771219/Models/Te...

Markdown styles

READY

READY

Sentiment Analysis example #3

Positivity/Negativity of IMDB reviews

Useful links:

- http://ai.stanford.edu/~amaas/data/sentiment/ (http://ai.stanford.edu/~amaas/data/sentiment/)
- http://www.aclweb.org/anthology/P11-1015 (http://www.aclweb.org/anthology/P11-1015)

Data:

- Example data from IMDB

Tools:

- elaborate data prepration including spellchecking
- sk-learn: bow, stopwords, bigrams, liguistic features

READY

0. Set-Up

Keytab

READY

```
Ticket cache: FILE:/tmp/krb5cc_1443748855
Default principal: 776859@BNZNAG.NZ.THENATIONAL.COM
Valid starting       Expires              Service principal
11/02/2018 17:14:24  11/03/2018 03:14:24  krbtgt/BNZNAG.NZ.THENATIONAL.COM@BNZNAG.NZ.THENATIONAL.CO
M
        renew until 11/09/2018 17:14:24
```

Pypark

READY

```
['seaborn-darkgrid', 'Solarize_Light2', 'seaborn-notebook', 'classic', 'seaborn-ticks', 'grayscale', 'bmh', 'seaborn-talk', 'dark_background', 'ggplot', 'fivethirtyeight', '_classic_test', 'seaborn-colorblind', 'seaborn-deep', 'seaborn-whitegrid', 'seaborn', 'seaborn-poster', 'seaborn-bright', 'seaborn-muted', 'seaborn-paper', 'seaborn-white', 'fast', 'seaborn-pastel', 'seaborn-dark', 'tableau-colorblind10', 'seaborn-dark-palette']
/usr/lib64/python2.7/site-packages/pandas/_libs/__init__.py:4: RuntimeWarning: numpy.dtype size
 changed, may indicate binary incompatibility. Expected 96, got 88
```

```
  from .tslib import iNaT, NaT, Timestamp, Timedelta, OutOfBoundsDatetime
/usr/lib64/python2.7/site-packages/pandas/__init__.py:26: RuntimeWarning: numpy.dtype size chang
ed, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs import (hashtable as _hashtable,
/usr/lib64/python2.7/site-packages/pandas/core/dtypes/common.py:6: RuntimeWarning: numpy.dtype s
ize changed, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs import algos, lib
/usr/lib64/python2.7/site-packages/pandas/core/util/hashing.py:7: RuntimeWarning: numpy.dtype si
ze changed, may indicate binary incompatibility. Expected 96, got 88
  from pandas._libs import hashing, tslib
```

Python 3

READY

```
['Solarize_Light2', '_classic_test', 'bmh', 'classic', 'dark_background', 'fast', 'fivethirtyeigh
t', 'ggplot', 'grayscale', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark-palette', 'seaborn
-dark', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'seaborn-paper',
 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 'seaborn-white', 'seaborn-whi
tegrid', 'seaborn', 'tableau-colorblind10']
```

READY

1. Import data

- Some IMDB reviews
- note that you can have multiple reviews for the same film - hence you cannot use the film id as index

READY

```
%pyspark
import os
from os import pardir, path
import io

rootdir = '/data/disk1/776859/Shared/ExtData/aclImdb/'
dirs = [rootdir + 'train/pos', rootdir + 'train/neg', rootdir + 'test/pos', rootdir + 'test/neg']
dirs

mylist  = []
counter = 0

for adir in dirs:
    for afile in os.listdir(adir):
        if afile.endswith(".txt"):
            # Get review id, score from name
            parts        = afile[:-4].split('_')
            review_id    = int(parts[0])
            review_score = int(parts[1])

            # Get review text
            #print(afile)
            f            = io.open(path.join(adir, afile), encoding='utf-8')
            review_text  = f.read()

            # store in Intermediate list
            mylist.append([counter, review_score, review_text, review_id])
            counter = counter + 1
        else:
            continue
```

READY

```
%pyspark
```

```
reviews         = pd.DataFrame(mylist)
reviews.columns = ["id", "rawscore", "text", "film"]

reviews.shape
```

```
(50000, 4)
```

```
%pyspark
cn.show(reviews, type='ta', mr=10, textlen=400)
```

READY

EXTRACT ONLY: The number of rows is limited to 10, original shape is (50000, 4)

id	rawscore	text
2	9	Due to reading bad reviews and being told by friends that they couldn't believe how />People have skulls thicker than Ned's helmet if they go to see a movie like this ar
3	8	An ok movie about downs syndrome. A mother has twins one is very sick the other agree too) so they take it to court & the family gets broken up & the grandparents l
4	9	People forget that there have been several King Kong ripoffs- Congo, King Kong Vs based on Bigfoot (not like King Kong)and archeologists have been fascinated it, at
5	7	I never saw this when I was a kid, so this was seen with fresh eyes. I had never hea great. My daughter was entranced. I loved watching Walken in this role thinking abo
6	7	I saw the last five or ten minutes of this film back in 1998 or 1999 one night when I found out some Internet stores sold it. Then, being a world-class procrastinator, I st

```
%pyspark
cn.show(reviews, type='si', mr=10)
```

READY

	id	rawscore	text	film
0	0	7	Why this is called "Mistresses" is a puzzle, b...	1845
1	1	8	It is not known whether Marilyn Monroe ever me...	12066
2	2	9	Due to reading bad reviews and being told by f...	256
3	3	8	An ok movie about downs syndrome. A mother has...	708
4	4	9	People forget that there have been several Kin...	10230
...	...	...	...	...
49995	49995	2	The evil bikie gang in this movie were called ...	4334
49996	49996	2	In the XXII century an architect by the name o...	10646
49997	49997	3	Role-reversal remake of 1942's "The Major and ...	6482
49998	49998	3	The Forgotten (AKA: Don't Look In The Basement...	105
49999	49999	1	"More" is yet another addition into the countl...	3308

50000 rows × 4 columns

```
%pyspark
cn.show(reviews[reviews.film == 10487], type='si', mr=10)
```

READY

	id	rawscore	text	film
6589	6589	7	Saving Grace is a feel good movie with it's he...	10487
18876	18876	2	OMG! The only reason I'm giving this movie a 2...	10487
31427	31427	7	PRICE OF HONOR aka PRICE OF POWER is an ambiti...	10487

	id	rawscore	text	film
49667	49667	3	What? - that was it? The town sheriff (John Ag...	10487

READY

2. Prepare data

READY

2.0 Check data - clean for nulls

READY

Overall info, with non-nulls

```
%pyspark
reviews.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 4 columns):
id          50000 non-null int64
rawscore    50000 non-null int64
text        50000 non-null object
film        50000 non-null int64
dtypes: int64(3), object(1)
memory usage: 1.5+ MB
```

READY

Count nulls

```
%pyspark

reviews.isnull().sum()
```

```
id          0
rawscore    0
text        0
film        0
dtype: int64
```

READY

No nulls or NaNs

READY

No 5 or 6 score

- OK. Should actually not be an issue

READY

2.1 Take a sample to speed up (especially first time around)

READY

```
%pyspark

reviews_clean = reviews.sample(n=1000, replace=False, random_state=0)
reviews_clean.shape
```

```
(1000, 4)
```

READY

2.2 Prepare data

READY

Actions:

- Remove odd characters and numbers
- Fix spelling

Extra bits to replace/remove

READY

```
%pyspark

extra_repl = {
    r"\bxxxxxx\b":    "",    # card numbers, abuses
    r"\bxxxxx\b":     "",
    r"\bxxxx\b":      "",
    r"\bxxx\b":       "",
    r"\bxx\b":        "",

    r"\bone\b":       "",    # score leakage
    r"\btwo\b":       "",
    r"\bthree\b":     "",
    r"\bfour\b":      "",
    r"\bfive\b":      "",
    r"\bsixe\b":      "",
    r"\bseven\b":     "",
    r"\beight\b":     "",
    r"\bnine\b":      "",
    r"\bten\b":       "",
}
```

