# 지도학습 기반, G&I 문장 경계선 구분하기 위한 코드/코퍼스 작성

1829008 김민영

# 목차

# 1. Preprocessing

먼저, gniCorpus를 읽어와 엔터키로 split해 gniRaws를 살펴보면 다음과 같다.

```
In [3]: corpus_root = "C:/Users/KimMinyoung/Downloads/sentence_tokenized"
        gniCorpus = nltk.corpus.PlaintextCorpusReader(corpus_root, ".*₩.txt", encoding="utf-8")

In [4]: gniRaws = gniCorpus.raw().split('₩n') #엔터키로 split
        gniRaws[:10]

Out[4]: ['Title: Survey of the Applications of NGS to Whole-Genome Sequencing and Expression Profiling',
         '',
         'Recently, the technologies of DNA sequence variation and gene expression profiling have been used widely as
        approaches in the expertise of genome biology and genetics.',
         '',
         'The application to genome study has been particularly developed with the introduction of the next-generation
        DNA sequencer (NGS) Roche/454 and Illumina/Solexa systems, along with bioinformation analysis technologies of
        whole-genome de novo assembly, expression profiling, DNA variation discovery, and genotyping.',
         '',
         'Both massive whole-genome shotgun paired-end sequencing and mate paired-end sequencing data are important st
        eps for constructing de novo assembly of novel genome sequencing data.',
         '',
         'It is necessary to have DNA sequence information from a multiplatform NGS with at least 2× and 30× depth s
        equence of genome coverage using Roche/454 and Illumina/Solexa, respectively, for effective an way of de novo
        assembly.',
         '']
```

# 1. Preprocessing

gniRaw를 살펴보면, 빈 리스트가 포함되어 있는 것을 확인할 수 있다. 이는 다음페이지와 같이 제거 할 수 있다.

```
In [5]: gniRaws = [nltk.word_tokenize(x) for x in gniRaws]
        gniRaws[:10]

Out[5]: [['Title',
          ':',
          'Survey',
          'of',
          'the',
          'Applications',
          'of',
          'NGS',
          'to',
          'Whole-Genome',
          'Sequencing',
          'and',
          'Expression',
          'Profiling'],
         [],
         ['Recently',
          ',',
          'the',
          'technologies',
          'of',
          'DNA',
          'sequence',
          'variation',
          'and',
          'gene',
          'expression',
          'profiling',
          'have',
```

# 1. Preprocessing

if len(x)>2를 사용하여, 2개 이하인 줄 단위 문장들을 버리면, 앞의 문제를 해결할 수 있다.

```
In [29]: gniSents = [x for x in gniRaws if len(x) > 2] # 임의로 1개 이하인 줄 단위 문장들은 버림
         gniSents[:10]

Out[29]: [['Title',
           ':',
           'Survey',
           'of',
           'the',
           'Applications',
           'of',
           'NGS',
           'to',
           'Whole-Genome',
           'Sequencing',
           'and',
           'Expression',
           'Profiling'],
          ['Recently',
           ',',
           'the',
           'technologies',
           'of',
           'DNA',
           'sequence',
           'variation',
           'and',
           'gene',
           'expression',
           'profiling',
           'have',
           'been',
```

