Sentiment Analysis in Simple Steps

Sharmila G Sivakumar

Aim

Explore the Natural Language Capabilities of Python

- Simple Example of Sentiment Analysis

What We Have

- Polarity dataset v2.0
  - Movie Review Data made available by Cornell University
  - 1000 postive reviews and 1000 negative reviews
  - http://www.cs.cornell.edu/people/pabo/movie-review-data

```
from glob import glob
neg = [open(f).read() for f in
glob('review_polarity/txt_sentoken/neg/*.txt')]
pos = [open(f).read() for f in
glob('review_polarity/txt_sentoken/pos/*.txt')]
print len(neg), len(pos)

>>> 1000, 1000
```

A Sample

- Neg - "this film is extraordinarily horrendous and i'm not going to waste any more words on it . \n"
- Pos - "kolya is one of the richest films i've seen in some time . \nzdenek sverak plays a confirmed old bachelor (who's likely to remain so) , who finds his life as a czech cellist increasingly impacted by the five-year old boy that he's taking care of . \nthough it ends rather abruptly-- and i'm whining , 'cause i wanted to spend more time with these characters-- the acting , writing , and production values are as high as , if not higher than , comparable american dramas . \nthis father-and-son delight-- sverak also wrote the script , while his son , jan , directed-- won a golden globe for best foreign language film and , a couple days after i saw it , walked away an oscar . \nin czech and russian , with english subtitles . \n"

Action Plan – Bag of Words

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

- **GetTerms** - Reduce each review to the list of words
- **Filtering** - Remove unnecessary words that will not add value for sentiment analysis
  - *is, among, but, and, it, that*
- **BaseWord** - Convert all inflections to their root word
  - *fry, fries, fried -> fry*
  - *going, go , went, goes -> go*
  - *movies, movie -> movie*
- **MakeFeatures** - Use the words thus extracted from a review as features to indicate the positiveness or negativeness of that review
- **Classifier** - Train a classifier to predict positivity

Our Samples Transformed – Neg

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

- "this film is extraordinarily horrendous and i'm not going to waste any more words on it . \n"
- ['film', 'extraordinarily', 'horrendous', u'go', 'waste', u'word']

Our Samples Transformed – Pos

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

- "kolya is one of the richest films i've seen in some time . \nzdenek sverak plays a confirmed old bachelor (who's likely to remain so) , who finds his life as a czech cellist increasingly impacted by the five-year old boy that he's taking care of . \nthough it ends rather abruptly-- and i'm whining , 'cause i wanted to spend more time with these characters-- the acting , writing , and production values are as high as , if not higher than , comparable american dramas . \nthis father-and-son delight-- sverak also wrote the script , while his son , jan , directed-- won a golden globe for best foreign language film and , a couple days after i saw it , walked away an oscar . \nin czech and russian , with english subtitles . \n"
- ['kolya', u'rich', u'film', "'ve", u'see', 'time', 'zdenek', 'sverak', u'play', u'confirm', 'old', 'bachelor', 'likely', 'remain', u'find', 'life', 'czech', 'cellist', 'increasingly', u'impact', 'five-year', 'old', 'boy', u'take', 'care', u'end', 'rather', 'abruptly', u'whine', 'cause', u'want', 'spend', 'time', u'character', u'act', u'write', 'production', u'value', 'high', u'high', 'comparable', 'american', u'drama', 'father-and-son', 'delight', 'sverak', 'also', u'write', 'script', 'son', 'jan', u'direct', u'win', 'golden', 'globe', 'best', 'foreign', 'language', 'film', 'couple', u'day', 'saw', u'walk', 'away', 'oscar', 'czech', 'russian', 'english', u'subtitle']

Extract individual terms

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

- Simplest – review.split()
  - Issues with punctuation characters
    - *“end-of-sentence. Vs amazon.com”*
    - *“father**'s** books”*
- A Seasoned Library – Topia Term Extractor
  - Content Term Extraction using POS Tagging

```
tagger.tokenize('This is a "simple" example.')
['This', 'is', 'a', '"', 'simple', '"', 'example',
'.']
tagger('This is a simple example.')
[['This', 'DT', 'This'],  ['is', 'VBZ', 'is'],  ['a',
'DT', 'a'], ['simple', 'JJ', 'simple'],
 ['example', 'NN', 'example'],  ['.', '.', '.']]
```

Term Extractor – Sample Review

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

```
from topia.termextract import tag
tagger = tag.Tagger()
tagger.initialize()  #Loads a lexicon
sample_review = "this film is extraordinarily
horrendous and i'm not going to waste any more words
on it. \n"
tagger(sample_review)

[['this', 'DT', 'this'], ['film', 'NN', 'film'],
['is', 'VBZ', 'is'], ['extraordinarily', 'RB',
'extraordinarily'], ['horrendous', 'JJ',
'horrendous'], ['and', 'CC', 'and'], ['i', 'NN', 'i'],
["'m", 'VBP', "'m"], ['not', 'RB', 'not'], ['going',
'VBG', 'going'], ['to', 'TO', 'to'], ['waste', 'NN',
'waste'], ['any', 'DT', 'any'], ['more', 'JJR',
'more'], ['words', 'NNS', 'word'], ['on', 'IN', 'on'],
['it', 'PRP', 'it'], ['.', '.', '.']]
```

