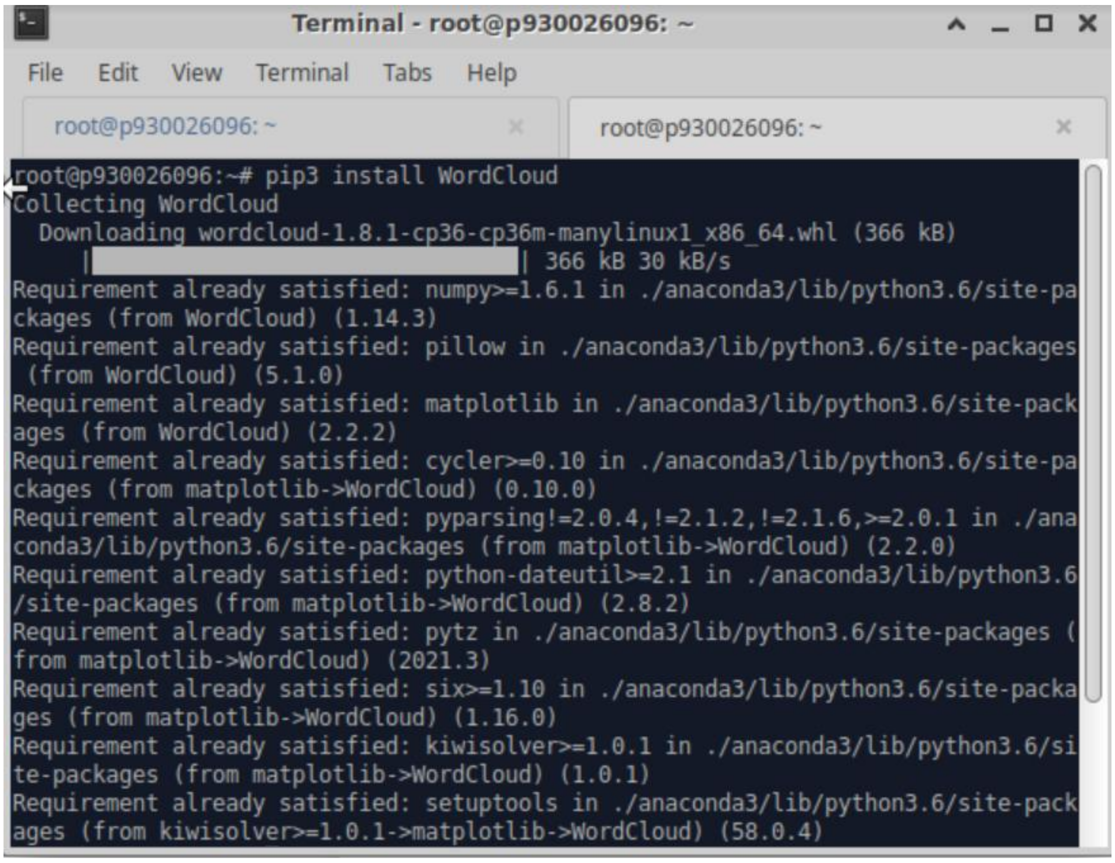
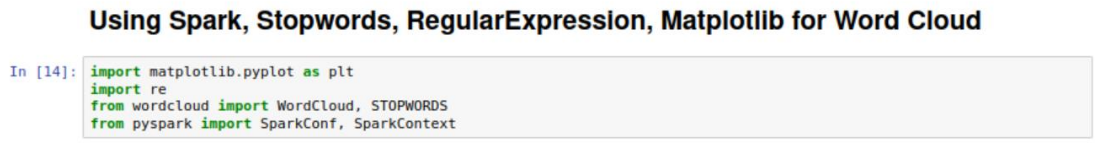
Basic



```
root@p930026096:~# pip3 install WordCloud
Collecting WordCloud
  Downloading wordcloud-1.8.1-cp36-cp36m-manylinux1_x86_64.whl (366 kB)
     |████████████████████████████████| 366 kB 30 kB/s
Requirement already satisfied: numpy>=1.6.1 in ./anaconda3/lib/python3.6/site-pa
ckages (from WordCloud) (1.14.3)
Requirement already satisfied: pillow in ./anaconda3/lib/python3.6/site-packages
 (from WordCloud) (5.1.0)
Requirement already satisfied: matplotlib in ./anaconda3/lib/python3.6/site-pack
ages (from WordCloud) (2.2.2)
Requirement already satisfied: cycler>=0.10 in ./anaconda3/lib/python3.6/site-pa
ckages (from matplotlib->WordCloud) (0.10.0)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in ./ana
conda3/lib/python3.6/site-packages (from matplotlib->WordCloud) (2.2.0)
Requirement already satisfied: python-dateutil>=2.1 in ./anaconda3/lib/python3.6
/site-packages (from matplotlib->WordCloud) (2.8.2)
Requirement already satisfied: pytz in ./anaconda3/lib/python3.6/site-packages (
from matplotlib->WordCloud) (2021.3)
Requirement already satisfied: six>=1.10 in ./anaconda3/lib/python3.6/site-packa
ges (from matplotlib->WordCloud) (1.16.0)
Requirement already satisfied: kiwisolver>=1.0.1 in ./anaconda3/lib/python3.6/si
te-packages (from matplotlib->WordCloud) (1.0.1)
Requirement already satisfied: setuptools in ./anaconda3/lib/python3.6/site-pack
ages (from kiwisolver>=1.0.1->matplotlib->WordCloud) (58.0.4)
```

Install the WordCloud library in python.

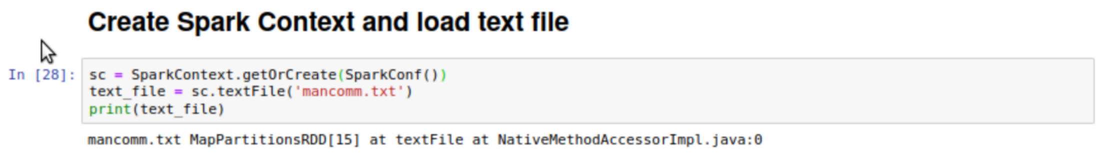
Run the first lab WordCloud.