# 1. Preprocessing

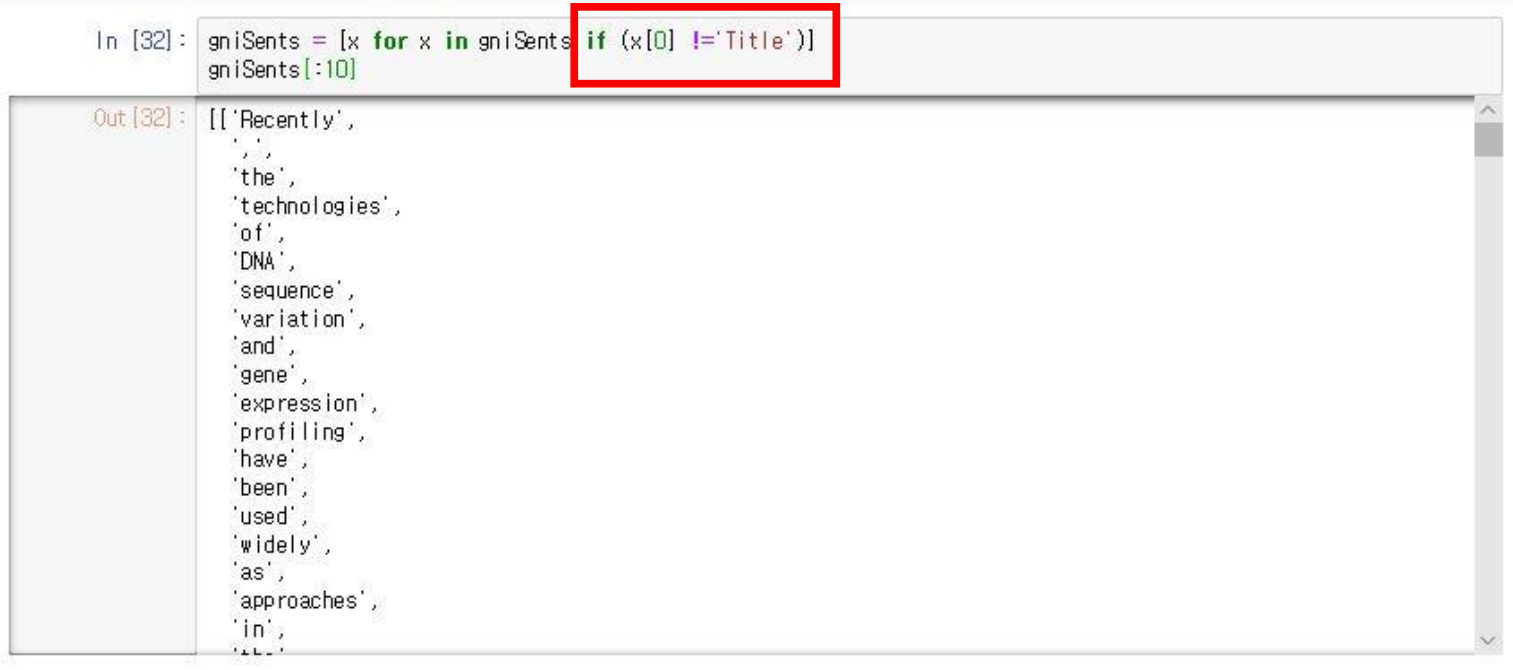
또한, Title 문장들이 들어가 있는데 이를 다음페이지와 같이 제거할 수 있다.

```
In [30]: gniTitle = [x for x in gniSents if (x[0] =='Title')]
         gniTitle[:10]

Out[30]: [['Title',
           ':',
           'Survey',
           'of',
           'the',
           'Applications',
           'of',
           'NGS',
           'to',
           'Whole-Genome',
           'Sequencing',
           'and',
           'Expression',
           'Profiling'],
          ['Title', ':', 'Roche/454', 'pyrosequencing', 'technology'],
          ['Title', ':', 'Illumina/Solexa', 'technology'],
          ['Title', ':', 'Novel', 'whole', 'genome', 'de', 'novo', 'assembly'],
          ['Title',
           ':',
           'SNP',
```

# 1. Preprocessing

Title 문장을 제거한 결과이다. 잘 제거가 된 것을 확인할 수 있다.



In [32]:
```
gniSents = [x for x in gniSents if (x[0] !='Title')]
gniSents[:10]
```

Out[32]:
```
[['Recently',
  ',',
  'the',
  'technologies',
  'of',
  'DNA',
  'sequence',
  'variation',
  'and',
  'gene',
  'expression',
  'profiling',
  'have',
  'been',
  'used',
  'widely',
  'as',
  'approaches',
  'in',
```

# 2. training

punc_features 함수

다음은 punc_features 함수이다.

```
In [52]: def punct_features(tokens, i):
             return {'next-word-capitalized': tokens[i+1][0].isupper(),
                     'prev-word': tokens[i-1].lower(),
                     'punct': tokens[i],
                     'prev-word-is-one-char': len(tokens[i-1]) == 1}
```

punc_features 함수는 아래 네가지를 반환한다.

next-word-capitalized : 다음 단어가 대문자인지
prev-word : 이전 단어
punct : punctuation mark
prev-word-is-one-char : 이전 단어가 1글자 인지

# 2. train & test

1) gni corpus

boundaries -

```
In [173]: tokens = []
          boundaries = set()
          offset = 0
          for sent in gniSents:
                  tokens.extend(sent)
                  offset += len(sent)
                  boundaries.add(offset-1)

In [174]: boundaries

Out[174]: {131072,
           655360,
           393218,
           524296,
           131092,
           786456,
           26,
           393242,
           786466,
```

featuresets -

```
In [175]: featuresets = [(punct_features(tokens, i), (i in boundaries_set))
                  for i in range(1, len(tokens)-1)
                          if tokens[i] in '.?!']

In [176]: featuresets[:10]

Out[176]: [({'next-word-capitalized': True,
             'prev-word': 'genetics',
             'punct': ',',
             'prev-word-is-one-char': False},
            True),
           ({'next-word-capitalized': True,
             'prev-word': 'genotyping',
             'punct': ',',
             'prev-word-is-one-char': False},
            True),
           ({'next-word-capitalized': True,
             'prev-word': 'data',
             'punct': ',',
             'prev-word-is-one-char': False},
            True),
```

