NLP and Text Mining Basics

Venkat Reddy

Contents

- What is text mining
- Corpus
- Preparing text data for analysis
- NLTK package
- Basic summary
- DTM

Text Mining

- **This vacuum cleaner sucks**
- **I never had such pizza before, not sure about future either**
- No action, no drama, no comedy, no romance, just pure horror
- The food was not good, it was bad
  The food was not bad, it was good

What is Text mining?

What is text mining

- Making sense out of text data
- Datamining on text input
- Exploratory data analysis on text data
- Also known as Text Analytics
- What is NLP – Natural Language Processing

Two major categories of text mining

- Text mining for prediction
- Text mining for pattern recognition

Text mining for pattern recognition

- Basic exploratory analysis to find the patterns or clusters.
- Some patterns might not mean anything to business
- Analogues to unsupervised learning
- Data has to be large in this case
- Example:
  - Document clustering
  - Document summarisation
  - Theme extraction

Text mining for prediction

- Human experts classify a set of documents/text. We predict new document's category
- Data can be small
- Analogues to supervised learning
- Example:
  - Sentiment Analysis
  - Classification of text into various categories

Data Preparation for Text Mining

Text data is Unstructured

- **This vacuum cleaner sucks**
- **I never had such pizza before, not sure about future either**
- No action, no drama, no comedy, no romance, just pure horror
- The food was not good, it was bad
  The food was not bad, it was good

Text data is Unstructured

- Numerical structured data
  - rows and columns.
  - For every record(row) we have information well organised in the form of columns.
  - Each column captures a specific section of information
  - Every record has almost all columns available
  - Easy to perform mathematical and statistical computations

Text data is unstructured

- Most of the text data has one or two columns
- Whole data is in one column
- Each record might have different length
- Difficult to arrange it as a dataset
- Text data is not very well structured.
- Direct computation on text data not easy

Computers don't Understand Language

- Direct computation on text data not easy.
- Computers don't understand a **Sentence** or a **Word** or any **Underlying Emotion**.
- We need to bring the data structure to a point where computers can convert text data into number, process the numbers, convert those numbers back to Text data.

Giving structure to Unstructured text

- Text data is unstructured. We need to add some basic structure.
- How to prepare data for computations.
- How to represent the text?
- There is a lot of pre-processing required before jumping on to analysis

Corpus

- Corpus is collection of text or sometimes text files.
- Each document is an entity/observation/ record in corpus.
- You can think corpus as a raw data frame for text data.

NLTK package

- The package used for all the NLP tasks in python is NLTK
- NLTK has rich documentation and examples
- The package has functions for all text mining tasks like
  - Reading data
  - Tokenizing
  - Stemmers
  - Taggers
  - Parsers
  - WordNet
  - Evaluation Metrics
- We will use NLTK package to process text

LAB: Update NLTK

```
#Step-1 : Write this code and stop it after 2 mins
import nltk
nltk.download('all')

#Step-2 :
#Run below code and get the temp folder.
nltk.data.path

#The temp folder looks path like this
#C:\\Users\\Admin\\AppData\\Roaming\\nltk_data
#C:\\Users\\r14\\AppData\\Roaming\\nltk_data

#Setp-3
#Copy all 10 folders from this folder nltk_data_venkat
#Run below code after copying
nltk.download('all')
```

LAB: Update NLTK - Shortcut

- copy NLTK folder manually from share drive(Already downloaded by a user)
- Go find your lib folder (C:\Users\StatInfer\Anaconda3\Lib)
- Update the lib folder (append files don’t delete and paste)

Reading the data as Corpus

- We can read data using pandas and create corpus
- But for text mining it is better to use NLTK own reader
- We can utilise NLTK's one of many corpus-reader-functions to read in our text data.
- If we want to read Plain Text data as corpus, NLTK's PlaintextCorpusReader(File_Directory , File_Name_Pattern) function would be good choice.
- There are some other variants of Corpus Reader available to read-in specific different structures of data or directory.
- NLTK has downloadable 'Set of Corpus' or Corpora.
- We can utilise sample NLTK corpora and perform our analytics on them or use Stemmers or Lemmatizers from corpus.

