# 4 Applications of Regular Expressions that every Data Scientist should know (with Python code)!

BEGINNER LISTICLE PYTHON TEXT UNSTRUCTURED DATA

## Overview

- Regular Expressions or Regex is a versatile tool that every Data Scientist should know about
- Regex can automate various mundane data processing tasks
- Learn about 4 exciting applications of Regex and how to implement them in Python

## Introduction

You have seen them, heard about them and probably have already used them for various tasks, without even realizing what happens under the hood? Yes, I'm talking about none other than – Regular Expressions; the quintessential skill for a data scientist's tool kit!

Regular Expressions are useful for numerous practical day to day tasks that a data scientist encounters. They are used everywhere from data pre-processing to natural language processing, pattern matching, web scraping, data extraction and what not!

That's why I wanted to write this article, to list some of those mundane tasks that you or your data team can automate with the help of Regular Expressions.

In case your basics of Regex are a bit hazy, I recommend you to read this article for a quick recap:

- [Beginners Tutorial for Regular Expressions in Python]

## Table of Contents



## Extracting emails from a Text Document

A lot of times, the sales and marketing teams might require finding/extracting emails and other contact information from large text documents.

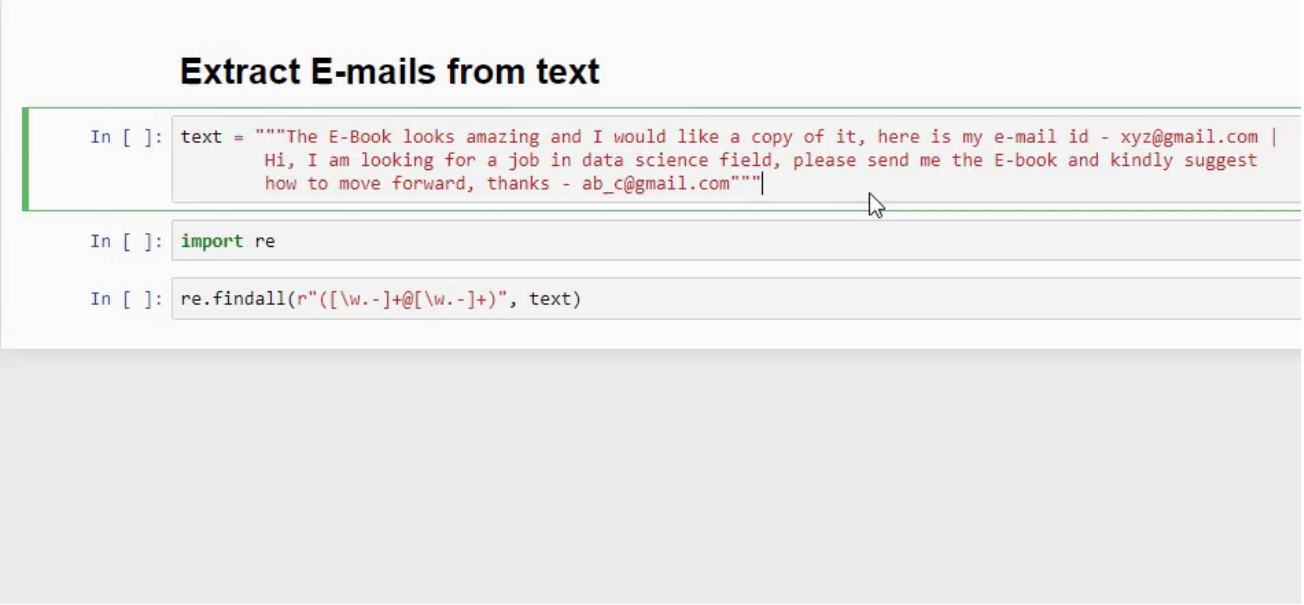
Now, this can be a cumbersome task if you are trying to do it manually! This is exactly the kind of situations when Regex really shines. Here's how you can code a basic email extractor:

```python
import re

# insert your text here
text = ""

re.findall(r"[\w.-]+@[\w.-]+", text)
```

view raw

re_email.py hosted with ❤ by GitHub

Just replace the "text" with the text of your document and you are good to go. Here is an example output we got:




Amazing, isn't it? In case you want to directly read a file and process it, you can simply add the file reading code to the Regex code:

```python
import re

# give your filename here
with open("filename.txt", "r") as fp:
  text = fp.read()
```

```
6
7  re.findall(r"[\w.-]+@[\w.-]+", text)
```

