Week 8: Unstructured data; preprocessing

Objective: In this lab session, you will learn:

1. Image preprocessing
2. Text preprocessing

We will use Google Colab - Python IDE. You will need to log in to use the Google Colab. You can use your university email or your personal.

**Image preprocessing**

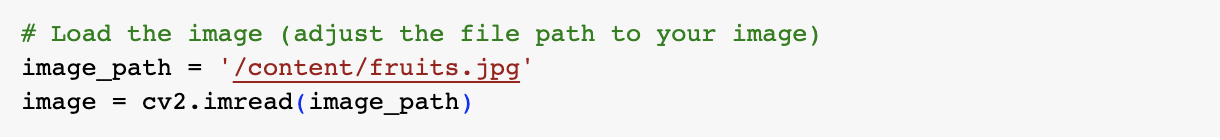
Images, like other forms of data, can also pass through the data pipelines, meaning images can be ingested, transformed and loaded into storage. This section is mainly focused on ingestion and transformation.

1. Download the fruit image from the BB.
2. Upload the data to your Colab.
3. Import all the necessary libraries.

A white background with words

Description automatically generated

1. Ingest image data



You may want to copy the image data file path from collab and replace ‘/content/fruits.jpg’.

1. Display the image

A screenshot of a computer

Description automatically generated

OpenCV loads images in BGR format, as such, there is a need to change BGR to RGB.

Now, let’s begin to transform(preprocess) the image.

1. Resize the image to a certain width \* height

A screenshot of a computer program

Description automatically generated

Here, the image was resized to 500 x 500, saved and then displayed.

1. Rotate the image

A screen shot of a computer code

Description automatically generated

Here, the image was rotated 90 degrees clockwise, saved and then displayed. We can rotate it to other degrees to enhance the quality of the image.

1. Denoise the image

A screenshot of a computer program

Description automatically generated

Here, the image was denoised using GaussianBlur, saved and then displayed.

You may have noticed that we've been using the original image\_rgb file separately for each individual processing technique. Instead, let's try applying all the techniques in sequence—first resizing, then rotating, and finally denoising—to obtain a fully processed image.

1. Apply resize, rotation and denoise on an image.

A screenshot of a computer program

Description automatically generated

Here, we applied the techniques in a specific order: resizing the image, rotating, and finally denoising. However, you can follow a different sequence. Instead, choose the order and particular techniques that best enhance the image quality based on your needs.

The image was processed, saved and then displayed.

1. Image Annotation

A close-up of a computer screen

Description automatically generated

We manually created metadata(annotation) by making a dictionary with the image name as the key, containing keywords and descriptions.

1. Save the metadata as a JSON file.

A close-up of words

Description automatically generated

Here, the metadata is saved as a JSON file for later use.

**Image Feature Extraction**

Extract features such as mean intensity and norm intensity.

1. Calculate the mean and norm intensity of the processed image.

A close up of words

Description automatically generated

Here, calculate the processed data's mean and norm intensity. We saved the mean and norm as variables, each for later use.

1. Extract shape features

A screenshot of a computer program

Description automatically generated

Here, we started by converting the image to grayscale, and then we computed the area, perimeter, centroid, and bounding box as examples of shape features.

1. Metadata for all the image features extracted from tasks 12-13.

A screenshot of a computer

Description automatically generatedHere, the shape features (area, perimeter, centroid, bounding box) were first put together and then combined with mean and norm intensity. All the image features were extracted and put together as a dictionary and metadata file for the processed image. The features extracted were saved as a JSON file for later use.

**Text processing**

This section focuses on text data's ingestion and transformation(preprocessing).

1. Download the document named 101551 from the BB.
2. Upload the data to your Colab.
3. Import all the necessary libraries.

A computer screen shot of a library

Description automatically generated

1. Download stopwords and punkt tokeniser

A green text on a white background

Description automatically generated

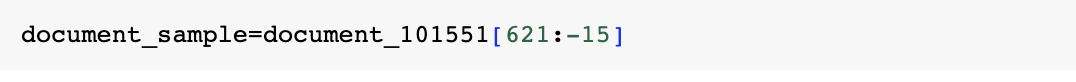
Here, we download the list of stop words for the English language. Note that each language typically has its own set of rules, so different languages may require their specific dictionary to identify stop words.

