CSCI 4152/6509
Natural Language Processing

Lab 6:

Python NLTK Tutorial 2

Lab Instructor: Sigma Jahan and Mayank Anand

Faculty of Computer Science

Dalhousie University

Lab Overview

- Part-of-speech (POS) taggers:
  - HMM and CRF, Brill
- Tree model and Text chunker for capturing;
- Named-entity recognition (NER);
- Jupyter and JupyterHub

NLTK Tagging and Chunking Overview

- Representing Tagged Tokens

```
from nltk.tag import str2tuple

tagged_token = str2tuple('fly/NN')
print(tagged_token)
# ('fly', 'NN')
print(tagged_token[0])
# 'fly'
print(tagged_token[1])
# 'NN'
```

```
from nltk.tag import str2tuple

sent = '''
  The/AT grand/JJ jury/NN commented/VBD on/IN a/AT
  number/NN of/IN other/AP topics/NNS ,/, AMONG/IN
  them/PPO the/AT Atlanta/NP and/CC Fulton/NP-tl
  County/NN-tl purchasing/VBG departments/NNS which/WDT
  it/PPS said/VBD "/" ARE/BER well/QL operated/VBN
  and/CC follow/VB generally/RB accepted/VBN
  practices/NNS which/WDT inure/VB to/IN the/AT
  best/JJT interest/NN of/IN both/ABX governments/NNS
  "/" ./.
  '''
print([str2tuple(t) for t in sent.split()])
# [('The', 'AT'), ('grand', 'JJ'), ('jury', 'NN'),
# ('commented', 'VBD'),
# ('on', 'IN'), ('a', 'AT'), ('number', 'NN'), ...
#       ('.', '.')]
```

Reading Tagged Corpora

- Brown Corpus in a text editor:

```
The/at Fulton/np-tl County/nn-tl Grand/jj-tl Jury/nn-tl
said/vbd Friday/nr an/at investigation/nn of/in
Atlanta's/np$ recent/jj primary/nn election/nn
produced/vbd ``/`` no/at evidence/nn ''/'' that/cs
any/dti irregularities/nns took/vbd place/nn ./.
```

- NLTK interface to corpus:

```
from nltk.corpus import brown

print(brown.tagged_words())
# [('The', 'AT'), ('Fulton', 'NP-TL'), ...]

print(brown.tagged_words(tagset='universal'))
# [('The', 'DET'), ('Fulton', 'NOUN'), ...]
```

Exploring Penn-Treebank Corpus

- First, install the full NLTK corpus using Python console:

```
python3
```

and then add the following lines:

```
import nltk
nltk.download('treebank')
quit()
```

Reading the Penn Treebank (Wall Street Journal sample):

```
from nltk.corpus import treebank

print(treebank.fileids()) # doctest: +ELLIPSIS
# ['wsj_0001.mrg', 'wsj_0002.mrg', 'wsj_0003.mrg',
#    'wsj_0004.mrg', ...]
```

```
print(treebank.words('wsj_0003.mrg'))
# ['A', 'form', 'of', 'asbestos', 'once', 'used', ...]

print(treebank.tagged_words('wsj_0003.mrg'))
# [('A', 'DT'), ('form', 'NN'), ('of', 'IN'), ...]

# doctest: +ELLIPSIS +NORMALIZE_WHITESPACE
print(treebank.parsed_sents('wsj_0003.mrg')[0])
    (S
      (S-TPC-1
        (NP-SBJ
          (NP (NP (DT A) (NN form))
              (PP (IN of) (NP (NN asbestos))))
          (RRC ...)...)...)
      ...
      (VP (VBD reported)
          (SBAR (-NONE- 0) (S (-NONE- *T*-1))))
      (. .))
```

Note: *If you have access to a full installation of the Penn Treebank, NLTK can be configured to load it as well. For your own purposes, you can download the* `ptb` *package, and in the directory* `nltk_data/corpora/ptb` *place the BROWN and WSJ directories of the Treebank installation (symlinks work as well).*

Ready-made POS Tagger

```
from nltk import tag

sent = ['Today','you',"'ll",'be','learning','NLTK','.']
tagged_sent = tag.pos_tag(sent)
print(tagged_sent)
# [('Today', 'NN'), ('you', 'PRP'), ("'ll", 'MD'),
#  ('be', 'VB'), ('learning', 'VBG'), ('NLTK', 'NNP'),
#  ('.', '.')]
```

The Penn Treebank POS tags are covered in the class, and a list of tags without punctuation is also available at:
`http://www.ling.upenn.edu/courses/Fall_2003/ling001/penn_treebank_pos.html`

