3조 – POS(Parts of Speech) 태깅하기 (3장 74-79pg)

sentence = 'The brown fox is quick and he is jumping over the lazy dog'

sentence 에 ‘The brown fox is quick and he is jumping over the lazy dog’라는 string을 지정한다.

# recommended tagger based on PTB

import nltk

tokens = nltk.word\_tokenize(sentence)

tagged\_sent = nltk.pos\_tag(tokens, tagset='universal')

print tagged\_sent

먼저 nltk 를 호출하여 변수 sentence의 nltk.word\_tokenize 함수를 tokens값으로 정한다.

이후, 변수 tokens와 tagset의 nltk.pos\_tag 함수를 tagged\_sent 값으로 정한다.

Tagged\_sent 값을 출력한다.

from pattern.en import tag

tagged\_sent = tag(sentence)

print tagged\_sent

pattern.en에서 tag로부터 호출한다. 변수 sentence의 내용에 대한 tag함수를 tagged\_sent로 새로 지정한 후, Tagged\_sent 값을 출력한다.

# building your own tagger

# preparing the data

from nltk.corpus import treebank

data = treebank.tagged\_sents()

train\_data = data[:3500]

test\_data = data[3500:]

print train\_data[0]

nltk.corpus로부터 treebank를 호출한다

Treebank.tagged\_sents()의 함수를 data에 저장한다.

Data 에 저장된 treebank의 0 부터 3500 까지의 정보를 train\_data에 저장한다.

Data 에 저장된 treebank의 3500부터 마지막까지의 정보를 test\_data에 저장한다.

Train\_data 에서 0번째(즉 첫 번째 정보를) 출력한다.

# default tagger

from nltk.tag import DefaultTagger

dt = DefaultTagger('NN')

print dt.evaluate(test\_data)

print dt.tag(tokens)

nltk.tag 로부터 DefaultTagger를 호출한다.

Dt 에 DefaultTagger 함수에 변수 ‘NN(Noun)’ 이 입력된 값을 저장한다.

Dt.evaulate 함수에서 변수(test\_data: data에 저장된 treebank의 3500부터 마지막까지의 정보)의 값을 출력한다.

이어 dt.tag 함수에서 변수(tokens)의 값을 출력한다.

# regex tagger

from nltk.tag import RegexpTagger

nltk.tag 로부터 RegexpTagger를 호출한다.

# define regex tag patterns

patterns = [

(r'.\*ing$', 'VBG'), # gerunds

(r'.\*ed$', 'VBD'), # simple past

(r'.\*es$', 'VBZ'), # 3rd singular present

(r'.\*ould$', 'MD'), # modals

(r'.\*\'s$', 'NN$'), # possessive nouns

(r'.\*s$', 'NNS'), # plural nouns

(r'^-?[0-9]+(.[0-9]+)?$', 'CD'), # cardinal numbers

(r'.\*', 'NN') # nouns (default) ...

]

rt = RegexpTagger(patterns)

print rt.evaluate(test\_data)

print rt.tag(tokens)

이제 패턴을 지정한 후, 이를 RegexpTagger 함수의 변수로 넣어야 한다.

여기서 패턴을 지정할 때 gerunds…부터 nn 까지를 영어 문법에서의 suffix에 기반하여 패턴을 지정하였다. 예를 들어 ing로 끝나는 단어는 gerunds(동명사)에 속할 것이고, ould로 끝나는 단어는 modals(조동사)로 분류될 것이다. 이러한 규칙성에 대한 각 코딩을 pattern이라는 명령어에 저장한다. 이후, RegexpTagger 함수의 변수 pattern에 대한 결과값을 rt에 저장한다.

Rt.evaulate 함수에 변수 test\_data 값을 출력하고, 이어 rt.tag 함수에 변수 tokens 값을 출력한다.

## N gram taggers

from nltk.tag import UnigramTagger

from nltk.tag import BigramTagger

from nltk.tag import TrigramTagger

nltk.tag 로부터 UnigramTagger를 불러온다

nltk.tag 로부터 BigramTagger를 불러온다

nltk.tag 로부터 TrigramTagger를 불러온다.

ut = UnigramTagger(train\_data)

bt = BigramTagger(train\_data)

tt = TrigramTagger(train\_data)

ut에 UnigramTagger 함수의 변수 train\_data 의 값을 저장한다.

bt에 BigramTagger 함수의 변수 train\_data 의 값을 저장한다.

tt에 TrigramTagger 함수의 변수 train\_data의 값을 저장한ㄷ.

print ut.evaluate(test\_data)

print ut.tag(tokens)

ut.evaulate 함수의 test\_data 변수에 대한 값을 출력한다

ut.tag 함수의 tokens 변수에 대한 값을 출력한다.

print bt.evaluate(test\_data)

print bt.tag(tokens)

bt.evaluate 함수의 test\_data 변수에 대한 값을 출력한다.

bt.tag 함수의 tokens 변수에 대한 값을 출력한다.

print tt.evaluate(test\_data)

print tt.tag(tokens)

tt.evaluate 함수의 test\_data 변수에 대한 값을 출력한다.

tt.tag 함수의 tokens 변수에 대한 값을 출력한다.

def combined\_tagger(train\_data, taggers, backoff=None):

for tagger in taggers:

backoff = tagger(train\_data, backoff=backoff)

return backoff

먼저 def 이하, backoff 까지의 코딩은 loop가 지정되어 있음을 알 수 있다.

backoff의 값이 None인 변수와 train\_data, taggers 변수에 대한 combined\_tagger 함수 값을 define 했을 때, tagger를 적용한다. 이때 backoff 는 변수 train\_data, backoff=backoff 에 대한 tagger 함수를 값을 저장한다. 이 과정이 끝난 뒤에는 다시 backoff 로 돌아간다.

ct = combined\_tagger(train\_data=train\_data,

taggers=[UnigramTagger, BigramTagger, TrigramTagger],

backoff=rt)

combined\_tagger 함수에 대해 train\_data 값이 저장된 train\_data 와, UnigramTagger, BigramTagger, TrigramTagger 각각의 값이 모두 지정된 taggers와, rt 값이 저장된 backoff에 대한 세가지 변수의 값을 ct에 저장한다.

print ct.evaluate(test\_data)

print ct.tag(tokens)

ct.evaluate 함수에 대한 변수 test\_data에 대한 값을 출력한다.

Ct.tag 함수에 대한 변수 tokens에 대한 값을 출력한다.

from nltk.classify import NaiveBayesClassifier, MaxentClassifier

from nltk.tag.sequential import ClassifierBasedPOSTagger

nbt = ClassifierBasedPOSTagger(train=train\_data,

classifier\_builder=NaiveBayesClassifier.train)

print nbt.evaluate(test\_data)

print nbt.tag(tokens)

nltk.classify에서 NaiveBayesclassifier 와 MaxentClassifier를 불러온다.

Nltk.tag.sequential 에서 ClassifierBasedPOSTagger 를 불러온다.

Train\_data 값이 저장된 train과, NaiveBayesClassifier.train 값이 저장된 classifier\_builder의 변수들에 대한 ClassifierBasedPOSTagger 함수 값을 nbt에 저장한다.

이후, 해당 nbt.evaluate 함수에 대한 변수 test\_data 값을 출력한다.

이어서 nbt.tag 함수에 대한 변수 tokens 값을 출력한다.