2018년 네이버 NLP Challenge 개체명 인식 데이터를 활용한 분류 모형 제작

20171218 김다빈

1. 데이터 불러오기

파일 'entity_data_utf8.txt'는 line 단위로 어절 정보를 명시하고, 좌측 번호로 문장 내 위치를 표기한 텍스트 데이터다. 필자는 입력-출력 쌍이 명시된 훈련 데이터를 확보하기 위해 해당 파일을 하나의 문자열로 불러왔으며, re 정규표현 모듈의 함수를 활용해 tagged_words 형식으로 데이터로 변환하였다. 이러한 과정으로 얻은 tagged_words 리스트는 다음 사진자료와 같이 어절과 태그로 이루어진 tuple 원소로 구성되었다. 해당 데이터는 random 모듈의 shuffle 함수를 적용해 어절 데이터 순서를 무작위로 뒤섞었으며, 이를 통해 데이터 학습과정 중 표본편향의 오류가 발생할 위험을 미리 방지하였다.

```
import nltk, re, os

# load data & make 'tagged_words'
data = open('./entity_data_utf8.txt', encoding='utf8').read()
tagged_words = [nltk.tag.str2tuple(line, sep='\t')
                for line in re.findall('^\d+\t(.*$)', data, re.M)]

# random shuffle tagged_words
import random
random.shuffle(tagged_words)
```

- 문자코드 UTF-8의 텍스트 파일을 불러오기 위해 open 함수의 논항에 encoding='utf8'을 입력하였으며, 하나의 문자열로 데이터를 확보하기 위해 read 메서드를 사용하였다.
- re 모듈의 findall 함수를 사용하여 각 line의 우측에 위치한 '어절\t태그' 꼴의 텍스트를 모두 추출하였고, str2tuple 함수를 이용해 (어절, 태그) 꼴의 원소로 변환하였다.
- random 모듈의 shuffle 함수를 활용하여 tagged_words의 원소 순서를 무작위로 재배치하였다.

radom.shuffle 이전

```
print(tagged_words[:15])
```

```
[('비토리오', 'PER'), ('양일', 'DAT'), ('만에', '-'), ('영사관', 'ORG'), ('감호', 'CVL'), ('용퇴,', '-'), ('항룡', '-'), ('압력설', '-'), ('의심만', '-'), ('가율', '-'), ('이', '-'), ('음경동맥의', '-'), ('직경이', '-'), ('8', 'NUM'), ('19mm입니다', 'NUM')]
```

random.shuffle 이후

```
print(tagged_words[:15])
```

```
[('30m', 'NUM'), ('팀에는', 'ORG'), ('데포르마시옹될', '-'), ('강한', '-'), ('구성하고', '-'), ('앓고', '-'), ('도마경기로', 'CVL'), ('우리의', '-'), ('일은', '-'), ('란초미라지', 'LOC'), ('연구원', '-'), ('일부분에서', '-'), ('그', '-'), ('커졌다', '-'), ('헤브론', 'LOC')]
```

2. Feature Extraction A : 길이 - 어절

첫번째로 추출한 feature는 어절의 길이다. 어절의 길이는 어절이 내포하는 구체적인 내용과 직접적으로 연관되지는 않지만, 각 어절에 존재하는 어휘유형의 특성을 반영할 수 있다. 예를 들어 '시간(TIM)' 이나 '날짜(DAR)'와 관련된 어휘는 '하루', '이틀'과 같이 짧은 길이로 이루어져 있지만, '의학용어/IT용어 등 일반용어(TRM)'는 비교적 긴 길이로 나타난다. 이러한 특성을 고려하여 필자는 각 어절의 길이 값을 추출한 'word_len' feature를 추가하였으며, 구현 코드와 결과는 다음과 같다.

```
# feature extraction A : Word Length

def word_features(word):
    # (조건4) 어절의 길이 추출
    return {'word_len': len(word)}
```

해당 함수는 앞서 구현한 tagged_words 리스트 원소의 어절 정보에 적용하여 featuresets 데이터를 만들었다. 또한 추후 기계학습 모형에 대한 테스트 과정에서 학습 데이터의 어절정보가 중복되어 활용되는 것을 방지하기 위해 featuresets를 둘로 분리하였다. featuresets의 90%는 train_set, 10%는 test_set으로 할당하였으며, 전자의 데이터로는 Naive Bayes Classifier 지도 기계학습을, 후자의 데이터로는 classifier 에 대한 accuracy(정확도) 평가를 진행했다. 이와 관련된 코드는 다음과 같다.