Demo: Corpus Reader

```
from nltk.corpus import PlaintextCorpusReader
#defining our corpus directory:
dirname_politics = "D:/Google Drive/Training/Datasets/News Group Data
Text/mini_newsgroups/talk.politics.misc"

#Reading the data with corpus:
politics_corpus = PlaintextCorpusReader(dirname_politics, '.*')

#All file names in the directory
politics_corpus.fileids()

#Few news examples
politics_corpus.raw('176869')
politics_corpus.raw('176878')

politics_corpus.words('179097')[1:100]
```

Preparing Data for text mining

statinfer

RAW Text cleaning data stages

- The raw data need to be cleaned to a great extent
- There are many steps in cleaning the data
  - Tokenizing
  - Stemming
  - Stop word removal
  - Lemmatising
  - Punctuation
- The final accuracy largely depends on quality of input data. The data preparation takes more time than final analysis

Preparing Data for text mining

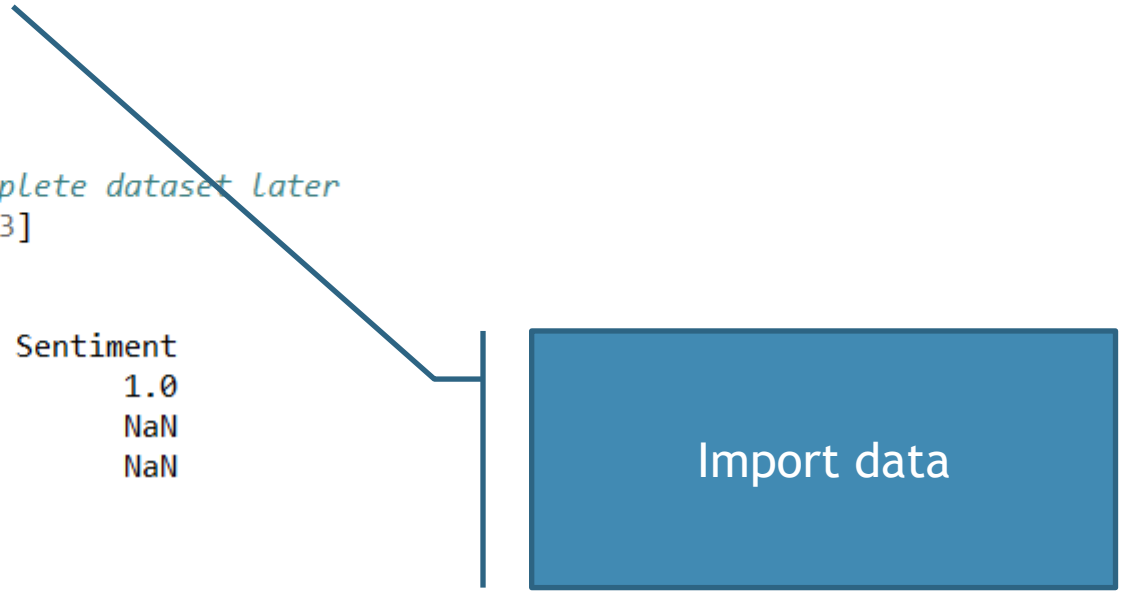
- There are some common words in every document. They might not have any meaning
  - a, an, the, this, is, was
- Sometimes we refer the same word in different ways.
  - U.S.A, United States, States, USA
- Few words in the document have same root but used in different ways
  - Buying, bought, buy
- Few documents have numbers
- Special characters and punctuation
- Upper case and lower case