Ready-made NE (Named Entity) Chunker

```
from nltk import chunk, tag

sent = ['Today','you',"'ll",'be','learning','NLTK','.']
tagged_sent = tag.pos_tag(sent)

tree = chunk.ne_chunk(tagged_sent)
print(tree)
# Tree('S',
#   [('Today', 'NN'), ('you', 'PRP'),
#    ("'ll", 'MD'), ('be', 'VB'), ('learning', 'VBG'),
#    Tree('ORGANIZATION', [('NLTK', 'NNP')]),
#    ('.', '.')])
```

```
ne_subtrees = tree.subtrees(filter=lambda t:
  t.label() in ["ORGANIZATION", "PERSON",
    "LOCATION","DATE","TIME", "MONEY","PERCENT",
    "FACILITY","GPE"])
ne_subtrees_list = [tree for tree in ne_subtrees]
print(ne_subtrees_list)
# [Tree('ORGANIZATION', [('NLTK', 'NNP')])]

ne_phrases = [' '.join(word for word, pos in
  tree.leaves()) for tree in ne_subtrees_list]
print(ne_phrases)
# ['NLTK']
```

Step 1. Logging in to server timberlea

- Login to server timberlea
- Change directory to csci4152 or csci6509
- Create directory `lab6` and `cd` to it:

```
mkdir lab6
cd lab6
```

Step 2: Training HMM POS Tagger

- Training an HMM POS Tagger
- Let us train it using `treebank` corpus
- Type the following into `hmm_tagger.py`

```
# Import the toolkit and tags
from nltk.corpus import treebank
# Import HMM module
from nltk.tag import hmm
```

```
# Train data - pretagged
train_data = treebank.tagged_sents()<slice_first_3000>
# Test data - pretagged
test_data = treebank.tagged_sents()
                          <slice_other_than_first_3000>

print(train_data[0])

# Setup a trainer with default(None) values
# And train with the data
trainer = hmm.HiddenMarkovModelTrainer()
tagger = trainer.train_supervised(train_data)

print(tagger)
# Prints the basic data about the tagger

print(tagger.tag("Today is a good day.".split()))
```

```
print(tagger.tag(
            "Joe met Joanne in New Delhi.".split()))

print(tagger.tag(
       "Chicago is the birthplace of Marry".split()))

print(tagger.evaluate(test_data))
```

- Submit `hmm_tagger.py` using the `submit-nlp` command

Step 3: Training CRF POS Tagger

- Install the CRF Suite with:

```
pip install --user python-crfsuite
```

- Training CRF based tagger:

```
# Import the toolkit and tags
from nltk.corpus import treebank

# Import CRF module
from nltk.tag import crf

# Train data - pretagged
train_data = treebank.tagged_sents()<same_as_previous>

# Train data - pretagged
test_data = treebank.tagged_sents()<same_as_previous>
```

```
# Setup a trainer with default(None) values
# Train with the data
tagger = crf.CRFTagger()
tagger.train(train_data,'model.crf.tagger')

print(tagger)
# Prints the basic data about the tagger
print(tagger.evaluate(test_data))
# <your comment section>
#
#
```

- Complete `crf_tagger.py` and submit using the `submit-nlp` command

Step 4: Brill Tagger Demo

```
from nltk.tbl.demo import postag

postag(num_sents=None, train=0.7665)
# if we set num_sents to None, it will use the whole
# treebank corpus.  We want this, so we can compare
# the results to the CRF and HMM we tested earlier.
# If we set train ratio to 0.7665, the train set will
# have 3000 sentences, just like in previous taggers.
# The other params are default.

# <your_comments_here>
#
#
```

- Complete `brill_demo.py` and submit using the `submit-nlp` command

Step 5: Named Entity Chunking Task

```
from nltk import FreqDist
# import treebank
# import ne chunker

data = # load treebank data
chunkd_data = # chunk the data
chunkd_trees = # select subtrees which are NE

word_fd = FreqDist([' '.join(word for word, pos in
           tree.leaves()) for tree in chunkd_trees])

print("Three most common named entities are: ")
for token, freq in word_fd.most_common(3):
    print("%s : %d"%(token, freq))
```

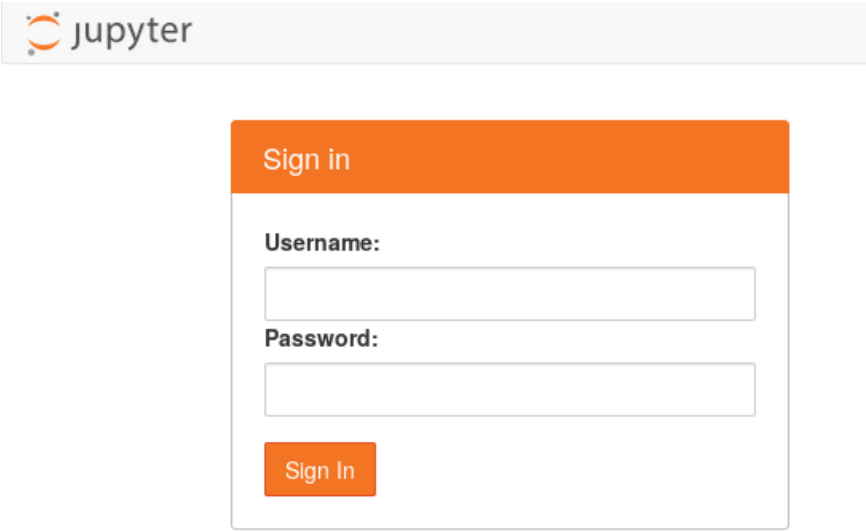
- Submit `ne_chunker_exercise.py` using the `submit-nlp` command