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
```

```
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
nltk.classify.accuracy(classifier, test_set)
```

```
0.6948231444743226
```

```
classifier.show_most_informative_features(15)
```

```
Most Informative Features
                word_len = 19               TRM : -      =   6918.8 : 1.0
                word_len = 22               TRM : -      =   3382.1 : 1.0
                word_len = 18               TRM : -      =   3108.9 : 1.0
                word_len = 23               TRM : -      =   2033.7 : 1.0
                word_len = 20               TRM : -      =   1989.7 : 1.0
                word_len = 21               TRM : -      =   1828.5 : 1.0
                word_len = 13               MAT : -      =    413.5 : 1.0
                word_len = 14               EVT : -      =    208.6 : 1.0
                word_len = 24               ORG : NUM    =    204.8 : 1.0
                word_len = 12               ORG : -      =    204.0 : 1.0
                word_len = 43               FLD : -      =    184.6 : 1.0
                word_len = 29               TRM : -      =    155.5 : 1.0
                word_len = 11               ORG : -      =    151.4 : 1.0
                word_len = 17               ORG : -      =    125.1 : 1.0
                word_len = 15               MAT : -      =    121.9 : 1.0
```

어절의 길이 feature로만 구현한 classifier의 정확도는 약 0.695로, 70%에 가까운 수치가 출력되었다. 영향력이 높은 상위 15개의 features를 확인했을 때, 문자길이 20 전후의 어절은 주로 '의학용어/IT용어 등 일반용어(TRM)'로 나타났다. 또한 '기관 및 단체 회의/회담(ORG)' 내용을 나타내는 어절는 주로

10 개 이상의 문자로 이루어져 있음을 추측할 수 있다. 이처럼 'word_len' 데이터는 아주 간단한 어절 정보임에도 불구하고 높은 정확도를 보여주며, 다른 features와 함께 작동할 때에도 강한 영향력을 나타내 첫번째 feature로 선정하였다.

3. Feature Extraction B : 한국어 자음모음 분리

다음으로 추출한 feature는 어절의 첫번째 문자 초성(자음)이다. 'entity_data_utf8.txt' 데이터를 확인한 결과, 일반적으로 어휘는 어절의 앞부분에, 문법 형태소는 어절의 뒷부분에 나타난다는 것을 확인하였다. 따라서 자소 정보는 어절의 첫번째 글자에서 추출하는 것이 가장 효과적일 것이라고 추측했으며, 앞서 확보한 'word_len' feature과 함께 초성/중성/종성의 feature extraction 결과를 비교 확인하였다.

```
def jaso(a):
    initial = ['ㄱ', 'ㄲ', 'ㄴ', 'ㄷ', 'ㄸ', 'ㄹ', 'ㅁ',
               'ㅂ', 'ㅃ', 'ㅅ', 'ㅆ', 'ㅇ', 'ㅈ', 'ㅉ',
               'ㅊ', 'ㅋ', 'ㅌ', 'ㅍ', 'ㅎ']
    midial = ['ㅏ', 'ㅐ', 'ㅑ', 'ㅒ', 'ㅓ', 'ㅔ', 'ㅕ',
              'ㅖ', 'ㅗ', 'ㅘ', 'ㅙ', 'ㅚ', 'ㅛ', 'ㅜ',
              'ㅝ', 'ㅞ', 'ㅟ', 'ㅠ', 'ㅡ', 'ㅢ', 'ㅣ']
    final = ['', 'ㄱ','ㄲ', 'ㄳ', 'ㄴ', 'ㄵ', 'ㄶ', 'ㄷ',
             'ㄹ', 'ㄺ', 'ㄻ', 'ㄼ', 'ㄽ', 'ㄾ', 'ㄿ', 'ㅀ',
             'ㅁ', 'ㅂ', 'ㅄ', 'ㅅ', 'ㅆ', 'ㅇ', 'ㅈ', 'ㅊ',
             'ㅋ', 'ㅌ', 'ㅍ', 'ㅎ']

    Chr_ord = ord(a) - 44032
    ini_index = Chr_ord // (21 * 28)
    mid_index = (Chr_ord // 28) % 21
    final_index = Chr_ord % 28
    return [initial[ini_index], midial[mid_index], final[final_index]]
```

- 한국에 자음모음 분리작업을 위해 강의자료로 공유받은 jaso1 함수를 그대로 사용하였다.

feature extraction B : Word Length, First Letter Jaso

1) 초성 추출 결과

```
def word_features(word):
    # (조건4) 어절의 길이 추출
    features = {'word_len': len(word)}

