Vertica ML Python Workshop

Exercise 7: Working with Text Data

Ouali Badr

December 6, 2019

Executive Summary

> *"Science knows no country, because knowledge belongs to humanity, and is the torch which illuminates the world."*
>
> **Louis Pasteur**

VERTICA ML PYTHON allows the users to use Vertica advanced analytics and Machine Learning with a Python front-end Interface. In this exercise, you'll learn some basics to begin your fantastic Data Science Journey with the API. As a summary:

- Create a dictionary of words
- Create new feature by extracting relevant words

Contents

1 Presentation

Text data are very rich and can lead to the creation of many different features. One of the most common operation is to build a dictionary of words and pick the most relevant ones. However Text Data One can lead to the creation of too many dummies. It can be solved with a PCA.

Extracting information from text data is a very important tasks. Most of the time, ML can not be applied on text. We need to find a way to structure it and to make it interpretable.

During this exercise, we will use some online retail data to extract information from the description.

2 Functions used during the Exercise

2.1 str_contains

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].str_contains(self, pat: str)
```

Verify if a regular expression is in each of the column elements. The column will be transformed.

Parameters

- **pat:** *<str>*
  The regular expression.

Returns

The parent vDataframe.

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["name"]

#Output
                                        name
0              Allison, Miss. Helen Loraine
1      Allison, Mr. Hudson Joshua Creighton
2  Allison, Mrs. Hudson J C (Bessie Wald...
3                    Andrews, Mr. Thomas Jr
4                  Artagaveytia, Mr. Ramon
...                                      ...
Name: name, Number of rows: 1234, dtype: varchar(164)

titanic["name"].str_contains(' ([A-Za-z]+)\.')

#Output
```

```
      name
0     True
1     True
2     True
3     True
4     True
...    ...
Name: name, Number of rows: 1234, dtype: boolean
```

2.2 str_count

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].str_count(self, pat: str)
```

Compute the number of times a regular expression is in each of the column elements. The column will be transformed.

Parameters

- **pat:** *<str>*
  The regular expression.

Returns

The parent vDataframe.

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["name"]

#Output
                                      name
0               Allison, Miss. Helen Loraine
1      Allison, Mr. Hudson Joshua Creighton
2  Allison, Mrs. Hudson J C (Bessie Wald...
3                    Andrews, Mr. Thomas Jr
4                  Artagaveytia, Mr. Ramon
...                                    ...
Name: name, Number of rows: 1234, dtype: varchar(164)

titanic["name"].str_count(' ([A-Za-z]+)\.')

#Output
     name
```

```
0        1
1        1
2        1
3        1
4        1
...    ...
Name: name, Number of rows: 1234, dtype: int
```

2.3 str_extract

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].str_extract(self, pat: str)
```

Extract the regular expression in each of the column elements. The column will be transformed.

Parameters

- **pat:** *<str>*
  The regular expression.

Returns

The parent vDataframe.

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["name"]

#Output
                                        name
0               Allison, Miss. Helen Loraine
1      Allison, Mr. Hudson Joshua Creighton
2  Allison, Mrs. Hudson J C (Bessie Wald...
3                    Andrews, Mr. Thomas Jr
4                   Artagaveytia, Mr. Ramon
...                                      ...
Name: name, Number of rows: 1234, dtype: varchar(164)

titanic["name"].str_extract(' ([A-Za-z]+)\.')

#Output
       name
0     Miss.
```

```
1        Mr.
2       Mrs.
3        Mr.
4        Mr.
...      ...
Name: name, Number of rows: 1234, dtype: varchar(164)
```

2.4 str_replace

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].str_replace(self, to_replace: str, value: str = "")
```

Replace the regular expression in each of the column elements by another value. The column will be transformed.

Parameters

- **to_replace:** *<str>*
  The regular expression to replace.
- **value:** *<str>*, optional
  The new value.

Returns

The parent vDataframe.

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["name"]

#Output
                                        name
0               Allison, Miss. Helen Loraine
1      Allison, Mr. Hudson Joshua Creighton
2  Allison, Mrs. Hudson J C (Bessie Wald...
3                    Andrews, Mr. Thomas Jr
4                   Artagaveytia, Mr. Ramon
...                                     ...
Name: name, Number of rows: 1234, dtype: varchar(164)

titanic["name"].str_replace(' ([A-Za-z]+)\.')

#Output
```

```
                                        name
0                     Allison, Helen Loraine
1           Allison, Hudson Joshua Creighton
2  Allison, Hudson J C (Bessie Waldo Dan...
3                          Andrews, Thomas Jr
4                        Artagaveytia, Ramon
...                                      ...
Name: name, Number of rows: 1234, dtype: varchar(672)
```

