A

Synopsis/Project Report

On

Creating a WordCloud using NLP and TF-IDF in Python

Submitted in partial fulfilment of the requirement for the IV semester

Bachelor of Computer Science

By

Lalit Dumka

Under the Guidance of

Mr. Ravindra Koranga

Assistant Professor

Department of CSE

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

GRAPHIC ERA HILL UNIVERSITY, BHIMTAL CAMPUS

SATTAL ROAD, P.O. BHOWALI,

DISTRICT- NAINITAL-263132

2022- 2023

<u>STUDENT'S DECLARATION</u>

I **Lalit Dumka** here by declare the work, which is being presented in the project, entitled "**Creating a Wordcloud using NLP and TF-IDF in Python**" In partial fulfillment of the requirement for the award of the degree **B. Tech** in the session **2022-2023**, is an authentic record of my own work carried out under the supervision of "**Mr. Ravindra Koranga", Assistant Professor, Department of CSE, Graphic Era Hill University, Bhimtal.**

The matter embodied in this project has not been submitted by us for the award of any other degree.

Date:

Lalit Dumka

CERTIFICATE

The project report entitled "Creating a wordcloud using NLP and TF-IDF in Python " Being submitted by Lalit Dumka to Graphic Era Hill University Bhimtal Campus for the award of Bonafede work carried out by them. They have worked under my guidance and supervision and fulfilled the requirement for the submission of the report.

(Mr. Ravindra Koranga)
Project Guide

(Dr. Ankur Bisht)
(HOD, CSE Dept.)

ACKNOWLEDGEMENT

I take immense pleasure in thanking Honorable **“Mr Ravindra Koranga”** (**Assistant Professor, CSE, GEHU Bhimtal Campus**) to permit us and carry out this project work with his excellent and optimistic supervision. This has all been possible due to his novel inspiration, able guidance and useful suggestions that helped me to develop as a creative researcher and complete the research work, in time.

Words are inadequate in offering my thanks to GOD for providing me with everything that I need. I again want to extend thanks to our President **“Prof. (Dr.) Kamal Ghanshala”** for providing us with all infrastructure and facilities to work in need without which this work could not be possible.

Many thanks to Professor **“Dr Manoj Chandra Lohani”** (Director Gehu Bhimtal), other faculties for their insightful comments, constructive suggestions, valuable advice, and time in reviewing this thesis.

Finally, yet importantly, I would like to express my heartiest thanks to our beloved parents, for their moral support, affection and blessings. I would also like to sincerely thank all our friends and well-wishers for their help and wishes for the successful completion of this research.

Lalit Dumka

Karanpratap Singh Kharka

TABLE OF CONTENTS

Declaration…………………………………………………………………………..I

Certificate…………………………………………………………………………II

Acknowledgement………………………………………………………………..III

Abstract……………………………………………………………………………IV

CHAPTER 1: INTRODUCTION…………………………………………….

1.1 Objective……………………………………………………….

1.2 Background and Motivations………………………………….

1.3 Problem Statement…………………………………………….

1.4 Project Organization…………………………………………..

CHAPTER 2: WORKING……………………………………………...…

2.1 Block Diagram of TF – IDF ……………………………...

2.2 Steps in pre-processing of data…………………………….

CHAPTER 3: TECHNOLOGY USED…………………………………….

3.1 Basic modules of the project………………………………….

CHAPTER 4: CODE……………………...………………………………….

4.1 Screenshots of Jupyter Notebook with code and outputs……..

CHAPTER 5: OUTPUT …………………………………………………….

5.1 WordCloud image……………………………….……………

5.2 Text File output………………………………………………

CHAPTER 6: BENEFITS.………………………………………………….

CHAPTER 7: CONCLUSION……………………………………………………..

PROJECT ABSTRACT

In this small project, I aim to leverage Natural Language Processing (NLP) techniques and the TF-IDF algorithm to generate an insightful word cloud from a given text corpus. WordClouds are powerful visual representations that highlight the most significant words in a document, providing a quick overview of its main themes.

To accomplish this, I will employ Python and its popular NLP libraries, such as NLTK (Natural Language Toolkit) and scikit-learn. The project will involve several key steps. Firstly, I will preprocess the text data by removing stop words, punctuation, and performing tokenization. Then, utilizing the TF-IDF algorithm, I will compute the importance score for each word in the corpus. The TF-IDF score takes into account both the term frequency and inverse document frequency, helping to identify words that are prevalent in a specific document while being relatively rare across the entire corpus.

INTRODUCTION

The objective of this small project is to create a wordcloud using Natural Language Processing (NLP) techniques and the TF-IDF algorithm in Python. NLP is a field of artificial intelligence that focuses on the interaction between computers and human language. TF-IDF, short for Term Frequency-Inverse Document Frequency, is a widely used numerical statistic in NLP that evaluates the importance of a word within a document relative to a larger corpus.

By combining the power of NLP and TF-IDF, we can create a wordcloud that not only showcases the most prominent words in a text corpus but also emphasizes words that are unique to a particular document while being infrequent in the overall corpus. This approach allows us to capture the essence of the text and visualize its main themes at a glance.

The resulting wordcloud can be a valuable tool in various applications, including sentiment analysis, content summarization, and data exploration. It enables researchers, analysts, and data scientists to quickly identify the significant keywords in a document and gain valuable insights from large volumes of text.

OBJECTIVE

The objective of this project is to leverage Natural Language Processing (NLP) techniques and the TF-IDF algorithm in Python to create a wordcloud that effectively summarizes the key themes and important words within a given text corpus.

This project aims to provide a practical demonstration of how NLP techniques and the TF-IDF algorithm can be applied to create a wordcloud that effectively summarizes the main themes and important keywords within a given text corpus. The resulting wordcloud can serve as a valuable tool for data exploration, content summarization, and information visualization in various domains.

