

# 04-basic-text-analysis

October 21, 2021

## 1 Basic Text Analysis

### 1.1 What is Text Mining and Why it's so Important

Text is one of the most widespread forms of sequence data. It can be understood as either a sequence of characters or a sequence of words, but it's most common to work at the level of words. According to industry estimates, only 21% of the available data is present in a structured form. Data is being generated as we speak, as we tweet, as we send messages on WhatsApp and in various other activities. The majority of this data exists in the textual form, which is highly unstructured in nature.

Despite having high dimension data, the information present in it is not directly accessible unless it is processed (read and understood) manually or analyzed by an automated system. In order to produce significant and actionable insights from text data, it is important to get acquainted with the basics of Text Analysis.

Text Mining is the process of analysis of texts written in natural language and extract high-quality information from text. It involves looking for interesting patterns in the text or to extract data from the text to be inserted into a database. Text mining tasks include text categorization, text clustering, concept/entity extraction, production of granular taxonomies, sentiment analysis, document summarization, and entity relation modeling (i.e., learning relations between named entities). Developers have to prepare text using lexical analysis, POS (Parts-of-speech) tagging, stemming and other Natural Language Processing techniques to gain useful information from text.

### 1.2 Package Required

**To start**, install the packages you need to run the code in this notebook.

```
[2]: # Python Regular Expression (RegEx)
import re
# Operating System Module
import os
# numpy library
import numpy as np
# pandas library
import pandas as pd
# matplotlib library
import matplotlib.pyplot as plt
# if uising a Jupyter notebook, include:
%matplotlib inline
```

```
# Natural Language Toolkit
import nltk
# The BeautifulSoup Library for WEB Scraping
from bs4 import BeautifulSoup

import codecs
from sklearn import feature_extraction
```

### 1.3 What is a Corpus

A Corpus is defined as a **collection of text documents** for example a data set containing news is a corpus or the tweets containing Twitter data is a corpus. So corpus consists of documents, documents comprise paragraphs, paragraphs comprise sentences and sentences comprise further smaller units which are called Tokens.

Acquiring a domain-specific corpus will be essential to producing a language-aware data product that performs well. Naturally the next question should then be “How do we construct a dataset with which to build a language model?”.

While in the next chapters we will see how to use existing corpus, it is still necessary to give a brief hint on how to extract texts from the web independently and which are the main libraries that we can use for this purpose.

### 1.4 Processing Raw Text

The most important source of texts is undoubtedly the Web. It’s convenient to have existing text collections to explore, such as the corpora we’ll see in the next paragraph. However, you probably have your own text sources in mind, and need to learn how to access them.

#### 1.4.1 Processing HTML Files

The first type of structured text document you’ll look at is HTML—a markup language commonly used on the web for human-readable representation of information. An HTML document consists of text and predefined tags (enclosed in angle brackets <>) that control the presentation and interpretation of the text. The tags may have attributes.

Reference: [this](#)

#### 1.4.2 Urllib

Urllib is a Python library that allows the developer to open and parse information from HTTP or FTP protocols. Urllib offers some functionality to deal with and open URLs, namely: - `urllib.request`: opens and reads URLs. - `urllib.error`: catches the exceptions raised by `urllib.request`. - `urllib.parse`: parses URLs. - `urllib.robotparser`: parses robots.txt files.

You don’t need to install Urllib since it is a part of the built-in Python library.

```
[3]: from urllib import request

def freq_words_2(url, n):
```

```
html = request.urlopen(url).read().decode('utf8')
text = BeautifulSoup(html, 'html.parser').get_text()
fd = nltk.FreqDist(word.lower() for word in nltk.word_tokenize(text))
return [word for (word, _) in fd.most_common(n)]
```

```
[4]: page = "https://static.nytimes.com/email-content/CB_sample.html"
freq_words_2(page, 5)
```

```
[4]: ['the', ',', 'and', '.', 'to']
```

```
[5]: from nltk.corpus import stopwords

def freq_words_3(url, n):
    stop_words = set(stopwords.words('english'))
    html = request.urlopen(url).read().decode('utf8')
    text = BeautifulSoup(html, 'html.parser').get_text()
    fd = nltk.FreqDist(word.lower() for word in nltk.word_tokenize(text) if not
    ↪word in stop_words and word.isalpha())
    return [word for (word, _) in fd.most_common(n)]
```

```
[6]: freq_words_3(page, 5)
```

```
[6]: ['new', 'york', 'i', 'the', 'time']
```

### 1.4.3 BeautifulSoup

**BeautifulSoup** is a Python library that is used to extract information from XML and HTML files. BeautifulSoup is considered a parser library. Parsers help the programmer obtain data from an HTML file. One of BeautifulSoup's strengths is its ability to detect page encoding, and hence get more accurate information from the HTML text. Another advantage of BeautifulSoup is its simplicity and ease.