Pre-processing and spelling fix

READY

```
%pyspark
from symspell.symspellpy   import SymSpell, Verbosity

dictionary_path     = path.join("/data/disk1/776859/Shared/Python/external/symspell/", "frequency_

##
## create object

initial_capacity                = 83000
# maximum edit distance per dictionary precalculation
max_edit_distance_dictionary    = 2
prefix_length                   = 7
```

```
sym_spell = SymSpell(initial_capacity, max_edit_distance_dictionary, prefix_length)

##
## load dictionary

term_index  = 0  # column of the term in the dictionary text file
count_index = 1  # column of the term frequency in the dictionary text file
if not sym_spell.load_dictionary(dictionary_path, term_index, count_index):
    print("Dictionary file not found")
```

Test 1

READY

```
%pyspark
max_edit_distance_lookup = 2

#input_term  = train_data_x.iloc[6]
input_term = u"Cann yu put tha platic taable inn the cornr"
prep        = cta.preprocessor(input_term, extra_repl2=extra_repl2)
suggestions = sym_spell.lookup_compound(prep, max_edit_distance_lookup)

print("Raw: {}\n".format(input_term))
print("---")
print("Step 1: {}\n".format(prep))
print("---")
print("Step 2: {}, {}, {}\n".format(suggestions[0].term, suggestions[0].count, suggestions[0].dist

# display suggestion term, edit distance, and term frequency
#for suggestion in suggestions:
```

```
Raw: Cann yu put tha platic taable inn the cornr
---
Step 1: cann yu put tha platic taable inn the cornr
---
Step 2: can you put the plastic table inn the corner, 35421202, 7
```

Test 2

READY

```
%pyspark
reload(cta)
max_edit_distance_lookup = 2

for n in range(0, 5):
    input_term  = reviews_clean['text'].iloc[n]
    prep        = cta.preprocessor(input_term, extra_repl=extra_repl)
    suggestions = sym_spell.lookup_compound(prep, max_edit_distance_lookup)

    print("---------------- {}".format(n))
    print("Raw: {}\n".format(input_term))
    print("---")
    print("Step 1: {}\n".format(prep))
    print("---")
    print("Step 2: {}, {}, {}\n".format(suggestions[0].term, suggestions[0].count, suggestions[0].
```

```
th the exception of jay underwood character in which i do believe it was his best performance co
nsidering roles such as the not quite human movies in which the story is told in a way that he h
as no character he cant act and people have made good decisions to not go see his movies this is

 why he is most likely not going to be any huge roles unless he sparks his career in a most rare
 but interesting way
---
Step 2: this is the worst movie i have ever seen in my entire life period this movie goes beyond
 ridiculous it is like the director wants to get his ass sued by the actors for wrongfully misre
presenting their roles as the fantastic i believe the movie should have been released in comic b
ook stores in order for the only how you should say it desperate geeks who cant get enough chees
iness and want to see more and more crap movies in conclusion to my paradox statements and thesi
```

```
s i do believe this movie has had great disadvantages to the futures of the cast contributors wi
th the exception of jay underwood character in which i do believe it was his best performance co
nsidering roles such as the not quite human movies in which the story is told in a way that he h
as no character he cant act and people have made good decisions to not go see his movies this is
 why he is most likely not going to be any huge roles unless he sparks his career in a most rare
 but interesting way, 18380, 0
```

Test for timing

READY

```
%pyspark
test = reviews_clean[:10]

test['clean'] = test['text'].apply(lambda t: sym_spell.lookup_compound(cta.preprocessor(t, extra_r
```

```
<string>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#ind
exing-view-versus-copy
```

READY

The spelling correction takes time

- 5 sec per long text -> 700 an hour -> 7 hours for 5,000
- you may bypass it the 1st time around and do only the preprocessing

READY

```
%pyspark

max_edit_distance_lookup = 2

# preprocessing + spelling
#reviews_clean['clean'] = reviews_clean['text'].apply(lambda t: sym_spell.lookup_compound(cta.prep

# Preprocessing only
reviews_clean['clean'] = reviews_clean['text'].apply(lambda t: cta.preprocessor(t, extra_repl=extr
```

READY

2.3 Deal with class imbalances

READY

Logistic regression should take a string label, to create independent regressions for each label

READY

```
%pyspark
reviews_clean['numscore'] = reviews_clean['rawscore'].apply(lambda s: 0 if (s < 4) else 1 if (s <
text_score                = ['1bad', '2neutral', '3good']
reviews_clean['score']    = reviews_clean['numscore'].apply(lambda n: text_score[n])
```

```
%pyspark
cn.show(reviews_clean, type='si', mr=10)
```

READY

	id	rawscore	text	film	clean	numscore	score
11841	11841	10	Smallville episode Justice is the best episode...	1558	smallville episode justice is the best episode...	2	3good
19602	19602	1	I can't believe they got the actors and actres...	825	i can not believe they got the actors and actr...	0	1bad
45519	45519	1	Bad dialog, slow story, scenes that drag on an...	4169	bad dialog slow story scenes that drag on and ...	0	1bad
25747	25747	7	Countless TV displays and the memorable appear...	9088	countless tv displays and the memorable appear...	1	2neutral
42642	42642	1	this is the worst movie i have ever seen in my...	9870	this is the worst movie i have ever seen in my...	0	1bad
...	...	...	...	...	...	...	...
46754	46754	3	Am I the only one who thought the point of thi...	5895	am i the only who thought the point of this fi...	0	1bad
11830	11830	9	'Loulou' delights in the same way an expensive...	11891	loulou delights in the same way an expensive ...	2	3good
27232	27232	9	A classic late 50's film. The superannuated he...	3321	a classic late film the superannuated headline...	2	3good
24445	24445	2	Yes there are worse movies out there. Most of ...	4447	yes there are worse movies out there most of t...	0	1bad
27336	27336	7	One of eastwood's best movies after he had sep...	7855	of eastwood best movies after he had separate...	1	2neutral

1000 rows × 7 columns

READY

Check the score populations:

