컴파일러 설계

NLTK : 텍스트 분석 기초

Prepared by Prof. Youn-Sik Hong

목차

텍스트 분석 과정

- **문장 경계 인식 – 문장 분리**

- **토큰화 (tokenize) : 문장을 단어 단위로 쪼갬**
- **POS 태깅 : 품사(POS, parts-of-speech) 구분**
- **불용어(stop word) 제거**
- **텍스트 정규화(normalization)**
  - 철자 수정(spelling correction)
  - 어간 추출(stemming)
  - 표제어 추출(lemmatization)
- **개체 이름 인식(NER, named entity recognition)**
- **단어 중의성 해결(disambiguation)**

문장 경계 인식 – 문장 분리 (1/2) – nltk(text-analysis). ipynb

- **sent_tokenize : 텍스트를 여러 개 문장으로 구분**

```
In this book authored by Sohom Ghosh and Dwight Gunning, .. insights from it.
The first four chapter will introduce you to the basics of NLP.
Later chapters will describe how to deal with complex NLP prajects.
If you want to get early access of it, you should book your order now.
```

```
from nltk.tokenize import sent_tokenize

sentence = open("data_ch1/file.txt", "r").read()

sent_list  = sent_tokenize(sentence)
len(sent_list)
```

실제 원본 텍스트는 **Newline** 문자 없이 죽~ 이어져 있음.

문장 경계 인식 – 문장 분리 (2/2)

- **LineTokenizer : newline으로 문장이 구분되어 있을 때만 적용 가능**

```
from nltk.tokenize import LineTokenizer

sent = """
In this book authored by Sohom Ghosh and Dwight Gunning, .. insights from it. ₩n
The first four chapter will introduce you to the basics of NLP. ₩n
Later chapters will describe how to deal with complex NLP prajects. ₩n
If you want to get early access of it, you should book your order now.
"""

lTokenizer = LineTokenizer()
sent_list = lTokenizer.tokenize(sent)
sent_list[-1]
```

토큰화 : 문장을 단어(token) 단위로 쪼갬 (1/2)

- **word_tokenize : 구두점도 단어로 구분**

```
from nltk import word_tokenize

print(sent_list[0])
print()
words = word_tokenize(sentence)
print(words[:12])
```

```
In this book authored by Sohom Ghosh and Dwight Gunning, we shall learnning how to pracess Natueral
Language and extract insights from it.

['In', 'this', 'book', 'authored', 'by', 'Sohom', 'Ghosh', 'and', 'Dwight', 'Gunning', ',', 'we']
```

토큰화 : 문장을 단어(token) 단위로 쪼갬 (2/2)

- **TweetTokenizer : 이모티콘을 토큰으로 인식**

```
from nltk.tokenize import TweetTokenizer

tTokenizer = TweetTokenizer()
print("Tweet Tokenizer output :")
print(tTokenizer.tokenize("This is a cooool #dummysmiley: :-) :-P <3"))
```

```
Tweet Tokenizer output :
['This', 'is', 'a', 'cooool', '#dummysmiley', ':', ':-)', ':-P', '<3']
```

품사 태깅 (POS tagging) (1/3)

■ **품사 태깅 : 품사(POS, parts of speech)를 사용해 문장의 단어를 분류**

```
import nltk

text = nltk.word_tokenize("And now for something completely different")
print(nltk.pos_tag(text))
```

```
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'), ('completely'
, 'RB'), ('different', 'JJ')]
```

- 'And' : **CC** (관계 접속사, coordinating conjunction)
- 'now', 'completely' : **RB** (부사, adverb)
- 'for' : **IN** (전치사, preposition)
- 'something' : **NN** (명사, noun)
- 'different' : **JJ** (형용사, adjective)

```
>>> nltk.help.upenn_tagset('RB')
RB: adverb
    occasionally unabatingly maddeningly adventurously professedly
    stirringly prominently technologically magisterially predominately
    swiftly fiscally pitilessly ...
```

품사 태깅 (POS tagging) (2/3)

POS tagging은 단어를 품사로 분류. 이렇게 분류하는 게 얼마나 효과적일까? 'woman'(명사), 'bought'(동사), 'over'(전치사), 및 'the'(관사)에 대해 유사 단어를 찾아보자.

```
text = nltk.Text(word.lower() for word in nltk.corpus.brown.words())
```

```
text.similar('woman')
```

```
man time day year car moment world house family child country boy
state job place way war girl work word
```

```
text.similar('bought')
```

동사

```
made said done put had seen found given left heard was been brought
set got that took in told felt
```

```
text.similar('over')
```

전치사

```
in on to of and for with from at by that into as up out down through
is all about
```

```
text.similar('the')
```