Using Spark, Stopwords, RegularExpression, Matplotlib for Word Cloud

```
In [14]: import matplotlib.pyplot as plt
         import re
         from wordcloud import WordCloud, STOPWORDS
         from pyspark import SparkConf, SparkContext
```

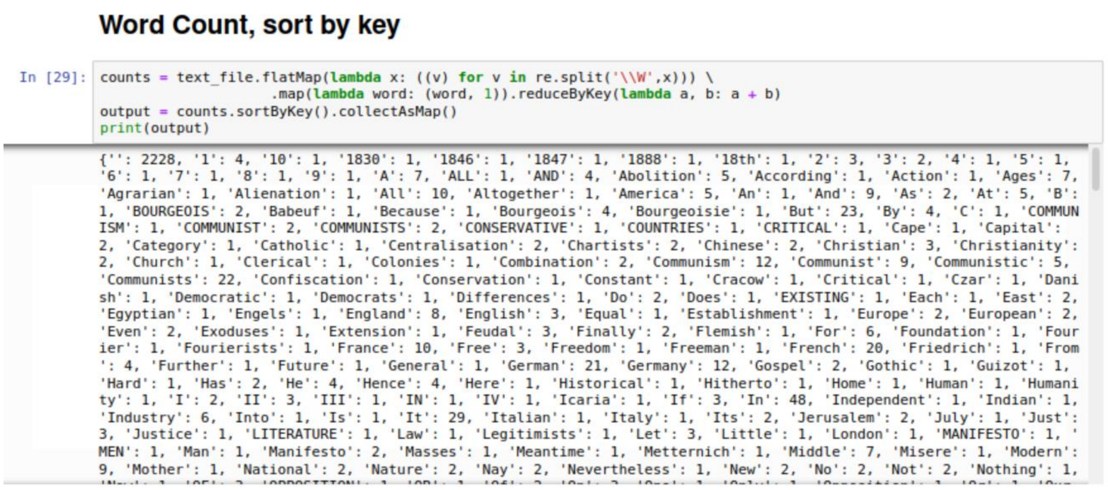
Import libraries.



Create Spark Context and load text file

```
In [28]: sc = SparkContext.getOrCreate(SparkConf())
         text_file = sc.textFile('mancomm.txt')
         print(text_file)

         mancomm.txt MapPartitionsRDD[15] at textFile at NativeMethodAccessorImpl.java:0
```

Create a spark context and load text file.

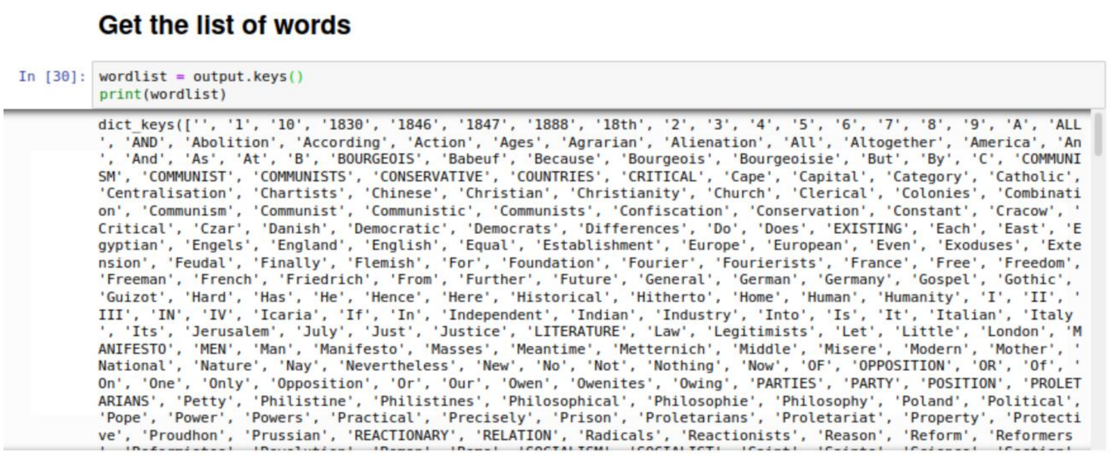


Word Count, sort by key

```
In [29]: counts = text_file.flatMap(lambda x: ((v) for v in re.split('\\W',x))) \
                        .map(lambda word: (word, 1)).reduceByKey(lambda a, b: a + b)
         output = counts.sortByKey().collectAsMap()
         print(output)
```

```
{'': 2228, '1': 4, '10': 1, '1830': 1, '1846': 1, '1847': 1, '1888': 1, '18th': 1, '2': 3, '3': 2, '4': 1, '5': 1,
'6': 1, '7': 1, '8': 1, '9': 1, 'A': 7, 'ALL': 1, 'AND': 4, 'Abolition': 5, 'According': 1, 'Action': 1, 'Ages': 7,
'Agrarian': 1, 'Alienation': 1, 'All': 10, 'Altogether': 1, 'America': 5, 'An': 1, 'And': 9, 'As': 2, 'At': 5, 'B':
1, 'BOURGEOIS': 2, 'Babeuf': 1, 'Because': 1, 'Bourgeois': 4, 'Bourgeoisie': 1, 'But': 23, 'By': 4, 'C': 1, 'COMMUN
ISM': 1, 'COMMUNIST': 2, 'COMMUNISTS': 2, 'CONSERVATIVE': 1, 'COUNTRIES': 1, 'CRITICAL': 1, 'Cape': 1, 'Capital':
2, 'Category': 1, 'Catholic': 1, 'Centralisation': 2, 'Chartists': 2, 'Chinese': 2, 'Christian': 3, 'Christianity':
2, 'Church': 1, 'Clerical': 1, 'Colonies': 1, 'Combination': 2, 'Communism': 12, 'Communist': 9, 'Communistic': 5,
'Communists': 22, 'Confiscation': 1, 'Conservation': 1, 'Constant': 1, 'Cracow': 1, 'Critical': 1, 'Czar': 1, 'Dani
sh': 1, 'Democratic': 1, 'Democrats': 1, 'Differences': 1, 'Do': 2, 'Does': 1, 'EXISTING': 1, 'Each': 1, 'East': 2,
'Egyptian': 1, 'Engels': 1, 'England': 8, 'English': 3, 'Equal': 1, 'Establishment': 1, 'Europe': 2, 'European': 2,
'Even': 2, 'Exoduses': 1, 'Extension': 1, 'Feudal': 3, 'Finally': 2, 'Flemish': 1, 'For': 6, 'Foundation': 1, 'Four
ier': 1, 'Fourierists': 1, 'France': 10, 'Free': 3, 'Freedom': 1, 'Freeman': 1, 'French': 20, 'Friedrich': 1, 'From
': 4, 'Further': 1, 'Future': 1, 'General': 1, 'German': 21, 'Germany': 12, 'Gospel': 2, 'Gothic': 1, 'Guizot': 1,
'Hard': 1, 'Has': 2, 'He': 4, 'Hence': 4, 'Here': 1, 'Historical': 1, 'Hitherto': 1, 'Home': 1, 'Human': 1, 'Humani
ty': 1, 'I': 2, 'II': 3, 'III': 1, 'IN': 1, 'IV': 1, 'Icaria': 1, 'If': 3, 'In': 48, 'Independent': 1, 'Indian': 1,
'Industry': 6, 'Into': 1, 'Is': 1, 'It': 29, 'Italian': 1, 'Italy': 1, 'Its': 2, 'Jerusalem': 2, 'July': 1, 'Just':
3, 'Justice': 1, 'LITERATURE': 1, 'Law': 1, 'Legitimists': 1, 'Let': 3, 'Little': 1, 'London': 1, 'MANIFESTO': 1, '
MEN': 1, 'Man': 1, 'Manifesto': 2, 'Masses': 1, 'Meantime': 1, 'Metternich': 1, 'Middle': 7, 'Misere': 1, 'Modern':
9, 'Mother': 1, 'National': 2, 'Nature': 2, 'Nay': 2, 'Nevertheless': 1, 'New': 2, 'No': 2, 'Not': 2, 'Nothing': 1,
```

Count the word and sorted by key.