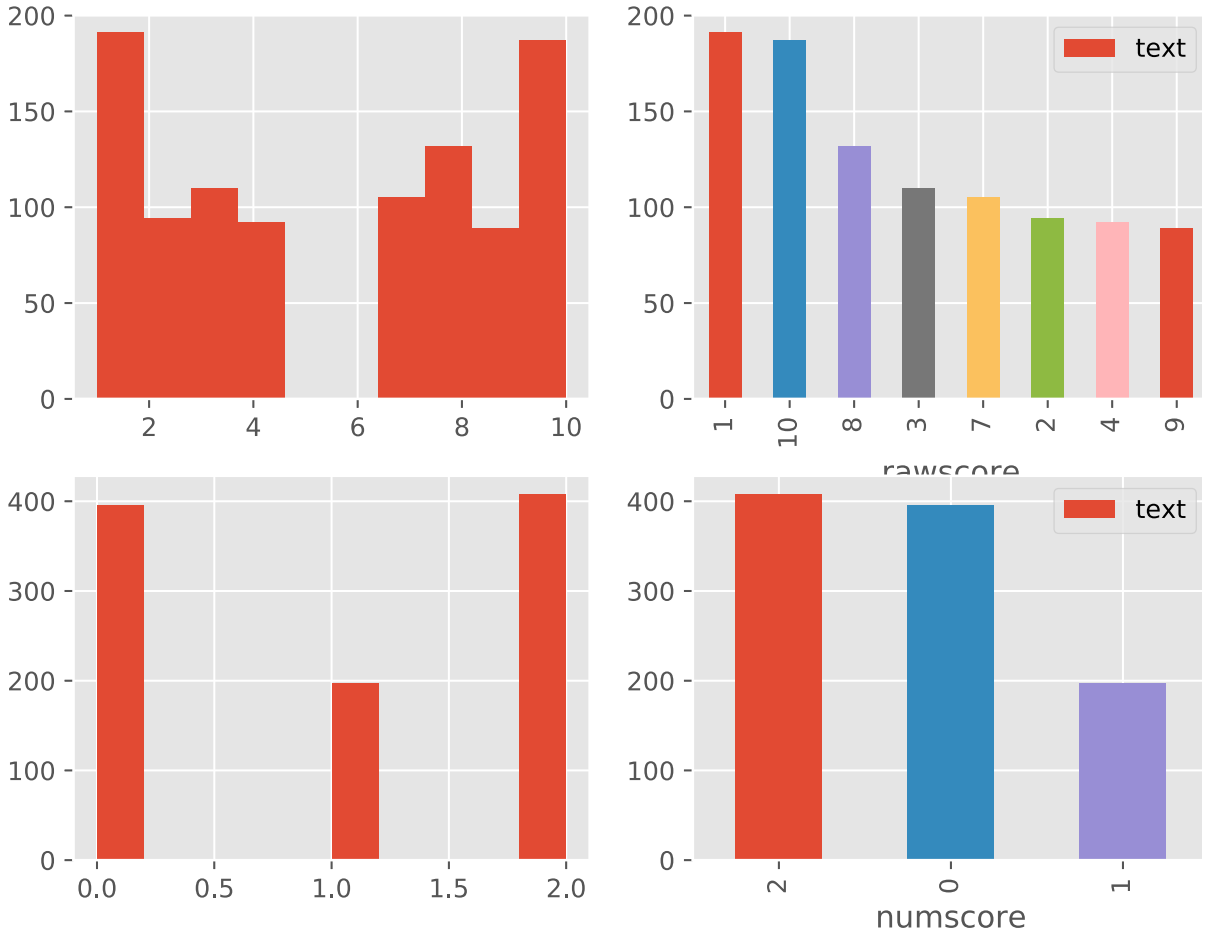
Note how unbalanced the scores are

```
%pyspark

# Order by score
fig1    = plt.figure(figsize=(8, 6))
ax1     = fig1.add_subplot(2, 2, 1)
ax2     = fig1.add_subplot(2, 2, 2)
ax3     = fig1.add_subplot(2, 2, 3)
ax4     = fig1.add_subplot(2, 2, 4)

##
## Old score

# Ordered by score
reviews_clean['rawscore'].hist(ax=ax1)

# Ordered by population
score = reviews_clean['rawscore'].unique()

ordered = reviews_clean.groupby('rawscore').count()['text'].sort_values(ascending=False).reset_ind
#cn.show(ordered, type='simple')
ordered.plot.bar(ax=ax2, x='rawscore', y='text')

# ##
# ##Aggregate score

# # Ordered by score
```

READY

```
reviews_clean['numscore'].hist(ax=ax3)

# Ordered by population
score = reviews_clean['numscore'].unique()

ordered = reviews_clean.groupby('numscore').count()['text'].sort_values(ascending=False).reset_ind
#cn.show(ordered, type='simple')
ordered.plot.bar(ax=ax4, x='numscore', y='text')

##
## Graph it
cn.plt_fig(fig1, align='l')

##
## Checking population
```

```
/usr/lib64/python2.7/site-packages/statsmodels/tsa/statespace/tools.py:59: RuntimeWarning: nump
y.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from . import (_representation, _kalman_filter, _kalman_smoother,
/usr/lib64/python2.7/site-packages/statsmodels/tsa/kalmanf/kalmanfilter.py:33: RuntimeWarning: n
umpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
  from . import kalman_loglike
/usr/lib64/python2.7/site-packages/scipy/signal/_max_len_seq.py:8: RuntimeWarning: numpy.dtype s
ize changed, may indicate binary incompatibility. Expected 96, got 88
  from ._max_len_seq_inner import _max_len_seq_inner
/usr/lib64/python2.7/site-packages/scipy/signal/_upfirdn.py:36: RuntimeWarning: numpy.dtype size
 changed, may indicate binary incompatibility. Expected 96, got 88
  from ._upfirdn_apply import _output_len, _apply
/usr/lib64/python2.7/site-packages/scipy/ndimage/measurements.py:36: RuntimeWarning: numpy.dtype
 size changed, may indicate binary incompatibility. Expected 96, got 88
  from . import _ni_label
/usr/lib64/python2.7/site-packages/scipy/signal/spectral.py:10: RuntimeWarning: numpy.dtype size
 changed, may indicate binary incompatibility. Expected 96, got 88
  from .  spectral import  lombscargle
```



READY

No problem - quite evenly distributed

READY

2.4 Create training and testing sets

READY

Note that the score is quite often in the text ('I score this 7')

- OK if numeric. Will be removed anyway as we strip non-characters out
- Not ok if text: 'This film gets a one/two/three'

Create feature (x), get target (y)

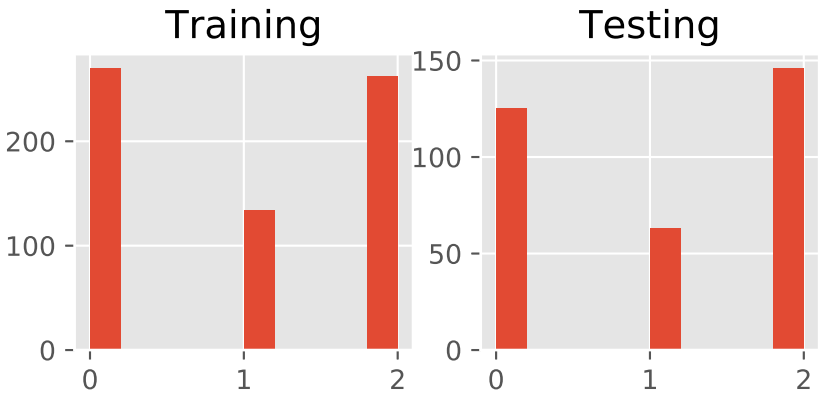
READY

```
%pyspark
```

```
from sklearn.model_selection import train_test_split

# Split
train_data_x, test_data_x, train_y, test_y = train_test_split(reviews_clean['clean'], reviews_clea

train_data_y = train_y['score']
test_data_y  = test_y['score']
```

READY

```
%pyspark

fig1    = plt.figure(figsize=(5, 2))
ax1     = fig1.add_subplot(1, 2, 1)
ax2     = fig1.add_subplot(1, 2, 2)


train_y['numscore'].hist(ax=ax1)
test_y['numscore'].hist(ax=ax2)

ax1.set_title("Training")
ax2.set_title("Testing")

cn.show(fig1)
```