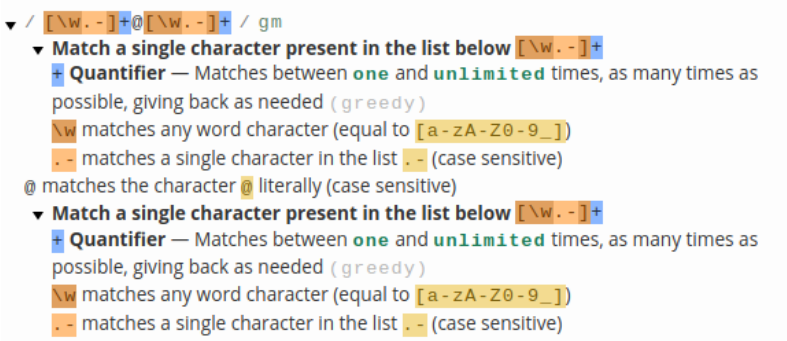
view raw

re_file_email.py hosted with ❤ by GitHub

The code might look scary but it is actually very simple to understand. Let me break it down for you. We use *re.findall()* to extract all the strings from the document which follows the following format:

`any character a-z, any digit 0-9 and symbol '_' followed by a '@' symbol and after this symbol we can again have any character, any digit and especially a dot.`

Here is an image that would give you a better understanding of the same




Wasn't that really simple? That's the thing about Regex, it lets you perform really complex tasks with simple expressions!

## Regular Expressions for Web Scraping (Data Collection)

Data collection is a very common part of a Data Scientist's work and given that we are living in the age of internet, it is easier than ever to find data on the web. One can simply scrape websites like Wikipedia etc. to collect/generate data.

But web scraping has its own issues – the downloaded data is usually messy and full of noise. This is where Regex can be used effectively!


Suppose this is the HTML that you want to work on:

```
html = """<table class="vertical-navbox nowraplinks" style="float:right;clear:right;width:22.0em;margin:0 0 1.0em 1.0em;backgrou
<div class="NavFrame collapsed" style="border:none;padding:0"><div class="NavHead" style="font-size:105%;background:transparent;
<ul><li><a href="/wiki/Statistical_classification" title="Statistical classification">Classification</a></li>
<li><a href="/wiki/Cluster_analysis" title="Cluster analysis">Clustering</a></li>
<li><a href="/wiki/Regression_analysis" title="Regression analysis">Regression</a></li>
<li><a href="/wiki/Anomaly_detection" title="Anomaly detection">Anomaly detection</a></li>
<li><a href="/wiki/Automated_machine_learning" title="Automated machine learning">AutoML</a></li>
<li><a href="/wiki/Association_rule_learning" title="Association rule learning">Association rules</a></li>
<li><a href="/wiki/Reinforcement_learning" title="Reinforcement learning">Reinforcement learning</a></li>
<li><a href="/wiki/Structured_prediction" title="Structured prediction">Structured prediction</a></li>
<li><a href="/wiki/Feature_engineering" title="Feature engineering">Feature engineering</a></li>
<li><a href="/wiki/Feature_learning" title="Feature learning">Feature learning</a></li>
<li><a href="/wiki/Online_machine_learning" title="Online machine learning">Online learning</a></li>
<li><a href="/wiki/Semi-supervised_learning" title="Semi-supervised learning">Semi-supervised learning</a></li>
<li><a href="/wiki/Unsupervised_learning" title="Unsupervised learning">Unsupervised learning</a></li>
<li><a href="/wiki/Learning_to_rank" title="Learning to rank">Learning to rank</a></li>
<li><a href="/wiki/Grammar_induction" title="Grammar induction">Grammar induction</a></li></ul>
</div></div></div></td>
</tr><tr><td style="padding:0 0.1em 0.4em">
<div class="NavFrame collapsed" style="border:none;padding:0"><div class="NavHead" style="font-size:105%;background:transparent;
<ul><li><a href="/wiki/Decision_tree_learning" title="Decision tree learning">Decision trees</a></li>
<li><a href="/wiki/Ensemble_learning" title="Ensemble learning">Ensembles</a>
<ul><li><a href="/wiki/Bootstrap_aggregating" title="Bootstrap aggregating">Bagging</a></li>
<li><a href="/wiki/Boosting_(machine_learning)" title="Boosting (machine learning)">Boosting</a></li>
<li><a href="/wiki/Random_forest" title="Random forest">Random forest</a></li></ul></li>
<li><a href="/wiki/K-nearest_neighbors_algorithm" title="K-nearest neighbors algorithm"><i>k</i>-NN</a></li>
<li><a href="/wiki/Linear_regression" title="Linear regression">Linear regression</a></li>
<li><a href="/wiki/Naive_Bayes_classifier" title="Naive Bayes classifier">Naive Bayes</a></li>
<li><a href="/wiki/Artificial_neural_network" title="Artificial neural network">Artificial neural networks</a></li>
<li><a href="/wiki/Logistic_regression" title="Logistic regression">Logistic regression</a></li>
<li><a href="/wiki/Perceptron" title="Perceptron">Perceptron</a></li>
<li><a href="/wiki/Relevance_vector_machine" title="Relevance vector machine">Relevance vector machine (RVM)</a></li>
```

```
33  <li><a href="/wiki/Support-vector_machine" title="Support-vector machine">Support vector machine (SVM)</a></li></ul>
34  </div></div></div></td></table>"""
```