    # (조건3) 어절의 첫번째 문자 초성 추출
    if re.match('[가-힣]', word[0]):
        features['first_letter_jaso'] = jaso(word[0])[0]
    else:
        features['first_letter_jaso'] = '<NOTKOR>'

    return features
```

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
nltk.classify.accuracy(classifier, test_set)
```

```
0.7029654562891366
```

2) 중성 추출 결과

```
def word_features(word):
    # (조건4) 어절의 길이 추출
    features = {'word_len': len(word)}

    # (조건3) 어절의 첫번째 문자 중성 추출
    if re.match('[가-힣]', word[0]):
        features['first_letter_jaso'] = jaso(word[0])[1]
    else:
        features['first_letter_jaso'] = '<NOTKOR>'

    return features
```

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
nltk.classify.accuracy(classifier, test_set)
```

```
0.6989695180428365
```

3) 종성 추출 결과

```
def word_features(word):
    # (조건4) 어절의 길이 추출
    features = {'word_len': len(word)}

    # (조건3) 어절의 첫번째 문자 종성 추출
    if re.match('[가-힣]', word[0]):
        features['first_letter_jaso'] = jaso(word[0])[2]
    else:
        features['first_letter_jaso'] = '<NOTKOR>'

    return features
```

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
```

```
nltk.classify.accuracy(classifier, test_set)
```

```
0.6987814738900694
```

위 세가지 코드의 결과를 확인했을 때, 기계학습모델의 정확도를 상승시키는 실질적인 feature는 음절의 첫번째 문자 초성이다. word_len과 초성정보를 담은 first_letter_jaso 정보를 학습한 classifier에 대해 show_most_informative_features 메서드를 적용한 결과는 다음과 같다.

```
classifier.show_most_informative_features(10)
```

```
Most Informative Features
                word_len = 19              TRM : -      =   5020.7 : 1.0
                word_len = 22              TRM : -      =   3385.3 : 1.0
                word_len = 18              TRM : -      =   2277.1 : 1.0
                word_len = 23              TRM : -      =   1965.3 : 1.0
                word_len = 21              TRM : -      =   1848.2 : 1.0
                word_len = 20              TRM : -      =   1401.1 : 1.0
       first_letter_jaso = 'ㄲ'             PLT : DAT    =    900.9 : 1.0
       first_letter_jaso = 'ㅋ'             FLD : DAT    =    681.5 : 1.0
       first_letter_jaso = 'ㅃ'             PLT : NUM    =    351.0 : 1.0
                word_len = 13              EVT : -      =    307.3 : 1.0
```

7번째, 9번째로 큰 영향력을 미치는 첫번째 문자 초성 'ㄲ'와 'ㅃ'는 '꽃', '뿌리' 등의 식물 관련 어휘(PLT)의 어간부 특성을 포착하고 있다. 또한 초성 'ㅋ'는 '클래식', '코미디' 등 학문분야의 어휘정보(FLD)에 빈번히 사용되는 문자유형을 반영한다. 이는 직관만으로 판단하기는 어려운 정보로, word_features 함수의 두번째 추출 feature로 설정하였다.

4. Feature Extraction C : N-gram - 음절

마지막으로 추출한 feature는 음절에 대한 N-gram 데이터이다. 이번 지도 기계학습 과정에서 어절 단위 데이터를 통해 학습하는 내용은 '어절 (혹은 어휘)의 내용 분류' 태그이므로, 그 무엇보다도 어절에 드러난 어간 내용을 파악하는 것이 중요하다. 따라서 'entity_data_utf8.txt'에 나타나는 모든 어절의 시작지점에 대해 uni-gram, bi-gram, tri-gram 음절의 어근 정보를 추출하였으며, nltk.FreqDist 를 활용하여 고빈도 20000개의 어근 리스트(common_roots)를 만들었다.
추가적으로 특정 내용의 어휘에 자주 함께 사용되는 문법 형태소가 존재할 것이라고 추측하여 고빈도 2000개의 조사 리스트(common_postpos)를 생성하였다. 어간부만큼 어휘내용에 직접적인 관련성이 없을 것이라고 판단하여 어근의 1/10 크기 데이터를 활용했지만, 인물명은 '이', '가', '에게', 시간어휘는 '까지', '부터' 등 특정한 조사가 사용될 것이라고 추측하여 uni-gram , bi-gram, tri-gram 음절의 조사 정보를 추출하였다.
word_features 함수에서는 각 어절이 고빈도 20000개 어근 리스트의 원소로 시작될 경우 'root' feature 항목을 추가하여 해당 어근 데이터를 부여했다. 또한 고빈도 2000개 조사 리스트의 원소로 끝나는 경우 'postpos' feature 항목을 추가하여 접사 데이터를 입력했다. 앞서 살펴본 과정과 마찬가지로 완성한 word_features 함수는 어절 단위의 featuresets을 만드는 데에 활용하였으며, 90%는 train_set 데이터로, 10%는 test_set 데이터로 할당하였다. train_set 을 학습한 classifier에 대해 정확도를 확인한 결과, 약 0.86의 수치가 산출되었다.

