Zolzaya Luvsandorj

Jun 20, 2020

https://towardsdatascience.com/simple-wordcloud-in-python-2ae54a9f58e5

Simple word cloud in Python

💡 ***Word cloud*** *is a technique for visualising frequent words in a text where the size of the words represents their frequency.*

One easy way to make a word cloud is to search 'word cloud' on Google to find one of those free websites that generate a word cloud. You can possibly customise how it looks like. Quick and easy!

What's more exciting is that you can build one yourself in Python 🤸. This post will show how to create a word cloud like the example below.

word cloud from section 3. Fancier word cloud ✿

0. Python setup 🔧

I assume the reader (👀 yes, you!) has access to and is familiar with Python including installing packages, defining functions and other basic tasks. If you are new to Python, this is a good place to get started.

I have used and tested the scripts in Python 3.7.1 in Jupyter Notebook.

Let's make sure you have the following libraries installed before we get started:

☐ **To create a word cloud:** *wordcloud*

☐ **To import an image:** *pillow* (will later import is as *PIL*)

☐ **To scrape text from Wikipedia:** *wikipedia*

Last package is optional, you can instead load up or create your own text data without having to pull text via web scraping.

1. Data 🎁

As our sample text, we will use scraped text from a Wikipedia page on 'Web scraping'.

```
# Import packages
import wikipedia
import re# Specify the title of the Wikipedia page
wiki = wikipedia.page('Web scraping')# Extract the plain text content of the
page
text = wiki.content# Clean text
text = re.sub(r'==.*?==+', '', text)
text = text.replace('\n', '')
```

🔗 I have explained what this script does in a separate post on scraping. In short, this script will pull out the plain text content in the paragraphs and assign it to *text* string.

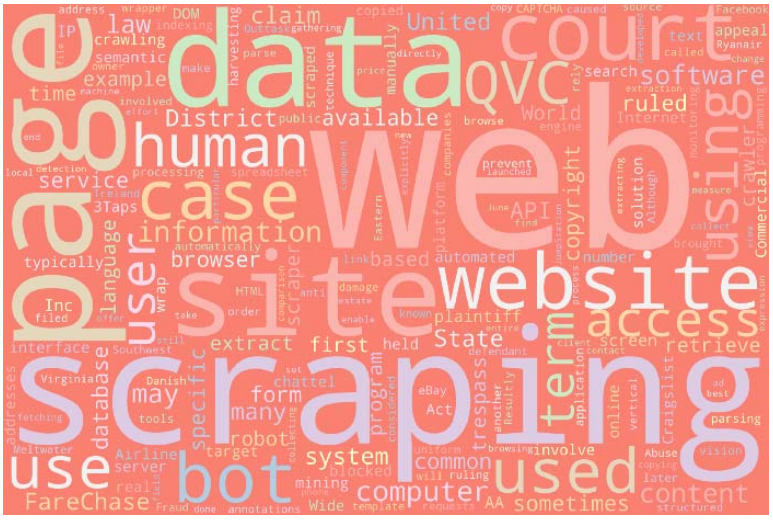
2. Word cloud ☁☐

Firstly, let's prepare a function that plots our word cloud:

```
# Import package
import matplotlib.pyplot as plt# Define a function to plot word cloud
def plot_cloud(wordcloud):
    # Set figure size
    plt.figure(figsize=(40, 30))
    # Display image
    plt.imshow(wordcloud)
    # No axis details
    plt.axis("off");
```

Secondly, let's create our first word cloud and plot it:

```
# Import package
from wordcloud import WordCloud, STOPWORDS# Generate word cloud
wordcloud = WordCloud(width= 3000, height = 2000, random_state=1,
background_color='salmon', colormap='Pastel1', collocations=False, stopwords =
STOPWORDS).generate(text)# Plot
plot_cloud(wordcloud)
```



Ta-da! We just built a word cloud! Here are some notes regarding the arguments for *WordCloud* function:

■ **width/height:** You can change the word cloud dimension to your preferred width and height with these.

■ **random_state:** If you don't this set this to a number of your choice, you are likely to get a slightly different word cloud every time you run the same script on the same input data. By setting this parameter, you ensure reproducibility of the exact same word cloud. You could play around with random numbers until you find the one that results in the word cloud you like.

■ **background_colour:** 'white' and 'black' are common background colours. If you would like to explore more colours, this may come in handy. Please note that some colours may not work. Hope you will find something you fancy.

■ **colormap:** With this argument, you can set up the colour theme that the words are displayed in. There are many beautiful Matplotlib colormaps to choose from. Some of my favourites are 'rainbow', 'seismic', 'Pastel1' and Pastel2'.