READY

```
%pyspark
cn.show(train_data_x[:4], type='st')
```

	clean
1083	george lopez never caught my interest in his stand up comedy and he still does not but this show is a work of art it is not ever show where the jokes keep you laughing every time you remember it and jokes that re memorable at that this show just has an upbeat look to it and the characters range from an old short drunk to an dyslexic teenager i do not know who writes this show but that person does a great job if they had just continued the show i am sure that it would get a positive response from the critics of this great country if you are looking for a good traditional comedy then george lopez is the show for you the bad thing is the title george lopez really imagine the fresh prince of bel air being will smith c mon man but otherwise this show is genius

	clean
39913	help i have ended up in cinema hell what a completely stupid film this is really nothing is good about it let us spit it out the story is incredibly far fetched an anti eu terrorist group is chasing a bunch of guys who drive around western europe carrying a delivery of see through bags full of xtc pills and the worst thing is they are serious about it the level of acting should put great shame on all faces involved some money eyed guy decided to let every talk english so that the international market would catch on ugliest advertising ever the french and dutch native tongues talking smart make all but sense and the result is laughable the soundtrack is totally misplaced and ill chosen the camera edit and effects work is supposed to be of some post noir road movie kind of style but is hardly worth some thing and not meant to accompany this story read anti story hidde maas the hero of wildschut never fails to convince a true actor usually i would give an extra point just for the sake of him being around but no sorry not this time i would just not forgive my self
12359	i loved it and was so shattered that there not making another season i wish they would it was the best show ever there is probably not any chance of them deciding to not cancel the show is there ha ha i wish there was though i would be so so excited i really would i miss it and was especially shattered not to know what happens to jason i think they should make another it i also think its silly that you have to writr lines to post a comment it makes your comment drag on and no will read it i really want to know what would have happened between jason and nicole maybe they could make a spin off
29561	this film was incredible looked high budget but felt heartfelt and original like an indie the most amazing part of this film were the astonishing performances by david beazely mark hildreth and paul anthony who plays the main role he carried this film with ease humor and charisma balanced with a huge depth the cinematography was really beautiful even though some of the subject was quite ugly it was not very realistic in that way but it did not have to be to make a larger point it was really great seeing alan cumming in this too the script was tight and propelled very nicely with some of the best acting i have seen in a while go see this

Check proportions

READY

```
%pyspark

np.size(test_data_x)/(np.size(test_data_x) + np.size(train_data_x) + 0.0), np.size(test_data_x) +
```

```
(0.334, 1000)
```

READY

3 Do Initial classification

We use an apriori model without n-fold. Proper model discovery done later.
This is really about testing the features and the pipeline.

Features:

- TFIDF, Note that we allow bow and bigrams
- Text len (1)
- SentiWorNet (3)

- Part Of SPeech (7)

Features selection

- SelectKBest
- see https://github.com/scikit-learn/scikit-learn/blob/master/doc/modules/feature_selection.rst#id93 (https://github.com/scikit-learn/scikit-learn/blob/master/doc/modules/feature_selection.rst#id93)

Choice of model:

- Good pointers:
  - http://streamhacker.com/2012/11/22/text-classification-sentiment-analysis-nltk-scikitlearn/ (http://streamhacker.com/2012/11/22/text-classification-sentiment-analysis-nltk-scikitlearn/)
  - https://streamhacker.com/2010/06/16/text-classification-sentiment-analysis-eliminate-low-information-features/ (https://streamhacker.com/2010/06/16/text-classification-sentiment-analysis-eliminate-low-information-features/)
- I setlled on SVM with SGD after a bit of experiment
- The code can easily be modified to try other models (BernoulliNB or MultinomialNB)

READY

3.1 Create Test Pipeline

READY

Explore these:

- Text length as feature
- Stemmer (NLTK)
- Lemmatization (TextBlob)
- Part of Speech (NLTK)
- Sentiment as feature (For sentiment analytics, not topics classification)
  - SentiWordNet
  - TextBlob .sentiment

READY

For n-fold, you really want to be looping only on the model choice, not on redoing the features

Hence the n-fold pipeline should work from the features after applying the Stemmer, Lematization or preprocessor

Vectorizer / Transformer options

READY

Define stemmer

READY

```
%pyspark
import nltk.stem
from sklearn.feature_extraction.text    import TfidfTransformer, TfidfVectorizer, CountVectorizer

english_stemmer = nltk.stem.SnowballStemmer('english')

class StemmedTfidfVectorizer(TfidfVectorizer):
    def build_analyzer(self):
        analyzer = super(TfidfVectorizer, self).build_analyzer()

        return lambda doc: (english_stemmer.stem(w) for w in analyzer(doc))
```

Define Lemmatization

READY

```
%pyspark
from nltk import word_tokenize
from nltk.stem import WordNetLemmatizer

class LemmaTokenizer(object):
    def __init__(self):
        self.wnl = WordNetLemmatizer()

    def __call__(self, doc):
        return [self.wnl.lemmatize(t) for t in word_tokenize(doc)]
```

READY

text size adds 1 features, ling_stats adds 10 features

Get features and labels - Features: TFIDF (Standard, Stemmer or Lemmization) + len, POS, SentiWordNet

READY

```
%pyspark

from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.multiclass import OneVsRestClassifier
from sklearn.preprocessing import FunctionTransformer

########################################
## TFIDF features
########################################

## Can be customized for Lemmatization, Stemmer or plain

# Lemmatization
count_vect      = CountVectorizer(tokenizer=LemmaTokenizer(), **vect_options)
#count_vect      = CountVectorizer(**vect_options)

# Transformer: Switch for Stemmer
tfidf_transformer   = TfidfTransformer(**trans_options)                                    # 2n
#tfidf_transformer  = TfidfVectorizer(**cio.merge2Dicts(trans_options, vect_options))      # Pl
#tfidf_transformer  = StemmedTfidfVectorizer(**cio.merge2Dicts(trans_options, vect_options))  # St

# Full tfidf pipeline
tfidf_stats         = Pipeline([
                            ('vect',    count_vect),
                            ('tfidf',   tfidf_transformer),
                            ])

########################################
## Text Len feature
########################################
def get_text_length(x):
```

```
    return np.array([len(t) for t in x]).reshape(-1, 1)

len_stats            = Pipeline([
                              ('count',   FunctionTransformer(get_text_length, validate=False))
                              ])

########################################
## Part-of-speech + Sentiment features
########################################
ling_stats          = Pipeline([
                              ('ta',      cta.SentimentVectorizer())
                              ])

##
## Al in one pipeline
##

# To test len + sentiment features only
sentpipe = FeatureUnion([
        #('text',   tfidf_stats),
        ('length', len_stats),
        ('ling',   ling_stats),
        ])

# Full cancidate pipeline
featpipe = FeatureUnion([
        ('text',   tfidf_stats),
        ('length', len_stats),
        ('ling',   ling_stats),
        ])
```

NB Feats

READY

```
%pyspark

nbfeats  = 4000   # Experiment with this -> 4,000 for a large corpus seems reasonnable, but check
```

READY

We force a low number of features (should be around 4,000 at least)

Create pipeline with features filtering and model

READY

```
%pyspark

from sklearn.pipeline                   import Pipeline
from sklearn.feature_selection          import SelectKBest, chi2
from sklearn.naive_bayes                import MultinomialNB, BernoulliNB
from sklearn.linear_model               import SGDClassifier, LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.svm                        import NuSVC

# TDF-IDF Features
# strip_accents='unicode' will slow things down a bit - set it to None instead if you want speed
# See http://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVector
#tfidf          = StemmedTfidfVectorizer(sublinear_tf=True, min_df=4, norm='l2', ngram_range=(1, 2
#tfidf          = TfidfVectorizer(sublinear_tf=True, min_df=4, norm='l2', ngram_range=(1, 2nnor),
#tfidf          = StemmedTfidfVectorizer(sublinear_tf=True, ngram_range=(1, 2), strip_accents='uni
#tfidf          = TfidfVectorizer(sublinear_tf=True, ngram_range=(1, 2), strip_accents=None, min_d
#tfidf          = TfidfTransformer(use_idf=True)

# Choice of model
model_choice    = LogisticRegression(random_state=0)
#model_choice    = LinearSVC()
#model_choice   = SGDClassifier(loss='hinge', penalty='l2') # Relying on good sci-kit defaults!
```

```
#model_choice   = BernoulliNB(fit_prior=False)       # Unigrams mostly
#model_choice   = MultinomialNB(fit_prior=False)     # Bigrams mostly
#model_choice   = NuSVC(gamma='scale')

# Filter: main tfidf features (based on correlation)
model_filter        = SelectKBest(chi2, k=nbfeats)

# Full Pipeline
full_clf = Pipeline([
            ('allf_1',   featpipe),       # Feature preparation
            ('chi2_1',   model_filter),
            ('clf_1',    model_choice),
            ])

# Test Pipeline
sent_clf = Pipeline([
            ('allf_2',   sentpipe),       # Feature preparation
            #('chi2_2',   model_filter),  # No need for filtering
            ('clf_2',    model_choice),
            ])
```