view raw

html.py hosted with ❤ by GitHub

It is from a wikipedia page and has links to various other wikipedia pages. The first thing that you can check is what topics/pages does it have link for?

Let's use the following regex:

```
import re re.findall(r">([\w\s()]*?)</a>", html)
```

Once you use the above code, you will quickly get the list of topics:

```
['Machine learning',
 'data mining',
 '',
 'Classification',
 'Clustering',
 'Regression',
 'Anomaly detection',
 'AutoML',
 'Association rules',
 'Reinforcement learning',
 'Structured prediction',
 'Feature engineering',
 'Feature learning',
 'Online learning',
 'Unsupervised learning',
 'Learning to rank',
 'Grammar induction',
 'Supervised learning',
 'classification',
 'regression',
 'Decision trees',
 'Ensembles',
 'Bagging',
 'Boosting',
 'Random forest',
 'Linear regression',
 'Naive Bayes',
 'Artificial neural networks',
 'Logistic regression',
 'Perceptron',
```

Similarly, you can extract the links to all these pages by using the following regex:

```
import re re.findall(r"\/wiki\/[\w-]*", html)
```

Once you execute the above code, you will get the links to all these wikipedia pages.

```
['/wiki/Machine_learning',
 '/wiki/Data_mining',
 '/wiki/File',
 '/wiki/Statistical_classification',
 '/wiki/Cluster_analysis',
 '/wiki/Regression_analysis',
 '/wiki/Anomaly_detection',
 '/wiki/Automated_machine_learning',
 '/wiki/Association_rule_learning',
 '/wiki/Reinforcement_learning',
 '/wiki/Structured_prediction',
 '/wiki/Feature_engineering',
 '/wiki/Feature_learning',
 '/wiki/Online_machine_learning',
 '/wiki/Semi-supervised_learning',
 '/wiki/Unsupervised_learning',
```

Note that if you just combine each of the above link with **http://wikipedia.com** you'll be able to navigate to all these wikipedia pages. For example, to go to the page about **Unsupervised Learning** on wikipedia you

can use this link:

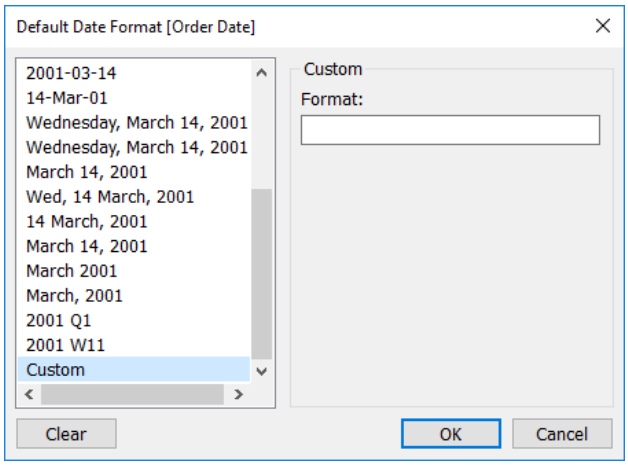
```
https://en.wikipedia.org/wiki/Unsupervised_learning
```

The first part of the URL is simply a template to which you can attach the second part that we just extracted from the HTML page.
You can learn more about extracting information of web scraped documents using Regex from this article:

- Beginners Tutorial for Regular Expressions in Python

## Working with Date-Time features

Most of the real world data has some kind of Date or Time column associated with it. Such columns carry useful information for the model, but since Date and Time have multiple formats available it becomes difficult to work with such data.




Can we use regex in this case to work with these different formats? Let us find out!