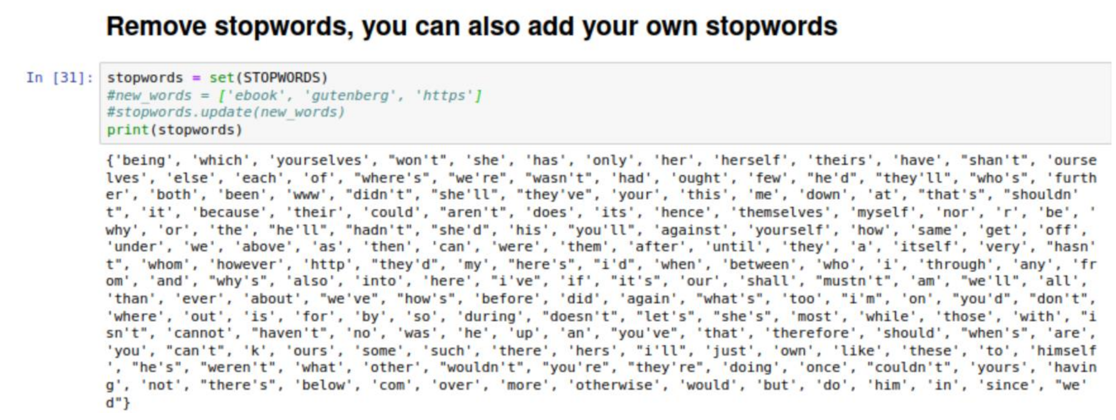


Get the list of words

```
In [30]: wordlist = output.keys()
         print(wordlist)
```

```
dict_keys(['', '1', '10', '1830', '1846', '1847', '1888', '18th', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'ALL
', 'AND', 'Abolition', 'According', 'Action', 'Ages', 'Agrarian', 'Alienation', 'All', 'Altogether', 'America', 'An
', 'And', 'As', 'At', 'B', 'BOURGEOIS', 'Babeuf', 'Because', 'Bourgeois', 'Bourgeoisie', 'But', 'By', 'C', 'COMMUNI
SM', 'COMMUNIST', 'COMMUNISTS', 'CONSERVATIVE', 'COUNTRIES', 'CRITICAL', 'Cape', 'Capital', 'Category', 'Catholic',
'Centralisation', 'Chartists', 'Chinese', 'Christian', 'Christianity', 'Church', 'Clerical', 'Colonies', 'Combinati
on', 'Communism', 'Communist', 'Communistic', 'Communists', 'Confiscation', 'Conservation', 'Constant', 'Cracow', '
Critical', 'Czar', 'Danish', 'Democratic', 'Democrats', 'Differences', 'Do', 'Does', 'EXISTING', 'Each', 'East', 'E
gyptian', 'Engels', 'England', 'English', 'Equal', 'Establishment', 'Europe', 'European', 'Even', 'Exoduses', 'Exte
nsion', 'Feudal', 'Finally', 'Flemish', 'For', 'Foundation', 'Fourier', 'Fourierists', 'France', 'Free', 'Freedom',
'Freeman', 'French', 'Friedrich', 'From', 'Further', 'Future', 'General', 'German', 'Germany', 'Gospel', 'Gothic',
'Guizot', 'Hard', 'Has', 'He', 'Hence', 'Here', 'Historical', 'Hitherto', 'Home', 'Human', 'Humanity', 'I', 'II', '
III', 'IN', 'IV', 'Icaria', 'If', 'In', 'Independent', 'Indian', 'Industry', 'Into', 'Is', 'It', 'Italian', 'Italy
', 'Its', 'Jerusalem', 'July', 'Just', 'Justice', 'LITERATURE', 'Law', 'Legitimists', 'Let', 'Little', 'London', 'M
ANIFESTO', 'MEN', 'Man', 'Manifesto', 'Masses', 'Meantime', 'Metternich', 'Middle', 'Misere', 'Modern', 'Mother', '
National', 'Nature', 'Nay', 'Nevertheless', 'New', 'No', 'Not', 'Nothing', 'Now', 'OF', 'OPPOSITION', 'OR', 'Of', '
On', 'One', 'Only', 'Opposition', 'Or', 'Our', 'Owen', 'Owenites', 'Owing', 'PARTIES', 'PARTY', 'POSITION', 'PROLET
ARIANS', 'Petty', 'Philistine', 'Philistines', 'Philosophical', 'Philosophie', 'Philosophy', 'Poland', 'Political',
'Pope', 'Power', 'Powers', 'Practical', 'Precisely', 'Prison', 'Proletarians', 'Proletariat', 'Property', 'Protecti
ve', 'Proudhon', 'Prussian', 'REACTIONARY', 'RELATION', 'Radicals', 'Reactionists', 'Reason', 'Reform', 'Reformers
```

Get the list of words



Remove stopwords, you can also add your own stopwords

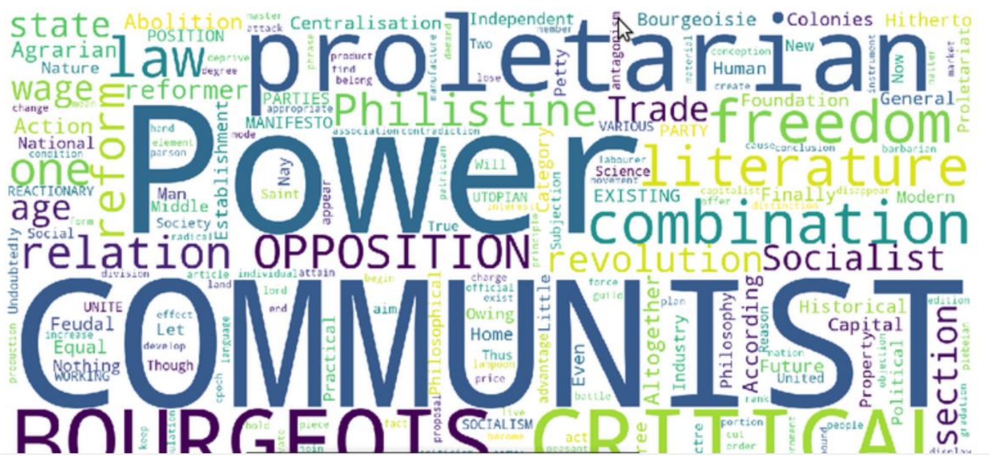
```
In [31]: stopwords = set(STOPWORDS)
         #new_words = ['ebook', 'gutenberg', 'https']
         #stopwords.update(new_words)
         print(stopwords)
```

```
{'being', 'which', 'yourselves', "won't", 'she', 'has', 'only', 'her', 'herself', 'theirs', 'have', "shan't", 'ourse
lves', 'else', 'each', 'of', "where's", "we're", "wasn't", 'had', 'ought', 'few', "he'd", "they'll", "who's", 'furth
er', 'both', 'been', 'www', "didn't", "she'll", "they've", 'your', 'this', 'me', 'down', 'at', "that's", "shouldn'
t", 'it', 'because', 'their', 'could', "aren't", 'does', 'its', 'hence', 'themselves', 'myself', 'nor', 'r', 'be', '
why', 'or', 'the', "he'll", "hadn't", "she'd", 'his', "you'll", 'against', 'yourself', 'how', 'same', 'get', 'off',
'under', 'we', 'above', 'as', 'then', 'can', 'were', 'them', 'after', 'until', 'they', 'a', 'itself', 'very', "hasn'
t", 'whom', 'however', 'http', "they'd", 'my', "here's", "i'd", 'when', 'between', 'who', 'i', 'through', 'any', 'fr
om', 'and', "why's", 'also', 'into', 'here', "i've", 'if', "it's", 'our', 'shall', "mustn't", 'am', "we'll", 'all',
'than', 'ever', 'about', "we've", "how's", 'before', 'did', 'again', "what's", 'too', "i'm", 'on', "you'd", "don't",
'where', 'out', 'is', 'for', 'by', 'so', 'during', "doesn't", "let's", "she's", 'most', 'while', 'those', 'with', "i
sn't", 'cannot', "haven't", 'no', 'was', 'he', 'up', 'an', "you've", 'that', 'therefore', 'should', "when's", 'are',
'you', "can't", 'k', 'ours', 'some', 'such', 'there', 'hers', "i'll", 'just', 'own', 'like', 'these', 'to', 'himself
', "he's", "weren't", 'what', 'other', "wouldn't", "you're", "they're", 'doing', 'once', "couldn't", 'yours', 'havin
g', 'not', "there's", 'below', 'com', 'over', 'more', 'otherwise', 'would', 'but', 'do', 'him', 'in', 'since', "we'
d"}
```