1. Load the document file

A computer code with red and green text

Description automatically generated with medium confidence

1. Filter out some parts of the document and create data out of that.

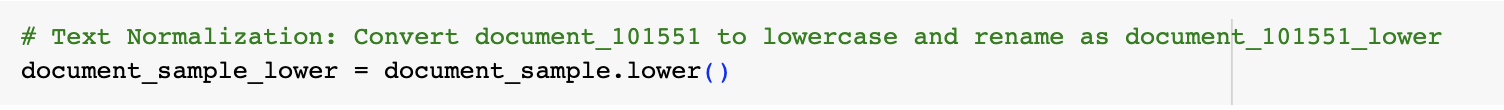


We created new text data by filtering out the first 621 and last 15 characters.

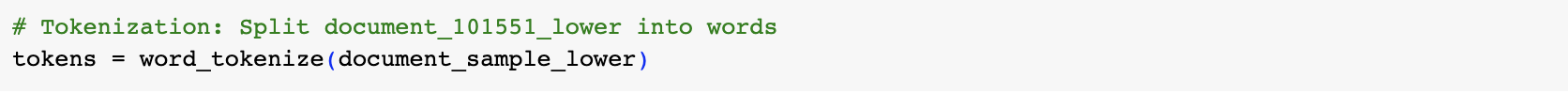
Preprocessing Techniques:

* Text Normalization (Lowercasing)
* Tokenization & Removal of Punctuation and Stop Words
* Stemming reduces a word to its root or base form by removing prefixes and suffixes.
* Lemmatization reduces word to their dictionary base form (lemma)

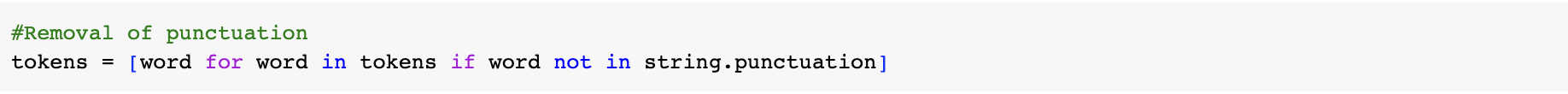
1. Convert the sample document to lowercase all through.



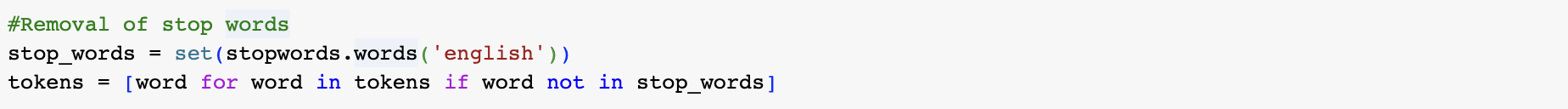
1. Tokenise the sample document.



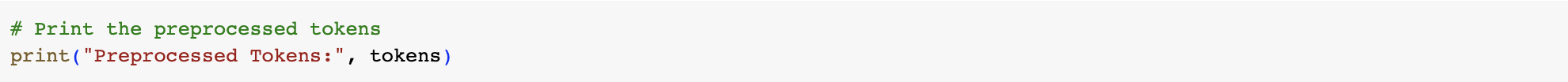
1. Remove all punctuation marks from the tokens.



1. Remove stop words from the tokens.



1. Print the processed tokens.



1. Apply stemming on the tokens.

A screenshot of a computer program

Description automatically generated

Here, we initialise a stemmer and a lemmatiser because they have to do with rules; as such, a sort of word dictionary that helps define suffixes, prefixes, and root words is required.

1. Apply lemmatisation to the tokens

A close-up of words

Description automatically generated

Extra Exercise.

Convert the sample document from task 7 into a numerical representation (i.e., vectorisation).

1. Using Bag of Words (BoW)
2. Using Term Frequency-Inverse Document Frequency (TF-IDF)

**Metadata and labelling for text data**

1. Create metadata for the processed text data.

**A close-up of a white page

Description automatically generated**

Here, we created a metadata document explaining the vectorisation process and labelled the text data.

1. Save the metadata file for the text data as a JSON file.