READY

3.2 Full Features

READY

3.2.1 Training

READY

Takes 4 min with linguistic features

```
%pyspark
tr_subset = None
ts_subset = None

full_clf = full_clf.fit(train_data_x[:tr_subset], train_data_y[:tr_subset])
```

READY

Hit rate

```
%pyspark

train_predicted = pd.Series(full_clf.predict(train_data_x[:ts_subset]), index=train_data_x[:tr_sub

np.mean(train_predicted == train_data_y)
```

```
0.8033033033033034
```

READY

3.2.2 Testing

READY

Full features

```
%pyspark

test_predicted = pd.Series(full_clf.predict(test_data_x[:ts_subset]), index=test_data_x[:ts_subset

np.mean(test_predicted == test_data_y[:ts_subset])
```

```
0.6826347305389222
```

READY

> **The hit rate drops quite a bit from Training to Testing**
>
> - Nothing optimized yet

READY

3.3 Len + Ling features

READY

3.3.1 Training

READY

Takes 4 min with linguistic features

```
%pyspark
tr_subset = None
ts_subset = None

sent_clf = sent_clf.fit(train_data_x[:tr_subset], train_data_y[:tr_subset])
```

READY

Hit rate

```
%pyspark

train_predicted_sent = pd.Series(sent_clf.predict(train_data_x[:ts_subset]), index=train_data_x[:t

np.mean(train_predicted_sent == train_data_y)
```

```
0.41291291291291293
```

READY

3.3.2 Testing

READY

Len + Ling features

```
%pyspark

test_predicted_sent = pd.Series(sent_clf.predict(test_data_x[:ts_subset]), index=test_data_x[:ts_s

np.mean(test_predicted_sent == test_data_y[:ts_subset])
```

```
0.39221556886227543
```

READY

Very poor hit rates - check the features

READY

3.4 Detailed Performance metrics

See http://scikit-learn.org/stable/modules/model_evaluation.html (http://scikit-learn.org/stable/modules/model_evaluation.html)

READY

If we don't rebalance the classes, then f1 score is really poor for the small classes

- These all are effectively part of a larget art category
- Some classes such as journalism have very few instances
- **journalism / crafts (slippery categories) are the worse**

Train - Full

READY

```
%pyspark
from sklearn import metrics

res = metrics.classification_report(train_data_y[:tr_subset], train_predicted)
cn.show(res)
```

```
             precision    recall  f1-score   support
       1bad       0.81      0.97      0.89       270
   2neutral       1.00      0.12      0.21       134
      3good       0.79      0.98      0.87       262
avg / total       0.84      0.80      0.74       666
```

Train: Len + Ling

READY

```
%pyspark
from sklearn import metrics

res = metrics.classification_report(train_data_y[:tr_subset], train_predicted_sent)
cn.show(res)
```

```
             precision    recall  f1-score   support
       1bad       0.43      0.81      0.56       270
   2neutral       0.30      0.02      0.04       134
      3good       0.38      0.21      0.27       262
avg / total       0.38      0.41      0.34       666
```

Test - Full

READY

```
%pyspark
```

```
res = metrics.classification_report(test_data_y[:ts_subset], test_predicted)
print(res)
```

```
             precision    recall  f1-score   support

      1bad       0.65      0.86      0.74       125
   2neutral       0.50      0.02      0.03        63
      3good       0.72      0.82      0.77       146
avg / total       0.65      0.68      0.62       334
```

Test: Len + Ling

READY

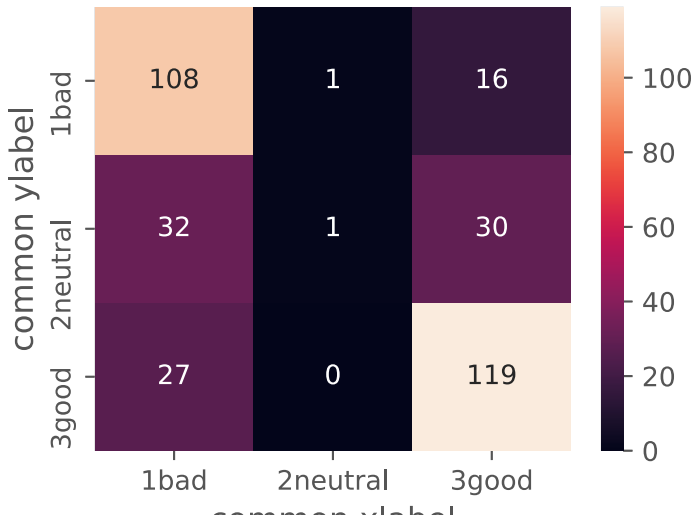
```
%pyspark


res = metrics.classification_report(test_data_y[:ts_subset], test_predicted_sent)
print(res)
```

```
             precision    recall  f1-score   support

      1bad       0.39      0.76      0.52       125
   2neutral       0.00      0.00      0.00        63
      3good       0.41      0.25      0.31       146
avg / total       0.33      0.39      0.33       334
```

Test - Full

READY

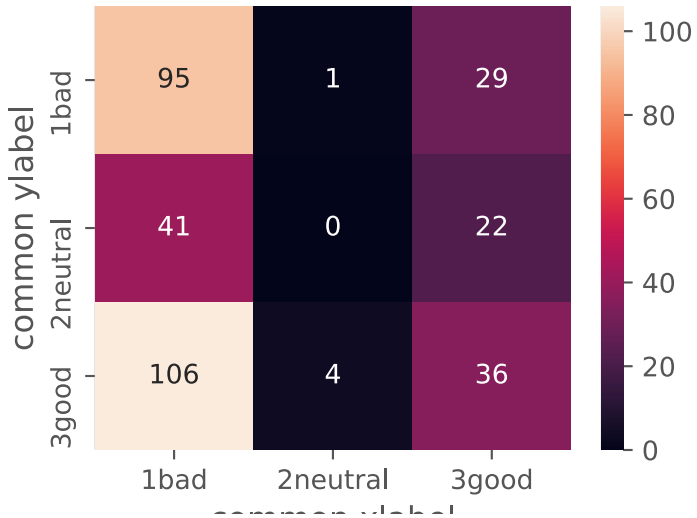
```
%pyspark
import seaborn as sns
from sklearn.metrics import confusion_matrix

names    = list(np.unique(test_predicted))
conf_mat = confusion_matrix(test_data_y[:ts_subset], test_predicted)

fig, ax = plt.subplots(figsize=(4, 3))
sns.heatmap(conf_mat,
            annot=True,
            fmt='d',
            xticklabels=names,
            yticklabels=names)

# Not working
ax.set(xlabel='common xlabel', ylabel='common ylabel')

cn.show(fig)
```



Test - Len + Ling

READY

```
%pyspark
import seaborn as sns
from sklearn.metrics import confusion_matrix

names    = list(np.unique(test_predicted_sent))
conf_mat = confusion_matrix(test_data_y[:ts_subset], test_predicted_sent)

fig, ax = plt.subplots(figsize=(4, 3))
sns.heatmap(conf_mat,
            annot=True,
            fmt='d',
            xticklabels=names,
            yticklabels=names)

# Not working
ax.set(xlabel='common xlabel', ylabel='common ylabel')

cn.show(fig)
```