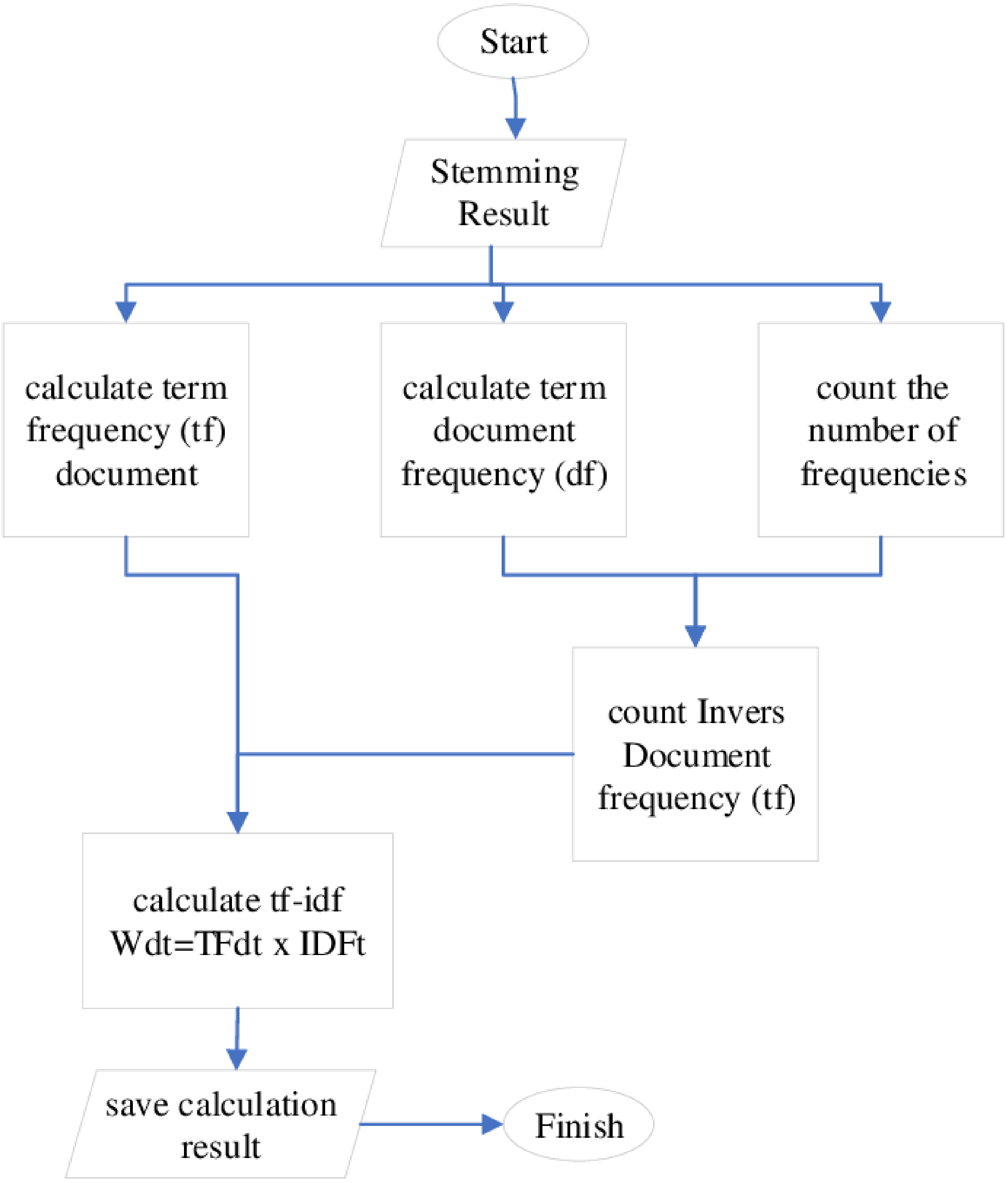
PROBLEM STATEMENT

In a large text corpus, there is a need to identify and extract the most significant keywords that are representative of the main themes within the corpus. Traditional keyword extraction methods based solely on word frequency may not accurately capture the importance and relevance of specific terms within the corpus.

The problem is to develop a solution that effectively analyzes the text corpus and utilizes NLP and TF-IDF to compute the importance score for each word. By considering both the term frequency and inverse document frequency, the solution should accurately identify keywords that are highly relevant to individual documents and relatively rare across the entire corpus.

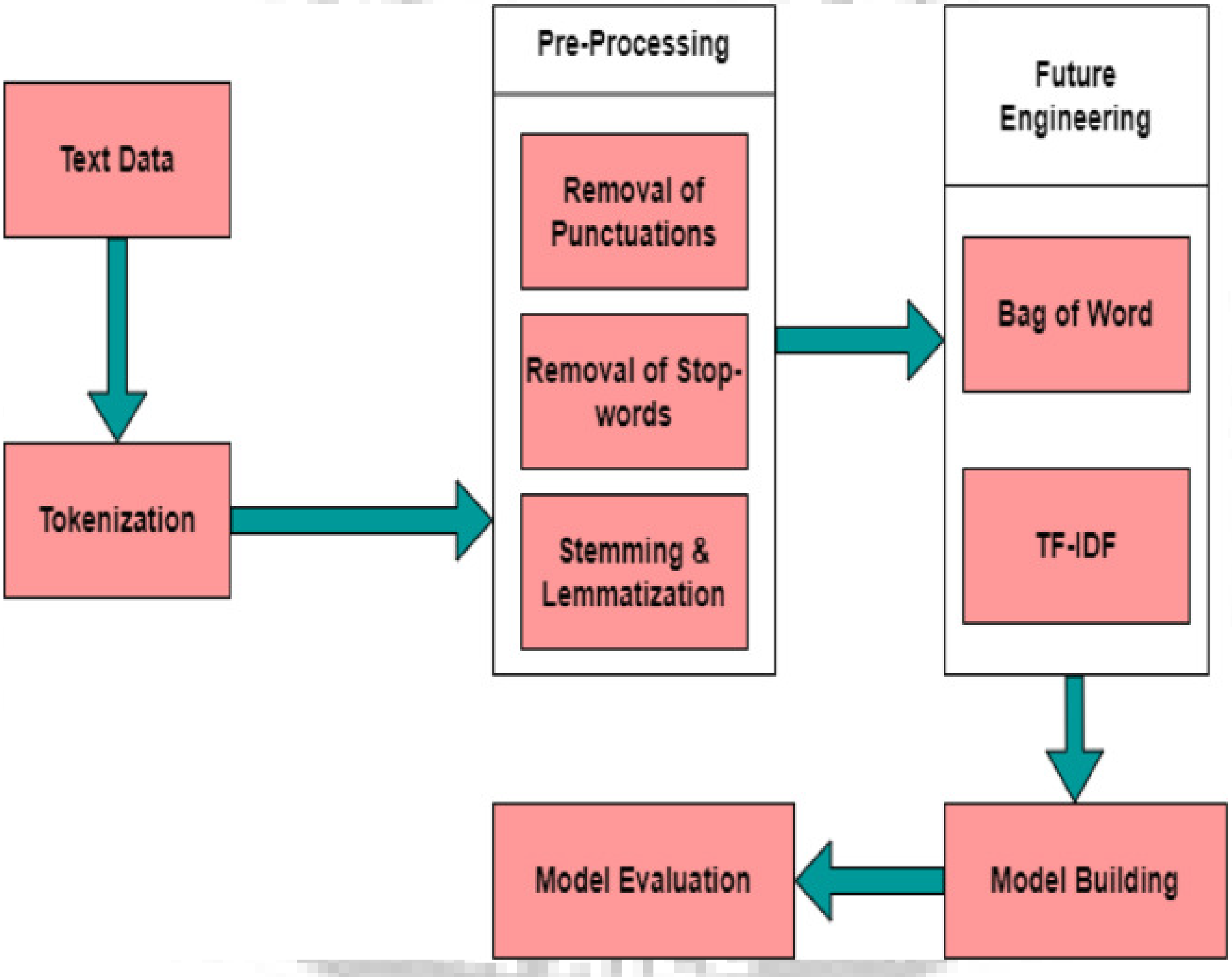
The solution should provide a means to preprocess the text data by removing stop words, punctuation, and performing tokenization. It should then calculate the TF-IDF scores for each word in the corpus, using appropriate weighting schemes to ensure the significance of each term is accurately represented.