# 2. training

2) brown corpus

boundaries -

```
In [168]: brownSents = [x for x in brownSents if len(x) > 2]
          from nltk.corpus import brown
          brownSents = brown.sents()
          tokens_brown = []
          boundaries_brown = set()
          offset = 0
          for sent in brownSents:
                  tokens_brown.extend(sent)
                  offset += len(sent)
                  boundaries_brown.add(offset-1)

In [51]: boundaries_brown

Out[51]: {786444,
          917517,
          131086,
          262160,
          524305,
          655380,
```

featuresets -

```
In [169]: featuresets_brown = [(punct_features(tokens_brown, i), (i in boundaries_brown))
                  for i in range(1, len(tokens_brown)-1)
                          if tokens_brown[i] in '.?!']

In [55]: featuresets_brown[:10]

Out[55]: [({'next-word-capitalized': True,
            'prev-word': 'place',
            'punct': '.',
            'prev-word-is-one-char': False},
           True),
          ({'next-word-capitalized': True,
            'prev-word': 'conducted',
            'punct': '.',
            'prev-word-is-one-char': False},
           True),
```

# 2. training

accuaracy

- gniCorpus

```
In [85]: size = int(len(featuresets) * 0.1)
         train_set, test_set = featuresets[size:], featuresets[:size]
         classifier1 = nltk.NaiveBayesClassifier.train(train_set)
         nltk.classify.accuracy(classifier1, test_set)
         print("Sentence Segmentation:", nltk.classify.accuracy(classifier1, test_set))

Sentence Segmentation: 0.9828516989520483
```

gniCorpus에 대해, 9:1로 train set, test set을 분리하여 학습시키고 측정한 Accuracy는 약 98.3%이다.

- brownCorpus

```
In [170]: train_set, test_set = featuresets_brown, featuresets
          classifier2 = nltk.NaiveBayesClassifier.train(train_set)
          nltk.classify.accuracy(classifier2, test_set)
          print("Sentence Segmentation:", nltk.classify.accuracy(classifier2, test_set))

Sentence Segmentation: 0.9802813323595719
```

brownCorpus를 train set으로 하고, gniCorpus를 test set으로 하여 학습시키고 측정한 Accuracy는 약 98.0%이다.

-> gniCorpus를 1:1로 분리하여 학습시킨 정확도가 0.2%정도 더 높은 것을 확인할 수 있다.

# 3. 결과 해석

show_most_informative_features() 함수

gniCorpus -

```
In [94]: classifier1.show_most_informative_features(10000)
Most Informative Features
   next-word-capitalized = True              True : False  =    359.4 : 1.0
              prev-word = 'eq'               False : True  =    100.6 : 1.0
              prev-word = ')'                True : False  =     47.1 : 1.0
   next-word-capitalized = False             False : True  =     31.0 : 1.0
              prev-word = 'data'             True : False  =     18.9 : 1.0
                  punct = '?'                False : True  =     16.6 : 1.0
   prev-word-is-one-char = True              True : False  =     14.3 : 1.0
              prev-word = 'disequilibrium'   False : True  =     10.7 : 1.0
              prev-word = 'retrieval'        False : True  =     10.7 : 1.0
              prev-word = 'patients'         True : False  =      7.7 : 1.0
              prev-word = 'body'             False : True  =      6.4 : 1.0
              prev-word = 'harm'             False : True  =      6.4 : 1.0
              prev-word = 'etc'              False : True  =      4.9 : 1.0
              prev-word = 'chip'             False : True  =      4.6 : 1.0
              prev-word = 'descriptor'       False : True  =      4.6 : 1.0
              prev-word = 'x'                False : True  =      4.6 : 1.0
              prev-word = 'u'                False : True  =      4.6 : 1.0
              prev-word = 'database'         True : False  =      4.6 : 1.0
              prev-word = 'methods'          True : False  =      4.6 : 1.0
              prev-word = 'region'           True : False  =      3.5 : 1.0
```