READY

Issues:

- The sentiment analysis gets rather confused
- No scoring for neutral
  - maybe model or unbalanced sets
  - or small size of training set

READY

4 Audit Features:

READY

4.1 Ling Features:

READY

4.1.1 Check features correlation:

- We are explicitly recreating the features - before Chi2 selection

READY

Takes a few minutes - otherwise done in the pipeline: see 4.3 for a way to access these from pipeline

```
%pyspark

# There must be a better way to do that, as this was effectiovely done in the pipeline
sentfeatures    = sentpipe.fit_transform(train_data_x[:tr_subset])

# Check dimensions
sentfeatures.shape
```

```
(666, 5)
```

READY

Retrieve feature names

```
%pyspark
a =    "len"
b =    sent_clf.named_steps['allf_2'].transformer_list[1][1].named_steps['ta'].get_feature_names()

# Get feature names
lingfeats = np.append(a, b)
lingfeats
```

```
array(['len', 'sent', 'sent_neut', 'sent_pos', 'sent_neg'], dtype='|S11')
```

READY

```
%pyspark
cn.show(pd.DataFrame(sentfeatures, columns=lingfeats[:2]), type='st', precision=4, mr=10)
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 360, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 1, in <module>
  File "/usr/lib64/python2.7/site-packages/pandas/core/frame.py", line 379, in __init__
    copy=copy)
  File "/usr/lib64/python2.7/site-packages/pandas/core/frame.py", line 536, in _init_ndarray
    return create_block_manager_from_blocks([values], [columns, index])
  File "/usr/lib64/python2.7/site-packages/pandas/core/internals.py", line 4866, in create_block
_manager_from_blocks
    construction_error(tot_items, blocks[0].shape[1:], axes, e)
  File "/usr/lib64/python2.7/site-packages/pandas/core/internals.py", line 4843, in construction
_error
    passed, implied))
ValueError: Shape of passed values is (5, 666), indices imply (2, 666)
```

READY

4.2 Full Features:

READY

4.2.1 Check features correlation:

- We are explicitly recreating the features - before Chi2 selection

Takes a few minutes - otherwise done in the pipeline: see 4.3 for a way to access these from pipeline

READY

```
%pyspark

# There must be a better way to do that, as this was effectiovely done in the pipeline

allfeatures    = featpipe.fit_transform(train_data_x[:tr_subset]).toarray()
tfidffeatures  = tfidf_stats.fit_transform(train_data_x[:tr_subset]).toarray()

# Check dimensions
(allfeatures.shape, tfidffeatures.shape)
```

```
((666, 9419), (666, 9414))
```

READY

```
%pyspark

count_vect.get_feature_names()[-100:]
```

```
[u'worst seen', u'worst thing', u'worst wa', u'worth', u'worth cent', u'worth commenting', u'worth
 living', u'worth look', u'worth price', u'worth renting', u'worth seeing', u'worth time', u'worth
 viewing', u'worth watch', u'worth watching', u'worthless', u'worthwhile', u'worthy', u'wound', u'w
ounded', u'wow', u'wrapped', u'wreck', u'wrist', u'write', u'writer', u'writer come', u'writer dire
ctor', u'writer like', u'writer probably', u'writes', u'writing', u'writing bad', u'writing directi
ng', u'writing film', u'written', u'written acted', u'written directed', u'wrong', u'wrong love',
 u'wrong make', u'wrong movie', u'wrong reason', u'wronged', u'wrongly', u'wrote', u'wtf', u'wwi',
 u'wwii', u'wyler', u'x', u'xian', u'y', u'ya', u'yash', u'yeah', u'yeah yeah', u'year', u'year abs
olutely', u'year ago', u'year doe', u'year later', u'year old', u'year say', u'year seeing', u'year
 wa', u'yearn', u'yearning', u'yeh', u'yell', u'yelling', u'yellow', u'yes', u'yes doe', u'yesterda
y', u'york', u'young', u'young anthony', u'young boy', u'young child', u'young frank', u'young gir
l', u'young lady', u'young man', u'young son', u'young woman', u'younger', u'younger brother', u'yo
ungest', u'youth', u'yummy', u'yup', u'z', u'zack', u'zero', u'zip', u'zombie', u'zombie film', u'z
ombie movie', u'zone']
```

Check features most correlated to labels

READY

```
%pyspark

from sklearn.feature_selection import chi2

nlevel    = 10
mylist1   = []
mylist2   = []

# Labels
labels          = np.unique(train_data_y[:ts_subset])

for label in sorted(labels):
  features_chi2 = chi2(tfidffeatures, train_data_y[:tr_subset] == label)
```

```
  indices       = np.argsort(features_chi2[0])

  feature_names = np.array(count_vect.get_feature_names())[indices]
  pvals         = features_chi2[1][indices]

  unigrams      = [v for v in feature_names if len(v.split(' ')) == 1]
  bigrams       = [v for v in feature_names if len(v.split(' ')) == 2]

  # Most correlated
  mylist1.append(
        pd.DataFrame(
            [[
            label,
            "{}".format('\n.'.join(unigrams[-nlevel:])),
            "{}".format('\n.'.join(bigrams[-nlevel:])),
            pvals[-nlevel:]
            ]])
        )

  # Least correlated
  mylist2.append(
        pd.DataFrame(
            [[
            label,
            "{}".format('\n.'.join(unigrams[:nlevel])),
            "{}".format('\n.'.join(bigrams[:nlevel])),
            pvals[:nlevel]
            ]])
        )

# One pass
topcors1         = pd.concat(mylist1, ignore_index=True)
topcors1.columns = ["label", "MC unigrams", "MC bigrams", "PVals"]

botcors1         = pd.concat(mylist2, ignore_index=True)
```

Top correlation -> should make sense

READY

```
%pyspark
cn.show(topcors1, type='st', fs=90)
```

	label	MC unigrams	MC bigrams	PVals
0	1bad	excellent .stupid .waste .terrible .avoid .suck .worse .bad .awful .worst	little girl .terrible acting .awful movie .script awful .wa worse .movie seen .b movie .bad movie .wa bad .worst movie	[0.16841197 0.16815004 0.16488232 0.16239962 0.14247848 0.13870823 0.09866842 0.05297738 0.03948299 0.02963886]
1	2neutral	disease .ex .underground .pegg .lewis .midnight .demille .creep .unger .bates	life soon .film doe .quite good .film waste .good supporting .character actor .sub genre .really character .end really .film new	[0.18833442 0.18211488 0.17726838 0.17231993 0.17084895 0.16994284 0.15438761 0.14693694 0.13826712 0.11703646]
2	3good	fantastic .excellent .season .amazing .worst .beauty .beautiful .awful .bad .loved	waste time .kung fu .wa great .enjoyable film .b movie .actor play .highly recommended .best episode .worst movie .film great	[0.19373814 0.19067286 0.17146499 0.14971733 0.145898 0.143787 0.12891684 0.08348358 0.06483642 0.06452859]

Bottom correlation -> should be neutral

READY

```
%pyspark
cn.show(botcors1, type='st', fs=90)
```

	label	MC unigrams	MC bigrams	PVals
0	1bad	pad .facet .absurdist .number .waltz .lay .introduce .accuracy .stretch .comedy	just dreadful .real good .tv film .better wa .movie wish .u feeling .rolling eye .silly plot .watching episode .try make	[0.9999969 0.99999552 0.99998143 0.99995445 0.99987883 0.99986165 0.99972519 0.99957518 0.99955172 0.99949966]
1	2neutral	walking .spent .away .dr .ala .ham .total .effect .finishing .got	make love .alien visitor .director michael .doe like .wa meant .just seen .wa quite .motion picture .unintentionally funny .come close	[0.9999821 0.99991386 0.99982072 0.99979112 0.99976365 0.99975336 0.99945151 0.99939326 0.9993239 0.99931615]
2	3good	worry .distortion .university .celluloid .aggressive .ben .danny .jennifer .charge .surprised	run steam .just showing .civil war .talented cast .life felt .patient asylum .look quite .hand table .production seen .wa canadian	[0.999995 0.99997593 0.99995914 0.99995088 0.99992466 0.9998367 0.99967196 0.99963542 0.99961156 0.99955011]

READY

Are the features clean and do thy make sense?

