDS')

# process each sentence, parsing and scoring relative to the gold standard
chunkscore = ChunkScore()
for sentence in sentences['SENTS']:
    test = Token(WORDS=sentence['WORDS'])
    chunkparser.parse(test)
    chunkscore.score(sentence['TREE'], test['TREE'])

# display the precision, recall and F measure
print chunkscore
```

The `nltk.parser.chunk` module supports the development of chunkers in two ways. First, you can use the `trace=1` optional argument of the `RegexpChunkParser` constructor to show the chunks after each rule has been applied. Second, you can use the `chunkscore.incorrect()` and `chunkscore.missed()` methods to report any false positive and false negative chunks (see the chunking tutorial for more details).