brownCorpus -

```
In [171]: classifier2.show_most_informative_features(10000)
Most Informative Features
              prev-word = 'what'             False : True  =     21.9 : 1.0
              prev-word = ')'                True : False  =     17.7 : 1.0
                  punct = '?'                False : True  =     16.7 : 1.0
                  punct = '!'                False : True  =     16.7 : 1.0
              prev-word = 'huh'              False : True  =     14.3 : 1.0
   prev-word-is-one-char = True              True : False  =     11.3 : 1.0
   next-word-capitalized = False             False : True  =     10.7 : 1.0
              prev-word = 'pure'             False : True  =     10.2 : 1.0
              prev-word = 'turtle'           False : True  =     10.2 : 1.0
              prev-word = 'she'              False : True  =     10.2 : 1.0
              prev-word = 'ada'              False : True  =     10.2 : 1.0
              prev-word = 'stupid'           False : True  =     10.2 : 1.0
              prev-word = 'clause'           False : True  =     10.2 : 1.0
              prev-word = 'productivity'     False : True  =     10.2 : 1.0
              prev-word = 'morale'           False : True  =     10.2 : 1.0
              prev-word = 'consulted'        False : True  =     10.2 : 1.0
              prev-word = 'paredon'          False : True  =     10.2 : 1.0
              prev-word = 'slater'           False : True  =     10.2 : 1.0
```

# 3. 결과 해석

show_most_informative_features()의 의미

- False : True 중, False 의 비율이 높은 것은 문장 경계로 구분되지 않을 가능성이 높은 경우이다.

- False : True 중, True의 비율이 높은 것은 문장 경계로 구분될 가능성이 높은 경우이다.

# 3. 결과 해석

1) 다음 단어가 대문자로 시작하는 경우, 문장으로 인식될 가능성이 높으며, 다음 단어가 대문자로 시작하지 않는 경우, 문장이 아닐 가능성이 높다.

gni Corpus 사례
next-word-capitalized = True  True : False = 359.4 : 1.0
next-word-capitalized = False  False : True = 31.0 : 1.0

brown Corpus 사례
next-word-capitalized = False  False : True = 10.7 : 1.0

2) ')' 로 끝나는 경우, 문장으로 인식될 가능성이 높았다. 이는 gni Corpus에서 (Fig.2)등 추가적으로 설명하는 사례가 많았기 때문이 확률이 좀더 높았다.

gni Corpus 사례
prev-word = ')' True : False = 47.1 : 1.0

brown 사례
prev-word = ')' True : False = 17.7 : 1.0

# 3. 결과 해석

3) 이전 단어가 한 단어로 끝나는 경우 문장 경계로 구분되지 않을 확률이 높다.

gni Corpus 사례
prev-word = ' x ' False : True = 4.6 : 1.0
prev-word = ' u ' False : True = 4.6 : 1.0

brown Corpus 사례
prev-word = 'i' False : True = 8.8 : 1.0

-> 이는 수식이 많이 포함된 gni Corpus와 일상문이 많은 brown Corpus 사이에 약간의 차이가 존재하긴 하다는 것을 보여준다.

# 3. 결과 해석

4) etc로 끝나는 경우, 이후 문장 경계로 구분되지 않을 확률이 높다.

GNI corpus 사례
prev-word = 'etc' False : True = 4.9 : 1.0

brown corpus 사례
prev-word = 'etc.' True : False = 3.7 : 1.0

5) 물음표(?)로 끝나는 경우, 이후 문장 경계로 구분되지 않을 확률이 높다.

brown corpus 사례
punct = ' ? ' False : True = 16.6 : 1.0

GNI corpus 사례
punct = '?' False : True = 16.6 : 1.0

* show_most_informative_features 함수를 통해 그 외 여러가지 특징들을 살펴볼 수 있었다.

# 4. 확인

gni-10-2-81.txt

원본 pdf)

and all CHB + JPT populations are outlying gain or loss status. The next decisive outlier is CNVR371.1 (chr. 1), in which only the YRI population has non-neutral CNVs.

sentence_tokenize txt)

The next decisive outlier is CNVR371.1 (chr.
1), in which only the YRI population has non-neutral CNVs.

-> 문제점 : (chr.1)에서, chr. 에 문장분리가 되어버렸다. 이는 약어사전을 만들 때 누락된 것으로 보인다. chr.를 추가하면 해결가능할 것 같다.

