00-general-introduction

October 9, 2021

1 Introduction to Unstructured Data Analysis

with Applications in Banking and Finance

1.1 Why?

1.1.1 The importance of Unstructured Data

When we think of financial data, our thinking almost always ends up displaying infinite historical series of numbers (prices, interest rates, volatility) or other personal information recorded in the tables of a database. Generally speaking financial data usually come in as **structured data**. They are commonly used for insights, analytics and reporting in financial institutions.

Structured data respect a predetefined set of rules so it is possible to define the type (date, name, number, characters, address) and the reciprocal relationships. Structured data depends on a schema and can be represented by rows and columns and stored in a central repository, typically a relational database, from which it can be retrieved separately or in a variety of combinations for processing and analysis.

On the other hand, **unstructured data**, such as call transcripts, emails text, transactional data are largely an area not yet fully exploited due to accessibility and processing challenges.

The combination of **Natural Language Processing (NLP)** and **Machine Learning (ML)** can offer powerful results. NLP is a machine's ability to interpret and analyze human language, both spoken and written, allowing us to extract context, meaning and intent from unstructured data. This capability, in combination with ML, provides an avenue to derive signals and generate actionable insights with a level of precision and scale that is significantly higher than ever before.

1.2 What?

1.2.1 Information Extraction

In order to act on unstructured data, ML models have to perform one of the crucial processes called **Information Extraction(IE)**. Information Extraction is the process of retrieving key information intertwined within the unstructured data. In other words, **extracting structured data from the unstructured data**. Unfortunately, when it comes to mining these sources for usable data, it's not quite so quick and easy. Sure, you can search documents for specific text, but what does that really tell you? Beyond word or phrase frequency, not much else. For these reasons you have to resort to sophisticated techniques of Text Analysis which employ a variety of methodologies to process the text, one of the most important of these being **Natural Language Processing (NLP)**.

1.2.2 Natural Language Processing

As we have already said, although textual data is abundantly available, the entanglement of natural language makes it particularly difficult to extract useful information from them. Natural Language Processing (NLP), a field of Artificial Intelligence (AI), analyzes, understands, and derives meaning from unstructured data. NLP focuses on the interaction between data science and human language, and is scaling to lots of industries. Today NLP is booming thanks to the huge improvements in the access to data and the increase in computational power, which are allowing practitioners to achieve meaningful results in areas like healthcare, media, finance and human resources, among others. NLP is used for Named Entity Recognition (NER) and Sentiment Analysis as well as Automatic Text Summarization, Parts-of-Speech tagging, and more.

1.2.3 Machine Learning

Machine learning is an artificial intelligence (AI) technology which provides systems with the ability to automatically learn from experience without the need for explicit programming, and can help solve complex problems. It is seen as a part of artificial intelligence. Machine learning algorithms build a model based on sample data, known as "training data", in order to make predictions or decisions without being explicitly programmed to do so. Machine learning algorithms are used in a wide variety of applications, such as in medicine, email filtering, speech recognition, and computer vision, where it is difficult or unfeasible to develop conventional algorithms to perform the needed tasks.

1.3 How?

1.3.1 Learning Tools

Python 3.X

We have chosen to use the Python 3.X programming language. Python is in some way similar to MATLAB, but contains a lot of modern software engineering tools that have become standard in the software industry and that should be adopted also for numerical computing projects. Python is at present also experiencing an exponential growth in popularity within the scientific computing community.

To set up your python environment, you'll first need to have a python on your machine. There are various python distributions available and we have chosen one that works very well for data science: **Anaconda**. Anaconda comes with its own Python distribution which will be installed along with it.

Data Science often requires you to work with a lot of scientific packages like scipy and numpy, data manipulation packages like pandas and IDEs and interactive Jupyter Notebook.Now, you don't need to worry about any python package most of them come pre-installed and if you want to install a new package, you can do that simply by using conda or pip.

To download an Anaconda distribution, you can use the official download page and you can select your platform and then choose the installer. For this, you can choose which version you want and whether 32-bit or 64-bit.

To test your installation, on Windows, click on Start and then Anaconda Navigator in the program list (or search for Anaconda in the search bar and select Anaconda Navigator). On a Mac, open up the finder, and in the Applications folder, double click on Anaconda-Navigator.

Package Managers

Anaconda will give you two package managers- pip and conda. When some packages aren't available with conda, you can use pip to install them. Note that using pip to install packages also available to conda may cause an installation error.

Jupyter Notebook

A notebook is a document like this one! A notebook integrates code and its output into a single document that combines visualizations, narrative text, mathematical equations, and other rich media.

In other words: it's a single document where you can run code, display the output, and also add explanations, formulas, charts, and make your work more transparent, understandable, repeatable, and shareable. As part of the open source Project Jupyter, Jupyter Notebooks are completely free. You can download the software on its own, or as part of the Anaconda data science toolkit.

1.3.2 Google Colab

Although it is not essential to work in a colab environment (all the course notebooks are in fact designed to be able to run without problems locally on your pc), it is useful to know some basic elements of the interaction with colab. In particular, in the cells below you will find two examples for the use of external files. In the first case it is shown how to load a text file from your local PC into the google virtual machine. The second example relates to the opposite operation: let's create a simple pandas dataframe into the colab environment and export it in csv format to the local machine.

How Upload a File on Google Colab

[5]:
```
if 'google.colab' in str(get_ipython()):
    from google.colab import files
    uploaded = files.upload()
    path = ''
else:
    path = './data/'
```

[6]:
```
with open(path + "carroll-alice.txt", "r") as f:
    alice = f.read()

alice[:392]
```

[6]:
```
"[Alice's Adventures in Wonderland by Lewis Carroll 1865]\n\nCHAPTER I. Down the
Rabbit-Hole\n\nAlice was beginning to get very tired of sitting by her sister on
the\nbank, and of having nothing to do: once or twice she had peeped into
the\nbook her sister was reading, but it had no pictures or conversations
in\nit, 'and what is the use of a book,' thought Alice 'without pictures
or\nconversation?'"
```

How Download a File on Google Colab

[7]:
```
import pandas as pd

cars = {'Brand': ['Honda Civic','Toyota Corolla','Ford Focus','Audi A4'],
        'Price': [22000,25000,27000,35000]
        }

df = pd.DataFrame(cars, columns= ['Brand', 'Price'])
```

[8]:
```
if 'google.colab' in str(get_ipython()):
    # if we run in google environment first we save in virtual machine...
    df.to_csv ('export_dataframe.csv', index = False, header=True)
    # ...then we download to local machine
    from google.colab import files
    files.download("export_dataframe.csv")
else:
    # if we are working in local we save directly with the usual method
    df.to_csv ('./data/export_dataframe.csv', index = False, header=True)
```

1.4 End Notes

So what will I learn from these lessons? Obviously, the duration of this seminar will not allow to deepen all the themes with due attention. Some important themes will remain out of our presentation (eg use of transformers in language understanding). What we will try to give you is just a starting point in the hope that you will find these topics fascinating and have fun learning them.

So let's start ….