09_Numerical_Data_Processing

April 22, 2024

Programming Techniques in Computational Linguistics II – FS24

1 Lecture 9: Numerical Data Processing

1.1 Topics

- Python Libraries
  - `NumPy`
  - `pandas`
  - `seaborn`

1.2 Learning Objectives

You should:

- …understand how you can create and use `ndarray` objects.
- …know how to apply basic mathematical operations on `ndarray` objects.
- …be familiar with basic pandas functionalities.
- …know how to gain an overview of a dataset.
- …know a few plot types and what they can be used to visualise.
- …know how to visualise pandas dataframes with seaborn.

2 NumPy

2.1 Motivation

In Computational Linguistics, we mainly focus on processing natural language. But that does not mean that we do not need to know about numerical data processing.

On the contrary, as you will see in more advanced courses, we almost always **represent natural language as numbers** to perform computations.

NumPy is a Python library that often helps us with our tasks as computational linguists because it:

- Adds support for large, multi-dimensional arrays and matrices
- Offers a large collection of high-level mathematical functions to operate on these arrays

2.2 NumPy ndarray

A `ndarray` object is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.

[ ]:
```python
import numpy as np
```

[ ]:
```python
arr = np.arange(16).reshape(4,4)
arr
```

[ ]:
```python
type(arr)
```

[ ]:
```python
arr.shape
```

2.3 NumPy ndarray Attributes

[ ]:
```python
# total number of elements in the array
arr.size
```

[ ]:
```python
# number of axes (dimensions) of the array
arr.ndim
```

[ ]:
```python
# type of the elements in the array
arr.dtype
```

[ ]:
```python
# site of each element of the array in bytes
arr.itemsize
```

2.4 NumPy ndarray Creation

[ ]:
```python
# create an array from a Python list
b = np.array([1, 2, 3])
b
```

[ ]:
```python
# create an array with ones / zeros
np.zeros(4)
```

[ ]:
```python
# create an array and fill with specified value
np.full(10, 42)
```

[ ]:
```python
# create an array with random values
np.random.rand(5)
```

2.5 NumPy Dimensions

2.5.1 Vector

[ ]:
```
vector = np.array([1, 2, 3])
vector.ndim
```

[ ]:
```
vector
```

2.5.2 Matrix

[ ]:
```
matrix = np.array([[1, 2, 3], [4, 5, 6]])
matrix.ndim
```

[ ]:
```
matrix
```

2.5.3 Tensor of rank 3

[ ]:
```
tensor = np.array([[[ 0,  1], [ 2,  3]],  [[ 4,  5],  [ 6,  7]],   [[ 8,  9], ␣
 ↪[10, 11]]])
tensor.ndim
```

[ ]:
```
tensor
```

2.5.4 Tensors of rank 4, 5, 6, …

[ ]:
```
tensor = np.arange(32).reshape(2,2,2,2,2)
tensor.ndim
```

[ ]:
```
tensor
```

2.6 Indexing and Slicing

`ndarray` objects behave very similarly to lists:

[ ]:
```
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
arr
```

[ ]:
```
# index an element
arr[1, 1]
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[ ]:
```
# index a row
arr[-2]
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[ ]:
```python
# index a column
arr[:, 1]
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])
```

[ ]:
```python
# slice a more complex subarray
arr[1:, 1:3]
```

2.7 Basic Operations

Arithmetic operators on arrays apply elementwise. A **new array** is created and filled with the result.

[ ]:
```python
a = np.array([20, 30, 40, 50])
b = np.array([0, 1, 2, 3])
```

[ ]:
```python
# subtraction
a - b
```

[ ]:
```python
# exponentiation
b ** 2
```

[ ]:
```python
# sine function
np.sin(a)
```

[ ]:
```python
# boolean
a[a > 20]
```

2.8 Axes

You can control along which axis an operation is applied.

[ ]:
```python
b = np.arange(12).reshape(3, 4)
b
```

[ ]:
```python
# sum of each column
b.sum(axis=0)
```

[ ]:
```python
# min of each row
b.min(axis=1)
```

2.9 Learning Goals

You should:

- …understand how you can create and use `ndarray` objects.
- …know how to apply basic mathematical operations on `ndarray` objects.

3 pandas

In computational linguistics, we often store our datasets in tabular formats such as .csv files (which you already talked about in the lecture on text formats).

To choose the right models for our data, we need to first get an **overview of our data**.

`pandas` is a Python library that helps us to explore and analyze our data because it provides:

- high-performance, easy-to-use data structures
- useful data analysis tools

pandas is built on top on NumPy.