We will start with a simple example, suppose you have a Date-Time value like this:

```
date = "2018-03-14 06:08:18"
```

Let's extract the "Year" from the date. We can simply use regex to find a pattern where 4 digits occur together:

```
import re re.findall(r"\d{4}", date)
```

The above code will directly give you the year from the date. Similarly, you can extract the month and day information all together in one go!

```
1  import re
2
3  re.findall(r"(\d{4})-(\d{2})-(\d{2})", date)
```

view raw

re_dates.py hosted with ❤ by GitHub

```
[('2018', '03', '14')]
```

You can do the same thing for extracting the time info like hour, minute and second. That was one example of a date format, what if you have a date like the following format?

```
12th September, 2019
```

You just need to update the above Regex code. In this case, instead of having all numbers in the date format, we now have some text in between. We also have couple of extra spaces which can be matched using the **\s operator** and the way to catch the digits will be the same like previous example.

```
import re

date2 = "This building was founded on 14th March, 1999. That's when the official foundation was laid."

re.findall(r"(\d{2})\w+\s(\w+),\s(\d{4})", date2)
```

```
[('14', 'March', '1999')]
```

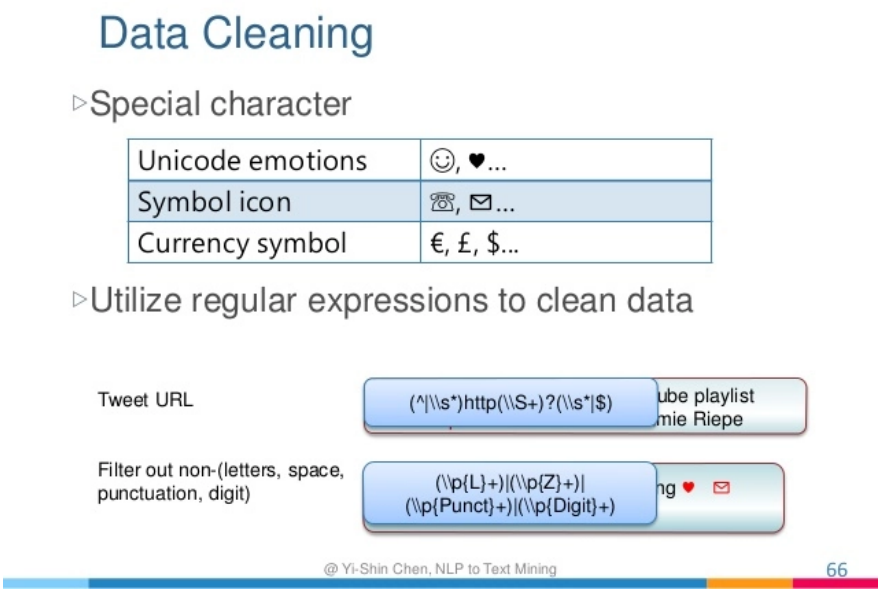
You can similarly create Regex for a variety of Date-Time formats and once your regex is ready, you can run the code using a loop or apply function in pandas over your columns of the dataset.
If you want to learn more about working with Date-Time features in python for machine learning, you can have a quick read here:

- [6 Powerful Feature Engineering Techniques For Time Series Data (using Python)]

## Using Regex for Text Pre-processing (NLP)

When working with text data, especially in NLP where we build models for tasks like text classification, machine translation and text summarization, we deal with a variety of text that comes from diverse sources.

For instance, we can have web scraped data, or data that's manually collected, or data that's extracted from images using OCR techniques and so on!




Source

As you can imagine, such diversity in data also implies good amount of **inconsistencies.** Most of this will not useful for our machine learning task as it just adds unnecessary noise and can be removed from the data. This is where Regex really comes handy!

Let's take the below piece of text as an example:

\xe2\x96\xb2 For Emergency Dial +91 9747200002 Patient Care Find a Doctor Get an Appointment Services and Facilities Get Directions Make an Enquiry Coronary Stents & Ortho Implants Give Feedback FAQ Medical Specialities Centre of Excellence Heart Centre Cardiology Cardio Thoracic & Vascular Surgery Cardiac Anesthesia Institute of Neuro Sciences Neurology Neurosurgery Institute of Liver and Gastro Sciences Medical Gastroenterology Surgical Gastroenterology BMH Cancer Centre Medical Oncology Radiation Oncology Surgical Oncology Nuclear Medicine PET-CT Scan Women care Obstetrics & Gynaecology Infertility & Laparoscopic Surgery Foetal Medicine Child care Paediatrics Neonatology Paediatric Surgery Child Guidance Orthopaedics, Spine & please call us at +91-495- 2777888 to check the status. Submit Treatment In India Online Appointment Health Check-up Packages Plan Your Visit FAQ's Patient Testimonial Happenings Healthcare Excellence Through Administration And Leadership \xe2\x86\x90\xc2\xa0\xc2\xa0Read More Media BMH inaugurated breast cancer awareness drive \xe2\x80\x9cBreast Cancer - Know it to beat it\xe2\x80\x9d on 25th October 2016. \xe2\x86\x90\xc2\xa0\xc2\xa0Read More Home About BMH Send an Enquiry Get an Appointment Find a Doctor Health Check-up Packages Careers Contact Us Sitemap BMH APP Download Our Apps. Everything You Need to Know All contents \xc2\xa9 copyright 1999-2016 Baby Memorial Hospital Ltd. All rights reserved. Follow Us On http://bit.ly/1RmnUT