Tokenizing

- A token is the technical name for a sequence of characters/words/sentences.
- Each “Entity" that is making a sentence or a paragraph when kept in a sequence would be called token.
- Word Token:
  - Each word is a token when a sentence is "tokenized" into words.
- Sentence Token
  - Each sentence is a token when a paragraph is tokenized.
- Tokenization is based on specific split rule:
  - word_tokenize: split would generally be ‘Space’
  - Sentence_tokenize : split would generally be ‘. {Capital letter}’

Demo: Tokenization

```
In [58]: import pandas as pd
    ...:
    ...: #importing data
    ...: User_restaurants_reviews = pd.read_csv("D:/Google Drive/Training/Datasets/User_Reviews/
User_restaurants_reviews.csv")
    ...: User_restaurants_reviews.shape
    ...: User_restaurants_reviews.head(20)
    ...:
    ...: ##############
    ...: #Lets take a small data, we will work on complete dataset later
    ...: user_data_tiny = User_restaurants_reviews[0:3]
    ...: user_data_tiny
Out[58]:
                                              Review  Sentiment
0                           Wow... Loved this place.        1.0
1  I learned that if an electric slicer is used t...        NaN
2                    But they don't clean the chiles?        NaN
```

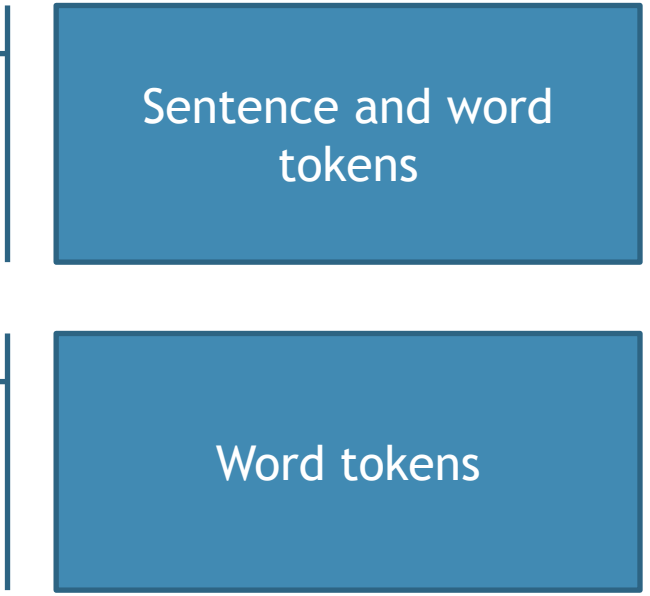


Demo: Tokenization

```
In [59]: from nltk.tokenize import sent_tokenize, word_tokenize
    ...:
    ...: example_text = user_data_tiny["Review"][0]
    ...: print(example_text)

In [60]: Wow... Loved this place.
sent_tokens = sent_tokenize(example_text)
    ...: print(sent_tokens)
    ...:
    ...: word_tokens = word_tokenize(example_text)
    ...: print(word_tokens)

In [61]: ['Wow...', 'Loved this place.']
['Wow', '...', 'Loved', 'this', 'place', '.']
```



Stop Words

- There are some common words in every document.
- These words are not really informative
- Most of the times they are irrelevant for document representation
  - Eg: a, an, the, this, is, was, for, are
- These words carries importance for humans but for analysis these words doesn’t give any insights.
- It’s better to remove these words form our documents.