관사

```
a his this their its her an that our any all one these my in your no
some other and
```

품사 태깅 (POS tagging) (3/3)

```
words = word_tokenize(sent_list[0])

print(sent_list[0])
nltk.pos_tag(words)
```

```
In this book authored by Sohom Ghosh and Dwight Gunning,
```

```
[('In', 'IN'),
('this', 'DT'),
('book', 'NN'),
('authored', 'VBN'),
('by', 'IN'),
('Sohom', 'NNP'),
('Ghosh', 'NNP'),
('and', 'CC'),
('Dwight', 'NNP'),
('Gunning', 'NNP'),
(',', ','),...
```

불용어 (stop word)

■ 불용어

- 문장 구성에서 구문론적 가치를 갖고 있음.
  - 그러나 무시할 수 있거나 최소한의 의미적 가치를 지닌 단어
- 구문론을 사용하지 않고 BoW(Bag of Words) 접근 방식을 사용하는 경우
  - 관심 단어를 제외한 불용어는 제거

```
nltk.download('stopwords')
from nltk.corpus import stopwords

stopwords = nltk.corpus.stopwords.words('english')
print(stopwords[:20])

words = word_tokenize(sentence)

new_words = [wd for wd in words if wd not in stop_words]
new_words[:10]
```

```
['i', 'me', 'my', 'myself', 'we', 'our',
"you'd", 'your', 'yours', 'yourself',
```

목차

텍스트 정규화

- **주로 replace 메소드를 사용해 표현을 통일**
  - 20 → 2020
  - US → United States

```
sample = "I visited US from UK on 01-08-20"
normalized_sample = sample.replace("US", "United States").replace("UK","United Kingdom").replace("-20", "-2020")
```

```
'I visited United States from United Kingdom on 01-08-2020'
```

철자 수정 (spelling correction) (1/2)

- **autocorrect 모듈 설치**
  - pip install autocorrect

```
from autocorrect import Speller

spell = Speller()
spell("I'm not sleapy and tehre is no place I'm giong to.")
```

```
"I'm not sleepy and there is no place I'm going to."
```

철자 수정 (2/2)

```
import string
nltk.download('punkt')
from autocorrect import Speller

spell = Speller()

corrected_sentence = ""
corrected_words = []

for wd in new_words:
    tmp = wd
    if wd not in string.punctuation:
        wd_c = spell(wd)
        if wd_c != wd:
            print(wd + " has been corrected to: "+ wd_c)
            tmp = wd_c

    corrected_sentence = corrected_sentence + " " + tmp
    corrected_words.append(tmp)
```

```
Sohom has been corrected to: Soom
Ghosh has been corrected to: Those
learnning has been corrected to: learning
pracess has been corrected to: process
Natueral has been corrected to: Natural
prajects has been corrected to: projects
```

어간 추출(Stemmer) (1/3)

- Stem : 어간. fishing, fished, fisher의 어간은 fish
- Stemming : 접미사를 제거하고 어간을 반환

```
from nltk import PorterStemmer, LancasterStemmer, word_tokenize

raw = """
My name is Maximus Decimus Meridius, commander of the Armies of the North,
General of the Felix Legions and loyal servant to the true emperor,
Marcus Aurelius. Father to a murdered son, husband to a murdered wife.
And I will have my vengeance, in this life or the next.
"""
tokens = word_tokenize(raw)

porter = PorterStemmer()
pStems = [porter.stem(t) for t in tokens]
print(pStems)

lancaster = LancasterStemmer()
lStems = [lancaster.stem(t) for t in tokens]
print(lStems)
```

Stemmer (2/3)

Porter Stemmer가 삭제한 접미사는 **‘s’, ‘es’, ‘e’, ‘ed’, ‘al’**
가능한 적게 제거

```
['My', 'name', 'is', 'maximu', 'decimu', 'meridiu', '
', 'gener', 'of', 'the', 'felix', 'legion', 'and',
cu', 'aureliu', ',', 'father', 'to', 'a', 'murder',
d', 'I', 'will', 'have', 'my', 'vengeanc', ',', 'in',
```

Lancaster Stemmer가 삭제한 접미사는 **‘us’, ‘e’, ‘th’, ‘eral’, ‘eded’, …**
- 가능한 많이 제거**(greedy)**

```
['my', 'nam', 'is', 'maxim', 'decim', 'meridi', ',',
n', 'of', 'the', 'felix', 'leg', 'and', 'loy', 'serv
h', 'to', 'a', 'murd', 'son', ',', 'husband', 'to',
g', ',', 'in', 'thi', 'lif', 'or', 'the', 'next', '.
```