Block Diagram of TF-IDF weight Calculation



Start
Stemming Result
calculate term frequency (tf) document
calculate term document frequency (df)
count the number of frequencies
count Invers Document frequency (tf)
calculate tf-idf Wdt=TFdt x IDFt
save calculation result
Finish

Steps in pre-processing of data



Resources and Technology used

Language and Libraries Used

Our project will be a wordcloud and it should be responsive for each accessible device. Hence, we are using these Technologies for making the project easier.

- ✓ **Python:** Python is a widely used programming language for NLP tasks due to its simplicity and availability of numerous libraries.
- ✓ **NLTK (Natural Language Toolkit):** NLTK is a popular Python library for natural language processing. It provides various tools and resources for tokenization, stemming, lemmatization, and other NLP tasks.
- ✓ **scikit-learn:** scikit-learn is a powerful machine learning library in Python. It includes efficient implementations of the TF-IDF vectorization algorithm.
- ✓ **WordCloud:** WordCloud is a Python library that allows you to create visually appealing word clouds from text data. It provides various customization options such as color palettes, mask shapes, and font styles.
- ✓ **Matplotlib:** Matplotlib is a widely used plotting library in Python. It can be used to visualize the generated word cloud.

Editor/Software Used

- ✓ VS code
- ✓ Jupyter Web

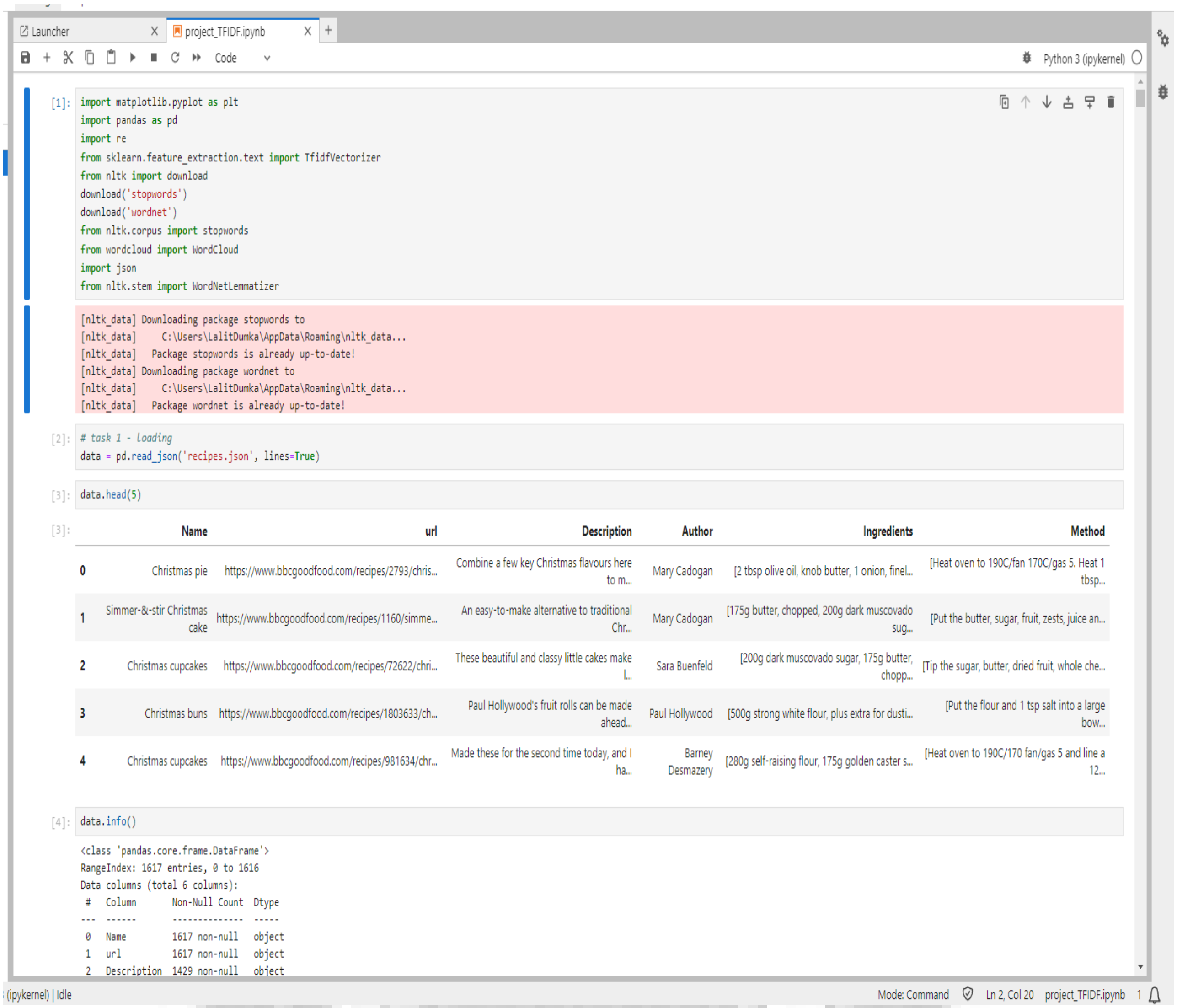
Basic modules of the project

Our project is divided into four modules:

- ✓ NLTK
- ✓ scikit-learn
- ✓ WordCloud
- ✓ Matplotlib

- **nltk:** The Natural Language Toolkit (NLTK) is a widely used library for natural language processing tasks. It provides various modules for text preprocessing, tokenization, stemming, lemmatization, and more.
- **sklearn:** The scikit-learn library is a powerful machine learning library that includes a Tf idf Vectorizer class. This class is used to convert text data into TF-IDF feature vectors, which are essential for computing TF-IDF scores.
- **wordcloud:** The wordcloud module is used to create visually appealing word clouds from text data. It provides the WordCloud class, which allows you to customize the appearance of the word cloud.
- **matplotlib:** Matplotlib is a popular plotting library in Python. It is used to visualize the generated word cloud.