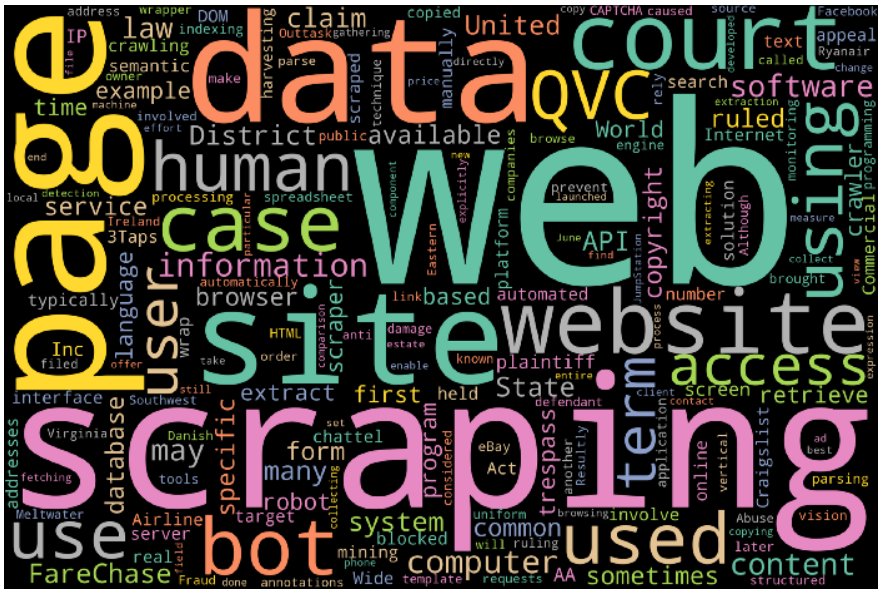
■□ **collocations:** Set this to *False* to ensure that the word cloud doesn't appear as if it contains any duplicate words. Otherwise, you may see 'web', 'scraping' and 'web scraping' as a collocation in the word cloud, giving an impression that words have been duplicated.

■□ **stopwords:** Stopwords are common words which provide little to no value to the meaning of the text. 'We', 'are' and 'the' are examples of stopwords. I have explained stopwords in more detail here (scroll to 'STEP3. REMOVE STOPWORDS' section). Here, we used *STOPWORDS* from the *wordcloud* package. To see the set of stopwords, use `print(STOPWORDS)` and to add custom stopwords to this set, use this template `STOPWORDS.update(['word1', 'word2'])`, replacing *word1* and *word2* with your custom stopwords before generating a word cloud.

There are other arguments that you can also customise. Check out the documentation for more information.

Let's generate another word cloud with a different *background_colour* and *colormap* 🎨. You could play with different combinations until you find the one that you like. I find the following combination quite nice:

```
# Generate wordcloud
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='black', colormap='Set2', collocations=False, stopwords =
STOPWORDS).generate(text)# Plot
plot_cloud(wordcloud)
```



Suppose we are happy with the word cloud and would like to save it as a .png file, we can do so using the code below:

```
# Save image
wordcloud.to_file("wordcloud.png")
```

3. Fancier word cloud ❁

By fancier word cloud, I mean those word clouds in custom shapes like the one shown at the beginning of this post. To create a fancy word cloud, we need to first find an image to use as a mask. It appears that the biggest challenge is to find the right image file. You may search for images with keywords: 'masking images for word cloud' on Google Images. During my search, I came across this source where a generous kaggler has shared some useful masking images. I used the 'upvote.png' to generate the word cloud at the start of this post with the following script (remember to save a copy of the masking image in the current directory before running the script):

```
# Import packages
import numpy as np
from PIL import Image# Import image to np.array
mask = np.array(Image.open('upvote.png'))# Generate wordcloud
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='white', colormap='Set2', collocations=False, stopwords =
STOPWORDS, mask=mask).generate(text)# Plot
plot_cloud(wordcloud)
```

You will notice that the only difference is that we have imported the image to a *numpy array* then added `mask=mask` in the *WordCloud*. One thing with masking is that it is best to set the background colour as 'white'. If we try changing to a different colour, the word cloud may not look as nice. Let's create one and see:

```
# Generate wordcloud
wordcloud = WordCloud(width = 3000, height = 2000, random_state=1,
background_color='navy', colormap='rainbow', collocations=False, stopwords =
STOPWORDS, mask=mask).generate(text)# Plot
plot_cloud(wordcloud)
```

What do you think of this compared to having a white background? I will let you be the judge of that.

I like word clouds and am planning to make one (definitely not about web scraping though! 🙈) to use as a poster to decorate my room. But it can also be used in other circumstances such as in presentations and documents as visual aid.

Would you like to access more content like this? Medium members get unlimited access to any articles on Medium. If you become a member using [my referral link](), a portion of your membership fee will directly go to support me.

Thank you for reading my post. I hope that you have learned something ✂□. In case you are interested, here are links to some of my other posts:

■ □ Two simple ways to scrape text from Wikipedia in Python

(Below lists a series of posts on Introduction to NLP)

■ □ Part 1: Preprocessing text in Python

■ □ Part 2: Difference between lemmatisation and stemming

■ □ Part 3: TF-IDF explained

■ □ Part 4: Supervised text classification model in Python

■ □ Part 5A: Unsupervised topic model in Python (sklearn)

■ □ Part 5B: Unsupervised topic model in Python (gensim)