Demo: Stop Words

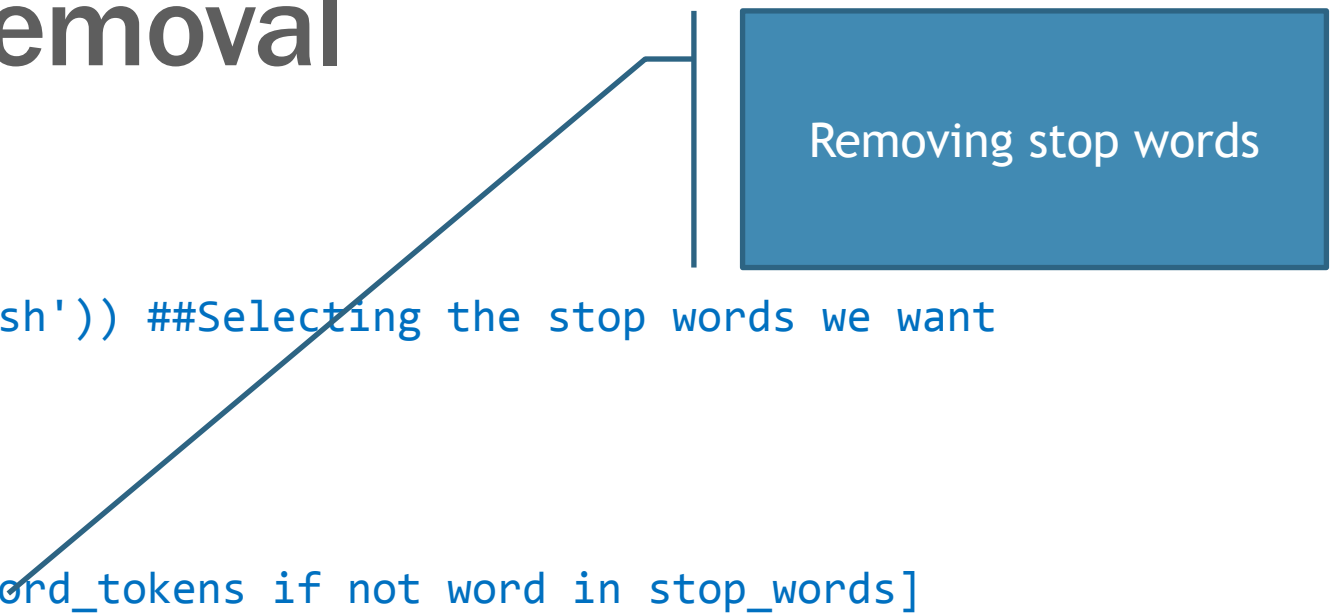
- This code is progression of Tokenization

```
In [66]: from nltk.corpus import stopwords
    ...: stop_words = set(stopwords.words('english')) ##Selecting the stop words we want
    ...: print(len(stop_words))
    ...: print(stop_words)
153
{'me', 'mightn', 'itself', 'won', 'himself', 'be', 'have', 'having', 'did', 'against', 'then', 'haven', 'by',
'hers', 'doesn', 'ours', 'of', 'while', 'your', 'y', 'those', 'does', 'weren', 'herself', 'so', 've',
'between', 'as', 'yours', 'are', 'wouldn', 'do', 'some', 'was', 'few', 'off', 'didn', 'themselves', 'their',
'now', 'aren', 'a', 'after', 'each', 'before', 'll', 's', 'over', 'needn', 'than', 'both', 'all', 'her',
'couldn', 'these', 'what', 'why', 'them', 'other', 'more', 'my', 'were', 'am', 'd', 'isn', 'its', 'when',
'shouldn', 'about', 'theirs', 'don', 'we', 'not', 'only', 'can', 'she', 'below', 'where', 'shan', 'at',
'until', 'will', 'which', 'i', 'but', 'through', 'doing', 'it', 'an', 'any', 'in', 'he', 'myself', 'how',
'again', 'further', 'there', 'too', 'for', 'above', 'o', 'mustn', 'and', 'this', 'if', 'that', 'm', 'should',
'has', 'our', 'from', 'had', 'own', 'under', 'ma', 'him', 'his', 'yourselves', 'ourselves', 'just', 'same',
'or', 'whom', 'wasn', 'very', 'here', 're', 'to', 'on', 'because', 'into', 'nor', 'hadn', 'been', 'they',
'once', 'most', 'the', 'up', 'out', 't', 'with', 'during', 'down', 'hasn', 'yourself', 'is', 'no', 'you',
'being', 'who', 'ain', 'such'}
```

