CSE 5525 POS Tutorial

We will be working through Chapter 5 of the NLTK Book, which can be found at:

http://nltk.org/book/ch05.html

One thing to remember about python is that spacing matters. Don't skip the indentation in the examples.

1) Open a terminal window, and run "python" (or run the ipython notebook).
2) Execute "from nltk import *" to pull the NLTK into python
3) Run "nltk.download()" and under models, install the HMM tagger and the MaxEnt Tagger
4) Work through the command lines in Section 5.1, which will guide you through the basics of how NLTK treats the tagging problem.
5) In Section 5.2, feel free to skip down to "Exploring Tagged Corpora"
6) Feel free to skip 5.3 (although it does explain the concept of dictionaries in python).
7) Work through Section 5.4.
8) If you have time, compare performance of the n-gram tagger (5.5, similar to the HMM tagger we will talk about Tuesday) and the transformation-based tagger (5.6, aka the Brill tagger we talked about).

Advanced tutorial:
Several students asked to look at advanced material such as twitter part of speech tagging. We can do so in NLTK by hacking around a bit in python. First:

```
cd nltk-data/corpora
wget http://ark-tweet-nlp.googlecode.com/files/twpos-data-v0.3.tgz
tar –xzf twpos-data-v0.3.tgz
```

This will create the twitter POS tagged data set. You can then open up python and load using the following:

```
from nltk import *

rd=TaggedCorpusReader(
'nltk_data/corpora/twpos-data-v0.3/oct27.splits/',
'oct27.train',sep='\t',
word_tokenizer=RegexpTokenizer('\n',gaps=True),
sent_tokenizer=RegexpTokenizer('\n\n', gaps=True))

rd_dev=TaggedCorpusReader('nltk_data/corpora/twpos-data-v0.3/oct27.splits/',
'oct27.dev',sep='\t',
word_tokenizer=RegexpTokenizer('\n', gaps=True),
sent_tokenizer=RegexpTokenizer('\n\n', gaps=True))
```

Now that you've read in the data you can look at the examples:

```
ts=rd.tagged_sents()
ts[0]
dev=rd_dev.tagged_sents()
dev_sents=rd_dev.sents()
```

Let's create a backoff tagger (from http://streamhacker.com/2008/11/03/part-of-speech-tagging-with-nltk-part-1/)

```
def backoff_tagger(tagged_sents, tagger_classes, backoff=None):
  if not backoff:
    backoff = tagger_classes[0](tagged_sents)
    del tagger_classes[0]

  for cls in tagger_classes:
    tagger = cls(tagged_sents, backoff=backoff)
    backoff = tagger

  return backoff

ubt_tagger = backoff_tagger(train_sents, [nltk.tag.UnigramTagger,
nltk.tag.BigramTagger, nltk.tag.TrigramTagger])
```

This creates a backoff tagger that first tries to use a trigram, then a bigram, then a unigram. It doesn't work great on this data:

```
ubt_tagger.evaluate(dev)
0.6328011611030478
ubt_tagger.tag(dev_sents[0])
[(u'@ciaranyree', None), (u'it', u'O'), (u'was', u'V'), (u'on', u'P'), (u'football', u'N'),
(u'wives', None), (u',', u','), (u'one', u'$'), (u'of', u'P'), (u'the', u'D'), (u'players', None),
(u'and', u'&'), (u'his', u'D'), (u'wife', u'N'), (u'own', u'N'), (u'smash', None), (u'burger',
None)]
```

Adding an affix tagger helps:

```
aubt_tagger=backoff_tagger(ts,[nltk.tag.AffixTagger,nltk.tag.UnigramTagger,
nltk.tag.BigramTagger, nltk.tag.TrigramTagger])
aubt_tagger.evaluate(dev)
0.6954177897574124
```

To do – look at the development data tags – are there things that the tags are missing? Could you look into other taggers in the cascade (e.g., RegexpTagger)?