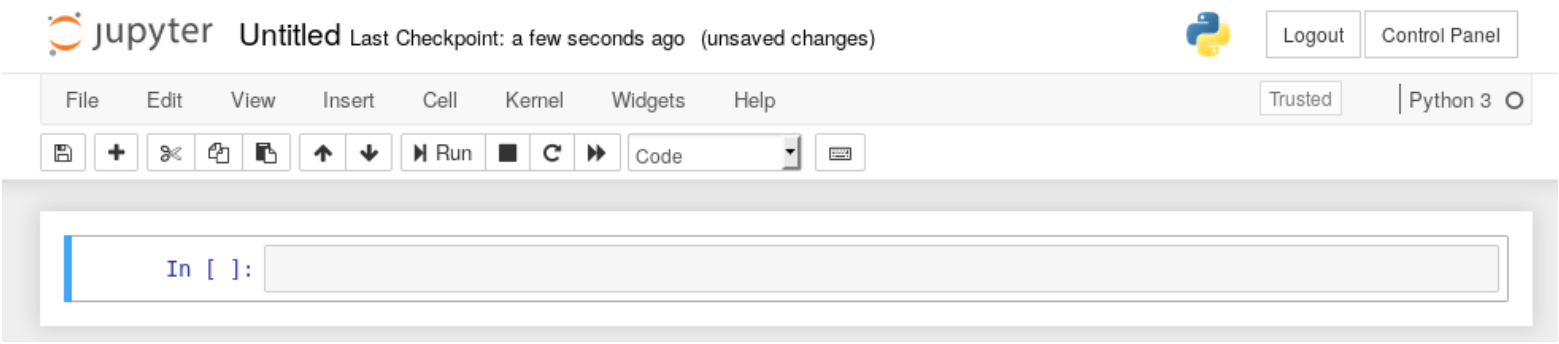
Jupyter Overview

- Jupyter notebook
  - Combines code and rich text elements, images, links, math equations
  - Brings everything together
  - Can be executed
  - Jupyter: acronym for Julia Python R
- Jupyter Notebook App
  - Used to produce notebook documents
  - Can install on your laptop
  - We use JupyterHub on Timberlea
  - Main components: kernels and dashboard

Step 6: Using JupyterHub on Timberlea

- Login to the JupyterHub on Timberlea





- Rename notebook to `first_notebook`
- Change Cell Type to Markdown
- Edit cell

```
# My First Jupyter Notebook

In this example, we will show how to demostrate
results of a POS tagger.
```

- Click “Run” or press Ctrl+Enter to reformatt the cell
- Enter in new cell code from `ne_chunker_exercise.py` with necessary updates, as before:

```
from nltk import FreqDist
# import treebank
# import ne chunker

data = # load treebank data
chunkd_data = # chunk the data
chunkd_trees = # select subtrees which are NE
```

```
word_fd = FreqDist([' '.join(word for word, pos in
           tree.leaves()) for tree in chunkd_trees])

print("Three most common named entities are: ")
for token, freq in word_fd.most_common(3):
    print("%s : %d"%(token, freq))
```

- Run the notebook, if necessary “Toggle” output to make it appear in the notebook
- You should see a page like this:

jupyter first_notebook Last Checkpoint: 38 minutes ago (autosaved)

Logout Control Panel

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

My First Jupyter Notebook

In this example, we will show how to demonstrate results of a POS tagger.

In [6]:
```
from nltk import FreqDist
# import treebank
[redacted]
# import ne chunker
[redacted]

#data = # load treebank data
data = [redacted]
#chunkd_data = # chunk the data
chunkd_data = [redacted]
#chunkd_trees = # select subtrees which are NE
[redacted]

word_fd = FreqDist([' '.join(word for word, pos in tree.leaves()) for tree in chunkd_trees])

print("Tree most common named entities are: ")
for token, freq in word_fd.most_common(3):
    print("%s : %d"%(token,freq))
```

```
Tree most common named entities are:
U.S. : 215
New York : 103
Japanese : 87
```

Final Steps

- Save the notebook
- Submit `first_notebook.ipynb` using the `submit-nlp` command

This is the end of Lab 6.