You can construct a BeautifulSoup object from a markup string, a markup file, or a URL of a markup document on the web:

```
[7]: from bs4 import BeautifulSoup

# Construct soup from a string
soup1 = BeautifulSoup("<HTML><HEAD><headers></HEAD><body></HTML>")
print(soup1.text)
```

```
<headers><body>
```

```
[8]: # Construct soup from a local file
soup2 = BeautifulSoup(open("./corpus/web/Democrats Poised For Senate Control As_
    ↪Counting Continues In Georgia_NPR.html"))
print(soup2.title)
```

```
<title>Democrats Poised For Senate Control As Counting Continues In Georgia :  
NPR</title>
```

```
[9]: print(soup2.title.string)
```

Democrats Poised For Senate Control As Counting Continues In Georgia : NPR

One common task is extracting all the URLs found within a page's tags:

```
[26]: for link in soup2.find_all('a'):  
       print(link.get('href'))
```

**When to use BeautifulSoup?** If you're just starting with webs scarping or with Python, Beautiful Soup is the best choice to go. Moreover, if the documents you'll be scraping are not structured, Beautiful Soup will be the perfect choice to use. If you're building a big project, Beautiful Soup will not be the wise option to take. Beautiful Soup projects are not flexible and are difficult to maintain as the project size increases.

```
[11]: import requests  
       from bs4 import BeautifulSoup  
  
base_url = 'http://www.nytimes.com'  
r = requests.get(base_url)  
soup = BeautifulSoup(r.text)  
  
for story_heading in soup.find_all(class_="story-wrapper"):  
    #print(story_heading)  
    if story_heading.a:  
        print(story_heading.a.text.replace("\n", " ").strip())  
    else:  
        print(story_heading.contents[0].strip())
```

Americans Stretch Across Political Divides to Welcome Afghan Refugees  
Taliban Claim Control Over Panjshir Valley; Resistance Vows to Fight On  
A military analysis raised questions about a deadly drone strike on a car in Kabul last Sunday.

Oregon and Idaho Are Running Out of I.C.U. Beds as Covid Cases Hit Records  
Here's why you might not be returning to the office until next year.

Congregations are marking the Jewish High Holy Days in the shadow of Covid, again.

Tracking the Coronavirus >

As Migrants Surge Toward Border, Court Hands Biden a Lifeline

An Unsung Pit Crew of California's Wildfires: Hotel Workers Left Behind

See the Caldor fire's march to the edge of South Lake Tahoe.

They Put Everything Into Their Homes. Not One Was Spared in the Flood.

California Bill Could Alter Amazon Labor Practices

The bill would rein in production quotas at warehouses that critics say are excessive and force workers to forgo bathroom breaks.

The Strange Tale of the Freedom Phone, a Smartphone for Conservatives

A 22-year-old Bitcoin millionaire wants Republicans to ditch their iPhones for a low-end handset that he hopes to turn into a political tool.

Can Anyone in the N.F.C. Stop Tom Brady and the Bucs From Repeating?

Rivalries and Rookies Bloom, but It's Still Patrick Mahomes's A.F.C.

Gail Collins and Bret Stephens History Can Close in on Us Awfully Fast

Margaret Renkl Southern Republicans Cannot Be Trusted With Public Health

Terri Gerstein Other People's Rotten Jobs Are Bad for Them. And for You.

Charles M. Blow From 'Ku Kluxism' to Trumpism

Maya Guzdar What Happened the Day After I Was Sexually Harassed at the Pentagon

Virginia López Glass Venezuela's New Lettuce-Based Economy Is Good Enough for Now

Kelly Corrigan How to Let Go of Your Irreplaceable, Unstoppable Daughter

The New York Times Join Kara Swisher for a Live Event

Coach Coughlin and the Caregiver's Love and Heartbreak

Jerome Karabel Let's Honor the True Spirit of Labor Day

Maureen Dowd Drowning Our Future in the Past

Jamelle Bouie How Has Joe Biden Become So Unpopular?