CODE AND OUTPUT



[1]:
```
import matplotlib.pyplot as plt
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import download
download('stopwords')
download('wordnet')
from nltk.corpus import stopwords
from wordcloud import WordCloud
import json
from nltk.stem import WordNetLemmatizer
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\LalitDumka\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\LalitDumka\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

[2]:
```
# task 1 - loading
data = pd.read_json('recipes.json', lines=True)
```

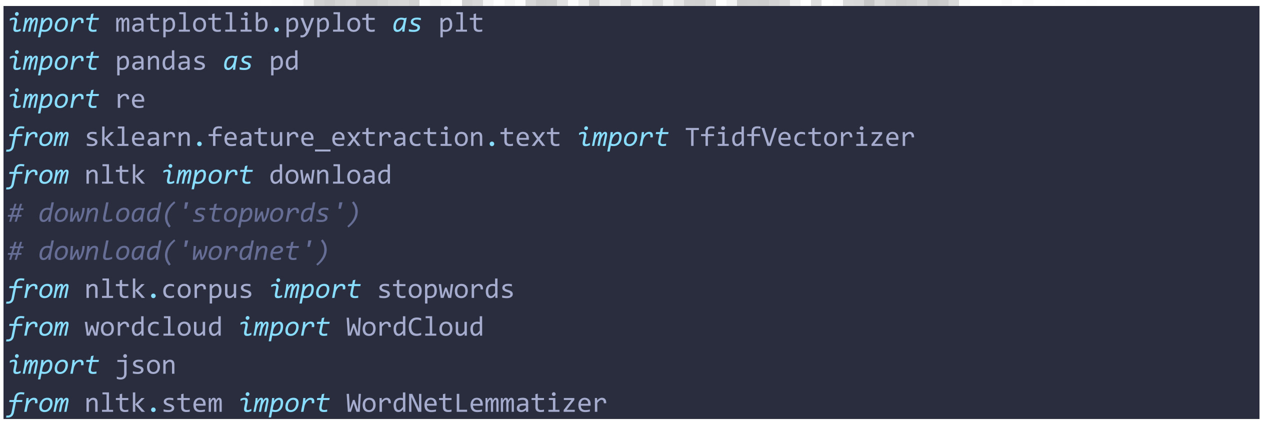
[3]:
```
data.head(5)
```

[3]:

	Name	url	Description	Author	Ingredients	Method
0	Christmas pie	https://www.bbcgoodfood.com/recipes/2793/chris...	Combine a few key Christmas flavours here to m...	Mary Cadogan	[2 tbsp olive oil, knob butter, 1 onion, finel...	[Heat oven to 190C/fan 170C/gas 5. Heat 1 tbsp...
1	Simmer-&-stir Christmas cake	https://www.bbcgoodfood.com/recipes/1160/simme...	An easy-to-make alternative to traditional Chr...	Mary Cadogan	[175g butter, chopped, 200g dark muscovado sug...	[Put the butter, sugar, fruit, zests, juice an...
2	Christmas cupcakes	https://www.bbcgoodfood.com/recipes/72622/chri...	These beautiful and classy little cakes make l...	Sara Buenfeld	[200g dark muscovado sugar, 175g butter, chopp...	[Tip the sugar, butter, dried fruit, whole che...
3	Christmas buns	https://www.bbcgoodfood.com/recipes/1803633/ch...	Paul Hollywood's fruit rolls can be made ahead...	Paul Hollywood	[500g strong white flour, plus extra for dusti...	[Put the flour and 1 tsp salt into a large bow...
4	Christmas cupcakes	https://www.bbcgoodfood.com/recipes/981634/chr...	Made these for the second time today, and I ha...	Barney Desmazery	[280g self-raising flour, 175g golden caster s...	[Heat oven to 190C/170 fan/gas 5 and line a 12...

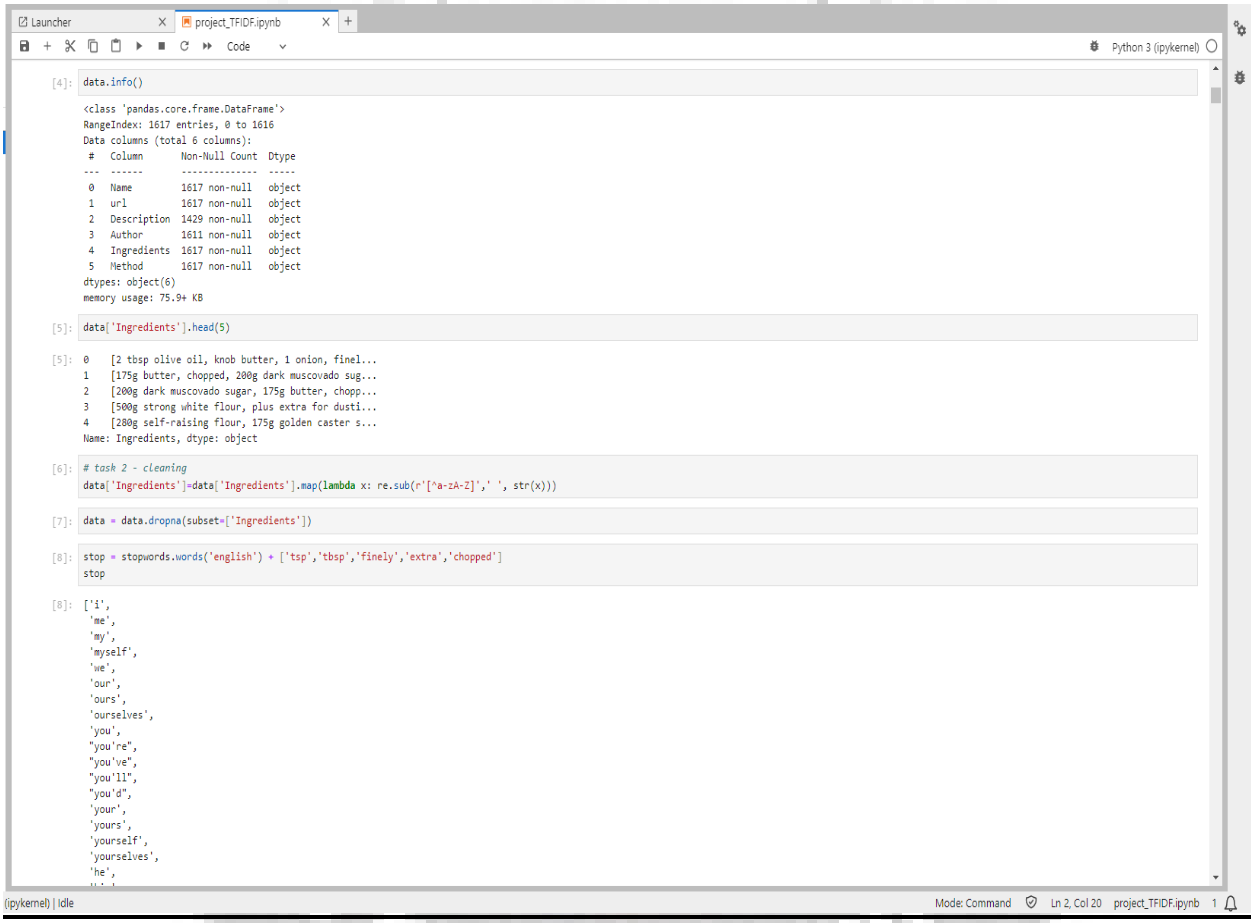
[4]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1617 entries, 0 to 1616
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Name         1617 non-null   object
 1   url          1617 non-null   object
 2   Description  1429 non-null   object
```