English general stopwords

statinfer

Demo: Stop Words removal




```
####################Stop Words
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english')) ##Selecting the stop words we want
print(len(stop_words))
print(stop_words)

#Removing the stopwords
filtered_sentence = [word for word in word_tokens if not word in stop_words]
print(filtered_sentence)

#The above code is simpler form of below code
filtered_sentence1=[]
for w in word_tokens:
    if w not in stop_words:
        filtered_sentence1.append(w)
print(filtered_sentence1)
```

Demo: Stop Words removal

```
############################Update with your own stop words
stop_words.update(['.', ',', '"', "'", '?', '!', ':', ';', '(', ')', '[', ']', '{', '}'])
print(len(stop_words))
print(stop_words)

filtered_sentence1 = [word for word in word_tokens if not word in stop_words]
print(filtered_sentence1)
```

Adding your own stopwords

Stemming

- Few words in the document have same root but used in different ways
- For example
  - Create, creating, created, creates
- Different words are derived from same root word.
- Same root word but different forms like plural form, adverb form, present tense, past tense form, continuous tense form.
- Need to be careful with tools. Sometimes they simply trim edges of all words.

Demo: Stemming

```
In [3]: from nltk.stem import PorterStemmer
   ...: stemmer = PorterStemmer()   #Defining the Stemmer
   ...:

In [4]: example_text1 = user_data_tiny["Review"][1]
   ...: print(example_text1)
   ...:
I learned that if an electric slicer is used the blade becomes hot enough to start to cook the prosciutto.

In [5]: word_tokens1 = word_tokenize(example_text1)
   ...: print(word_tokens1)
   ...:
['I', 'learned', 'that', 'if', 'an', 'electric', 'slicer', 'is', 'used', 'the', 'blade', 'becomes', 'hot',
'enough', 'to', 'start', 'to', 'cook', 'the', 'prosciutto', '.']

In [6]: stem_tokens=[stemmer.stem(word) for word in word_tokens1]
   ...: print(stem_tokens)
   ...:
['I', 'learn', 'that', 'if', 'an', 'electr', 'slicer', 'is', 'use', 'the', 'blade', 'becom', 'hot', 'enough',
'to', 'start', 'to', 'cook', 'the', 'prosciutto', '.']
```

Actual word tokens

Word tokens after stemming

Lemmatizing

- Stemming is very basic way of trimming the words.
  - {become} is stemmed as {becom}
- A very similar operation to stemming is called lemmatizing.
- Stemming gives us root stem which can often be **non-existent** word.
- Lemmatization gives us Lemma, which **can be looked up** in a dictionary.
- Mostly word and it's generated Lemma are very similar words.
- But sometimes, we wind up with a completely different word.

Demo: Lemmatizing

```
In [19]: from nltk.stem import WordNetLemmatizer
    ...: lemmatizer = WordNetLemmatizer()           #Choosing the Lemmatizer
    ...: Lemmatized_tokens = [lemmatizer.lemmatize(word) for word in word_tokens1]
    ...: print(Lemmatized_tokens)
['I', 'learned', 'that', 'if', 'an', 'electric', 'slicer', 'is', 'used', 'the', 'blade', 'becomes',
'hot', 'enough', 'to', 'start', 'to', 'cook', 'the', 'prosciutto', '.']
```

Demo: Lemmatizing

```
In [22]: print(Lemmatized_tokens)
['I', 'learned', 'that', 'if', 'an', 'electric', 'slicer', 'is', 'used', 'the', 'blade', 'becomes',
'hot', 'enough', 'to', 'start', 'to', 'cook', 'the', 'prosciutto', '.']

In [23]: print(stem_tokens)
['I', 'learn', 'that', 'if', 'an', 'electr', 'slicer', 'is', 'use', 'the', 'blade', 'becom', 'hot',
'enough', 'to', 'start', 'to', 'cook', 'the', 'prosciutto', '.']

In [24]:
```