Chris Windsor

Listen to 'Popcast'

Sign Up for the Watching Newsletter

Opinion

Virtual Event: Kids and Covid

Ex-Marine Sharpshooter Kills 4 and Fires at Deputies in Florida, Sheriff Says

If Gawker Is Nice, Is It Still Gawker?

Endangered Birds Have a Defender on N.Y.C. Beaches: The Plover Patrol

Texas Abortion Law Is the Culmination of Decades of Conservative Effort

Belarus Opposition Leader Sentenced to 11 Years in Prison

Chinese Social Media Site Suspends K-Pop Fan Accounts

Kanye West's 'Donda' Is No. 1, but Drake Waits in the Wings

Jacob Zuma, South Africa's Ex-President, Gets Medical Parole

Hurricane Larry Is Forecast to Bring Rough Surf to East Coast

Is Your 'Go Bag' Ready?

What Vaccinated People Need to Know About Breakthrough Infections

The Secret to Raising a Resilient Kid

How Burnout Became the Norm for American Parents

5-Minute Stress Resets to Try

Tony Cenicola/The New York Times

John G Mabanglo/EPA, via Shutterstock

Charlotte Hadden for The New York Times

Jeenah Moon for The New York Times

Marcio Jose Sanchez/Associated Press

Our Best Rosh Hashana and Yom Kippur Recipes

Bombay Frittata

Sheet-Pan Salmon and Eggplant With XO Sauce

Apple Jelly

Zucchini Panzanella  
The Best Kids Headphones  
Keep Your Tent Out of the Sun  
Everything You Need to Make Hot Pot at Home  
Learn More About Wirecutter  
Spelling Bee  
The Crossword  
Letter Boxed  
Tiles  
Vertex

```
[12]: base_url = 'https://www.theguardian.com/international'
r = requests.get(base_url)
soup = BeautifulSoup(r.text)

for story_heading in soup.find_all(class_="fc-item__standfirst"):
    #print(story_heading)
    if story_heading.a:
        print(story_heading.a.text.replace("\n", " ").strip())
    else:
        print(story_heading.contents[0].strip())
```

Taliban fighters pictured outside governor's compound, but Ahmad Massoud's rebels deny province has fallen  
Former world leaders and public figures say nationwide marches are modelled on US Capitol insurrection  
Real Vishnevsky battling two doppelgängers who seem to have changed their appearance as well as their names  
Storm wreaked havoc on offshore oil production platforms and onshore oil and gas processing plants  
Recent discoveries of mass graves have shed new light on the country's troubled colonial legacy  
Immortalised by Godard and Melville, the actor specialised in seductive tough guys - and blazed a trail through cinema history  
After building his own version of Middle-earth, Nicolas Gentile has thrown a 'ring' into Mount Vesuvius  
Mississippi is least vaccinated state in US; New Zealand reports 20 new cases for third day  
Some welcome the routine, while others are concerned about how it will work without everyone in at the same time  
England face a record run chase to beat India in the fourth Test. Join our writers for updates  
Giovani Lo Celso and Cristian Romero are in hot water with Tottenham after going against the club's wishes to play in the abandoned World Cup qualifier between Brazil and Argentina  
Tony Estanguet has already set up community programmes and wants to 'transform the city of Paris into a big Olympic and Paralympic park'  
From plummeting trade to drastic shortages of workers, needlessly leaving the

single market has been disastrous, says Guardian columnist Simon Jenkins

Those responsible for 20 years of strategic disaster won't actually be held responsible, says former military officer Frank Ledwidge

It may be the first Marvel film not to be approved in mainland China, but its American Chinese heritage hero and strong female representation are badass

Developers are harming cities like London and New York by asking the public to visit and help monetise their monstrosities

A warning call told residents of al-Jalaa apartment block that their homes were about to be destroyed. This is the story of the frantic evacuation that followed - told through recordings made by the people who lived there

With a tight campaign devised by a sports marketer and his face all over the streets, the SPD leader looks set to win

House in southern France yielded find of outstanding wall paintings dating from 1st century BC

20 men accused of involvement in 2015 massacre, but it is unclear whether the key accused will break their silence

An innovative community project has brightened buildings, 'brought people together' and provided an emotional outlet after traumatic journeys

Editorial in publications worldwide urges leaders to take measures to stop 'greatest threat to public health'

The figure for the three months to September was 1.3C above the long-term average and higher than the previous record set in 2020

