Text Analytics With Python

Section 1

Section 2

Section 3

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Lesson Objectives

- Learn Python libraries for text analytics

Text Analytics / NLP Libraries for Python

Library	Pros	Cons
NLTK	- 'the' library - modular	- Can be steep learning curve - Might not be high performant
TextBlob - Built on top of NLTK	- Easily accessible - Fast prototyping	- Performance may not be high
Stanford CoreNLP - Core java library with python wrappers	- Fast - Lot of use in production	
SpaCy - New		
Gensim - Topic modeling		

NLTK

>> Section 1

Section 2

Section 3

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

Python Libraries

- NLTK – Natural Language Tool Kit
  - Very popular and versatile library
  - http://www.nltk.org/
  - https://github.com/nltk

- NLTK Features:
  - Supports multiple algorithms
    - Lexical analysis : tokenization of text
    - Ngram analytics
    - Named entity recognition
  - Comes with data (50+ corpora / lexicons)

Installing NLTK

- NLTK is part of modern python stacks (like 'anaconda')

- Installing NLTK

```
// using pip tool
$  pip install nltk

// to install nltk dataset
$  python3
>   nltk.download()
# This will pop up a UI, select a directory to
# download data.  This directory will be referred
# as 'nltk_data_dir'. be sure to add this as follows

> nltk.data.path.append("/Users/sujee/data/nltk_data")
```

NLTK Dataset / Corpus

- NLTK (data) comes with pretty interesting datasets / corpus
- This is part of 'nltk.corpus' package

```
import nltk
from os.path import expanduser
nltk.data.path.append( expanduser("~") + "/data/nltk_data")
from nltk.corpus import words

print (words.readme())
words_en_basic = words.words('en-basic')
print ("words_en_basic : ", len(words_en_basic))
print (words_en_basic[:10])
#  words_en_basic : 850
# ['I', 'a', 'able', 'about', 'account', 'acid', 'across', 'act',
'addition', 'adjustment']

words_en = words.words('en')
print ("words_en : ", len(words_en))
print(words_en[:10])
# words_en :  235886
# ['A', 'a', 'aa', 'aal', 'aalii', 'aam', 'Aani', 'aardvark',
'aardwolf', 'Aaron']
```

NLTK Corpus : State of the Union

- Each corpus has a 'readme()' function
- State of the Union addresses from 1945 to 2006

```
from nltk.corpus import state_union

print(state_union.readme())
print(state_union.fileids())
# ['1945-Truman.txt', '1963-Kennedy.txt',... '1964-Johnson.txt', '1974-
Nixon.txt',... '1981-Reagan.txt',... '2000-Clinton.txt', '2001-GWBush-1.txt',
'2001-GWBush-2.txt', ... '2006-GWBush.txt']

# see all words in entier state of the union corpus
print (len(state_union.words()))
# 399822

# get one particular state of the union
gw2006 = state_union.raw('2006-GWBush.txt')
print (len(gw2006))
# 33411

# get only words for one SOTU
gw2006_words = state_union.words('2006-GWBush.txt')
print (len(gw2006_words))
# 6515

gw2006_sentences = state_union.sents('2006-GWBush.txt')
print(gw2006_sentences[:10])
```

NLTK Corpus : Novels

- These are public domain novels from Gutenberg project

```
from nltk.corpus import gutenberg

print (gutenberg.readme())

# Let's see what we have
print (gutenberg.fileids())
# ['austen-emma.txt', 'austen-persuasion.txt', 'austen-sense.txt', 'bible-
kjv.txt', 'blake-poems.txt', 'bryant-stories.txt', 'burgess-busterbrown.txt',
'carroll-alice.txt', 'chesterton-ball.txt', 'chesterton-brown.txt',
'chesterton-thursday.txt', 'edgeworth-parents.txt', 'melville-moby_dick.txt',
'milton-paradise.txt', 'shakespeare-caesar.txt', 'shakespeare-hamlet.txt',
'shakespeare-macbeth.txt', 'whitman-leaves.txt']

# get Moby Dick novel
moby_dick = gutenberg.raw('melville-moby_dick.txt')
print (moby_dick[1:1000])

# get the words for Moby Dick
moby_dick_words = gutenberg.words('melville-moby_dick.txt')
print (len(moby_dick_words))
# 260819
```

