TF-IDF calculation using Scikit-Learn

Ko, Youngjoong

Sungkyunkwan University

nlp.skku.edu, nlplab.skku.edu

成均館大學校 1398 SUNG KYUN KWAN UNIVERSITY

Contents

❖ NLTK Part-Of-Speech(POS) Tag

❖ Count based Word Representation
- Bag of Words(BoW)
- TF-IDF

❖ Scikit-Learn

❖ Assignment

POS Tag

❖ POS Tag

- A POS tag is a label assigned to each word in a text to indicate the part of speech.
  Ex) subject, verb, object, etc.

- In general, main components of sentences are subject, verb, object, and complement, and these are usually verbs or nouns.

- In this assignment, we use only verbs and nouns tags.
  - Verb : (VB, VBD, VBG, VBN, VBP, VBZ)
  - Noun : (NN, NNS, NNP, NNPS)

  (Details in page 4)

POS Tag

❖ NLTK POS Tag

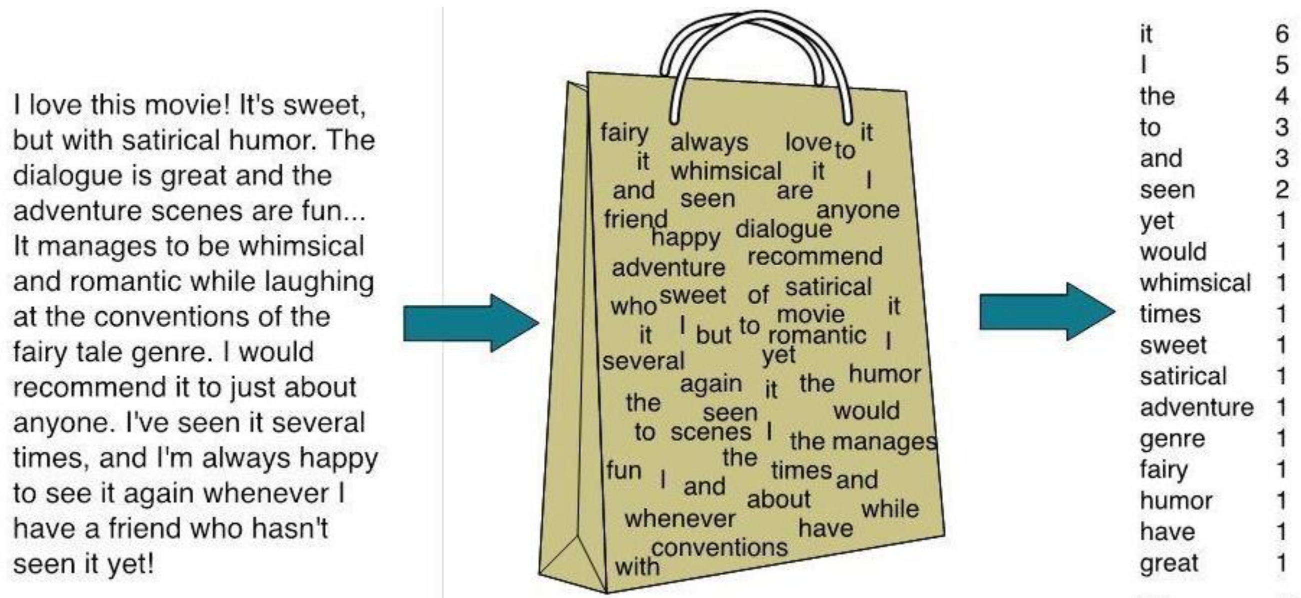
Tag	Description	Example
CC	coordinating conjunction	and
CD	cardinal number	1, third
DT	determiner	the
EX	existential there	*there* is
FW	foreign word	d'hoevre
IN	preposition/subordinating conjunction	in, of, like
JJ	adjective	big
JJR	adjective, comparative	bigger
JJS	adjective, superlative	biggest
LS	list marker	1)
MD	Modal	could, will
NN	noun, singular or mass	Door
NNS	noun plural	Doors
NNP	proper noun, singular	John
NNPS	proper noun, plural	Vikings
PDT	Predeterminer	*both* the boys
POS	possessive ending	friend*'s*