World's largest lizard moves from vulnerable to endangered on IUCN red list of threatened species

ShareAction says lack of plans to tackle climate crisis and biodiversity loss casts doubts on banking's sustainability pledges

Angela Merkel 'profoundly shaken' by death of Jan Hecker, 54, one of her top foreign policy advisers

English-born entertainer, 74, has cancelled her appearance at American film festival in France

Researchers solve mystery of why southern hemisphere whales switch suddenly but in north it is gradual

As Elon Musk launches his general-purpose humanoid, a look at some of movie's most helpful robots, from Alita to R2-D2

Netflix's new documentary series sells the first all-civilian flight to space as an exercise in philanthropy, but it's little more than a privilege-fuelled puff piece for the billionaire's adventures

Todd, 47, and India, 27, met when she walked into his office on her police duties. They now live together in St Augustine, Florida

A delicious fusion of Spanish and Italian cultures meld in this tomatoey noodle dish infused with chipotle and topped with soured cream and salsa

Find out why this beautifully rustic bean and pasta offcut stew is the 'Bill Murray of foods'

A week after the storm hit, thousands of residents face a continuing catastrophe amid faltering federal government assistance

During lockdown, more than 60% of us started new creative pursuits. Here's how to keep them up as autumn gets under way

Colleges offer support as young people aim to devote their lives to battling the

crisis

We'd like to hear from those living in Australia who have tested positive recently for Covid or are recovering

From the joyful to the sad, we would like to hear your stories about a point in time when your world turned on its axis

The long-running series in which readers answer other readers' questions on subjects ranging from trivial flights of fancy to profound scientific and philosophical concepts

You can send a news tip direct to Guardian journalists [here](#). For stories that need a high level of security then contact us [here](#)

The Guardian's picture editors select photo highlights from around the world

The Someone's Daughter exhibition, portraits of women involved with the criminal justice system, launches at Photo London, special exhibitors section, on 9 September

The singer, model, actor and former member of the group Girls Aloud has died aged 39 after being diagnosed with breast cancer

Tom Wood captures a moment of pilgrimage overlooking his birthplace of County Mayo

Michael Rosen, Paloma Faith, St Etienne and 'champing' holidays - the best photography commissioned by the Observer in August 2021

Artist Melanie Issaka's photograms explore what it means to be black, British and female

## 1.5 Regular Expressions

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern. RegEx can be used to check if a string contains the specified search pattern. Python has a built-in package called `re`, which can be used to work with Regular Expressions.

### 1.5.1 Why we need Regular Expression

Imagine you have a string object `s`. Now suppose you need to write Python code to find out whether `s` contains the substring `'123'`. There are at least a couple ways to do this. You could use the `in` operator:

```
[2]: s = 'foo123bar'
     '123' in s
```

```
[2]: True
```

If you want to know not only whether `'123'` exists in `s` but also where it exists, then you can use `.find()` or `.index()`. Each of these returns the character position within `s` where the substring resides:

```
[3]: s.find('123')
```

```
[3]: 3
```

```
[4]: s.index('123')
```



[4]: 3

In these examples, the matching is done by a straightforward character-by-character comparison. That will get the job done in many cases. But sometimes, the problem is more complicated than that.

For example, rather than searching for a fixed substring like '123', suppose you wanted to determine whether a string contains any three consecutive decimal digit characters, as in the strings 'foo123bar', 'foo456bar', '234baz', and 'qux678'.

Strict character comparisons won't cut it here. This is where regexes in Python come to the rescue.

### 1.5.2 The Re Module

Regex functionality in Python resides in a module named `re`.

For now, you'll focus predominantly on one function, `re.search()`.

```
re.search(<regex>, <string>)
```

This function searches looking for the first location where the pattern `<regex>` matches. If a match is found, then `re.search()` returns a match object. Otherwise, it returns `None`.

[5]: `import re`

```
re.search('123', s)
```

[5]: `<re.Match object; span=(3, 6), match='123'>`

For the moment, the important point is that `re.search()` did in fact return a match object rather than `None`. That tells you that it found a match. In other words, the specified `<regex>` pattern `123` is present in `s`. The interpreter displays the match object as `<_sre.SRE_Match object; span=(3, 6), match='123'>`.

This contains some useful information:

- `span=(3, 6)` indicates the portion of in which the match was found. In this example, the match starts at character position 3 and extends up to but not including position 6.
- `match='123'` indicates which characters from `<string>` matched.