- some actor / director names -> ok if that is what it takes!
- note worse in bottom unigram and worst in top

READY

4.2.2 Audit of some of the bad classifications

READY

Simple check

```
%pyspark
iscorrect = [test_predicted == test_data_y[:ts_subset]][0]

badones   = [(test_data_x.ix[i], test_data_y.ix[i], test_predicted.ix[i]) for i in iscorrect.index

for badone in badones[:20]:
    print('[Actual: %s, Predicted: %s]\n%s\n' % (badone[2], badone[1], badone[0]))
    print("---\n")
```

```
[Actual: 1bad, Predicted: 2neutral]
this is a better than average entry in the saint series it holds your interest and as mysteries
 should keeps you guessing until the end and has several suspects to choose from many films from
 the golden age are not for all tastes especially younger viewers they date themselves by clothi
ng cars settings etc who nowadays asks for a highball or wears a suit and tie everywhere and the
 legal process was so much simpler must have been a dearth of lawyers back then frankly much of
 value is missing from those days in any case go with it and enjoy it is good in an old fashione
d sense
---
[Actual: 1bad, Predicted: 3good]
this movie is breath taking and mind blowing but i think maybe it can only be appreciated by die
 hard rpg funs it is like a game problem is the plot is too game like and just has too many twis
ts the twists are excessive jude law gives a very good performance i really like him in this mov
ie just as jerome in gattaca and gigglo joe in a i
```

```
---
[Actual: 1bad, Predicted: 2neutral]
the movie had so much potential but due to technology constraints and also a weak script killed
```

Check misclassifications

READY

```
%pyspark

max_row = 3
min_pop = 10

for predicted in range(np.size(labels)):
    for actual in range(np.size(labels)):
        try:
            if (predicted != actual) and (conf_mat.shape[0] > 0):
                if conf_mat[actual, predicted] >= min_pop:
                    print("'{}' predicted as '{}' : {} examples, showing first {}".format(labels[a
                    #print (conf_mat.shape, predicted, actual)

                    extract = test_data_x[:ts_subset][(test_data_y[:ts_subset] == labels[actual])
                    cn.show(extract, type='st', fs=80)

                    print('')
        except:
            print("Unexpected error: {}".format(sys.exc_info()[0]))
            pass
```

'2neutral' predicted as '1bad' : 41 examples, showing first 3

	clean
32118	this is a better than average entry in the saint series it holds your interest and as mysteries should keeps you guessing until the end and has several suspects to choose from many films from the golden age are not for all tastes especially younger viewers they date themselves by clothing cars settings etc who nowadays asks for a highball or wears a suit and tie everywhere and the legal process was so much simpler must have been a dearth of lawyers back then frankly much of value is missing from those days in any case go with it and enjoy it is good in an old fashioned sense
14563	the movie had so much potential but due to technology constraints and also a weak script killed the main plot of the film the book version of the film was much better and well conceived if it had been done right in the beginning with sources from the book it could have been a very cool classic
27792	the thing that can be said about running out of time is that it is an immensely clever film it is interesting to note that the film writers are french which may explain the movie out of the norm vibe as it does not really fit in with what is commonly called hong kong cinema the movie concerns a thief who plans revenge on some criminal types using the assistance of an equally clever cop but first he has to convince the cop to join his personal crusade and so begins a series of games where the thief manuevers the cop into his plan quite a clever movie out of go to www nixflix com for a more detailed review of this movie or full length reviews of other foreign films

'3good' predicted as '1bad' : 106 examples, showing first 3

	clean
26779	this movie is breath taking and mind blowing but i think maybe it can only be appreciated by die hard rpg funs it is like a game problem is the plot is too game like and just has too many twists the twists are excessive jude law gives a very good performance i really like him in this movie just as jerome in gattaca and gigglo joe in a i
34110	ok so some of the plot points might be a bit obvious but over all an interesting idea which works towards a tight ending the acting is solid particularly lachy hulme who plays of the central characters in this ensemble piece he certainly has a screen presence and he is interesting to watch it has a low budget feel which works for the sort of thriller horror genre jacks belongs to the film does not try to take itself to seriously which adds to the overall charm the character of phil dave serrafin has to be of the most annoying character seen on screen since rupert pupkin king of comedy worth adding to a weekend pile of dvds

	clean
10487	i just saw this last night it was broadcast on the canadian broadcasting corporation passionate eye series it has been screened recently sept at the toronto international film festival as well as many others it is a quite remarkable film the filmmakers literally stumbled into the story being there to make a documentary about chavez himself instead they found themselves squarely in the middle of events as the coup unfolded they had unprecedented access to events and people and for the most part let the story unfold as it happens they of course have their own ideological perspective which they make evident but they keep themselves in the background and instead try to focus attention on the events the people and the background and history leading up to the coup as a film it is not ground breaking in a stylistic or aesthetic sense and that is i think the way it should be what we get to see what embedded journalism should really be what we get to see is a remarkable account of a country struggling to attain democracy a charismatic leader chavez who actually cares for his people a story about power and greed as a coalition of corporate military media interests combine to lead a coup of a democratically elected leader and unprecedented access to a historical event as it unfolds