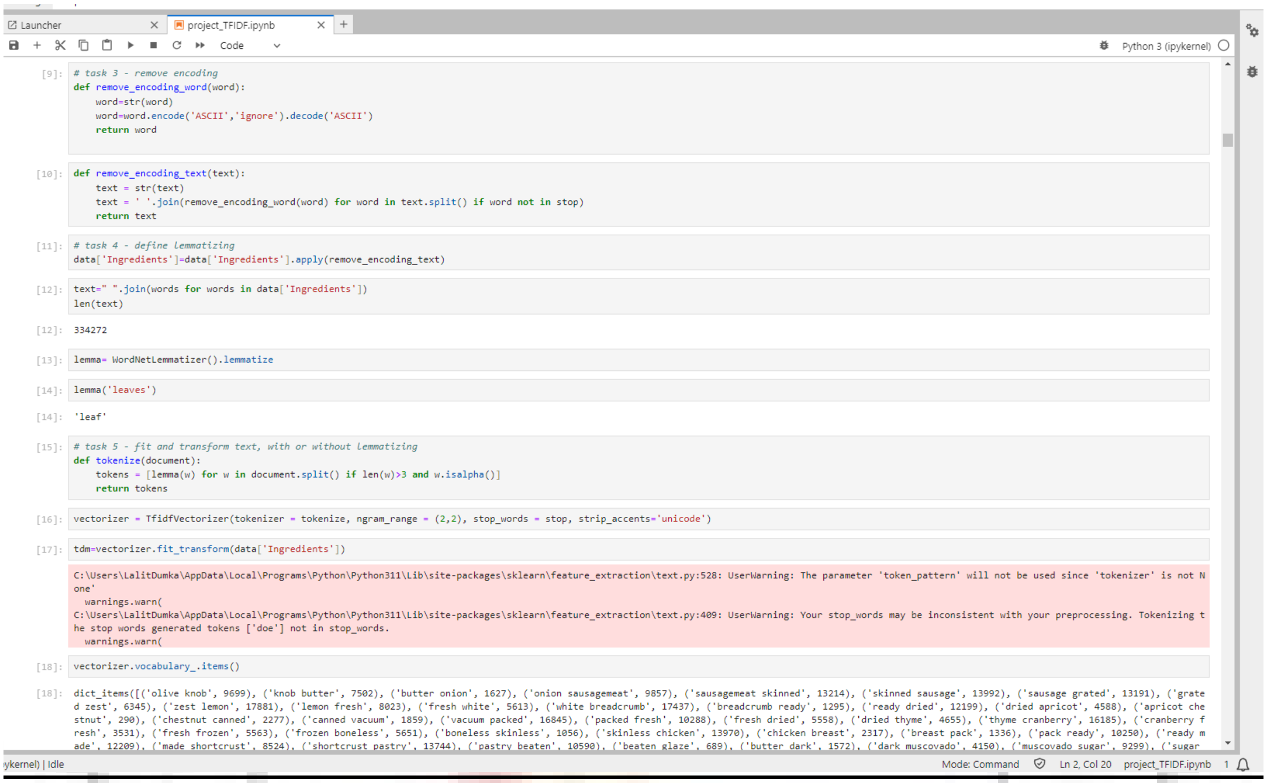
```
import matplotlib.pyplot as plt
import pandas as pd
import re
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import download
# download('stopwords')
# download('wordnet')
from nltk.corpus import stopwords
from wordcloud import WordCloud
import json
from nltk.stem import WordNetLemmatizer
```

```
# task 1 - loading
data = pd.read_json('recipes.json', lines=True)
data.head(5)
data.info()
data['Ingredients'].head(5)
```

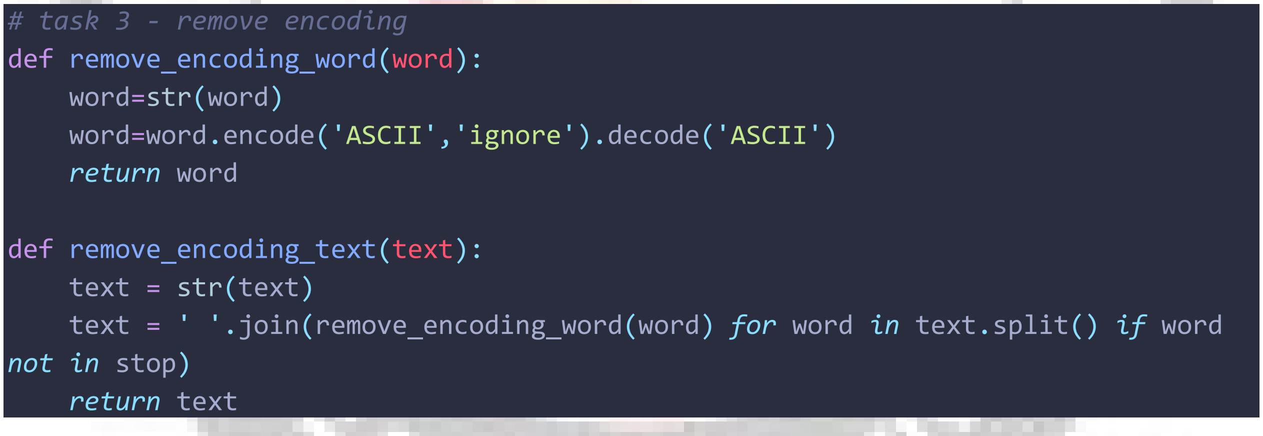


```
[4]: data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1617 entries, 0 to 1616
Data columns (total 6 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   Name         1617 non-null   object
 1   url          1617 non-null   object
 2   Description  1429 non-null   object
 3   Author       1611 non-null   object
 4   Ingredients  1617 non-null   object
 5   Method       1617 non-null   object
dtypes: object(6)
memory usage: 75.9+ KB

[5]: data['Ingredients'].head(5)

[5]: 0    [2 tbsp olive oil, knob butter, 1 onion, finel...
     1    [175g butter, chopped, 200g dark muscovado sug...
     2    [200g dark muscovado sugar, 175g butter, chopp...
     3    [500g strong white flour, plus extra for dusti...
     4    [280g self-raising flour, 175g golden caster s...
     Name: Ingredients, dtype: object

[6]: # task 2 - cleaning
     data['Ingredients']=data['Ingredients'].map(lambda x: re.sub(r'[^a-zA-Z]',' ', str(x)))

[7]: data = data.dropna(subset=['Ingredients'])

[8]: stop = stopwords.words('english') + ['tsp','tbsp','finely','extra','chopped']
     stop

[8]: ['i',
      'me',
      'my',
      'myself',
      'we',
      'our',
      'ours',
      'ourselves',
      'you',
      "you're",
      "you've",
      "you'll",
      "you'd",
      'your',
      'yours',
      'yourself',
      'yourselves',
      'he',
```

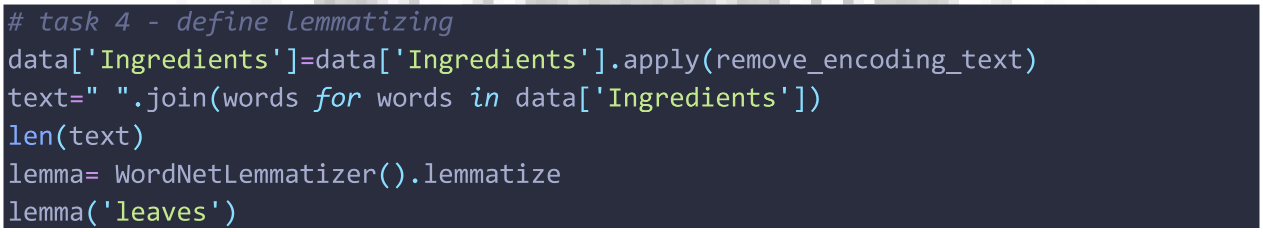
```
# task 2 - cleaning
data['Ingredients']=data['Ingredients'].map(lambda x: re.sub(r'[^a-zA-Z]',' ',
str(x)))
data = data.dropna(subset=['Ingredients'])
stop = stopwords.words('english') + ['tsp','tbsp','finely','extra','chopped']
stop
```