NLTK Tokenizing Text

- First step in analyzing text is splitting raw text into words
  - Called tokenizing
- Nltk.tokenize package offers few handy ones
  - Word_tokenize : gives out words
  - Workpunct_tokenize : numbers and punctuations in their own words
  - Sent_tokenize : splits into sentences

```
import nltk
from nltk.tokenize import word_tokenize
from nltk.tokenize import wordpunct_tokenize
from nltk.tokenize import sent_tokenize

text = """I went to Starbucks. And bought a latte for $4.50!
Yum :)"""

print(sent_tokenize(text))
# ['I went to Starbucks.',     'And bought a latte for $4.50!',   'Yum :-)']

print(word_tokenize(text))
# ['I', 'went', 'to', 'Starbucks', '.', 'And', 'bought', 'a', 'latte', 'for',
'$', '4.50', '!', 'Yum', ':', '-', ')']

print(wordpunct_tokenize(text))
# ['I', 'went', 'to', 'Starbucks', '.', 'And', 'bought', 'a', 'latte', 'for',
'$', '4', '.', '50', '!', 'Yum', ':-)']
```

Lab: TEXT-1 : NLTK Intro

- **Overview**:
  Get to know NLTK library

- **Builds on previous labs**:
  None

- **Approximate time**:
  15 mins

- **Instructions**:
  – 1-NLTK-Intro

Lab: Text-2 : Text Analytics With NLTK

- **Overview**:
  Analyzing raw text with NLTK library

- **Builds on previous labs**:
  TEXT-1 : NLTK intro

- **Approximate time**:
  15 mins

- **Instructions**:
  – 2-analyzing-text-with-nltk

Lab: Text-3 : Ngrams

- **Overview**:
  Analyzing text for Ngrams

- **Builds on previous labs**:
  TEXT-2 : NLTK

- **Approximate time**:
  15 mins

- **Instructions**:
  – 3-ngrams

TextBlob

>> Section 1

Section 2

Section 3

Licensed for personal use only for Vincent Chang <vincent.chang@macys.com> from Python ML @ Macy's 04/25/2108 @ 2018-04-25

TextBlob

- Simplified library for text processing in Python
- Built on NLTK & Pattern

- Features
  - Tokenization
  - Word / Phrase frequencies (ngrams)
  - Noun phrase extraction
  - Sentiment analysis
  - Classification

- https://textblob.readthedocs.io/en/dev/
- https://github.com/sloria/TextBlob/

Installing TextBlob

```
// using pip tool
$  pip install -U textblob

// to get dataset (same as NLTK)
$ python -m textblob.download_corpora

# This will pop up a UI, select a directory to
# download data.  This directory will be referred
# as 'nltk_data_dir'. be sure to add this as follows

> nltk.data.path.append("/Users/sujee/data/nltk_data")
```

TextBlob Usage

```
from textblob import TextBlob

import nltk
# setup nltk data
from os.path import expanduser
nltk.data.path.append( expanduser("~") + "/data/nltk_data")

text = """TextBlob aims to provide access to common text-
processing operations through a familiar interface. You can
treat TextBlob objects as if they were Python strings that
learned how to do Natural Language Processing."""

tb = TextBlob(text)
print(tb)
```

TextBlob Usage : Tokenizing

```
from textblob import TextBlob

...

text = """TextBlob aims to provide access to common text-processing
operations through a familiar interface. You can treat TextBlob
objects as if they were Python strings that learned how to do
Natural Language Processing."""

tb = TextBlob(text)

print(tb.words)
['TextBlob', 'aims', 'to', 'provide', 'access', 'to', 'common',
'text-processing', 'operations', 'through', 'a', 'familiar',
'interface', 'You', 'can', 'treat', 'TextBlob', 'objects', 'as',
'if', 'they', 'were', 'Python', 'strings', 'that', 'learned', 'how',
'to', 'do', 'Natural', 'Language', 'Processing']

print(tb.sentences)
[Sentence("TextBlob aims to provide access to common text-processing
operations through a familiar interface."), Sentence("You can treat
TextBlob objects as if they were Python strings that learned how to
do Natural Language Processing.")]
```

TextBlob Usage : Sentiment Analysis

- 'sentiment' returns a tuple (polarity, subjectivity)
- Polarity ranges **from -1.0 (very negative) to +1.0 (very positive)**
- Subjectivity ranges from **0.0 (very objective) to +1.0 (very subjective)**

```
from textblob import TextBlob

tweets = ["I love bigmacs",
          "I hate this traffic!",
          "American Idol is awesome!",
          "this song is lame",
          "Let's go to beach"]

for tweet in tweets:
    tb = TextBlob(tweet)
    print("{} ==> {}".format(tweet, tb.sentiment))

I love bigmacs ==> Sentiment(polarity=0.5, subjectivity=0.6)
I hate this traffic! ==> Sentiment(polarity=-1.0, subjectivity=0.9)
American Idol is awesome! ==> Sentiment(polarity=0.5, subjectivity=0.5)
this song is lame ==> Sentiment(polarity=-0.5, subjectivity=0.75)
Let's go to beach ==> Sentiment(polarity=0.0, subjectivity=0.0)
```