RegEx : Regular Expressions

- How to identify/ remove numbers from before analysis
- How to identify/ remove currency symbols?
- How to identify/ remove punctuation ?
- Http, email address, other sybmols etc.,

statinfer

RegEx : Regular Expressions

- As we are working with text data, a bit of manipulation is needed.
- Regular Expressions can help us finding pattern and replace or modify them.
- Regex is it's own language, and is basically the same no matter what programming language we are using with it.
- As Regex can be too vast, we will just go through very minimal to help us find and replace some bits and pieces in our data.
- We will use python package re and it's function re.sub()

Syntax

re.sub(pattern, replacement, string)

RegEx- Syntax

- Numeric String /^[0-9]+$/
- An Identifier (or Name) /[a-zA-Z_][0-9a-zA-Z_]*/
- An Image Filename /^\w+\.(gif|png|jpg|jpeg)$/I
- /^\w+([\.-]?\w+)*@\w+([\.-]?\w+)*(\.\w{2,3})+$/
- HTTP Address /^http:\/\/\S+(\/\S+)*(\/)?$/

Demo: RegEx

- Search and replace all numbers, currency and punctuations in a sentence.

```
In [35]: review22_text = User_restaurants_reviews["Review"][22]
    ...:
    ...: import re
    ...: #re.sub(regexpattern, replacement, string)
    ...: #Replacing numbers and currency with space
    ...: review22_text_cleaned=re.sub(r'\W+|\d+|_',  ' ',  review22_text)
    ...: print("Text after removing currency - \n " + review22_text_cleaned)
    ...:
    ...: print("Actual Text - \n " + review22_text)
    ...:
Text after removing currency -
 We ordered     Margaritas they couldnt get the machine to work because it was frozen so refunded
our money
Actual Text -
 We ordered 2 $.99 Margaritas - they couldnt get the machine to work because it was frozen so
refunded our money.
```

Removed numbers and currency

Case Study : The news articles text mining

Importing data as multiple documents

- As explained earlier, we can directly import from a text file by opening and reading the file.
- However, sometimes we have many documents in a folder, we can write a loop to iterate through directory with os package.
- We will see how to iterate through the directories to read our documents.

LAB: Importing Data

```
import os
path = 'D:/Google Drive/Training/Datasets/News Group Data Text/mini_newsgroups/sci.space'
doc_dict = {}

for subdir, dirs, files in os.walk(path):
    for file in files:
        file_path = subdir + os.path.sep + file
        f = open(file_path, 'r')
        text = f.read()
        doc_dict[file] = text

#print(doc_dict)
print(doc_dict.keys())
doc_dict['60804']
```

statinfer

Convert to Lower Case

- Convert everything to lower case.
- Will be easy to compare and process the words later

```
In [130]: for my_var in doc_dict:
     ...:     doc_dict[my_var] = doc_dict[my_var].lower()
     ...:
     ...: doc_dict['60804']
Out[130]: 'newsgroups: sci.space\npath: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!
fs7.ece.cmu.edu!europa.eng.gtefsd.com!emory!swrinde!zaphod.mps.ohio-state.edu!uwm.edu!linac!
uchinews!raistlin!runyon.cim.cdc.com!pbd\nfrom: pbd@runyon.cim.cdc.com (paul dokas)\nsubject: big
amateur rockets\norganization: icem systems, inc.\ndate: fri, 16 apr 1993 14:15:34 gmt\nmessage-id:
<c5ky9y.mkk@raistlin.udev.cdc.com>\nsender: usenet@raistlin.udev.cdc.com (news poster)\nlines: 23\n
```