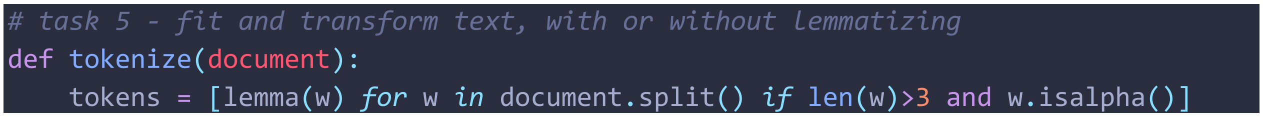
```
[9]: # task 3 - remove encoding
     def remove_encoding_word(word):
         word=str(word)
         word=word.encode('ASCII','ignore').decode('ASCII')
         return word

[10]: def remove_encoding_text(text):
          text = str(text)
          text = ' '.join(remove_encoding_word(word) for word in text.split() if word not in stop)
          return text

[11]: # task 4 - define lemmatizing
      data['Ingredients']=data['Ingredients'].apply(remove_encoding_text)

[12]: text=" ".join(words for words in data['Ingredients'])
      len(text)

[12]: 334272

[13]: lemma= WordNetLemmatizer().lemmatize

[14]: lemma('leaves')

[14]: 'leaf'

[15]: # task 5 - fit and transform text, with or without lemmatizing
      def tokenize(document):
          tokens = [lemma(w) for w in document.split() if len(w)>3 and w.isalpha()]
          return tokens

[16]: vectorizer = TfidfVectorizer(tokenizer = tokenize, ngram_range = (2,2), stop_words = stop, strip_accents='unicode')

[17]: tdm=vectorizer.fit_transform(data['Ingredients'])

C:\Users\LalitDumka\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_extraction\text.py:528: UserWarning: The parameter 'token_pattern' will not be used since 'tokenizer' is not N
one'
  warnings.warn(
C:\Users\LalitDumka\AppData\Local\Programs\Python\Python311\Lib\site-packages\sklearn\feature_extraction\text.py:409: UserWarning: Your stop_words may be inconsistent with your preprocessing. Tokenizing t
he stop words generated tokens ['doe'] not in stop_words.
  warnings.warn(

[18]: vectorizer.vocabulary_.items()

[18]: dict_items([('olive knob', 9699), ('knob butter', 7502), ('butter onion', 1627), ('onion sausagemeat', 9857), ('sausagemeat skinned', 13214), ('skinned sausage', 13992), ('sausage grated', 13191), ('grate
d zest', 6345), ('zest lemon', 17881), ('lemon fresh', 8023), ('fresh white', 5613), ('white breadcrumb', 17437), ('breadcrumb ready', 1295), ('ready dried', 12199), ('dried apricot', 4588), ('apricot che
stnut', 290), ('chestnut canned', 2277), ('canned vacuum', 1859), ('vacuum packed', 16845), ('packed fresh', 10288), ('fresh dried', 5558), ('dried thyme', 4655), ('thyme cranberry', 16185), ('cranberry f
resh', 3531), ('fresh frozen', 5563), ('frozen boneless', 5651), ('boneless skinless', 1056), ('skinless chicken', 13970), ('chicken breast', 2317), ('breast pack', 1336), ('pack ready', 10250), ('ready m
ade', 12209), ('made shortcrust', 8524), ('shortcrust pastry', 13744), ('pastry beaten', 10590), ('beaten glaze', 689), ('butter dark', 1572), ('dark muscovado', 4150), ('muscovado sugar', 9299), ('sugar
```



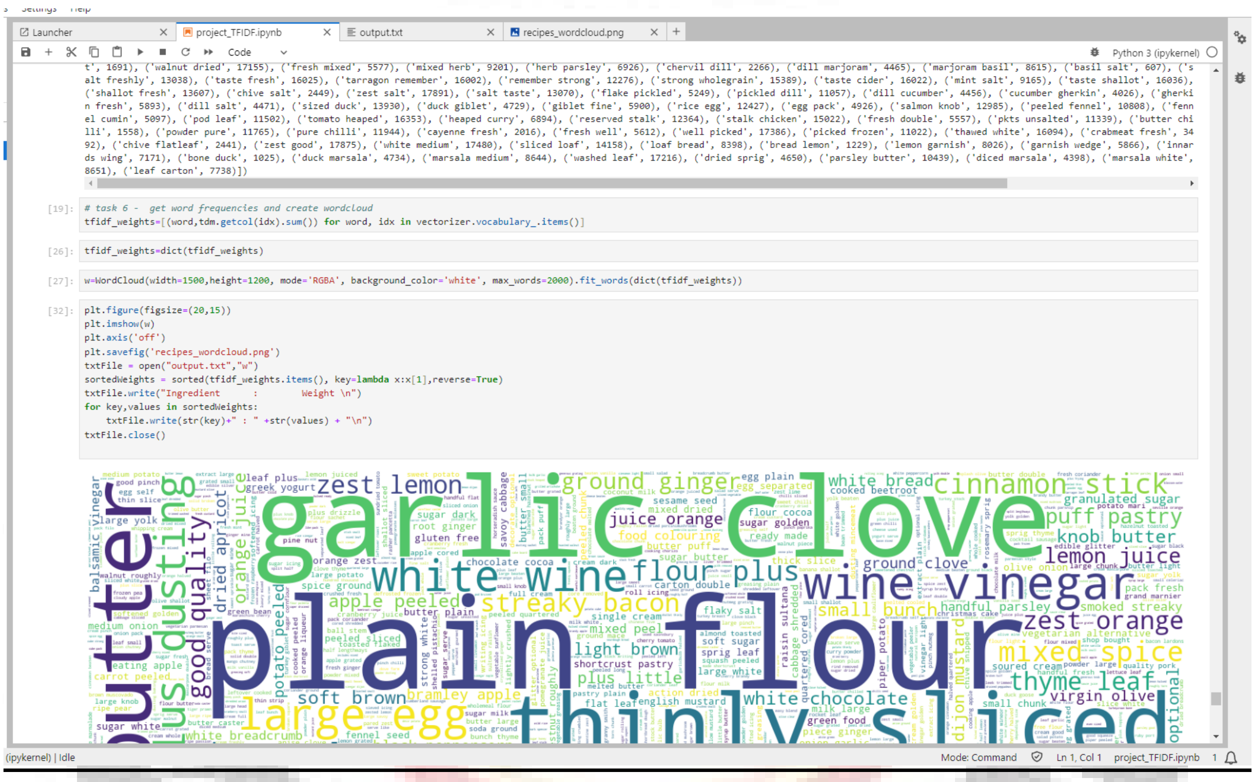
```
# task 3 - remove encoding
def remove_encoding_word(word):
    word=str(word)
    word=word.encode('ASCII','ignore').decode('ASCII')
    return word

def remove_encoding_text(text):
    text = str(text)
    text = ' '.join(remove_encoding_word(word) for word in text.split() if word
not in stop)
    return text
```



```
# task 4 - define lemmatizing
data['Ingredients']=data['Ingredients'].apply(remove_encoding_text)
text=" ".join(words for words in data['Ingredients'])
len(text)
lemma= WordNetLemmatizer().lemmatize
lemma('leaves')
```



```
# task 5 - fit and transform text, with or without lemmatizing
def tokenize(document):
    tokens = [lemma(w) for w in document.split() if len(w)>3 and w.isalpha()]
    return tokens
vectorizer = TfidfVectorizer(tokenizer = tokenize, ngram_range = (2,2),
stop_words = stop, strip_accents='unicode')
tdm=vectorizer.fit_transform(data['Ingredients'])
vectorizer.vocabulary_.items()
```



```
t', 1691), ('walnut dried', 17155), ('fresh mixed', 5577), ('mixed herb', 9201), ('herb parsley', 6926), ('chervil dill', 2266), ('dill marjoram', 4465), ('marjoram basil', 8615), ('basil salt', 607), ('s
alt freshly', 13038), ('taste fresh', 16025), ('tarragon remember', 16002), ('remember strong', 12276), ('strong wholegrain', 15389), ('taste cider', 16022), ('mint salt', 9165), ('taste shallot', 16036),
('shallot fresh', 13607), ('chive salt', 2449), ('zest salt', 17891), ('salt taste', 13070), ('flake pickled', 5249), ('pickled dill', 11057), ('dill cucumber', 4456), ('cucumber gherkin', 4026), ('gherki
n fresh', 5893), ('dill salt', 4471), ('sized duck', 13930), ('duck giblet', 4729), ('giblet fine', 5900), ('rice egg', 12427), ('egg pack', 4926), ('salmon knob', 12985), ('peeled fennel', 10808), ('fenn
el cumin', 5097), ('pod leaf', 11502), ('tomato heaped', 16353), ('heaped curry', 6894), ('reserved stalk', 12364), ('stalk chicken', 15022), ('fresh double', 5557), ('pkts unsalted', 11339), ('butter chi
lli', 1558), ('powder pure', 11765), ('pure chilli', 11944), ('cayenne fresh', 2016), ('fresh well', 5612), ('well picked', 17386), ('picked frozen', 11022), ('thawed white', 16094), ('crabmeat fresh', 34
92), ('chive flatleaf', 2441), ('zest good', 17875), ('white medium', 17480), ('sliced loaf', 14158), ('loaf bread', 8398), ('bread lemon', 1229), ('lemon garnish', 8026), ('garnish wedge', 5866), ('innar
ds wing', 7171), ('bone duck', 1025), ('duck marsala', 4734), ('marsala medium', 8644), ('washed leaf', 17216), ('dried sprig', 4650), ('parsley butter', 10439), ('diced marsala', 4398), ('marsala white',
8651), ('leaf carton', 7738)])
```

```
# task 6 -  get word frequencies and create wordcloud
tfidf_weights=[(word,tdm.getcol(idx).sum()) for word, idx in vectorizer.vocabulary_.items()]
```