As it's evident, the above text has a lot of inconsistencies like random phone numbers, web links, some strange unicode characters of the form "\x86…" etc. For our text classification task, we just need clean and pure text so let's see how to fix this.

We will write a function to clean this text using Regex:

```python
import re
import nltk
nltk.download('stopwords')

# download stopwords list from nltk
from nltk.corpus import stopwords

stop_words = set(stopwords.words('english'))

def clean_text(text):
    # converting to lowercase
    newString = text.lower()
    # removing links
    newString = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', newString)
    # fetching alphabetic characters
    newString = re.sub("[^a-zA-Z]", " ", newString)
    # removing stop words
    tokens = [w for w in newString.split() if not w in stop_words]
    # removing short words
    long_words=[]
    for i in tokens:
        if len(i)>=4:
            long_words.append(i)
    return (" ".join(long_words)).strip()
```

view raw

re_nlp.py hosted with ❤ by GitHub

Once you run the given code on the above text, you will see that the output is pretty clean:

emergency dial patient care find doctor appointment services facilities directions make enquiry coronary stents ortho implants give feedback medical specialities centre excellence heart centre cardiology cardio thoracic vascular surgery cardiac anesthesia institute neuro sciences neurology neurosurgery institute liver gastro sciences medical gastroenterology surgical gastroenterology cancer centre medical oncology radiation oncology surgical oncology nuclear medicine scan women care obstetrics gynaecology infertility laparoscopic surgery foetal medicine child care paediatrics neonatology paediatric surgery child guidance orthopaedics spine please call check status submit treatment india online appointment health check packages plan visit patient testimonial happenings healthcare excellence administration leadership read media inaugurated breast cancer awareness drive breast cancer know beat october read home send enquiry appointment find doctor health check packages careers contact sitemap download apps everything need know contents copyright baby memorial hospital rights reserved follow

So what did we do here? We basically applied a bunch of operations on our input string:

- Firstly, we removed all the irrelevant links from the text

```
# removing links newString = re.sub(r'(https|http)?:\/\/(\w|\.|\/|\?|\=|\&|\%)*\b', '', newString)
```

- We also removed all the digits (phone numbers)

```
# fetching alphabetic characters newString = re.sub("[^a-zA-Z]", " ", newString)
```

- Post that we used another useful package of Python **nltk** to remove stopwords such as “and, the, for” etc. from our text input

```
# removing stop words tokens = [w for w in newString.split() if not w in stop_words]
```

- We made sure to ignore any word that is smaller than 3 characters because such abbreviations like “CET”, “BHM” etc. do not add much information unless they are taken into context

```
# removing short words long_words=[] for i in tokens: if len(i)>=4: long_words.append(i) return (" ".join(long_words)).strip()
```

So that’s text pre-processing, isn’t it fascinating? You can learn all about text pre-processing from this intuitive blog:

- Ultimate guide to deal with Text Data (using Python) – for Data Scientists and Engineers

## End Note

In this article we saw some simple yet useful ways of using Regular expressions. Yet, we barely scratched the surface of the capabilities of this great tool.

I encourage you to dig deeper and understand how Regex works (as they can be quite confusing in the beginning!) rather than simply using them blindly.

Here are some resources that you can follow to know more about them:

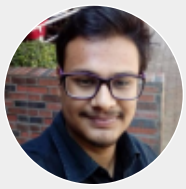
- Regex101 – It is a very useful website to visualize and understand how Regex works under the hood.

- Regex Documentation for Python 3
- Regex Quickstart CheatSheet – This is for those who want to quickly revise regex operators.

Have you used Regex before? Do you want to add an application to this list that I missed? Answer in comments below!

---

Article Url - https://www.analyticsvidhya.com/blog/2020/01/4-applications-of-regular-expressions-that-every-data-scientist-should-know-with-python-code/




## Mohd Sanad Zaki Rizvi

A computer science graduate, I have previously worked as a Research Assistant at the University of Southern California(USC-ICT) where I employed NLP and ML to make better virtual STEM mentors. My research interests include using AI and its allied fields of NLP and Computer Vision for tackling real-world problems.