Chunk parsing exercise

- Goal:
  - Write a programm that uses regular expressions to recognize noun groups/chunks.
- Starting point:
  - http://www.cnts.ua.ac.be/conll2000/chunking/
  - Check CoNLL 2002 homepage and learn about goals and data format
- Two different data sets are issued
  - Download training and test data
  - Evaluation script (in Perl)

Chunk parsing exercise

- Form of annotation:
  - Each line a word + its annotation
    - POS
    - NP/PP/VP chunks
- Sequence of words are annotated according to the IOB1 standard
  - I-XP: words inside a chunk
  - B-XP: beginning of a XP chunk
  - O: for all elements outside any chunk
  - XP stands for NP, PP, VP, ADJP, SBAR

Example of an chunk annotated sentence in CoNLL format

He PRP B-NP
reckons VBZ B-VP
the DT B-NP
current JJ I-NP
account NN I-NP
deficit NN I-NP
will MD B-VP
narrow VB I-VP
to TO B-PP
only RB B-NP
I-NP
1.8 CD I-NP
billion CD I-NP
in IN B-PP
September NNP B-NP
. . O

In order to use standard RE packages transform data into a string

- Input string
  "The/Det woman/NN will/MD give/VB Mary/NNP a/Det book/NN"
- Output string
  "The/B-NP woman/I-NP will/B-VP give/I-VP Mary/B-NP a/B-NP book/I-NP"

- Alternative:
  - In: "DET NN MD VB NNP Det NN"
  - Out: "B-NP I-NP … "
- Note: Gets easier input, probably more difficult output

In order to simplify the problem

- You might implement separate programs for recognizing each chunk type individually

Structure of program: read in data

- Read in each sentence from file (all tokens between \newline) using readline() function
- Two possibilies
  - Fetch two first elements (which are seprated by \tab)
  - Just fetch POS tag (second element)
- Append each such token to a string (initialized by „“)

Structure of program: define regular expressions (e.g., Python)

```
import re
# pattern for html tags of form <TAG> or </TAG> or both ending with digit
# if pattern is found bind it to pattern variable ?P<tag>

def apply_re(f):
    expr = re.compile(r"(?P<tag>(<(/)?[a-zA-Z]+(\d)?>))")
    file = open(f, 'r')
    for line in file.readlines():
        # searches only first match
        #res = expr.search(line)
        # searches all matches and binds it to a list
        res = expr.findall(line)
        if res == None:
            print "No match!"
        else:
            for tg in res:
                print tg
    return file.close()
```

Write result to output in CoNLL format

- Tranform each string to CoNLL format
- Append it to some output file

- Call Perl script conlleval.txt to evaluate your results