Remove stopwords and you can update the stopwords by update function.

Generate and show word cloud

```
In [32]: wordcloud = WordCloud(stopwords=stopwords, width=1600, height=800, background_color='white')\
                    .generate(" ".join(wordlist))

         plt.figure(figsize=(20,10))
         plt.imshow(wordcloud)
         plt.axis('off')
         plt.show()
```



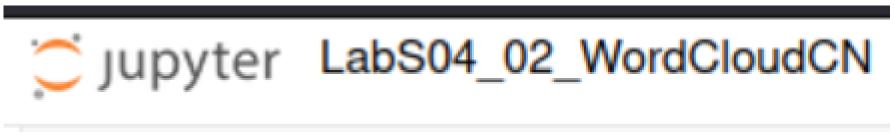
Generate and show the word cloud.

You can also include a picture mask for the result

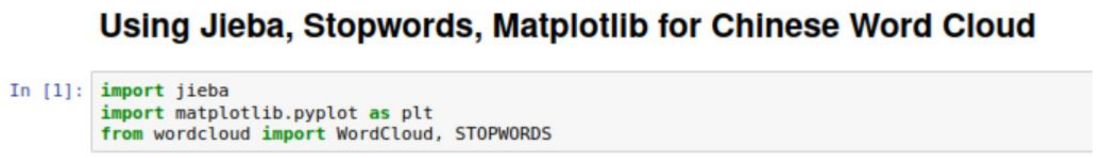
```
In [33]: import numpy as np
         from PIL import Image
         from wordcloud import ImageColorGenerator

         background_image = np.array(Image.open('heart.jpg'))
         #background_image = np.array(Image.open('balloon.jpg'))

         wordcloud = WordCloud(stopwords=stopwords, mask=background_image, width=1600, height=800, background_color='white')\
                    .generate(" ".join(wordlist))

         plt.figure(figsize=(20,10))

         image_colors = ImageColorGenerator(background_image)
         plt.imshow(wordcloud.recolor(color_func=image_colors))

         plt.axis('off')
         plt.show()
```

You can include a picture for the word cloud (change the background picture size from default rectangle to picture shape).

The word cloud is transformed into heart shape.

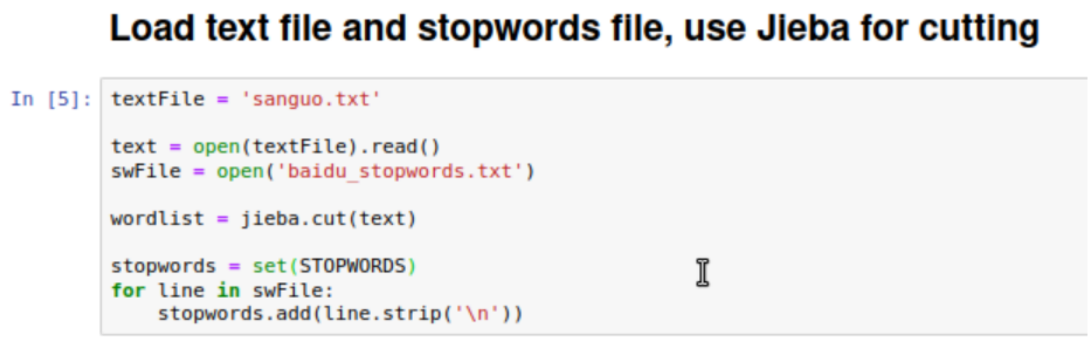

jupyter LabS04_02_WordCloudCN

Run the second lab WordCloudCN.


Using Jieba, Stopwords, Matplotlib for Chinese Word Cloud

```
In [1]: import jieba
        import matplotlib.pyplot as plt
        from wordcloud import WordCloud, STOPWORDS
```

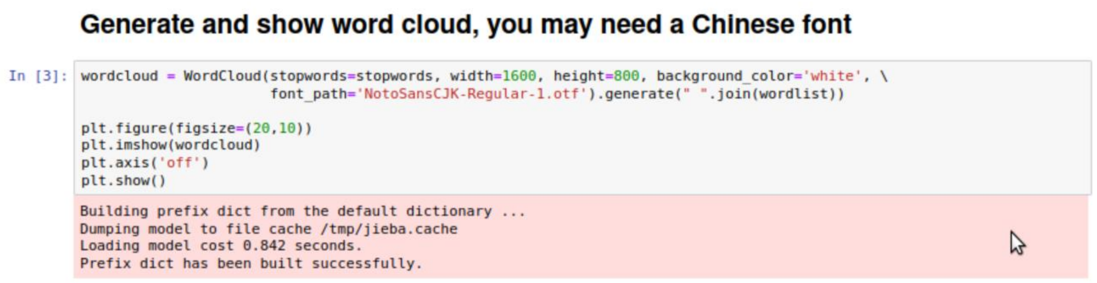
Import libraries.



Load text file and stopwords file, use Jieba for cutting

```
In [5]: textFile = 'sanguo.txt'

        text = open(textFile).read()
        swFile = open('baidu_stopwords.txt')

        wordlist = jieba.cut(text)

        stopwords = set(STOPWORDS)
        for line in swFile:
            stopwords.add(line.strip('\n'))
```

Load text file and stop words file, using Jieba for cutting.

