CMPE 409 Machine Translation

Worksheet(Week-07)

1 Download NLTK

Download NLTK package with following instructions

```
>>> import nltk
>>> nltk.download()
```

Note: you need some files from the link below:
https://github.com/japerk/nltk3-cookbook

2 Train a Unigram POS tagger

```
print ("Test Unigram")
from nltk.tag import UnigramTagger
from nltk.corpus import treebank
text= treebank.sents()[0]
print(text)


tagged_sentene= treebank.tagged_sents()
print(tagged_sentene)

train_sents= tagged_sents()[:3000]

#create the tagger with tagged words

tagger = UnigramTagger(train_sents) # now we have the tagger
result= tagger.tag(text)
print(result)

## Evaluate
test_sents = treebank.tagged_sents()[3000:]
eval=tagger.evaluate(test_sents)
print(eval)
```

3 Combining and Saving taggers

```
tagger1 = DefaultTagger('NN')
tagger2 = UnigramTagger(train_sents, backoff=tagger1)
print("The resul of combined taggers is: ")
print(tagger2.evaluate(test_sents))

## Saving and uploading

import pickle
f = open('tagger.pickle', 'wb')
pickle.dump(tagger, f)
f.close()
print("Tagger is saved")

f = open('tagger.pickle', 'rb')
tagger = pickle.load(f)
print (tagger is uploaded)
```

4 Bigram and Trigram taggers

```
from nltk.tag import BigramTagger, TrigramTagger
bitagger = BigramTagger(train_sents)
print("Result of bigram:", bitagger.evaluate(test_sents))

tritagger = TrigramTagger(train_sents)
print("Result of trigram", tritagger.evaluate(test_sents))
```

5 Combine ngram taggers

```
from tag_util import backoff_tagger
backoff = DefaultTagger('NN')
tagger = backoff_tagger(train_sents,
                            [UnigramTagger,
                             BigramTagger,
                             TrigramTagger],
                             backoff=backoff)
print("The performance is: ",tagger.evaluate(test_sents))
```

Note: This method is defined in "tag_util.py" file
https://github.com/japerk/nltk3-cookbook

6 Quadgram tagger

```
from taggers import QuadgramTagger
quadtagger = backoff_tagger(train_sents,
                            [UnigramTagger,
                             BigramTagger,
                             TrigramTagger,
                             QuadgramTagger],
                            backoff=backoff)
print("Quadgram: ", quadtagger.evaluate(test_sents))
```

7 Tagging with Regular Expression

```
from tag_util import patterns
from nltk.tag import RegexpTagger
tagger = RegexpTagger(patterns)
print("Tagging with Regular Expression:", tagger.evaluate(test_sents))
```

Note: this pattern can be found in tag _util.py
https://github.com/japerk/nltk3-cookbook

Test: modify the "pattern" file and test it again.

8 Using WordNet for Tagging

```
from taggers import WordNetTagger
wn_tagger = WordNetTagger()
print("Worden Tagging: ", wn_tagger.evaluate(train_sents))

print("apply chaining...")
from tag_util import backoff_tagger
from nltk.tag import UnigramTagger, BigramTagger,
                                        TrigramTagger
```

```
tagger = backoff_tagger(train_sents, [UnigramTagger,
                                      BigramTagger,
                                      TrigramTagger],
                                    backoff=wn_tagger)

print("Wordnet chaining: ",tagger.evaluate(test_sents))
```

9 Tagging Proper Names

```
from taggers import NamesTagger
nt = NamesTagger()
result= nt.tag(['Jacob'])
print("Tagging proper names: ", result)
```

Try this with some Turkish names.

10 Uploading

Show your work to your instructor and upload to learn

11 Training a tagger with NLTK-Trainer

This section is optional. You may try it at your home. The simplest way to run train_tagger.py is with the name of an NLTK corpus.

```
python train_tagger.py treebank
...........
.............
dumping TrigramTagger to /Users/jacob/nltk_data/
taggers/treebank_aubt.pickle
```

12 Resource

This worksheet is prepared from the following books:

- Jacob Perkins, **Python 3 Text Processing with NLTK 3 Cookbook, Packt Publishing**,ISBN: 9781782167853
- Steven Bird, Ewan Klein & Edward Loper, **Natural Language Processing with Python**, O'Reily, June, 2009