Certificate Course in Machine Learning using Python [6 Weeks]

Dashboard › My courses › Certificate Course in Machine Learning using Python [6 Weeks] › Day 19 › Text Classification

Text Classification

Attempt: 1

Text Classification

Text Classification

Text classification also known as text tagging or text categorization is the process of categorizing text into organized groups. By using Natural Language Processing (NLP), text classifiers can automatically analyze text and then assign a set of pre-defined tags or categories based on its content.

Natural Language Processing

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence. It is the technology that is used by machines to understand, analyses, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.

Applications of NLP

- **Question Answering**

  Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

- **Spam Detection**

  Spam detection is used to detect unwanted e-mails getting to a user's inbox.

- **Sentiment Analysis**

  Sentiment Analysis is also known as **opinion mining**. It is used on the web to analyze the attitude, behaviour, and emotional state of the sender.

- **Machine Translation**

  Machine translation is used to translate text or speech from one natural language to another natural language.

- **Speech Recognition**

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

- **Chatbot**

  Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

Text transformation

A text transformation is a technique that is used to control the capitalization of the text. Here the two major way of document representation is given.

- Bag of words
- Vector Space

Bag-of-Words Model

The bag-of-words model is used to represent text data for machine learning algorithms. The bag-of-words model is simple to understand and implement and has seen great success in language modeling and text classification.

A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

1. A vocabulary of known words.
2. A measure of the presence of known words.

Step 1: Collect Data

For example, An English man speaks some text to villagers:

- Good Job Done
- Very Good Job Done
- Bad Job Done
- Very Bad Job Done
- Bad Job

Step 2: Design the Vocabulary

Now we can make a list of all of the words in our model vocabulary. The unique words here (ignoring case and punctuation) are:

- Good
- Job
- Done
- Very
- Bad

Step 3: Sparse Matrix

Matrices that contain mostly zero values are called sparse matrix. The sparse matrix of above example can be constructed as:

Dictionary	Good	Job	Done	Very	Bad
Good Job Done	1	1	1	0	0
Very Good Job Done	1	1	1	1	0
Bad Job Done	0	1	1	0	1
Very Bad Job Done	0	1	1	1	1
Bad Job	0	1	0	0	1

The above matrix can be represented as sparse matrix:

Row/Column index	0	1	2	3	4
0	1	1	1	0	0
1	1	1	1	1	0
2	0	1	1	0	1
3	0	1	1	1	1
4	0	1	0	0	1

Text Processing using Python

CountVectorizer():

The CountVectorizer() function provides a simple way to both tokenize a collection of text documents and build a vocabulary of known words. It also used to encode new documents using that vocabulary.

Sample Text

```
simple_text=["Good Job Done", "Very Good Job Done", "Bad Job Done", "Very Bad Job Done", "Bad Job"]
```

- **Building the vocabulary using CountVectorizer()**

```
from sklearn.feature_extraction.text import CountVectorizer

vect=CountVectorizer()

#Fit : Learn the "Dictionary" of the data provided.

vect.fit(simple_text)
```

- **To see the dictionary made from data**

```
vect.get_feature_names()
```

get_feature_names() - Returns a list of feature names, ordered by their indices.

- **To prepare data matrix i.e. index having the value 1**

```
data_matrix=vect.transform(simple_text)

print(data_matrix)
```

- **To prepare dense matrix**

```
dense_matrix=data_matrix.toarray()

print(dense_matrix)
```

- **Converting the transformed data into dataframe**

```
import pandas as pd

df=pd.DataFrame(data_matrix.toarray(),columns=vect.get_feature_names())

print(df)
```

Note: These operations must be performed on the text data on which you want to perform Machine Learning. Now your training data is ready.

Next

PREVIOUS ACTIVITY

◄ Tkinter Presentation

Jump to...

NEXT ACTIVITY

Solving Text Classification Problem: Spam Detection (Mini Project 3) ►

Stay in touch