PRP	personal pronoun	I, he, it
PRP$	possessive pronoun	my, his
RB	Adverb	however, usually
RBR	adverb, comparative	Better
RBS	adverb, superlative	Best
RP	Particle	give *up*
TO	To	*to* go, *to* him
UH	Interjection	Uhhuhhuhh
VB	verb, base form	Take
VBD	verb, past tense	Took
VBG	verb, gerund/present participle	Taking
VBN	verb, past participle	Taken
VBP	verb, sing. present, non-3d	Take
VBZ	verb, 3rd person sing. Present	Takes
WDT	wh-determiner	Which
WP	wh-pronoun	who, what
WP$	possessive wh-pronoun	Whose
WRB	wh-abverb	where, when

Count based Word Representation

❖ Bag of Words(BoW)

- A method of numerical expression of text data that focuses only on the frequency of words without considering the order of words.



Count based Word Representation

❖ Bag of Words(BoW) based Document Features Extraction Method

- Doc0: I/PRP am/VBP a/DT boy/NN
- Doc1: I/PRP am/VBP a/DT girl/NN

1) Remove duplicates from all words in Doc0 and Doc1, and extract verbs and nouns. List each word in column form. (Sorted by alphabet)
   - ['am/VBP', 'boy/NN', 'girl/NN']
2) Give each of the following words a unique number (index).
   - { 'am/VBP' : 0, 'boy/NN' : 1, 'girl/NN' : 2 }
3) In each document, write the frequency in which the word appears in each number.
   - Index : 0 1 2
   - Doc0 : [1, 1, 0]
   - Doc1 : [1, 0, 1]

Count based Word Representation

❖ TF-IDF(Term Frequency-Inverse Document Frequency)

- TF(Term Frequency)
  : The number of times that term t occurs in document d ($tf_{t,d}$)
- DF(Document Frequency)
  : The number of documents that contain the term t (df_t)
- IDF(Inverse Document Frequency)
  : Inverse value of DF.

Term frequency		Document frequency		Normalization	
n (natural)	$\mathrm{tf}_{t,d}$	n (no)	1	n (none)	1
l (logarithm)	$1+\log(\mathrm{tf}_{t,d})$	t (idf)	$\log \frac{N}{\mathrm{df}_t}$	c (cosine)	$\frac{1}{\sqrt{w_1^2+w_2^2+\ldots+w_M^2}}$
a (augmented)	$0.5+\frac{0.5\times\mathrm{tf}_{t,d}}{\max_t(\mathrm{tf}_{t,d})}$	p (prob idf)	$\max\{0, \log \frac{N-\mathrm{df}_t}{\mathrm{df}_t}\}$	u (pivoted unique)	$1/u$
b (boolean)	$\begin{cases} 1 & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$			b (byte size)	$1/\mathit{CharLength}^{\alpha}$, $\alpha < 1$
L (log ave)	$\frac{1+\log(\mathrm{tf}_{t,d})}{1+\log(\mathrm{ave}_{t\in d}(\mathrm{tf}_{t,d}))}$				

Count based Word Representation

❖ TF (Term Frequency)

- Doc0 : I/PRP am/VBP a/DT boy/NN
- Doc1 : I/PRP am/VBP a/DT girl/NN
- Doc2 : who/WP is/VBZ a/DT boy/NN

- $tf_{t,d}$: The number of times that term t occurs in document d ($tf_{t,d}$)

	am/VBP	is/VBZ	boy/NN	girl/NN
Doc0	1	0	1	0
Doc1	1	0	0	1
Doc2	0	1	1	0

Count based Word Representation

❖ IDF (Inverse Document Frequency)

- Doc0 : I/PRP am/VBP a/DT boy/NN
- Doc1 : I/PRP am/VBP a/DT girl/NN
- Doc2 : who/WP is/VBZ a/DT boy/NN
  - $log\frac{N}{df_t}$: Inverse value of DF.
  - N : Total number of document
  - df_t : The number of documents that contain the term t (df_t)

