In [ ]:
```
N-gram(LM) - 문장생성, 띄어쓰기(숙제 - 과제란)
P(토큰2|토큰1) => LM
Tokenizing - Branch Entropy(-plogp)
토큰1+토큰2(활용)
    *
- Edit Dist.(음절 => 음소)
- Normalization

ABCDEFGHIJKLMNOPQR...
 *
A' 'BCD' ''

1. P(B|A) = P(A,B)/P(A) = freq(A,B)/freq(A)
=> Max of B[-1] == ' '
   P(B|' ')
2. * 이동
```

In [1]:
```
# 0330에 있음
from os import listdir

def fileids(path):
    fileList = list()

    path = path + ('' if path[-1] == '/' else '/')

    for f in listdir(path):
        if f.endswith('.txt'):
            fileList.append(path+f)

    return fileList
```

In [2]:
```
# 0330에 있음
import re

p1 = re.compile('[a-zA-Z0-9_-]+@(?:[.]?[a-zA-Z0-9_-]+)+')
p2 = re.compile('[{}]'.format(re.escape('''"'=·…ⓒ''')))
p3 = re.compile('[\(\[].+?[\)\]]')
p4 = re.compile('[a-zA-Z]+')
p5 = re.compile('\s+')
```

In [3]:
```
# 0330에 있음
corpus = list()

for f in fileids('news'):
    with open(f, 'r', encoding='utf8') as fp:
        corpus.append(p5.sub(' ',
                      p4.sub(' ',
                      p3.sub(' ',
                      p2.sub(' ',
                      p1.sub(' ', fp.read()))))).strip())
```

In [6]:
```
from nltk.tokenize import word_tokenize
from collections import Counter

tokens = Counter(word_tokenize('\n'.join(corpus)))

# '\n'.join(['문서1 내용', '문서2 내용'])
# word_tokenize('문서1 내용\n문서2 내용...')
```

```
# Counter([단어1, 단어2, ... ㅌ문서1, 단어1, 단어2, ... ㅌ문서2])
# tokens = {단어1:빈도, 단어2:빈도, ...}
```

In [8]:
```
from math import log

BE = lambda n1, n2:-(n1/n2)*log(n1/n2) # -p * log p
```

In [13]:
```
tokenList = lambda t:{k:v for k,v in tokens.items() if re.match(t,k)}
# tokens.items() => [(k:단어1,v:빈도), (단어2,빈도), ...]
#                     re.match(^t, k) t:대, k:대~~~~~~~
#                   {[t]단어1:빈도, [t]단어2:빈도, ...}
```

In [63]:
```
from collections import defaultdict

# 대표
q = '테슬라에서'
beList = list()
for i in range(len(q)):
    candidates = defaultdict(int)
    for k,v in tokenList(q[:i+1]).items():
        candidates[k[:len(q[:i+1])+1]] += v
#         i=0, t=>q[:i+1]=1, branch=>len(q[:1])+1=2

    be = 0
    for k,v in candidates.items(): # t+모든 가지수:freq
        e = BE(v, sum(tokenList(q[:i+1]).values()))
        be += e
#         print('{} - {:3f}'.format(k,e))
    beList.append(be)

for i in range(2, len(beList)): # [음절음절]음절(i=2)
    if beList[i-1] < beList[i]: # 2번째 음절 be < 3번째 음절 be
        print(q[:i], 'True')
    elif beList[i-1] > beList[i]: #
        print(q[:i], 'True')
    else:
        print(q[:i], 'False')
beList
# 대: 2.8, 대표: 2.23, 대표는:0
```

```
테슬 True
테슬라 True
테슬라에 False
```

Out[63]:
```
[1.5574745947975033, 0.0, 1.0608569471580214, 0, 0]
```

In [83]:
```
from math import sqrt
# CS = lambda n1, n2, n3:(n1/n2)**(1/n3)
CS = lambda n1, n2, n3:sqrt(n1/n2)
```

In [98]:
```
# Cohesion Score = Perplexity
q = '대통령에게' # 를, 의

n1 = sum(tokenList(q).values()) # [대통령~]으로 시작하는 모든 빈도의 합
n2 = sum(tokenList(q[:1]).values()) # [대~]로 시작하는 모든 빈도의 합
n3 = len(q)

CS(n1, n2, n3), n1, n2, n3
```

Out[98]:
```
(0.07615096238502143, 9, 1552, 5)
```

In [ ]:
```
테슬      (0.5883484054145521, 9, 26, 2)
테슬라    (0.5883484054145521, 9, 26, 3)
---------------------------------------
테슬라를   (0.2773500981126146, 2, 26, 4)
테슬라의   (0.3396831102433787, 3, 26, 4)

대통령    (0.51834888541436, 417, 1552, 3)
---------------------------------------
대통령은  (0.23402572772478114, 85, 1552, 4)
대통령에게 (0.07615096238502143, 9, 1552, 5)

-> 어근, 어간 (핵심되는 단어 추출)
-> 어미, 접사 (불필요한 단어를 제거)
```

In [ ]:
```
Tokenizing이 중요한대 => feature 추출(모델에 이용)
sent/word/Twitter/regex_tokenizer
morpheme analyzer, stemming+lemmatization(in ma)
n-gram(lm), be, cs, bpe   ===> 어떤 token 중요? (zipf)
                          ===> heap's
```