### 1.5.3 Python Regex Metacharacters

The real power of regex matching in Python emerges when `<regex>` contains special characters called metacharacters. These have a unique meaning to the regex matching engine and vastly enhance the capability of the search. Consider again the problem of how to determine whether a string contains any three consecutive decimal digit characters.

In a regex, a set of characters specified in square brackets (`[]`) makes up a character class. This metacharacter sequence matches any single character that is in the class, as demonstrated in the following example:

```
[17]: #
# [0-9] matches any single decimal digit character-any character between '0'
# → and '9', inclusive.
# The full expression [0-9][0-9][0-9] matches any sequence of three decimal
# → digit characters.
# On the other hand, a string that doesn't contain three consecutive digits
# → won't match!
#
pattern = '[0-9][0-9][0-9]'

mylist = ['gdash5622hjj', 'dafasfas', '654fdhaskjf', 'ashjdfuqo', '67yahd',
# → '9jhdk saf', '42hddh dh67', 'udyakh']
for l in mylist:
    if re.search(pattern, l):
        print(l)
```

```
gdash5622hjj
654fdhaskjf
```

With regexes in Python, you can identify patterns in a string that you wouldn't be able to find with the in operator or with string methods.

Take a look at another regex metacharacter. The dot (.) metacharacter matches any character except a newline, so it functions like a wildcard:

```
[18]: pattern = 'a.h'
for l in mylist:
    if re.search(pattern, l):
        print(l)
```

```
gdash5622hjj
ashjdfuqo
udyakh
```

Here, you're essentially asking, “Does s contain a ‘a’, then any character (except a newline), then a ‘h’?”.

A character class metacharacter sequence will match any single character contained in the class. You can enumerate the characters individually like this:

```
[19]: pattern = 'Nr[0-9]'

mylist = ['gdas askjd Nr59 dsafh', 'dafasfas', 'Nr47 adfd jads', 'dkajòqwo idf
# → Nr 78', '67yahd', '9jhdk saf', '42hddh dh67', 'udyakh']
for l in mylist:
    if re.search(pattern, l):
        print(l)
```

```
gdas askjd Nr59 dsafh
Nr47 adfd jads
```

You can complement a character class by specifying `^` as the first character, in which case it matches any character that isn't in the set. In the following example, `[^0-9]` matches any character that isn't a digit:

```
[13]: pattern = '[^0-9].'
```

```
words = 'All in all, the EU economy is forecast to grow by 4.6% in 2021 and \
        to strengthen to around 5.3% in 2022, 4.2% and 3.2% respectively, in_
        the euro area.'
```

```
#words = words.replace('.', '').replace(',', '')
```

```
mylist = words.split()
for l in mylist:
    if re.search(pattern, l):
        print(l, end= ' ')
```

All in all, the EU economy is forecast to grow by 4.6% in and to strengthen to around 5.3% in 4.2% and 3.2% respectively, in the euro area.

```
[17]: # match all non-blank characters
y = re.findall('[^ ]', words)
print(y)
# a simple way to remove all blank space in a string
print(''.join(y))
```

```
['A', 'l', 'l', 'i', 'n', 'a', 'l', 'l', ' ', 't', 'h', 'e', 'E', 'U', 'e', 'c',
'o', 'n', 'o', 'm', 'y', 'i', 's', 'f', 'o', 'r', 'e', 'c', 'a', 's', 't', 't',
'o', 'g', 'r', 'o', 'w', 'b', 'y', '4', '.', '6', '%', 'i', 'n', '2', '0', '2',
'1', 'a', 'n', 'd', 't', 'o', 's', 't', 'r', 'e', 'n', 'g', 't', 'h', 'e', 'n',
't', 'o', 'a', 'r', 'o', 'u', 'n', 'd', '5', '.', '3', '%', 'i', 'n', '2', '0',
'2', '2', ' ', '4', '.', '2', '%', 'a', 'n', 'd', '3', '.', '2', '%', 'r', 'e',
's', 'p', 'e', 'c', 't', 'i', 'v', 'e', 'l', 'y', ' ', 'i', 'n', 't', 'h', 'e',
'e', 'u', 'r', 'o', 'a', 'r', 'e', 'a', '.']
```