statinfer

Remove numbers

- Remove numbers if necessary
- Numbers might not carry any useful information in text analysis later

```
In [131]: import re
     ...: # Use regular expressions to do a find-and-replace
     ...:
     ...: for my_var in doc_dict:
     ...:     doc_dict[my_var] = re.sub(r'\d+',           # The pattern to search for
     ...:                               " ",              # The pattern to replace it with
     ...:                               doc_dict[my_var])  # The text to search
     ...:
     ...: doc_dict['60804']
Out[131]: 'newsgroups: sci.space\npath: cantaloupe.srv.cs.cmu.edu!crabapple.srv.cs.cmu.edu!fs
.ece.cmu.edu!europa.eng.gtefsd.com!emory!swrinde!zaphod.mps.ohio-state.edu!uwm.edu!linac!uchinews!
raistlin!runyon.cim.cdc.com!pbd\nfrom: pbd@runyon.cim.cdc.com (paul dokas)\nsubject: big amateur
rockets\norganization: icem systems, inc.\ndate: fri,   apr    : :  gmt\nmessage-id: <c ky
y.mkk@raistlin.udev.cdc.com>\nsender: usenet@raistlin.udev.cdc.com (news poster)\nlines:  \n\ni was
reading popular science this morning and was surprised by an ad in\nthe back.  i know that a lot of
```

Remove Punctuation

```
In [132]: import re
     ...: # Use regular expressions to do a find-and-replace
     ...: for my_var in doc_dict:
     ...:     doc_dict[my_var] = re.sub(r'\W+|\_',          # The pattern to search for
     ...:                               " ",                # The pattern to replace it with
     ...:                               doc_dict[my_var])  # The text to search
     ...:
     ...: doc_dict['60804']
     ...:
Out[132]: 'newsgroups sci space path cantaloupe srv cs cmu edu crabapple srv cs cmu edu fs ece cmu
edu europa eng gtefsd com emory swrinde zaphod mps ohio state edu uwm edu linac uchinews raistlin
runyon cim cdc com pbd from pbd runyon cim cdc com paul dokas subject big amateur rockets
organization icem systems inc date fri apr gmt message id c ky y mkk raistlin udev cdc com sender
```

Remove general English stop words

```
In [133]: from nltk.corpus import stopwords
     ...: from nltk import word_tokenize
     ...: stop = stopwords.words('english')
     ...: for my_var in doc_dict:
     ...:     doc_dict[my_var] = ' '.join([i for i in word_tokenize(doc_dict[my_var]) if i not in
stop])
     ...:
     ...: doc_dict['60804']
     ...:
Out[133]: 'newsgroups sci space path cantaloupe srv cs cmu edu crabapple srv cs cmu edu fs ece cmu
edu europa eng gtefsd com emory swrinde zaphod mps ohio state edu uwm edu linac uchinews raistlin
runyon cim cdc com pbd pbd runyon cim cdc com paul dokas subject big amateur rockets organization
icem systems inc date fri apr gmt message id c ky mkk raistlin udev cdc com sender usenet raistlin
```

Remove custom stop words

- This step is very important should not be ignored.
- We need to identify stop words in our corpus and eliminate them.
- This step takes a lot of time, some business understanding and context of data.
- For example
  - If you are working with the iPhone review data
  - apple, iPhone, phone, mobile, i-phone can be removed.

Remove custom stop words

```
In [134]: custom_stop = stopwords.words('english') + ['news', 'writes', 'told']
     ...: for my_var in doc_dict:
     ...:     doc_dict[my_var] = ' '.join([i for i in word_tokenize(doc_dict[my_var]) if i not in
custom_stop])
     ...:
     ...: doc_dict['60804']
     ...:
Out[134]: 'newsgroups sci space path cantaloupe srv cs cmu edu crabapple srv cs cmu edu fs ece cmu
edu europa eng gtefsd com emory swrinde zaphod mps ohio state edu uwm edu linac uchinews raistlin
runyon cim cdc com pbd pbd runyon cim cdc com paul dokas subject big amateur rockets organization
```