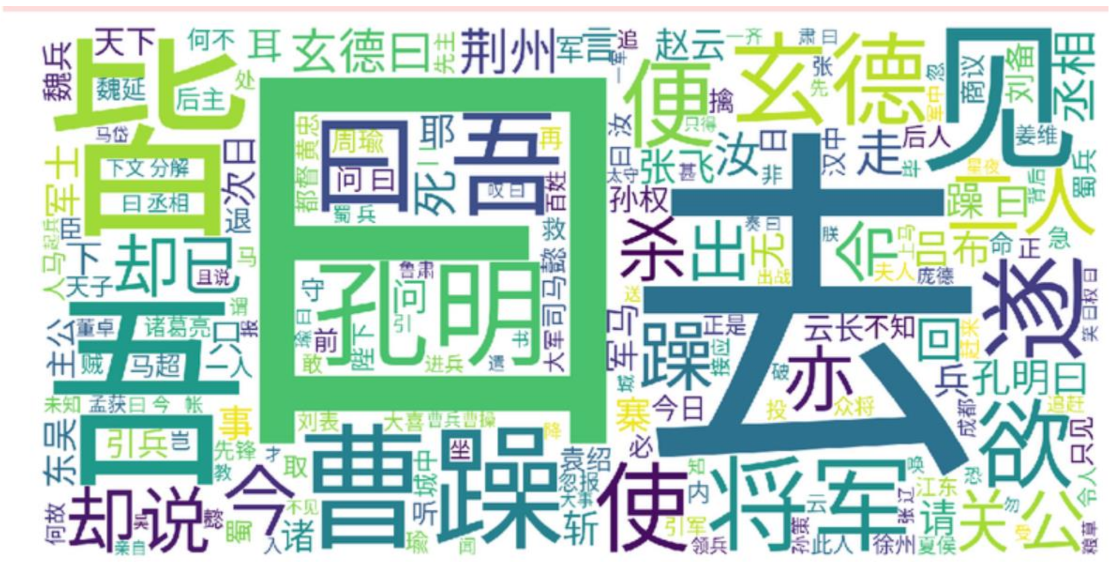


Generate and show word cloud, you may need a Chinese font

```
In [3]: wordcloud = WordCloud(stopwords=stopwords, width=1600, height=800, background_color='white', \
                              font_path='NotoSansCJK-Regular-1.otf').generate(" ".join(wordlist))

        plt.figure(figsize=(20,10))
        plt.imshow(wordcloud)
        plt.axis('off')
        plt.show()
```

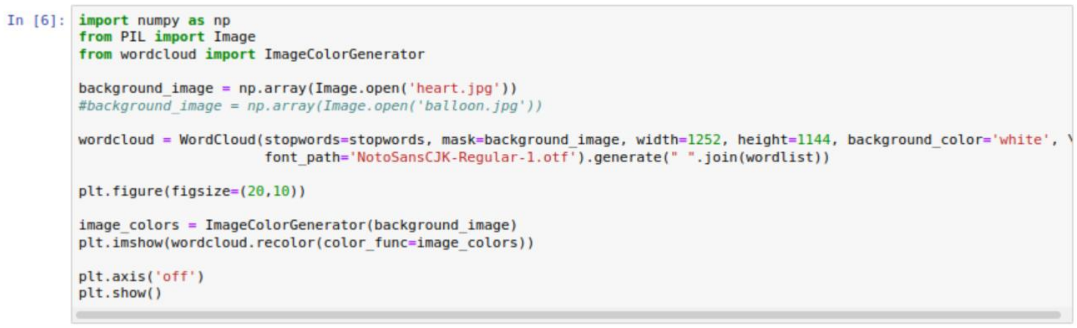
```
Building prefix dict from the default dictionary ...
Dumping model to file cache /tmp/jieba.cache
Loading model cost 0.842 seconds.
Prefix dict has been built successfully.
```

Generate and show word cloud, and you can use a Chinese font.



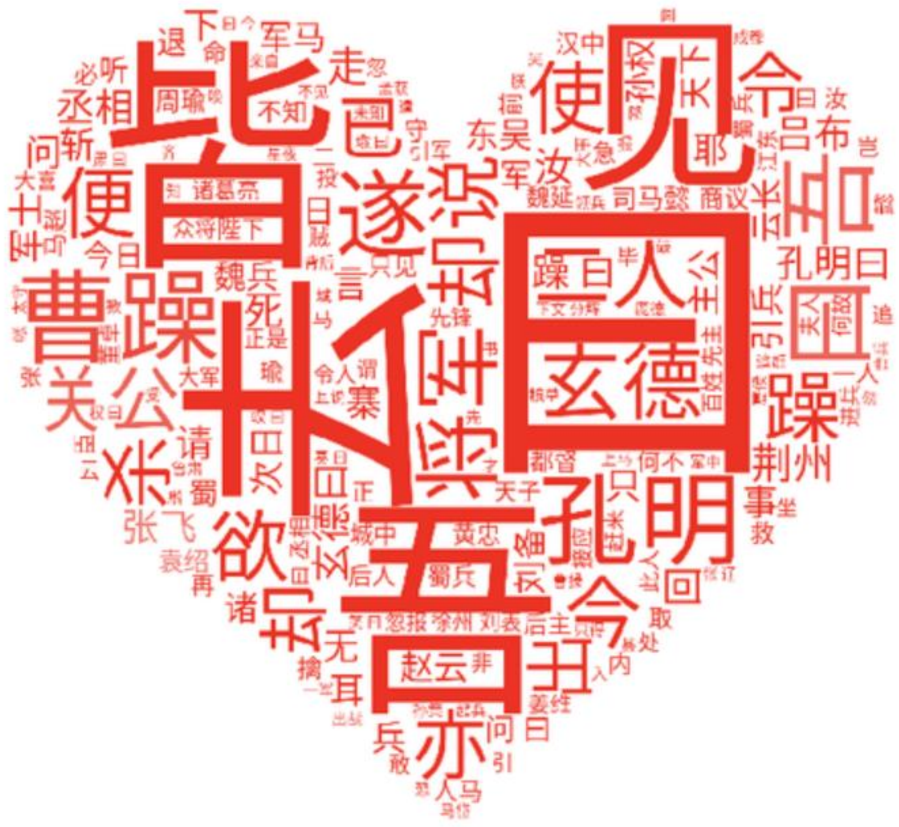
The default shape word cloud for Chinese text.

