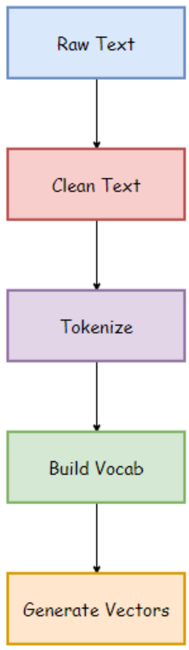
5. NLP – Bag of words, TF and IDF

Contents

5. NLP – Bag of words, TF and IDF

1. NLP – Components in NLP Bag-of-Words

- ✓ It is a method of extracting essential features from raw text.
- ✓ So that we can use it for machine learning models.
- ✓ A bag of words model converts the raw text into words, and it also counts the frequency for the words in the text.



1.1. Raw Text

- ✓ This is the original text on which we want to perform analysis.

1.2. Clean Text

- ✓ Since our raw text contains some unnecessary data like punctuation marks and stopwords, so we need to clean up our text.

1.3. Tokenize

- ✓ Tokenization represents the sentence as a group of tokens or words.

1.4. Building Vocab

- ✓ It contains total words used in the text after removing unnecessary data.

1.5. Generate Vocab

- ✓ It contains the words along with their frequencies in the sentences.

2. Use case

Let's take few sentences

- ✓ Jim and Pam travelled by bus.
- ✓ The train was late.
- ✓ The flight was full. Travelling by flight is expensive.

2.1. Creating a basic structure

Sentence 1	Sentence2	Sentence 3
Jim	The	The
and	train	flight
Pam	was	was
travelled	late	full
by		Travelling
the		by
bus		flight
		is
		expensive

2.2. Words with frequencies

Sentence1	Count	Sentence2	Count	Sentence3	Count
Jim	1	The	1	The	1
and	1	train	1	flight	2
Pam	1	was	1	was	1
travelled	1	late	1	full	1
by	1			Travelling	1
the	1			by	1
bus	1			is	1
				expensive	1

2.3. Combining all the words

Sentence	Frequency
and	1
bus	1
by	2
expensive	1
flight	2
full	1
is	1
jim	1
late	1
pam	1
the	3
train	1
travelled	1
travelling	1
was	1

2.4. Final model

	and	bus	by	expensive	flight	full	is	jim	Late	pam	The	train	travelled	travelling	was
S-1	1	1	1	0	0	0	0	1	0	1	1	0	1	0	0
S-2	0	0	0	0	0	0	0	0	1	0	1	1	0	0	1
S-3	0	0	1	1	2	1	1	0	0	0	1	0	0	1	1

Program Name: Bag of the words
demo1.py

```
from sklearn.feature_extraction.text import CountVectorizer

sentences = ["Jim and Pam travelled by bus.",
"The train was late",
"The flight was full. Travelling by flight is expensive"]

cv = CountVectorizer()

B_O_W = cv.fit_transform(sentences).toarray()

print(B_O_W)
```

Output

```
[[1 1 1 0 0 0 0 1 0 1 0 0 1 0 0]
 [0 0 0 0 0 0 0 0 1 0 1 1 0 0 1]
 [0 0 1 1 2 1 1 0 0 0 1 0 0 1 1]]
```

Applications

- ✓ This concept we can use in nlp applications
- ✓ Information retrieval from documents.
- ✓ Classifications of documents.

Limitations

- ✓ Semantic meaning:
    - It does not consider the semantic meaning of a word.
- ✓ Vector size:
    - For large documents, the vector size increase, which may result in higher computational time?
- ✓ Preprocessing:
    - In preprocessing, we need to perform data cleansing before using it.

3. Term Frequency-Inverse Document Frequency (TF-IDF)

- ✓ “**T**erm **F**requency – **I**nverse **D**ocument **F**requency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection.”

Here’s a sample of reviews about a particular horror movie:

- ✓ Review 1: This movie is very scary and long
- ✓ Review 2: This movie is not scary and is slow
- ✓ Review 3: This movie is spooky and good

TF: Term Frequency

$$TF = \frac{Frequency\ of\ the\ word\ in\ the\ sentence}{Total\ number\ of\ words\ in\ the\ sentence}$$

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

Inverse Document Frequency (IDF)

- ✓ IDF is a measure of how important a term is in a sentence
- ✓ We need the IDF value because computing just the TF alone is not sufficient to understand the importance of words

$$idf_t = \log \frac{number\ of\ documents}{number\ of\ documents\ with\ term\ 't'}$$

Example: Review 2

- ✓ IDF ('movie') = log(3/3) = 0
- ✓ IDF ('is') = log (3/3) = 0
- ✓ IDF ('not') = log (3/1) = 0.48
- ✓ IDF ('scary') = log(3/2) = 0.18
- ✓ IDF ('and') = log(3/3) = 0
- ✓ IDF ('slow') = log(3/1) =0.48

Calculating IDF

Term	Review 1	Review 2	Review 3	IDF
This	1	1	1	0.00
movie	1	1	1	0.00
is	1	2	1	0.00
very	1	0	0	0.48
scary	1	1	0	0.18
and	1	1	1	0.00
long	1	0	0	0.48
not	0	1	0	0.48
slow	0	1	0	0.48
spooky	0	0	1	0.48
good	0	0	1	0.48

- ✓ We can now compute the TF-IDF score for each word in the corpus.
- ✓ Words with a higher score are more important, and lower score are less important

Calculating final TF-IDF values:

✓ TF-IDF = TF * IDF

Example

Review 2

TF-IDF('this') = TF('this') * IDF('this') = 1/8 * 0 = 0

Term	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	0.000	0.000	0.000
movie	0.000	0.000	0.000
is	0.000	0.000	0.000
very	0.068	0.000	0.000
scary	0.025	0.022	0.000
and	0.000	0.000	0.000
long	0.068	0.000	0.000
not	0.000	0.060	0.000
slow	0.000	0.060	0.000
spooky	0.000	0.000	0.080
good	0.000	0.000	0.080