고빈도 20000개 어근 리스트 & 고빈도 2000개 조사 리스트 생성

```
words = re.findall('^\d+\t(\S*)', data, re.M)

roots_fdist = nltk.FreqDist()
postpos_fdist = nltk.FreqDist()

for word in words:
    # roots
    if len(word) > 0:
        roots_fdist[word[:1]] += 1
    if len(word) > 1:
        roots_fdist[word[:2]] += 1
    if len(word) > 2:
        roots_fdist[word[:3]] += 1

    # postpos
    if len(word) > 0:
        postpos_fdist[word[-1:]] += 1
    if len(word) > 1:
        postpos_fdist[word[-2:]] += 1
    if len(word) > 2:
        postpos_fdist[word[-3:]] += 1

common_roots = list(roots_fdist)[:20000]
common_postpos = list(postpos_fdist)[:2000]
```

feature extraction C : Word Length, First Letter Jaso, N-gram Roots & N-gram Postpositions

```
def word_features(word):
    # (조건4) 어절의 길이 추출
    features = {'word_len': len(word)}


    # (조건3) 어절의 첫번째 문자 초성 추출
    if re.match('[가-힣]', word[0]):
        features['first_letter_jaso'] = jaso(word[0])[0]
    else:
        features['first_letter_jaso'] = '<NOTKOR>'

    # (조건1) 고빈도 20000개 어근부, 고빈도 2000개 조사부 N-gram 포함 여부
    for root in common_roots:
        if word.startswith(root):
            features['root'] = root
    for postpos in common_postpos:
        if word.endswith(postpos):
            features['postpos'] = postpos

    return features
```

accuracy & show most informative features

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

```
0.8612704262960943
```

```
classifier.show_most_informative_features(20)
```

```
Most Informative Features
                  postpos = '상오'              TIM : -      =  12188.6 : 1.0
                  postpos = '올림픽'            EVT : -      =   7496.4 : 1.0
                  postpos = '신문,'             PER : -      =   5070.1 : 1.0
                 word_len = 19                  TRM : -      =   5038.1 : 1.0
                  postpos = '6일'               DAT : -      =   4773.5 : 1.0
                  postpos = '8일'               DAT : -      =   4690.1 : 1.0
                  postpos = '4년'               DAT : -      =   4662.3 : 1.0
                  postpos = '1일'               DAT : -      =   4606.7 : 1.0
                  postpos = '9일'               DAT : -      =   4606.7 : 1.0
                     root = '상오'              TIM : -      =   4452.7 : 1.0
                  postpos = '2일'               DAT : -      =   4412.2 : 1.0
                  postpos = '8년'               DAT : -      =   4398.3 : 1.0
                 word_len = 22                  TRM : -      =   4331.1 : 1.0
                     root = '.'                   - : CVL    =   4313.7 : 1.0
                     root = '10일'              DAT : -      =   4186.2 : 1.0
                  postpos = '%'                 NUM : -      =   4078.9 : 1.0
                     root = '6년'               DAT : -      =   4031.8 : 1.0
                  postpos = '언더파'            NUM : -      =   4027.2 : 1.0
                     root = '1일'               DAT : -      =   3921.4 : 1.0
                     root = 'pre'               TRM : -      =   3823.3 : 1.0
```

show_most_informative_features의 결과값을 확인하면, '어근' 및 '접사' 추출을 의도하고 만들었던 common_roots(고빈도 20000 어근)와 common_postpos(고빈도 2000 접사) 리스트가 실은 더 다양한 종류의 데이터로 구성되었음을 확인할 수 있다. 어근의 경우 하나의 line 을 차지하는 마침표 특수문자(.)를 uni-gram 데이터로 확보했으며, 'pre'라는 접두사를 tri-gram 음절 데이터로 저장했다. 이를 통해 'root' feature는 텍스트의 문장부호에 대해 어떠한 내용 분류에도 속하지 않는 어절로 분석하였

으며, ‘pre’ 로 시작하는 어절을 의학 및 IT 관련 용어로 판단했다. 또한 접사의 경우 ‘OO신문’, ‘OO올림픽’ 등 합성어를 구성하는 우측 어간을 포착하거나, ‘N일’, ‘N년’ 등 1의 자리 수로 구성된 날짜명칭을 저장하였다. 이를 통해 ‘postpos’ feature는 자주 사용되는 합성어에 대한 내용정보를 분석하고, ‘일’, ‘월’, ‘년’으로 끝나는 어절에 대해 날짜 및 시간정보를 나타낸다고 판정했다.