  Ex) am/VBP = $log\frac{3}{2} = 0.176$, girl/NN= $log\frac{3}{1} = 0.477$

	am/VBP	is/VBZ	boy/NN	girl/NN
IDF	0.176	0.477	0.176	0.477

Count based Word Representation

❖ TF-IDF(Term Frequency Inverse Document Frequency)

- $tf_{t,d} \times log\frac{N}{df_t}$

Ex) am/VBP : $(1 \times 0.176) = 0.176$

	am/VBP	is/VBZ	boy/NN	girl/NN
Doc0	0.176	0	0.176	0.477
Doc1	0.176	0	0.176	0
Doc2	0	0.477	0	0

Scikit-Learn

❖ Install scikit-learn

- Google, 'Colab'

  - Packages installed by default

- Anaconda

  - If your virtual environment is activated,

    $ conda install scikit-learn

  - If your virtual environment is deactivated

    $ conda install --name [virtual_environment_name] scikit-learn

- Python environment

  - $ pip install -U scikit-learn

Scikit-Learn

❖ Scikit-Learn Package

- One of the most widely-used Python package for data science and machine learning.
  - Benchmark dataset
  - Preprocessing
  - Supervised learning
  - Unsupervised learning
  - Evaluation and selection

Scikit-Learn

❖ Preprocessing with Scikit-Learn

- Subpackage sklearn.feature_extraction and sklearn.feature_extraction.text provide the following subsystems for document preprocessing:
- The following classes create vectors after automatically sorting each token.
  - DictVectorizer
    - Creates a BOW vector using a dictionary consisting of counts of each word.
  - CountVectorizer
    - Generates word tokens and then creates a BOW vector by counting the number of each token.
  - TFidfVectorizer
    - It is similar to CountVectorizer but creates a BOW vector using weights of words calculated by TF-IDF.

Scikit-Learn

❖ DictVectorizer

- DictVectorizer is provided by the feature_extraction subpackage.
- Creates a BOW vector using a dictionary consisting of counts of each word.

```
from sklearn.feature_extraction import DictVectorizer

Bag_Of_Word = DictVectorizer(sparse=False)
Frequency = [{'am':1, 'boy':1}, {'am':1, 'girl':1}]
output = Bag_Of_Word.fit_transform(Frequency)

print(output)
```

```
[[1. 1. 0.]
 [1. 0. 1.]]
```

```
print(Bag_Of_Word.feature_names_)
```

```
['am', 'boy', 'girl']
```

```
print(Bag_Of_Word.transform({'girl':2, 'female':3}))
```

```
[[0. 0. 2.]]
```

Scikit-Learn

❖ CountVectorizer

- CountVectorizer performs following tasks:

    1. Converting the document to a list of tokens.
    2. Counting the frequency of each token in each document.
    3. Converting each document to a BOW vector.

❖ CountVectorizer

```
from sklearn.feature_extraction.text import CountVectorizer
import nltk
from nltk.tokenize import word_tokenize

corpus = ['I am a boy',
          'I am a girl',
          'Who is a boy']

POS_corpus = list()

for sentence in corpus:
    pos_token = nltk.pos_tag(word_tokenize(sentence))
    POS_corpus.append(' '.join([t[0] for t in pos_token if t[1] in ['NN', 'NNS', 'NNP', 'NNPS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']]))

print(POS_corpus)
```

```
['am boy', 'am girl', 'is boy']
```

```
vect = CountVectorizer()
vect.fit_transform(POS_corpus)
print(vect.vocabulary_)
```

```
{'am': 0, 'boy': 1, 'girl': 2, 'is': 3}
```

```
print(vect.transform(POS_corpus).toarray().tolist())
```

```
[[1 1 0 0]
 [1 0 1 0]
 [0 1 0 1]]
```