You can also include a picture mask for the result

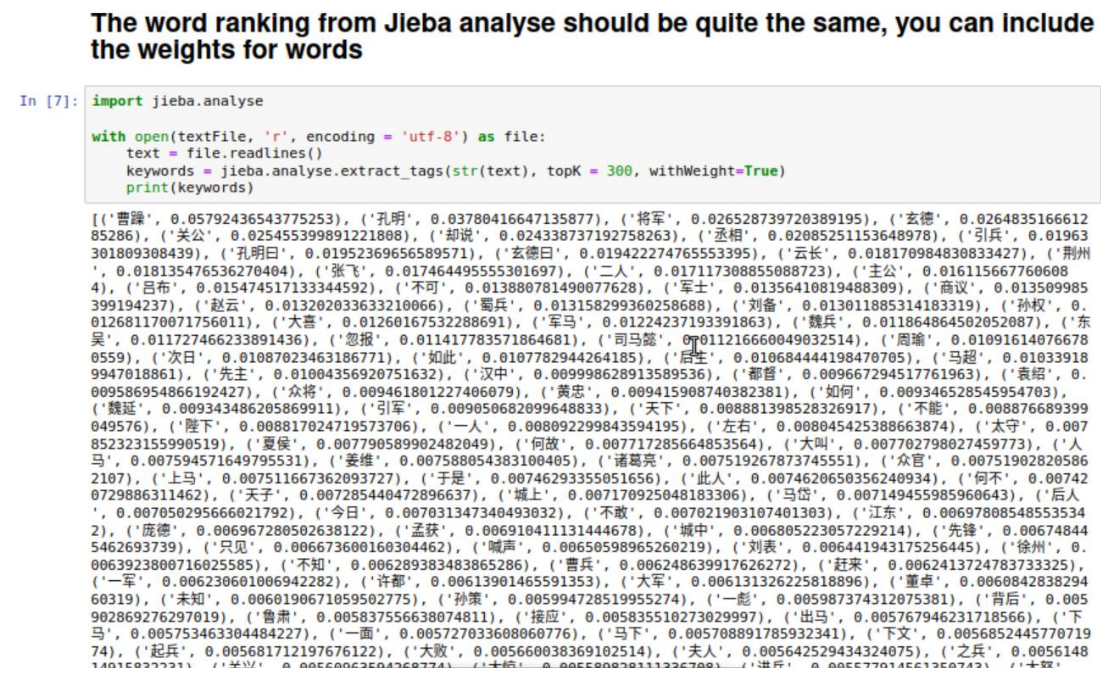


```
In [6]: import numpy as np
        from PIL import Image
        from wordcloud import ImageColorGenerator

        background_image = np.array(Image.open('heart.jpg'))
        #background_image = np.array(Image.open('balloon.jpg'))

        wordcloud = WordCloud(stopwords=stopwords, mask=background_image, width=1252, height=1144, background_color='white', \
                              font_path='NotoSansCJK-Regular-1.otf').generate(" ".join(wordlist))

        plt.figure(figsize=(20,10))

        image_colors = ImageColorGenerator(background_image)
        plt.imshow(wordcloud.recolor(color_func=image_colors))

        plt.axis('off')
        plt.show()
```

Include a picture mask for the result.



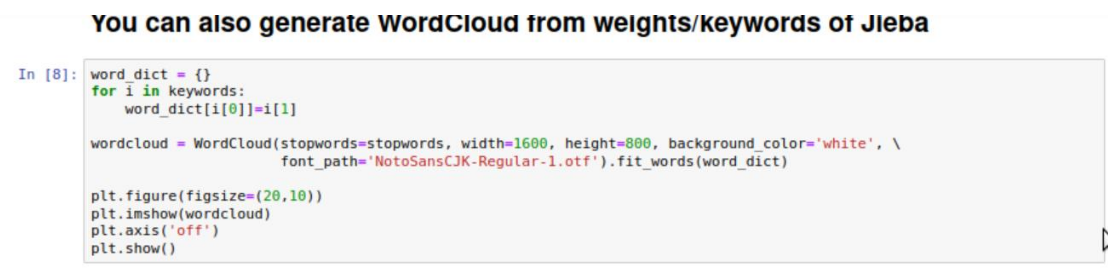
The word cloud with a heart shape as the background.


The word ranking from Jieba analyse should be quite the same, you can include the weights for words

```
In [7]: import jieba.analyse

        with open(textFile, 'r', encoding = 'utf-8') as file:
            text = file.readlines()
            keywords = jieba.analyse.extract_tags(str(text), topK = 300, withWeight=True)
            print(keywords)
```

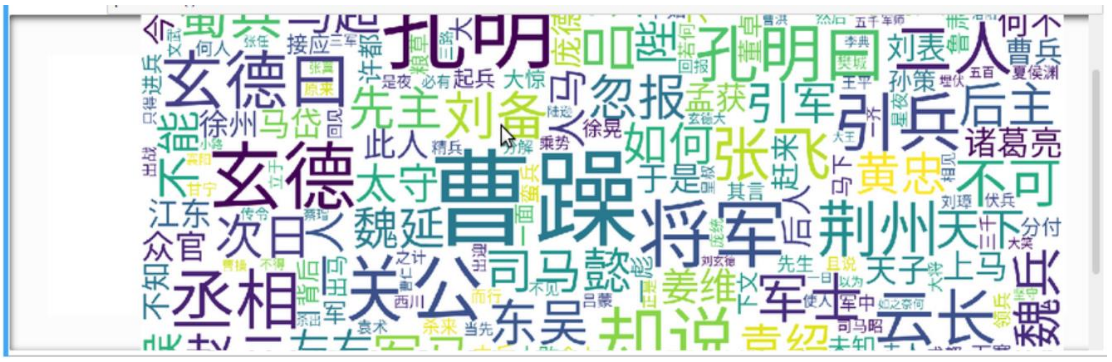
```
[('曹躁', 0.05792436543775253), ('孔明', 0.03780416647135877), ('将军', 0.026528739720389195), ('玄德', 0.0264835166612
85286), ('关公', 0.025455399891221808), ('却说', 0.024338737192758263), ('丞相', 0.02085251153648978), ('引兵', 0.01963
301809308439), ('孔明曰', 0.01952369656589571), ('玄德曰', 0.019422274765553395), ('云长', 0.018170984830833427), ('荆州
', 0.018135476536270404), ('张飞', 0.017464495555301697), ('二人', 0.017117308855088723), ('主公', 0.016115667760608
4), ('吕布', 0.015474517133344592), ('不可', 0.013880781490077628), ('军士', 0.013564108194888309), ('商议', 0.013509985
399194237), ('赵云', 0.013202033633210066), ('蜀兵', 0.013158299360258688), ('刘备', 0.013011885314183319), ('孙权', 0.
012681170071756011), ('大喜', 0.012601675322886691), ('军马', 0.012242371933918663), ('魏兵', 0.011864864502052087), ('东
吴', 0.011727466233891436), ('忽报', 0.011417783571864681), ('司马懿', 0.011216660049032514), ('周瑜', 0.01091614076678
0559), ('次日', 0.010870234631867771), ('如此', 0.0107782944264185), ('后主', 0.010684444198470705), ('马超', 0.01033918
9947018861), ('先主', 0.010043569207516332), ('汉中', 0.009998628913589536), ('都督', 0.009667294517761963), ('袁绍', 0.
009586954866192427), ('众将', 0.009461801227406079), ('黄忠', 0.009415908740382381), ('如何', 0.009346528545954703),
('魏延', 0.009343486205869911), ('引军', 0.009050682099648833), ('天下', 0.008881398528326917), ('不能', 0.008876689399
049576), ('陛下', 0.008817024719573706), ('一人', 0.008092299843594195), ('左右', 0.008045425388663874), ('太守', 0.007
852323155990519), ('夏侯', 0.007790589902482049), ('何故', 0.007717285664853564), ('大叫', 0.007702798027459773), ('人
马', 0.007594571649795531), ('姜维', 0.007588054383100405), ('诸葛亮', 0.007519267873745551), ('众官', 0.00751902820586
2107), ('上马', 0.007511667362093727), ('于是', 0.007462933550516560), ('此人', 0.007462065035624934), ('何不', 0.00742
0729886311462), ('天子', 0.007285440472896637), ('城上', 0.007170925048183306), ('马岱', 0.007149455985960643), ('后人
', 0.007050295666021792), ('今日', 0.007031347340493032), ('不敢', 0.007021903107401303), ('江东', 0.00697808548553534
2), ('庞德', 0.006967280502638122), ('孟获', 0.006910411131444678), ('城中', 0.006805223057229214), ('先锋', 0.00674844
5462693739), ('只见', 0.006673600160304462), ('喊声', 0.006505989652602219), ('刘表', 0.006441943175256445), ('徐州', 0.
006392380071602585), ('不知', 0.006289383483865286), ('曹兵', 0.006248639917626272), ('赶来', 0.006241372478373325),
('一军', 0.006230601006942282), ('许都', 0.006139014655913353), ('大军', 0.006131326225818896), ('董卓', 0.0060842838294
60319), ('未知', 0.006019067105950275), ('孙策', 0.005994728519955274), ('一彪', 0.005987374312075381), ('背后', 0.005
902869276297019), ('鲁肃', 0.005837556638074811), ('接应', 0.005835510273029997), ('出马', 0.005767946231718566), ('下
马', 0.005753463304484227), ('一面', 0.005727033608060776), ('马下', 0.005708891785932341), ('下文', 0.0056852445770719
74), ('起兵', 0.005681712197676122), ('大败', 0.005660038369102514), ('夫人', 0.005642529434324075), ('之兵', 0.0056148
14915832231), ('关兴', 0.005609635942687 74), ('大惊', 0.005589828111336708), ('进兵', 0.005577914561350743), ('大怒'
```