이처럼 영향력이 큰 features의 내용이 초기 예상과 다르게 나타난 점을 고려하여 ‘root’ 및 ‘postpos’ features의 이름을 ‘first_ngrams’, ‘last_ngrams’로 변경하였으며, 관련 변수 이름을 함께 수정한 코드는 다음과 같다.

```
words = re.findall('^\d+\t(\S*)', data, re.M)

first_ngrams_fdist = nltk.FreqDist()
last_ngrams_fdist = nltk.FreqDist()

for word in words:
    # roots
    if len(word) > 0:
        first_ngrams_fdist[word[:1]] += 1
    if len(word) > 1:
        first_ngrams_fdist[word[:2]] += 1
    if len(word) > 2:
        first_ngrams_fdist[word[:3]] += 1

    # postpos
    if len(word) > 0:
        last_ngrams_fdist[word[-1:]] += 1
    if len(word) > 1:
        last_ngrams_fdist[word[-2:]] += 1
    if len(word) > 2:
        last_ngrams_fdist[word[-3:]] += 1

common_fngrams = list(first_ngrams_fdist)[:20000]
common_lngrams = list(last_ngrams_fdist)[:2000]
```

```
def word_features(word):
    # (조건4) 어절의 길이 추출
    features = {'word_len': len(word)}


    # (조건3) 어절의 첫번째 문자 초성 추출
    if re.match('[가-힣]', word[0]):
        features['first_letter_jaso'] = jaso(word[0])[0]
    else:
        features['first_letter_jaso'] = '<NOTKOR>'

    # (조건1) 고빈도 20000개 어근부, 고빈도 2000개 조사부 N-gram
    for fngram in common_fngrams:
        if word.startswith(fngram):
            features['first_ngram'] = fngram
    for lngram in common_lngrams:
        if word.endswith(lngram):
            features['last_ngram'] = lngram

    return features
```

```
featuresets = [(word_features(word), tag)for word, tag in tagged_words]
size = int(len(featuresets)*0.9)
train_set, test_set = featuresets[:size], featuresets[size:]
classifier = nltk.NaiveBayesClassifier.train(train_set)
nltk.classify.accuracy(classifier, test_set)
```

```
0.8623422779668666
```

```
classifier.show_most_informative_features(20)
```

```
Most Informative Features
              last_ngram = '상오'             TIM : -      =  12290.8 : 1.0
              last_ngram = '올림픽'           EVT : -      =   7382.2 : 1.0
             first_ngram = '10년'             DAT : -      =   5364.2 : 1.0
                word_len = 19                 TRM : -      =   5184.7 : 1.0
              last_ngram = '신문,'            PER : -      =   5149.6 : 1.0
              last_ngram = '6일'              DAT : -      =   4770.6 : 1.0
              last_ngram = '10년'             DAT : -      =   4687.3 : 1.0
              last_ngram = '8일'              DAT : -      =   4673.4 : 1.0
              last_ngram = '4년'              DAT : -      =   4659.5 : 1.0
              last_ngram = '2일'              DAT : -      =   4645.6 : 1.0
              last_ngram = '3일'              DAT : -      =   4631.7 : 1.0
             first_ngram = '1차전'            EVT : -      =   4577.1 : 1.0
             first_ngram = '상오'             TIM : -      =   4502.6 : 1.0
              last_ngram = '9일'              DAT : -      =   4437.3 : 1.0
              last_ngram = '8년'              DAT : -      =   4354.0 : 1.0
             first_ngram = '.'                - : CVL      =   4298.5 : 1.0
             first_ngram = '[마이'            ORG : -      =   4236.1 : 1.0
             first_ngram = '6년'              DAT : -      =   4107.2 : 1.0
             first_ngram = '10일'             DAT : -      =   4074.2 : 1.0
              last_ngram = '언더파'           NUM : -      =   4060.4 : 1.0
```

이상으로 코드에 대한 수정을 모두 마쳤다. 앞서 살펴본 과정을 통해 <2018년 네이버 NLP Challenge 개체명 인식 데이터>를 학습한 Naive Bayes Classifier 를 생성하였으며, (1) 어절의 길이, (2) 어절 첫번째 문자의 초성(자음), (3) 어절의 양옆 N-gram 음절 데이터에 대한 feature extraction 을 통해 86% 이상의 기계학습모델의 정확도를 확보하였다.