Allinall,theEUeconomyisforecasttogrowby4.6%in2021andtostrenghtentoaround5.3%in2022,4.2%and3.2%respectively,intheeuroarea.

```
[21]: wordlist = [w for w in nltk.corpus.words.words('en') if w.islower()]
```

```
[22]: [w for w in wordlist if re.search('^zu', w)]
```

```
[22]: ['zuccarino',
       'zucchetto',
       'zucchini',
       'zudda',
       'zugtierlast',
       'zugtierlaster',
       'zuisin',
```

```
'zumatic',  
'zumbooruk',  
'zunyite',  
'zupanate',  
'zuurveldt',  
'zuza']
```

Let's find words ending with **zz** using the regular expressio

```
[23]: [w for w in wordlist if re.search('zz$', w)]
```

```
[23]: ['abuzz',  
      'bejazz',  
      'bizz',  
      'blizz',  
      'brizz',  
      'bruzz',  
      'buzz',  
      'fizz',  
      'frizz',  
      'fuzz',  
      'gizz',  
      'hizz',  
      'humbuzz',  
      'huzz',  
      'jazz',  
      'muzz',  
      'outbuzz',  
      'outjazz',  
      'razz',  
      'sizz',  
      'unfrizz',  
      'zizz']
```

```
[24]: [w for w in wordlist if re.search('^app.*ed$', w)]
```

```
[24]: ['appearanced',  
      'appellatived',  
      'appendaged',  
      'appendiculated',  
      'applied',  
      'appressed']
```

```
[25]: chat_words = sorted(set(w for w in nltk.corpus.nps_chat.words()))  
      [w for w in chat_words if re.search('^[ha]+$ ', w)]
```

```
[25]: ['a',
      'aaaaaaaaaaaaaaaa',
      'aaahhhh',
      'ah',
      'ahah',
      'ahahah',
      'ahh',
      'ahhahahahaha',
      'ahhh',
      'ahhhh',
      'ahhhhhh',
      'ahhhhhhhhhhhhhh',
      'h',
      'ha',
      'haaa',
      'hah',
      'haha',
      'hahaaa',
      'hahah',
      'hahaha',
      'hahahaa',
      'hahahah',
      'hahahaha',
      'hahahahaaa',
      'hahahahahahaha',
      'hahahahahahaha',
      'hahahahahahahahahahahahahahaha',
      'hahahhahah',
      'hahhahahaha']
```

It should be clear that + simply means “one or more instances of the preceding item”, which could be an individual character like m, a set like [fed] or a range like [d-f]. Now let’s replace + with \*, which means “zero or more instances of the preceding item”.

#### 1.5.4 Further Examples

```
[38]: text=['X-Sieve: CMU Sieve 2.3', 'X-DSPAM-Result: Innocent', 'X-Plane is behind_
      ↪schedule: two weeks']
```

```
[39]: regex = r'^X.*:'
      for t in text:
          y = re.findall(regex, t)
          print(y)
```

```
['X-Sieve:']
['X-DSPAM-Result:']
['X-Plane is behind schedule:']
```

```
[40]: regex = r'^X-\S+:'
      for t in text:
          y = re.findall(regex, t)
          print(y)
```

```
['X-Sieve:']
['X-DSPAM-Result:']
[]
```

Greedy and Lazy Matching

```
[11]: # greedy
      x = 'From: Using the : character'
      y = re.findall('^F.+:', x)
      print(y)
```

```
['From: Using the :']
```

```
[12]: # lazy (note the question mark before ':')
      x = 'From: Using the : character'
      y = re.findall('^F.+?:', x)
      print(y)
```

```
['From:']
```

```
[7]: text = "From stephen.marquard@uct.ac.za, giovanni.dellalunga@unibo.it Sat Jan 5 09:14:16 2008"

      y = re.findall(r'\S+@\S+', text)
      print(y)
```

```
['stephen.marquard@uct.ac.za,', 'giovanni.dellalunga@unibo.it']
```

```
[9]: y = re.findall(r'^From (\S+@\S+)', text)
      print(y)
```

```
['stephen.marquard@uct.ac.za,']
```

```
[20]: text = "From giovanni.dellalunga@unibo.it Sat Jan 5 09:14:16 2008"
      regex = r'^From .*@([^\s]*)'

      y = re.findall(regex, text)
      print(y)
```

```
['unibo.it']
```

## 1.6 References and Credits

*Bird S. et al.*, “Natural Language Processing with Python” O’Reilly (2009)

*Bengfort B. et al.*, “*Applied Text Analysis with Python*” O’Reilly (2018)