TextBlob Usage : Word Counts

```
from textblob import TextBlob

text = """It was a sunny day! We went to the dog park.  Lots
of dogs were running around.  My dog likes to run too; so he
had a great time.  I bought ice cream from the ice cream
truck. Yummy!It was a perfect sunny day!"""

tb = TextBlob(text)

print(tb.word_counts)
defaultdict(<class 'int'>, {'it': 2, 'was': 2, 'a': 3,
'sunny': 2, 'day': 2, 'we': 1, 'went': 1, 'to': 2, 'the': 2,
'dog': 2, 'park': 1, 'lots': 1, 'of': 1, 'dogs': 1, 'were':
1, 'running': 1, 'around': 1, 'my': 1, 'likes': 1, 'run': 1,
'too': 1, 'so': 1, 'he': 1, 'had': 1, 'great': 1, 'time': 1,
'i': 1, 'bought': 1, 'ice': 2, 'cream': 2, 'from': 1,
'truck': 1, 'yummy': 1, 'perfect': 1})

print(tb.word_counts['sunny'])
2
```

TextBlob Usage : Ngrams

```
text = """It was a sunny day! We went to the dog park.  Lots
of dogs were running around.  My dog likes to run too; so he
had a great time.  I bought ice cream from the ice cream
truck. Yummy!It was a perfect sunny day!"""

tb = TextBlob(text)

print(tb.ngrams(n=2))
```

```
[WordList(['It', 'was']), WordList(['was', 'a']),
WordList(['a', 'sunny']), WordList(['sunny', 'day']),
WordList(['day', 'We']), WordList(['We', 'went']),
WordList(['went', 'to']), WordList(['to', 'the']),
WordList(['the', 'dog']), WordList(['dog', 'park']),
WordList(['park', 'Lots']), WordList(['Lots', 'of']),
WordList(['of', 'dogs']), WordList(['dogs', 'were']),
WordList(['were', 'running'])
...
...]
```

TextBlob Usage : Language Detection and Translation

- Detect languages and even translate!
- Translation is powered by Google Translate API

```
text_en = "I just had dinner"

TextBlob(text_en).translate(to='es')
Acabo de cenar

TextBlob(text_en).translate(to='ja')
私はちょうど夕食

text_jp = u"私はちょうど夕食"
TextBlob(text_jp).detect_language()
ja
```

Lab: Text-4 : TextBlob

- **Overview**:
  Get familiar with TextBlob API

- **Builds on previous labs**:
  None

- **Approximate time**:
  15 mins

- **Instructions**:
  – 4-ngrams

Jump Point: TFIDF Theory

- Go to: Text Analytics Core : TF-IDF section

Lab: Text-5 : TF-IDF

- **Overview**:
  Calculate and understand TF-IDF scores

- **Builds on previous labs**:
  TEXT4 : TextBlob

- **Approximate time**:
  15 mins

- **Instructions**:
  – 5-tfidf

TF-IDF With SciKit Learn Library

- Scikit-Learn has a good TFIDF Implementation
  - sklearn.feature_extraction.text.TfidfVectorizer
- TFidfVectorizer
  - Can read a corpus (files / collection of strings)
  - And compute TFIDF
  - It gives 'document term matrix'

TF-IDF with SciKit Learn Code

```
from sklearn.feature_extraction.text import TfidfVectorizer

d0 = "the brown dog likes the white cow"
d1 = "the grass is brown"
d2 = "the spotted cow likes green grass"
documents = [d0,d1,d2]
tf = TfidfVectorizer(analyzer='word', ngram_range=(1,1),
                     min_df = 0, stop_words=None)
tfidf_matrix = tf.fit_transform(documents)
print(tfidf_matrix)  # document term matrix

document-term matrix
  (0, 8)0.521500948636
  (0, 0)0.335763711163
  ...
  (1, 8)0.373118805931
  (1, 0)0.480458397292

feature_names = tf.get_feature_names()
for i, feature in enumerate(feature_names):
    print(i,feature)
feature vectors
0 brown
1 cow
2 dog
...
```

Lab: Text-6 : TF-IDF With SciKit-Learn

- **Overview**:
  Calculate TF-IDF with SciKit-Learn

- **Builds on previous labs**:
  TEXT5 : TFIDF intro

- **Approximate time**:
  15 mins

- **Instructions**:
  – 6-tfidf-with-scikit-learn

Review Questions

Lesson Summary

- Learned Python libraries : TextBlob, NLTK, SciKit
- Implemented text analytics algorithms in Python