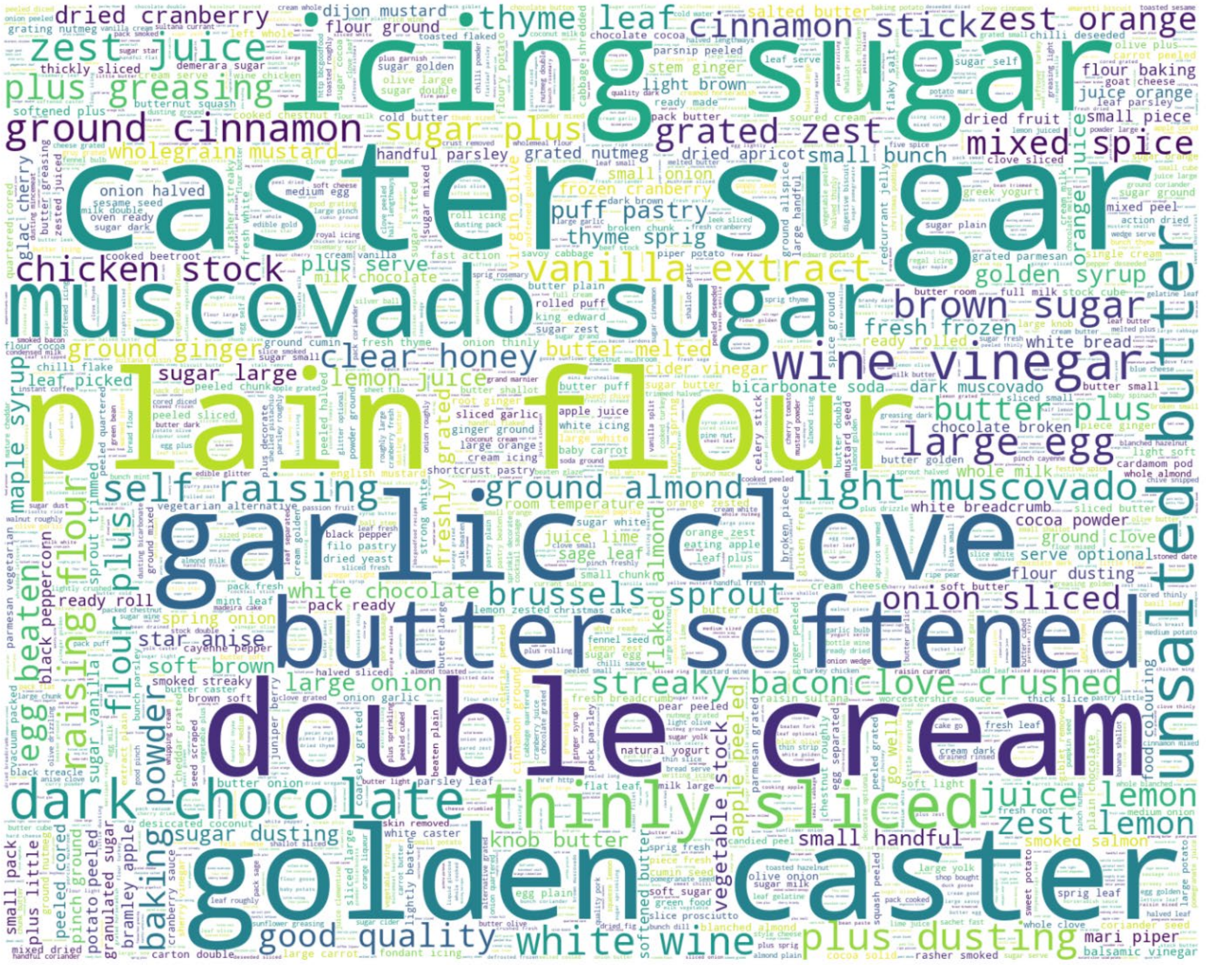
```
tfidf_weights=dict(tfidf_weights)
```

```
w=WordCloud(width=1500,height=1200, mode='RGBA', background_color='white', max_words=2000).fit_words(dict(tfidf_weights))
```

```
plt.figure(figsize=(20,15))
plt.imshow(w)
plt.axis('off')
plt.savefig('recipes_wordcloud.png')
txtFile = open("output.txt","w")
sortedWeights = sorted(tfidf_weights.items(), key=lambda x:x[1],reverse=True)
txtFile.write("Ingredient      :        Weight \n")
for key,values in sortedWeights:
    txtFile.write(str(key)+" : " +str(values) + "\n")
txtFile.close()
```

```
# task 6 -  get word frequencies and create wordcloud
tfidf_weights=[(word,tdm.getcol(idx).sum()) for word, idx in
vectorizer.vocabulary_.items()]
tfidf_weights=dict(tfidf_weights)
w=WordCloud(width=1500,height=1200, mode='RGBA', background_color='white',
max_words=2000).fit_words(dict(tfidf_weights))
plt.figure(figsize=(20,15))
plt.imshow(w)
plt.axis('off')
plt.savefig('recipes_wordcloud.png')
txtFile = open("output.txt","w")
sortedWeights = sorted(tfidf_weights.items(), key=lambda x:x[1],reverse=True)
txtFile.write("Ingredient      :        Weight \n")
for key,values in sortedWeights:
    txtFile.write(str(key)+" : " +str(values) + "\n")
txtFile.close()
```