# 4. 확인

gni-10-2-81.txt

For a set of I loci, the multilocus ANOVA estimators are - 문장1

for nc = (S1-S2/S1)/(n-1), where S1 is the total sample size and S2 is the sum of squared sample sizes of populations [21]. - 문장2

a weighted-average $F_{ST}$ from all SNPs is estimated for each gene [18]. For a set of $I$ loci, the multilocus ANOVA estimators are

$$\hat{F}_{IS} = \frac{\sum_{i=1}^{I}[n_c(MSI - MSG)]_i}{\sum_{i=1}^{I}[n_c(MSI + MSG)]_i},$$

$$\hat{F}_{ST} = \frac{\sum_{i=1}^{I}[MSP - MSI]_i}{\sum_{i=1}^{I}[MSP + (n_c - 1)MSI + n_c MSG)]_i},$$

$$\hat{F}_{IT} = \frac{\sum_{i=1}^{I}[MSP + (n_c - 1)MSI - n_c MSG)]_i}{\sum_{i=1}^{I}[MSP + (n_c - 1)MSI + n_c MSG)]_i}, \quad (2)$$

for $n_c = (S_1 - S_2/S_1)/(n-1)$, where $S_1$ is the total sample size and $S_2$ is the sum of squared sample sizes of populations [21]. For convenience, we denote the estimator $\hat{F}_{ST}$ by $F_{ST}$.

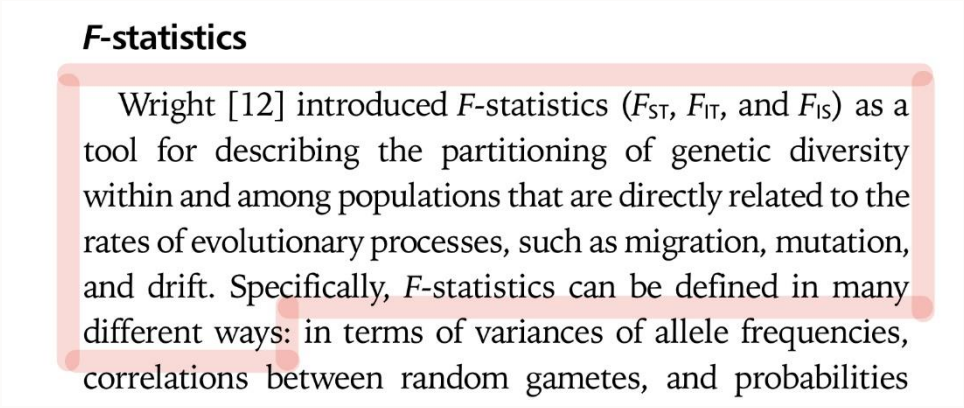
Rousset [21] explained that the multilocus estimators of

-> For ... are~ 과 ~for 사이에 수식 값이 있었어야하는데, 중간에 생략되어 한 문장이 두 문장으로 짤려있는 경우도 있었다.

# 4. 확인

gni-10-2-81.txt

Title: Wright [12] introduced F-statistics (FST, FIT, and FIS) as a tool for describing the partitioning of genetic diversity within and among populations that are directly related to the rates of evolutionary processes, such as migration, mutation, and drift.



***F*-statistics**

Wright [12] introduced *F*-statistics ($F_{ST}$, $F_{IT}$, and $F_{IS}$) as a tool for describing the partitioning of genetic diversity within and among populations that are directly related to the rates of evolutionary processes, such as migration, mutation, and drift. Specifically, *F*-statistics can be defined in many different ways: in terms of variances of allele frequencies, correlations between random gametes, and probabilities

-> 문제 : Title이 아닌데, Title 이라고 잘못 들어가있는 경우가 있는데 이는 Title 제거하는 과정에서 함께 사라져 누락되는 문제가 발생하기도 하였다.

# 4. 확인

gni-10-2-81.txt

Disease could affect the patterns of CNV diversity via natural selection, and higher copy numbers of the immunoregulatory and inflammatory cytokine gene CCL3L1, for example, are associated with lower risks of HIV infection and the progression to AIDS [11]; furthermore, in a genome-wide study, the level of population differentiation at this locus was found to be extraordinary compared to that of other CNVs, suggesting that natural selection may have influenced CCL3L1 copy number in humans [3].

-> ; 로 분리되어있는 경우 다른 문장으로 분리되지 않는 애매한 문제도 있었음

# 4. 확인

gni-10-2-81.txt

Following the work of Cockerham [16], F-statistics are defined in terms of the variance components - that is, the total variation in the genetic data is broken down into three components: (a) between subpopulations within the total population (we sometimes say 'between populations'); (b) between individuals within subpopulations; and (c) between gametes within individuals.

-> 이런 애매한 경우도 존재했다.

# 4. 확인

gni-10-2-106.txt

Pulse rate is known to be positively correlated with triglycerides and negatively correlated with high-density lipoprotein cholesterol [3].

-> gni-10-2-106.txt에서는 뒤에 reference 번호가 들어가 있는 것 이외에는 특별한 문제점이 보이지 않았다.