Lemmatizing

```
In [135]: from nltk.stem import WordNetLemmatizer
     ...: lemmatizer = WordNetLemmatizer()          #Choosing the Lemmatizer
     ...: for my_var in doc_dict:
     ...:     doc_dict[my_var] = ' '.join([lemmatizer.lemmatize(i) for i in
word_tokenize(doc_dict[my_var])])
     ...:
     ...: doc_dict['60804']
Out[135]: 'newsgroups sci space path cantaloupe srv c cmu edu crabapple srv c cmu edu f ece cmu edu
europa eng gtefsd com emory swrinde zaphod mp ohio state edu uwm edu linac uchinews raistlin runyon
cim cdc com pbd pbd runyon cim cdc com paul dokas subject big amateur rocket organization icem
system inc date fri apr gmt message id c ky mkk raistlin udev cdc com sender usenet raistlin udev
cdc com poster line reading popular science morning surprised ad back know lot ad back p fringe
science questionablely legal one really grabbed attention company name personal missle inc something
```

Stemming

- Few words in the document have same root but used in different ways.
  - Create, creating, created, creates
- Different words are derived from same root word.
- Same root word but different forms like plural form, adverb form, present tense, past tense form, continuous tense form.
- Need to be careful with tools. Sometimes they simply trim edges of all words.
- You may want to ignore stemming if you did lemmatizing

Document Term Matrix

Document Term Matrix

- Document – text document
- Can we consider each sentence as document? Can we call a sentence as a basic form of document
- We can create DTM and work with sklearn and other regular packages

- Doc1: Loved this place
- Doc2: At this place, crust is not good.
- Doc3: Loved it, good thin crust pizza.

Document Term Matrix

Doc1: Loved this place, good pizza
Doc2: At this place, crust is not good. pizza is not good.
Doc3: Loved it, good thin crust pizza.

Terms

Documents

	loved	this	place	at	crust	is	not	good	it	thin	pizza
Doc1	1	1	1					1			1
Doc2		1	1	1	1	2	2	2			1
Doc3	1				1			1	1	1	1

LAB: Document Term Matrix

Read the data

```
In [211]: import pandas as pd
     ...: User_restaurants_reviews = pd.read_csv("D:/Google Drive/Training/Datasets/User_Reviews/
User_restaurants_reviews.txt", sep='\t', header = None)
     ...: User_restaurants_reviews.shape
     ...: User_restaurants_reviews.head()
Out[211]:
                                                   0    1
0                            Wow... Loved this place.  1.0
1  I learned that if an electric slicer is used t...  NaN
2                  But they don't clean the chiles?  NaN
3                                Crust is not good.  0.0
4          Not tasty and the texture was just nasty.  0.0
```

LAB: Document Term Matrix

```
In [213]: from sklearn.feature_extraction.text import CountVectorizer
     ...: countvec = CountVectorizer()
     ...:

In [214]: Test_dtm = pd.DataFrame(countvec.fit_transform(user_data_tiny[0]).toarray(),
columns=countvec.get_feature_names(), index=None)
     ...: Test_dtm
     ...:
Out[214]:
   an  becomes  blade  but  chiles  clean  cook  don  electric  enough ...   \
0   0        0      0    0       0      0     0    0         0       0 ...
1   1        1      1    0       0      0     1    0         1       1 ...
2   0        0      0    1       1      1     0    1         0       0 ...

   prosciutto  slicer  start  that  the  they  this  to  used  wow
0           0       0      0     0    0     0     1   0     0    1
1           1       1      1     1    2     0     0   2     1    0
2           0       0      0     0    1     1     0   0     0    0

[3 rows x 26 columns]
```

LAB: Document Term Matrix

```
user_data_r100 =User_restaurants_reviews[0:100]
Test_DTM_r100 =
pd.DataFrame(countvec.fit_transform(user_data_r100[0]).toarray(),
columns=countvec.get_feature_names(), index=None)
```

```
#Lets look at the TDM
Test_DTM_r100
Test_DTM_r100[0:10]
Test_DTM_r100[0:20]
Test_DTM_r100[0:50]
Test_DTM_r100[1:80]
Test_DTM_r100
```

Conclusion

Conclusion

- Text is unstructured / semi structed data.
- Preparing data analysis is the key step in text mining
- Text mining involves lot of customisation
- We discussed very basic steps of data preparation and summarisation of text