3.1 pandas `Series`

pandas has two main classes to work with: `Series` and `DataFrame`

`Series` takes a list of values or a NumPy vector and creates an index for every element.

```
[ ]: import numpy as np
     import pandas as pd
```

```
[ ]: # create a Series from a list of values
     s = pd.Series([3, 5, np.nan])
     s
```

```
[ ]: # create a Series from numpy vector
     a = np.arange(5)
     s = pd.Series(a)
     s
```

3.2 pandas `DataFrame`

`DataFrame` takes a NumPy matrix and creates a table-like object with indeces and column headers.

```
[ ]: df = pd.DataFrame(np.random.rand(6, 4), columns=["first", "second", "third",␣
       ↪"fourth"])
     df
```

You can also pass a dictionary of objects that can be converted to series-like objects.

```
[ ]: df = pd.DataFrame({'A': 1.,
                        'B': pd.Timestamp('20230419'),
                        'C': pd.Series(1, index=[1, 2, 3, 4], dtype='float32'),
                        'D': np.array([3, 3, 3, 3], dtype='int32'),
                        'E': pd.Categorical(["test", "train", "test", "train"]),
                        'F': 'foo'})
```

```
df
```

Usually, we want to read in a csv File into a `DataFrame`object.

[ ]:
```
taxi = pd.read_csv("https://raw.githubusercontent.com/mwaskom/seaborn-data/
↪master/taxis.csv")
taxi.head()
```

3.2.1 Getting an Overview

`DataFrame` objects have a number of useful methods to get an overview over your data.

[ ]:
```
taxi.info()
```

[ ]:
```
taxi.describe()
```

3.3 Selecting and Indexing

[ ]:
```
# column "color" (of type Series)
taxi['color']
```

[ ]:
```
taxi['color'].value_counts()
```

[ ]:
```
# Last three rows dataset (of type DataFrame)
taxi[-3:]
```

[ ]:
```
# Select by label, rows 100 to 102, cols "pickup_zone" and "dropoff_zone"
taxi.loc[100:102, ['pickup_zone', 'dropoff_zone']]
```

[ ]:
```
# Select by position, rows 100 to 102, cols "pickup_zone" and "dropoff_zone"
taxi.iloc[100:103, [10, 11]]
```

[ ]:
```
taxi[taxi['distance'] > 30]
```

[ ]:
```
taxi.groupby("dropoff_borough").mean(numeric_only=True)
```

3.4 Learning Goals

You should:

- …be familiar with basic pandas functionalities.
- …know how to gain an overview of a dataset.

4 Seaborn

`pandas` is very useful to get some information about your data but often it is not enough to only look at bare numbers.

We also want to **visualize our data** to see the actual distribution of data points (e.g. to better identify outliers).

`seaborn` is a data visualization library based on matplotlib that:

- provides a high-level interface for drawing attractive and informative statistical graphics
- is closely integrated with pandas

4.1 Countplots

Visualize the number of trips to each dropoff borough.

```
[ ]: import matplotlib.pyplot as plt
     import seaborn as sns
```

```
[ ]: sns.countplot(y='pickup_borough', data=taxi) # hue='color', palette=['yellow',␣
       ↪'green']
     plt.show()
```

What are the most frequent pickup zones?

```
[ ]: # top_n = taxi["pickup_zone"].value_counts().iloc[:15].index

     sns.countplot(y='pickup_zone', data=taxi ) #, order=top_n)
     plt.show()
```

4.2 Scatterplots

How are distance and fare related?

```
[ ]: sns.scatterplot(x="distance", y="fare", data=taxi) #, hue="color",␣
       ↪palette=["yellow", "green"])
     plt.show()
```

4.3 Histplot

What is the distribution of total fares?

```
[ ]: sns.histplot(taxi["total"]) #, bins=50)
     plt.show()
```

4.4 Learning Goals

You should:

- …know a few plot types and what they can be used to visualise.
- …know how to visualise pandas dataframes with seaborn.

Numpy, Pandas and Seaborn all have extensive functionality. Consult the respective documentations to get the most out of them:

- NumPy

- Pandas
- Seaborn

5 Take-home messages

- You know basic functionalities and use-cases of the python libraries `numpy`, `pandas` and `seaborn`.
- Practice using them by solving exercise 7!

5.1 Admin

This Friday	26. April	**No** tutorial (Room is occupied!)
Next Wesnesday	1. May	**No** lecture (Labor day, University is closed)
Friday	3. May	Next tutorial: Numerical Data Processing + Midterm Feedback
Wednesday	8. May	Next lecture and **release of last graded exercise!**