Word Ranking from Jieba.


You can also generate WordCloud from weights/keywords of Jieba

```
In [8]: word_dict = {}
        for i in keywords:
            word_dict[i[0]]=i[1]

        wordcloud = WordCloud(stopwords=stopwords, width=1600, height=800, background_color='white', \
                             font_path='NotoSansCJK-Regular-1.otf').fit_words(word_dict)

        plt.figure(figsize=(20,10))
        plt.imshow(wordcloud)
        plt.axis('off')
        plt.show()
```

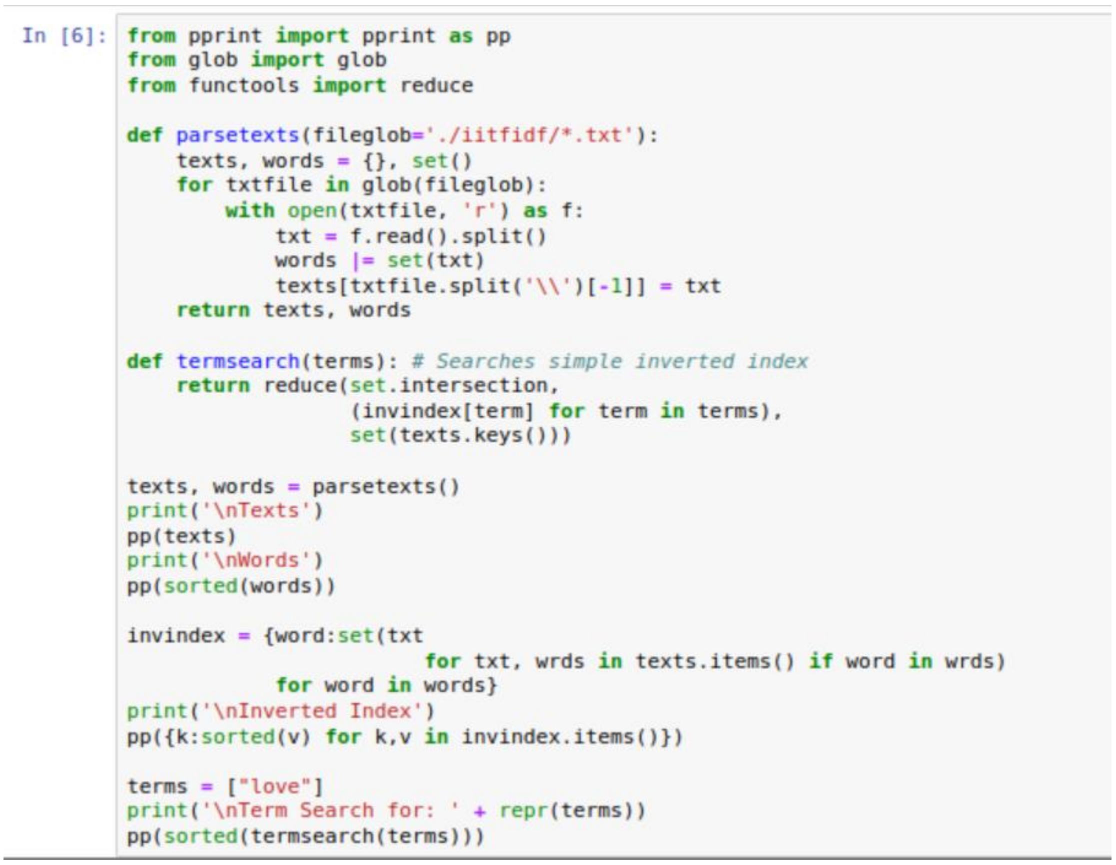
Generate word cloud from weights/keywords of Jieba.



Show the word cloud with weights.

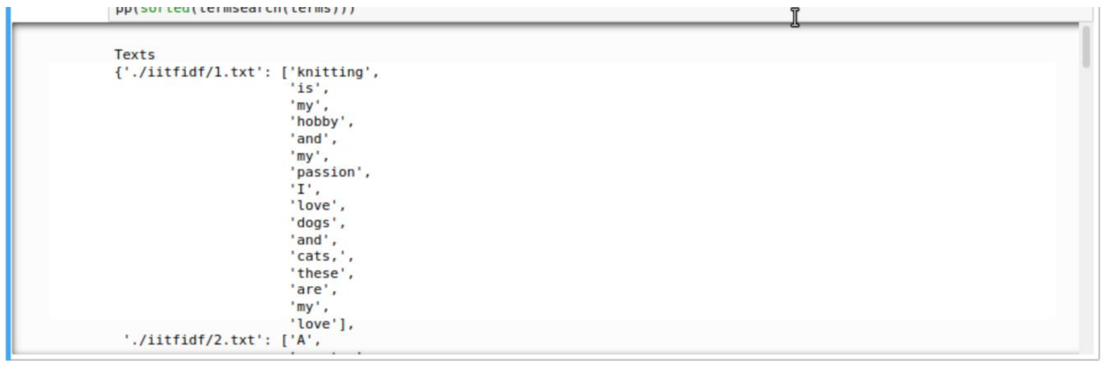
jupyter LabS04_03_SimpleInvIndex

Run the third Lab SimpleInvIndex.