2.5 str_slice

Library: vertica_ml_python.vDataframe[]

```
vDataframe[].str_slice(self, start: int, step: int)
```

Slice the column expression. The column will be transformed.

Parameters

- **start:** *<int>*
  Where to start on the expression.
- **step:** *<int>*
  The step to use (output expression length).

Example

```
from vertica_ml_python.learn.datasets import load_titanic
titanic = load_titanic(cur)
titanic["name"]

#Output
                                        name
0              Allison, Miss. Helen Loraine
1      Allison, Mr. Hudson Joshua Creighton
2  Allison, Mrs. Hudson J C (Bessie Wald...
3                    Andrews, Mr. Thomas Jr
4                   Artagaveytia, Mr. Ramon
...                                      ...
Name: name, Number of rows: 1234, dtype: varchar(164)

titanic["name"].str_slice(0, 5)

#Output
     name
0    Alli
```

```
1     Alli
21    Alli
3     Andr
23    Arta
...    ...
25  Name: name, Number of rows: 1234, dtype: varchar(20)
```

2.6 CountVectorizer

Create a CountVectorizer object which creates the dictionary of the different text columns. During the process, it will create a text index and compute the number of occurrences.

initialization

Library: vertica_ml_python.learn.preprocessing

```
1  class CountVectorizer(
       name: str,
3      cursor,
       lowercase: bool = True,
5      max_df: float = 1.0,
       min_df: float = 0.0,
7      max_features: int = -1,
       ignore_special: bool = True,
9      max_text_size: int = 2000)
```

Parameters

- **name:** *<str>*
  Name of the text index.
- **cursor:** *<object>*
  DB cursor.
- **lowercase:** *<bool>*, optional
  Convert all the elements to lowercase before processing.
- **max_df:** *<float>*, optional
  Keep the words which represent less than this float in the total dictionary distribution.
- **min_df:** *<float>*, optional
  Keep the words which represent more than this float in the total dictionary distribution.
- **max_features:** *<int>*, optional
  Keep only the top words of the dictionary
- **ignore_special:** *<bool>*, optional
  Ignore all the special characters to build the dictionary.
- **max_text_size:** *<int>*, optional
  The maximum size of the column which is the concatenation of all the text columns during the fitting.

Methods

The CountVectorizer object has 3 methods:

```
# Drop the text index
def drop(self)

# Fit the model with the input columns
def fit(self, input_relation: str, X: list)

# Build a vdf from the output relation
def to_vdf(self)
```

Attributes

The CountVectorizer object has two attributes:

```
self.stop_words # The words not added to the vocabulary because of some
    parameters
self.vocabulary # The final vocabulary
```

Example

```
from vertica_ml_python.learn.preprocessing import CountVectorizer

# We can build the model
model = CountVectorizer("name_voc", cur)
model.fit("titanic", ["Name"])

# We can export the dictionary
model.to_vdf()

# Output
       token                        df    cnt    rnk
0         mr    0.148163100524828421    734      1
1       miss    0.046023415421881308    228      2
2        mrs    0.037343560758982640    185      3
3    william    0.016148566814695196     80      4
4       john    0.013726281792490916     68      5
...      ...                     ...    ...    ...
Name: name_voc, Number of rows: 1841, Number of columns: 4
```

3 Questions

Turn on Jupyter with the 'jupyter notebook' command. Start the notebook exercise7.ipynb and answer to the following questions.

- **Question 1:** Build a dictionary of words and find the most relevant ones. What do you notice?

- **Question 2:** Build new features by extracting relevant words from the description.
- **Question 3:** Explain all the different other actions that you could do on text data.