In [ ]:
```
# class Model:
#     def dataloader(self, 어쩌고,...):
#         전처리
SPTokenizer(패키지이름;SP-SentencePiece) - WordPieceModel(Google)
WPM -> BPE(Byte Pair Encoding) ; 통계이용 (가장 많이 사용되는 분절)
```

In [ ]:
```
Huffman Encoding => a-z:26 -> 5bits(32)
                    26*5 => 130bits
                    z->, the -> T
            여러개의 낱자(가장 많이 나오는 쌍을 한 글자로 바꾸자)
            여러개의 토큰(               한 토큰으로 바꾸자)
      BPE 의 목적
```

In [100…
```
'가정폭력', '가', '정', '폭', '력', '가정', '정폭', '폭력', ...
          '가정' + '폭력' => '가정폭력'
WPM = 데이터를 wikipedia -> BPE(threshold) 1000개
```

Out[100]:
```
'가정폭력'
```

In [106…
```
D = {'low':5,'lower':2,'newest':5,'widest':3}

def preprocessing(d):
    result = dict()
    # 쪼개고, 단어의 마지막에 </w> 붙여주고
    for k,v in d.items():
        nk = ' '.join(list(k)+['</w>',]) # ['l', 'o', 'w'] + ['</w>']
        result[nk] = v
    return result

preprocessing(D)
```

Out[106]:
```
{'l o w </w>': 5,
 'l o w e r </w>': 2,
 'n e w e s t </w>': 5,
 'w i d e s t </w>': 3}
```

In [107…
```
d = preprocessing(D)
```

```
pairs = dict()
for k,v in d.items():
    for i in range(len(k.split())-1):
        bigram = ' '.join(k.split()[i:i+2])
        if bigram in pairs:
            pairs[bigram] += v
        else:
            pairs[bigram] = v
```

In [111…

```
bestkey = max(pairs, key=pairs.get) # max(key=> 누구를 기준으로 정렬)
```

In [116…

```
for k,v in d.items():
    nk = re.sub(bestkey, re.sub(' ', '', bestkey), k)
    print({nk:v})
```

```
{'l o w </w>': 5}
{'l o w e r </w>': 2}
{'n e w es t </w>': 5}
{'w i d es t </w>': 3}
```

In [123…

```
# d = preprocessing(D)
d = preprocessing(tokens)

for _ in range(100):
    # 바이그램(쌍) 찾는 부분
    pairs = dict()
    for k,v in d.items():
        for i in range(len(k.split())-1):
            bigram = ' '.join(k.split()[i:i+2])
            if bigram in pairs:
                pairs[bigram] += v
            else:
                pairs[bigram] = v
    # 빈도 가장 높은 쌍 찾는 부분
    bestkey = max(pairs, key=pairs.get)
    # merge하는 부분
    newd = dict()
    for k,v in d.items():
        nk = re.sub(bestkey, re.sub(' ', '', bestkey), k)
        newd[nk] = v
    d = newd
    print(re.sub(' ', '', bestkey))
```

다</w>
.</w>
니다</w>
에.</w>
습니다</w>
으.</w>
있습니다</w>
하.</w>
세.</w>
닫.</w>
다.
가.</w>
이.</w>
했다</w>
뉴스
기사
언론
언론사
구독
시.</w>
합니다</w>
로.</w>
인에.</w>
까.</w>
지.</w>
보세.</w>
메인에.</w>
라.</w>
해보세.</w>
추천
분류
섹션
섹션으.</w>
섹.</w>
추천합니다</w>
언론사에.</w>
기.</w>
주요
보러
보러가.</w>
보.</w>
주요뉴스
정.</w>
되었
해.</w>
되었습니다</w>
있다</w>
20
일.</w>
리.</w>
부.</w>
50
진.</w>
확인
했습니다</w>
페이.</w>
경.</w>
들.</w>
24
사.</w>
도.</w>

```
됩니다</w>
배.</w>
인.</w>
스.</w>
니다.
백.</w>
강.</w>
주세.</w>
활용
자.</w>
분석
바로
24시.</w>
바로가.</w>
공감
모두
서.</w>
대.</w>
확인해보세.</w>
흥미
후속
눌러
분류했습니다</w>
분류.</w>
구독해보세.</w>
50회
50회까.</w>
쏠쏠
쏠쏠정.</w>
흥미진.</w>
공감백.</w>
분석탁
분석탁.</w>
후속강.</w>
모두에.</w>
기사라.</w>
눌러주세.</w>
기사배.</w>
활용됩니다</w>
```

In [125…
```
from string import punctuation
```

In [145…
```
s = '''I'd like to learn more somthing.
       I'd like to learn more somthing.
       state-of-the-art'''
```

In [146…
```
len(set(word_tokenize(s.lower(), preserve_line=False))), \
list(set(word_tokenize(s.lower(), preserve_line=False)))
```

Out[146]:
```
(9,
 ['learn',
  'somthing',
  'i',
  'to',
  "'d",
  'more',
  'like',
  'state-of-the-art',
  '.'])
```

In [147…
```
list(set(re.sub('[{}]'.format(re.escape(punctuation)),
           ' ', s.lower()).split()))
```

Out[147]:
```
['learn',
 'state',
 'd',
 'somthing',
 'i',
 'to',
 'of',
 'the',
 'more',
 'like',
 'art']
```