자연어 처리 – AI 전문가 양성(기본) Day 1

02 Word Embedding – Count-based

INGLab

실습 소개

- **실습 목표**
  - Count 기반의 word embedding 방법 구현 및 비교
  - Scikit-learn을 이용한 word vectorizer

- **실습 내용**
  - Bag of Words (BOW) 구현
  - TF-IDF 구현
  - Scikit-learn을 이용한 TF-IDF vectorizer 사용
  - Scikit-learn을 이용한 BOW → Naïve Bayes classification

Word Embedding

- Bag of Words
  - 단어들의 순서는 고려하지 않고, 출현 빈도만 파악한 임베딩 방법

(1) 문장 부호 제거

“코로나 백신 어서 맞아야 할텐데 하지만 백신 구하기 어려워 코로나 끝났으면”

(2) 토큰화

‘코로나’, ‘백신’, ‘어서’, ‘맞아야’, ‘할텐데’, ‘하지만’, ‘백신’, ‘구’, ‘하기’, ‘ 어려워’, ‘코로나’, ‘끝났으면’

(3) 단어 집합(vocab) 생성

```
{'코로나': 0, '백신': 1, '어서': 2, '맞아야': 3, '할텐데': 4, '하지만': 5,
'구': 6, '하기': 7, '어려워': 8, '끝났으면': 9}
```

(4) 단어 등장 횟수 카운트하여 벡터화

```
[1, 2, 1, 1, 2, 1, 1, 1, 1, 1]
```

Word Embedding

- TF-IDF
  - 단어 빈도와 역 문서 빈도를 고려한 임베딩 방법

(1) TF (term-frequency): 특성 문서에서 특정 단어가 등장한 횟수

(2) DF(document-frequency): 특정 단어가 등장한 문서의 수

(3) IDF(inverse document-frequency): DF에 반비례하는 값

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

```
# term-frequency
def tf(t, d):
  return d.count(t)

# inverse-term-frequency
def idf(t):
  df = 0
  for docu in documents:
    if t in docu:
      df += 1
  return math.log(N/(df + 1))

# tf-idf
def tf_idf(t, d):
  return tf(t, d) * idf(t)
```

Scikit-learn

- BOW
  - Scikit-learn → feature_extraction → text → CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer

vector = CountVectorizer()
vector.fit_transform(corpus)
train_vector = vector.transform(train_texts).toarray()
```

- TF-IDF
  - Scikit-learn → feature_extraction → text → TfidfVectorizer

```
from sklearn.feature_extraction.text import TfidfVectorizer

vector = TfidfVectorizer()
vector.fit_transform(corpus)
train_vector = vector.transform(train_texts).toarray()
```

Scikit-learn

- Bernoulli Naïve Bayes Classification
  - 데이터 전처리
  - 문장 벡터화
  - 객체 정의
  - Train
  - Prediction

```
from sklearn.naive_bayes import BernoulliNB

nb = BernoulliNB()
nb.fit(train_vector, train_y)
y_pred = nb.predict(test_vector)

print(classification_report(test_y, y_pred))
```

	precision	recall	f1-score	support
0	0.98	1.00	0.99	491
1	0.98	0.88	0.93	67
accuracy			0.98	558
macro avg	0.98	0.94	0.96	558
weighted avg	0.98	0.98	0.98	558