Word Cloud and Text File Output



Ingredient :	Weight
caster sugar :	33.92679359870039
plain flour :	26.164699816027802
double cream :	24.47528017761138
golden caster :	23.665605324844833
icing sugar :	23.56736189792369
garlic clove :	20.5616648091688
muscovado sugar :	15.157442191497505
butter softened :	14.33284930512973
thinly sliced :	13.434276096688432
unsalted butter :	13.02225291911623
dark chocolate :	12.759546623175648
wine vinegar :	12.328195823929985
ground cinnamon :	11.836742570209484
large egg :	11.655002686716756
zest juice :	11.596074513899923
white wine :	11.50957441262325
plus dusting :	11.200198008258184
vanilla extract :	11.146409447587125
brown sugar :	11.135413088674557
light muscovado :	10.955134306343428
self raising :	10.824187358979268
raising flour :	10.782791060584968
plus greasing :	10.619582545840558
chicken stock :	10.54995369400423
clove crushed :	10.449428841281208
grated zest :	10.238750697266983
mixed spice :	9.882048828461663
cinnamon stick :	9.826696050610387
flour plus :	9.761752979175906
baking powder :	9.575576364877861
juice lemon :	9.218973352201182
brussels sprout :	8.760995074060489

```
egg beaten : 8.733133620281524
streaky bacon : 8.561729322633997
ground almond : 8.527966748756377
clear honey : 8.485596404076897
good quality : 8.45323965947161
sugar plus : 8.389880240431221
butter plus : 8.36841543181027
onion sliced : 8.300531309453822
thyme leaf : 8.263535749988357
zest orange : 8.260474585826856
zest lemon : 8.076860974104306
ground ginger : 7.923132778011525
dried cranberry : 7.616847676526653
maple syrup : 7.6098421312007485
lemon juice : 7.543426313041776
large onion : 7.530992783461033
golden syrup : 7.528255046680915
puff pastry : 7.3514946689895995
wholegrain mustard : 7.346843934074862
small bunch : 7.334888420314011
star anise : 7.186835264657585
fresh frozen : 7.16753473144454
vegetable stock : 7.055420180052844
plus serve : 6.950395768563531
butter melted : 6.935057863546623
sugar dusting : 6.877052007208772
freshly grated : 6.803062087205156
sugar large : 6.6841164198277
flaked almond : 6.615035320411994
soft brown : 6.589275376256462
apple peeled : 6.546330918694762
white chocolate : 6.537824204026434
knob butter : 6.475600035271045
thyme sprig : 6.473124535072162
small handful : 6.448344956228847
orange juice : 6.370504009714113
frozen cranberry : 6.356445501846716
peeled cored : 6.231054584465458
small pack : 6.134358043381125
grated nutmeg : 6.125389597761781
light brown : 6.070506759090989
cocoa powder : 6.067324945597459
bicarbonate soda : 6.054598831265656
mari piper : 5.970117673900321
dijon mustard : 5.943162595544985
juice orange : 5.929455258515979
flour dusting : 5.909574844108866
dried apricot : 5.908431948446099
dark muscovado : 5.889335563862861
plus little : 5.853880818141382
flour baking : 5.8113100082383
serve optional : 5.773989544631382
leaf picked : 5.653649277862263
small onion : 5.599796689010847
white bread : 5.4801556178746225
cider vinegar : 5.450944692684173
spring onion : 5.395562647257052
juice lime : 5.316331217423093
black peppercorn : 5.28922224312433
flour ground : 5.266958383465993
glac cherry : 5.264095966063564
sage leaf : 5.212046569688317
ground clove : 5.2104242762605635
pinch ground : 5.183622411466276
go well : 5.163376835303158
bramley apple : 5.134819390644449
potato peeled : 5.085394888209205
virgin olive : 5.08369970102348
smoked salmon : 5.03575046896877
whole milk : 4.993033020220411
small piece : 4.9839941319157495
salted butter : 4.943177210530218
ready roll : 4.934405070468728
sugar vanilla : 4.907390618514263
handful parsley : 4.9070226440403495
onion halved : 4.902345455249703
stem ginger : 4.893432850989686
white breadcrumb : 4.856425990554216
milk chocolate : 4.855886269249453
dried fruit : 4.820237069127879
chocolate broken : 4.776806296688027
food colouring : 4.760692389939962
ready rolled : 4.667419817096415
granulated sugar : 4.639177024147694
room temperature : 4.6368899141339295
pack ready : 4.543476184561189
softened butter : 4.539718513473184
sprout trimmed : 4.533009874572549
balsamic vinegar : 4.510122051580977
mixed peel : 4.497783151079771
smoked streaky : 4.490831354827359
Check the output.txt for more…
```

BENEFITS

- **Data Visualization:** Word clouds provide an intuitive and visually appealing representation of textual data. They help in quickly identifying the most frequent and important terms in a document or corpus.
- **Text Analysis:** By utilizing NLP techniques, such as tokenization, TF-IDF computation, and word cloud generation, the project allows you to gain insights into the underlying text data. It helps in identifying key themes, significant terms, or prominent features within the text.
- **Communication and Presentation:** Word clouds are often used for effective communication and presentation purposes. They condense large amounts of text into a visually engaging format, making it easier to convey information to an audience.
- **Feature Extraction:** The TF-IDF values computed during the project can serve as feature vectors for further analysis or machine learning tasks. These vectors can be used to represent documents or text snippets, enabling various text classification or clustering applications.

CONCLUSION

The project of creating a word cloud using NLP and TF-IDF in Python allows you to analyze and visualize text data in a meaningful way. By leveraging NLP techniques, such as tokenization and TF-IDF computation, you can identify the most important terms within the text and generate a visually appealing word cloud. The project offers benefits in terms of data visualization, text analysis, communication, and feature extraction. It can be a valuable tool for gaining insights from textual data and presenting information in a concise and visually appealing manner.