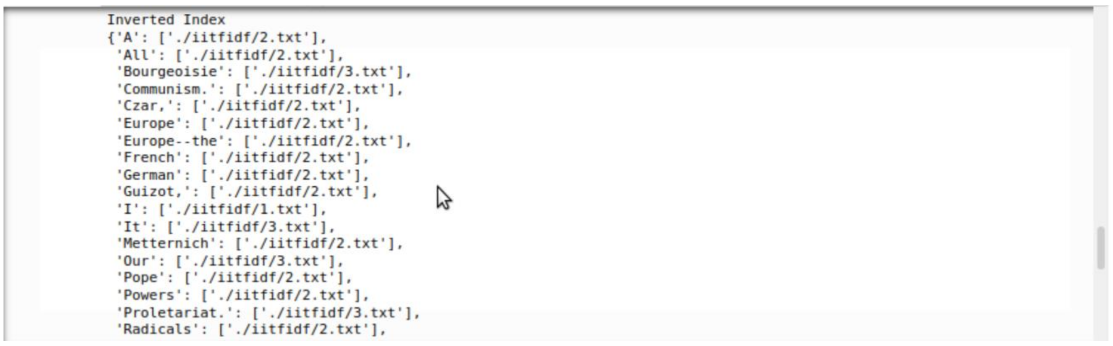
```
In [6]: from pprint import pprint as pp
        from glob import glob
        from functools import reduce

        def parsetexts(fileglob='./iitfidf/*.txt'):
            texts, words = {}, set()
            for txtfile in glob(fileglob):
                with open(txtfile, 'r') as f:
                    txt = f.read().split()
                    words |= set(txt)
                    texts[txtfile.split('\\')[-1]] = txt
            return texts, words

        def termsearch(terms): # Searches simple inverted index
            return reduce(set.intersection,
                          (invindex[term] for term in terms),
                          set(texts.keys()))

        texts, words = parsetexts()
        print('\nTexts')
        pp(texts)
        print('\nWords')
        pp(sorted(words))

        invindex = {word:set(txt
                                for txt, wrds in texts.items() if word in wrds)
                    for word in words}
        print('\nInverted Index')
        pp({k:sorted(v) for k,v in invindex.items()})

        terms = ["love"]
        print('\nTerm Search for: ' + repr(terms))
        pp(sorted(termsearch(terms)))
```

Read all the from a local directory and split each file. Get the filename and split words And define a function that can perform word searching.



```
pp(sorted(termsearch(terms)))

Texts
{'./iitfidf/1.txt': ['knitting',
                     'is',
                     'my',
                     'hobby',
                     'and',
                     'my',
                     'passion',
                     'I',
                     'love',
                     'dogs',
                     'and',
                     'cats,',
                     'these',
                     'are',
                     'my',
                     'love'],
 './iitfidf/2.txt': ['A',
```

The result of word splitting.

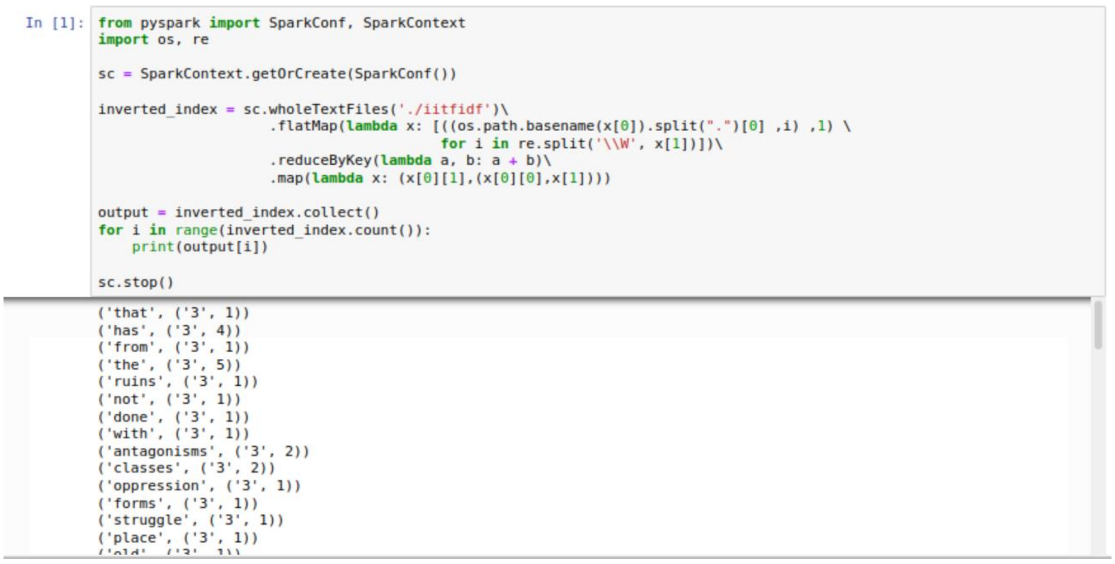


```
Inverted Index
{'A': ['./iitfidf/2.txt'],
 'All': ['./iitfidf/2.txt'],
 'Bourgeoisie': ['./iitfidf/3.txt'],
 'Communism.': ['./iitfidf/2.txt'],
 'Czar,': ['./iitfidf/2.txt'],
 'Europe': ['./iitfidf/2.txt'],
 'Europe--the': ['./iitfidf/2.txt'],
 'French': ['./iitfidf/2.txt'],
 'German': ['./iitfidf/2.txt'],
 'Guizot,': ['./iitfidf/2.txt'],
 'I': ['./iitfidf/1.txt'],
 'It': ['./iitfidf/3.txt'],
 'Metternich': ['./iitfidf/2.txt'],
 'Our': ['./iitfidf/3.txt'],
 'Pope': ['./iitfidf/2.txt'],
 'Powers': ['./iitfidf/2.txt'],
 'Proletariat.': ['./iitfidf/3.txt'],
 'Radicals': ['./iitfidf/2.txt'],
```

The result of inverse indexing.

jupyter LabS04_04_SparkInvIndex

Run the forth lab SparkInvIndex.



```
In [1]: from pyspark import SparkConf, SparkContext
        import os, re

        sc = SparkContext.getOrCreate(SparkConf())

        inverted_index = sc.wholeTextFiles('./iitfidf')\
                           .flatMap(lambda x: [((os.path.basename(x[0]).split(".")[0] ,i) ,1) \
                                               for i in re.split('\\W', x[1])])\
                           .reduceByKey(lambda a, b: a + b)\
                           .map(lambda x: (x[0][1],(x[0][0],x[1])))

        output = inverted_index.collect()
        for i in range(inverted_index.count()):
            print(output[i])

        sc.stop()
```

```
('that', ('3', 1))
('has', ('3', 4))
('from', ('3', 1))
('the', ('3', 5))
('ruins', ('3', 1))
('not', ('3', 1))
('done', ('3', 1))
('with', ('3', 1))
('antagonisms', ('3', 2))
('classes', ('3', 2))
('oppression', ('3', 1))
('forms', ('3', 1))
('struggle', ('3', 1))
('place', ('3', 1))
```

Using Spark to perform inverse indexing, the result contains the word, the document included this word and term frequency.