Simplify Term Extractor

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

- If your eyes glaze over, you are not alone. :)
- A simple Noun, Verb, Adjective Classification is sufficient.
- WordNet to the rescue
  - A Database of nouns, verbs, adjectives and adverbs
  - A python api is available through nltk

```
from nltk.corpus import wordnet
for synset in wordnet.synsets("progress"):
    print synset.pos(), synset.definition()
n gradual improvement or growth or development
n the act of moving forward (as toward a goal)
n a movement forward
v develop in a positive way
v move forward, also in the metaphorical sense
v form or accumulate steadily
```

Simplify Term Extractor

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> Classifier

```
from nltk.corpus import wordnet

def get_wordnet_pos(tag):
    if tag.startswith('J'):
        return wordnet.ADJ
    elif tag.startswith('V'):
        return wordnet.VERB
    elif tag.startswith('N'):
        return wordnet.NOUN
    elif tag.startswith('R'):
        return wordnet.ADV
    else:
        return ''

terms = [(rec[0], get_wordnet_pos(rec[1])) for rec in
        tagger(sample_review)]
print terms
```

```
[('this', ''), ('film', u'n'), ('is', u'v'), ('extraordinarily',
u'r'), ('horrendous', u'a'), ('and', ''), ('i', u'n'), ("'m", u'v'),
('not', u'r'), ('going', u'v'), ('to', ''), ('waste', u'n'), ('any',
''), ('more', u'a'), ('words', u'n'), ('on', ''), ('it', ''), ('.',
'')]
```

Filtering unnecessary words

GetTerms -> **Filtering** -> BaseWord -> MakeFeatures -> Classifier

```
stopwords = { word.strip() for word in
open('/path/to/english.stopwords').readlines()
             if word.strip()}
print list(stopwords)[:20]

['all', 'just', 'being', 'over', 'both', 'through',
'yourselves', 'its', 'before', 'herself', 'had', 'should',
'to', 'only', 'under', 'ours', 'has', 'do', 'them', 'his']

#Remove stop words
terms = [term for term in terms if term[0] not in stopwords]

#Remove words that are not noun/verb/adjective/adverb
terms = [ term for term in terms if  term[1]]
print terms

[('film', u'n'), ('extraordinarily', u'r'), ('horrendous',
u'a'), ('going', u'v'), ('waste', u'n'), ('words', u'n')]
```

Infering the root word

GetTerms -> Filtering -> **BaseWord** -> MakeFeatures -> Classifier

Lemmatization reduces inflectional forms and sometimes derivationally related forms of a word to a common base form.

done with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma .

am, are, is => be

car, cars, car's, cars' => car

Infering the root word

GetTerms -> Filtering -> **BaseWord** -> MakeFeatures -> Classifier

```
from nltk.stem.wordnet import
WordNetLemmatizer
lmtzr = WordNetLemmatizer()

lemmas = [ lmtzr.lemmatize(*params) for
params in terms]
print lemmas
```

```
['film', 'extraordinarily', 'horrendous',
u'go', 'waste', u'word']
```

Final bag_of_words function

GetTerms -> Filtering -> **BaseWord** -> MakeFeatures -> Classifier`

```
def bag_of_words(review):
    tags = tagger(review)
    terms = [(rec[0].lower(), get_wordnet_pos(rec[1])) for
rec in tags]
    terms = [term for term in terms if term[1]]
    terms = [ term for term in terms if term[0] not in
stopwords]
    terms = [lmtzr.lemmatize(*params) for params in terms]

    return terms

print bag_of_words(sample_review)

['film', 'extraordinarily', 'horrendous', u'go', 'waste',
u'word']
```

Remember?

GetTerms -> Filtering -> BaseWord -> **MakeFeatures** -> Classifier

```
from glob import glob

neg = [open(f).read() for f in
glob('review_polarity/txt_sentoken/neg/*.txt')]

pos = [open(f).read() for f in
glob('review_polarity/txt_sentoken/pos/*.txt')]
```

For each review

    Feature set => bag-of-words

    Tag = 'n'/'p' for every feature set

Deriving features

GetTerms -> Filtering -> BaseWord -> **MakeFeatures** -> Classifier

```
#We are going to extract features for all reviews.
def make_feature(word_list):
    return dict([(word, True) for word in word_list])


features = [(make_feature(bag_of_words(review)), 'p') for review in pos] +\
           [(make_feature(bag_of_words(review)), 'n') for review in neg]
```

Classification

GetTerms -> Filtering -> BaseWord -> MakeFeatures -> **Classifier**

```
import random

random.shuffle(features)

test_features = features[:200]

train_features = features[200:]

classifier = nltk.NaiveBayesClassifier.train(train_features)

classifier.classify(test_features[100][0]), test_features[100][1]

('p', 'p')

print(nltk.classify.accuracy(classifier, test_features))

0.69
```