Stemmer (3/3)

```
from nltk import PorterStemmer

porter = PorterStemmer()

corrected_words_stemmed = []
for wd in corrected_words:
    corrected_words_stemmed.append(porter.stem(wd))
corrected_words_stemmed[:20]
```

```
['In', 'book', 'authored', 'Soom', 'Those', 'Dwight', 'Gunning', ',', 'shall', 'learning', 'proces
s', 'Natural', 'Language', 'extract', 'insights', '.', 'The', 'first', 'four', 'chapter']
```

```
['In', 'book', 'author', 'soom', 'those', 'dwight', 'gun', ',', 'shall', 'learn', 'process', 'natu
r', 'languag', 'extract', 'insight', '.', 'the', 'first', 'four', 'chapter']
```

표제어 추출 (1/2) : 기본형, 원형 복원

(단어의) 기본형과 원형 복원

- Lemma: 단어의 기본형.
  - 어휘 목록 표제어 (lexicon headword)
  - 사전과 일치(dictionary matching)
- 원형 복원(lemmatization)
  - dictionary matching을 통해 기본형을 찾음
    - 사전에서 결과 단어를 찾을 수 있는 경우에만 접미사를 제거
    - 대문자로 시작하는 단어를 특수 문자로 인식해 어떤 처리도 하지 않음
  - Stemming(어간 추출)보다 속도가 느림
    - Stemming은 접미사를 제거하거나 대체하여 어간을 구함

```
from nltk import WordNetLemmatizer

wordlemma = WordNetLemmatizer()
print(wordlemma.lemmatize('cars'))
print(wordlemma.lemmatize('walking',pos='v'))
print(wordlemma.lemmatize('meeting',pos='n'))
print(wordlemma.lemmatize('meeting',pos='v'))
print(wordlemma.lemmatize('better',pos='a'))
```

```
car
walk
meeting
meet
good
```

WordNetLemmatizer (2/2)

```
from nltk import word_tokenize, PorterStemmer, WordNetLemmatizer

raw = "My name is Maximus Decimus Meridius, commander of the armies of the north
tokens = word_tokenize(raw)

porter = PorterStemmer()
stems = [porter.stem(t) for t in tokens]
print(stems)

lemmatizer = WordNetLemmatizer()
lemmas = [lemmatizer.lemmatize(t) for t in tokens]
print(lemmas)
```

```
['My', 'name', 'is', 'maximu', 'decimu', 'meridiu', ',', 'command', 'of', 'the',
 'armi', 'of', 'the', 'north', ',', 'gener', 'of', 'the', 'felix', 'legion', 'an
d', 'loyal', 'servant', 'to', 'the', 'true', 'emperor', ',', 'marcu', 'aureliu',
 '.', 'father', 'to', 'a', 'murder', 'son', ',', 'husband', 'to', 'a', 'murder',
 'wife', '.', 'and', 'I', 'will', 'have', 'my', 'vengeanc', ',', 'in', 'thi', 'l
ife', 'or', 'the', 'next', '.']
```

```
['My', 'name', 'is', 'Maximus', 'Decimus', 'Meridius', ',', 'commander', 'of', '
the', 'army', 'of', 'the', 'north', ',', 'General', 'of', 'the', 'Felix', 'legio
n', 'and', 'loyal', 'servant', 'to', 'the', 'true', 'emperor', ',', 'Marcus', 'A
urelius', '.', 'Father', 'to', 'a', 'murdered', 'son', ',', 'husband', 'to', 'a'
, 'murdered', 'wife', '.', 'And', 'I', 'will', 'have', 'my', 'vengeance', ',', '
in', 'this', 'life', 'or', 'the', 'next', '.']
```

개체 이름 인식 (NER)

- 여러 개 명사를 묶어(chunking) 하나의 이름으로 인식

```
sentence = "Seoul Botanical Garden is a well known place in Seoul, Korea."

words = nltk.word_tokenize(sentence)
tags = nltk.pos_tag(words)
chunks = nltk.ne_chunk(tags)
print(chunks)
```

```
(S
  (GPE Seoul/NNP)
  (PERSON Botanical/NNP Garden/NNP)
  is/VBZ
  a/DT
  well/RB
  known/VBN
  place/NN
  in/IN
  (GPE Seoul/NNP)
  ,/,
  (GPE Korea/NNP)
  ./.)
```

단어 중의성 해결

- 같은 단어이지만, 문장에 따라 뜻이 다를 수 있다.

```
from nltk.wsd import lesk

sentence1 = "Keep your savings in the bank"
print(lesk(word_tokenize(sentence1), 'bank'))

sentence2 = "It's so risky to drive over the banks of the road"
print(lesk(word_tokenize(sentence2), 'bank'))
```

```
Synset('savings_bank.n.02')
Synset('bank.v.07')
```

은행

도로 경사