Scikit-Learn

❖ TfidfVectorizer

- When TfidfVectorizer creates a vector from TF-IDF, the vector is normalized using L2 Norm.
  - L2 Norm : $\|x\|_2 := \sqrt{x_1^2 + x_2^2 + \cdots + x_n^2} \quad where\ \ x = (x_1, x_2, \cdots, x_n)$

- TfidfVectorizer uses $idf = log\left(\frac{N+1}{df_t+1}\right) + 1$ to calculate TF-IDF.

$$idf = log\left(\frac{N + \mathbf{1}}{df_t + \mathbf{1}}\right) + \mathbf{1}$$

  - The constant **"1"** is added to the numerator and denominator of $log\left(\frac{N}{df_t}\right)$ to prevent dividing by zero.
  - The constant **"1"** is added to $log\left(\frac{N+1}{df_t+1}\right)$ to prevent terms which occur in all documents in a training set from being entirely ignored.

Scikit-Learn

❖ TfidfVectorizer

```python
from sklearn.feature_extraction.text import TfidfVectorizer
import nltk
from nltk.tokenize import word_tokenize

corpus = ['I am a boy',
          'I am a girl',
          'Who is a boy']

POS_corpus = list()

for sentence in corpus:
    pos_token = nltk.pos_tag(word_tokenize(sentence))
    POS_corpus.append(' '.join([t[0] for t in pos_token if t[1] in ['NN', 'NNS', 'NNP', 'NNPS', 'VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']]))

print(POS_corpus)
```

```
['am boy', 'am girl', 'is boy']
```

```python
tfidfvect = TfidfVectorizer()
tfidfvect.fit_transform(POS_corpus)
print(tfidfvect.vocabulary_)
```

```
{'am': 0, 'boy': 1, 'girl': 2, 'is': 3}
```

```python
print(tfidfvect.transform(POS_corpus).toarray().tolist())
```

```
[[0.70710678 0.70710678 0.         0.        ]
 [0.60534851 0.         0.79596054 0.        ]
 [0.         0.60534851 0.         0.79596054]]
```

Assignment

❖ Assignment

1) The attached JSON file in assignment 2 consists of 300 articles with multiple sentences.

   - See page 21 for JSON input file format.

2) Calculate TF-IDF vector of 300 articles using only nouns and verbs from given data.

   - Verbs (VB, VBD, VBG, VBN, VBP, VBZ) and nouns (NN, NNS, NNP, NNPS)

   - See page 4 for POS tags

3) You MUST create an output text file.

   - See page 22 for an example of output file.

4) You can use only Scikit-Learn for calculating TF-IDF and PyKomoran for morpheme analysis respectively.

Assignment

❖ Submit File

1) Python code file (.py) (python version 3.x)
   - Format: “Student Number_Name_TFIDF.py”.
     - Ex) “2020000000_홍길동.py”

2) TEXT file (.txt)
   - Format: “Student Number_Name_TFIDF.txt”.

JSON Input File

Article Topic

Article

```
"business": [
"Ad sales boost Time Warner profit  Quarterly profits at US media …
"Dollar gains on Greenspan speech  The dollar has hit …
…
(중략)
…
"GM, Ford cut output as sales fall …
"Ebbers denies WorldCom fraud …
  ],
…
(중략)
…
"politics": [
"Labour plans maternity pay rise…
"Watchdog probes e-mail deletions…
…
(중략)
…
"Blair stresses prosperity goals…
"Guantanamo man 'suing government'…
  ]
```

- JSON input file includes 300 articles on business, politics, and tech topics.

An Example of Output File

Student Number_Name_TFIDF.txt

Article

Tab

Article Topic

```
(business1)
0.0    0.0    0.4311        0.0    0.0    0.0    0.0    …(하략)…

(business2)
0.0    0.0    0.0    0.0    0.1334    0.0    0.0    …(하략)…
…
(중략)
…



…
(중략)
…

(politics1)
0.0    0.0    0.0    0.0    0.0    0.0    0.3311    …(하략)…
…
(중략)
…
(politics100)
0.1176    0.0    0.3311    0.0    0.0    0.0    0.0    …(하략)…
```

- This is an example.
- This can be different with actual answers.