'1bad' predicted as '3good' : 29 examples, showing first 3

	clean
39006	before the days of home video stan laurel pre hardy comedy kill or cure was known to the extent that it was known at all only because a few sequences were included in robert youngson compilation film laurel and hardy laughing s youngson knew what he was doing the best gags in kill or cure were brought intact into his compilation while the rest of this only mildly funny comedy remained on youngson cutting room floor laurel portrays a commercial traveller hawking a patent medicine cried professor i o dine knox all that name is the funniest joke in this movie which is not sayin much i should point out that this movie dates from the shank of prohibition during prohibition quite a lot of americans purchased patent medicine if it had ahem medicinal properties so if knox all contains alcohol stan job in this movie is less desperate than the which he and ollie famously had in big business selling christmas trees in the summer too bad for this movie that it is not nearly so funny as big business we see stan but do not hear him in this silent film delivering a spirited sales talk to a man who seems to be paying attention until we learn that they are standing outside a deaf mute institution and this man is deaf a haughty woman emerges from the gates stan quickly tries to engage her attention by wiggling his fingers at her of course she is not deaf and she promptly whacks him with her umbrella i found this sequence offensive not because it involves deaf people the deaf are not the butt of the joke but because it abets the very widely held misconception among hearing people that they can communicate with the deaf by merely waggling their fingers randomly and performing charades without actually learning the highly complex grammar of sign language more amusingly a spinster in this movie has a pet canary named rudolph as in valentino and there is a gag involving trick photography to enable a man to hide behind an object that is narrower than his body i have seen this device in several cartoons and live action movies but kill or cure is i think the earliest movie to use it that i have seen so far stan laurel an under rated actor does bit of physical business here that is worthy of chaplin or keaton in which he conveys his emotions and a change in his demeanour while walking away from the camera with his back to us still laurel never really became a first rate comedian until he united with oliver hardy to form the greatest comedy team ever kill or cure barely rates out of
44542	at the beginning of loggerheads we are introduced to pairs of seemingly unrelated characters to make matters even more confusing we are informed via titles on the screen that the action is taking place in separate time lines between the years and it takes a great deal of time but eventually we come to see how the pairs are related mark austin a young man in his s gay and hiv positive is estranged from his conservative parents elizabeth and rev robert austin mark is now a drifter and arrives in kure beach north carolina a seaside town where he meets george sensitively played by michael kelly a gay motel owner and they eventually become involved with each other meanwhile mark birth mother grace played by bonnie hunt has come to the point in her life where she has decided to find the son she gave up for adoption when she was similarly mark adoptive mother also has decided to track her estranged son as she misses him despite the misgivings of her homophobic minister husband loggerheads we are told is based on a true story and that perhaps is its achilles heel director writer tim kirkman tries too hard to create scenes fraught with dramatic tension where there is very little to be found take mark and georgethey are both sensitive souls who have little to disagree about there is some slight tension when grace faces off against an adoption agency director who is forbidden by law to give her any information about her lost son as well as a slight conflict with her mother who denies that she disapproved of her when she became pregnant as a teenager no sparks fly either between elizabeth and robert since the good reverend has adamantly insisted from the beginning that he has no intention of reconciling with his son loggerheads is similar to brokeback mountain in that the gay couple are the good guys and the straight males for example the kure beach cop and the reverend are the baddies the biggest letdown of the movie is that there is no interaction and hence no dramatic conflict between mark and either of his mothers mark is already dead before either the birth or adoptive mother has a chance to reconcile with him kirkman theme is both a plea for tolerance and an exhortation for family members to express their heartfelt feelings before it is too late kirkman sentiments are for the most part well intentioned but they do not make for good drama loggerheads moves along at a snail pace without providing any new revelations or suspense regarding such topics as aids adoption and homophobia ultimately loggerheads fails due to a lack of originality

	clean
18852	i wonder who how and more importantly why the decision to call richard attenborough to direct the most singular sensation to hit broadway in many many years he is an academy award winning director yes he won for ghandi you moron jeremy irons is an academy winning actor do you want to see him play rocky balboa he has experience with musicals really oh what a lovely war have you forgotten to answer your question yes the film is a disappointment clear and simple not an ounce of the live energy survived the heavy handedness of the proceedings every character danced beautifully they were charming but their projection was theatrical i felt nothing but when i saw it on stage i felt everything the film should have been cast with stars unknown newcomers but stars with compelling unforgettable faces even the most invisible of the group great actors who could dance beautifully well michael douglas was in it true i forgot i am absolutely wrong and you are absolutely right nothing like a richard attenborough michael douglas musical

'2neutral' predicted as '3good' : 22 examples, showing first 3

	clean
31672	i suppose it is quite an achievement to be able to present to an audience a true tail about a frail man a tail in which the protagonist will spend the majority of the film on his back in a bed and totally unable to move and yet the achievement is in the film effectiveness as a dramatic piece as a recollection of a true story and i guess as an argument as to why people should have the right to die if they so wish to but the film is not a political statement and perhaps thankfully it shies away from too many scenes of debate although it does include for the sake of argument anyway more so this is of those foreign language films that presents its lead character as a cripple whom can do nothing but talk to the people around him and yet is able to come across as engaging and compelling anyway so rather than be an out and out argument the film is more a sweet yet timely dramatic piece about another person wanting something or in search of something the only difference is that by attaining this goal it would mean the termination of a someone life and it would be achieved by not physically going anywhere javier bardem plays real life galician ramon sampedro an individual who at a much earlier age dived into a clearing of water that was too shallow for such activity this rendered him bedridden for the rest of his life and his wish to die is the focus of chilean director alejandro amenabar film we have seen so many films in our lives in which characters have certain goals or targets must meet before the film is over to provide a satisfactory experience for the audience but the change of pace in the sea inside is gentle it does not involve young energetic attractive heroes going off to do battle in far off places but a real person after something that means so much to them even if you do have a strong policy either pro or anti euthanasia you may find yourself hoping ramon gets what he wants at the end of it all anyway the film sets its tone very early on with ramon giving a speech on why he wants to die to watching family members immediately introducing the situation and subject to the watching audience who may not know what the film is about interestingly some of the family members are anti what he wishes which might place any audience member that feels strongly about the subject in their respective shoes but the purpose of this set up is to tell the audience no this is not man after something who incidentally has the whole of his family backing him to the end quite easily the film could have gone down a route in which it is the sampedro family vs everybody else but some are anti ramon idea some are too young to acknowledge what is really going on and others are seemingly too distraught to even have an opinion other than they just want ramon to stick around a bit longer they love him after all the sea inside is a following of a story revolving around a victim of sorts ramon is a quadriplegic and it is his perspective we see things from this is something that may disjoint viewers or have the film come across as quite odd given we are being presented with a film from the point of view of a victim rather than an instigator or a lead character in a film that is always inducing the cause in the cause and effect drive but this is no criticism and it is a credit to the director for delivering such an approach in the effective manner in which it is the film asks questions it offers a scenario to its audience if you were in ramon position what would you do or think about or dream of or talk about consequently dreams about lawyer juila rueda are not so much shot for the audience pleasure as much as they are an ever so slight window into man escapist fantasies from his predicament the study of julia intensifies somewhat later on when a she begins to share certain similarities with ramon and that is when she begins to have strokes that are a result of a disease of her own this trait seems timely in the progression of their relationship and adds a further ingredient of connection on top of an already engaging friendship this is because julia feels the physical pain and restriction not in a sense that she is not able to get up and walk but i think she realises the value of life given how emotionally bad she felt beforehand while the film is based on a true story and covers the subject of euthanasia it feels like more of a down to earth drama about a man in a situation in which he is prepared to fight for what he wants but must do so verbally it is refreshing to see films like the sea inside as it not only references history and gives us an insight into that but as a stand alone film delivers on an emotional and engaging level

	clean
1220	in clarksberg was a famous speed trap town much revenue was generated by the sheriff department catching speeders the ones who tried to outrun the sheriff well that gave the sheriff a chance to push them off the clarksberg curve with his plymouth cruiser for example in the beginning of the movie a couple of servicemen on leave trying to get back to base on time are pushed off to their deaths if i recall correctly then day a stranger drove into town possibly the coolest hot rodder in the world michael mccord even his name is a car name as in mccord gaskets in possibly the ultimate hot rod a black flamed ford coupe the colors of death evil and hellfire he gets picked up for speeding by the sheriff on purpose he checks out the lay of the land he is the brother of of the sheriff victims he knows how his brother died the clarksberg government is all in favor of the sheriff there is only way to get justice served for the killing of his brother and to fix things so this is not a ever gonna happen again to anyone recreate the chase and settle the contest hot rodder style to the death he goes out to the curve and practices the sheriff knows mccord knows the race begins this is a movie to be remembered by anyone who ever tried to master maneuvering on a certain stretch of road
48613	popular actors are paired in showtime like de niro and murphy playing a police and an actor although they share the most screen time together they do not reach their expected performances such films like analyse this meet the parents dr doolittle and nutty professor the film starts interestingly but becomes a thin ordinary comedy which is quite disappointing it feels like especially de niro does not enjoy being in showtime thats a shame because he really shows his feelings thats what i think unfortunately if the idea had been used cleverly this could have been a more interesting piece of work and could be followed with a sequel like any other popular hollywood movie well i do not think there would be a sequel to this at least casting de niro and murpy again out of

READY

4.3 Show log-regression coefficients

READY

```
%pyspark
full_clf.named_steps
```

```
{'clf_1': LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=0, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False), 'chi2_1': SelectKBest(k=4000, score_func=<function chi2 at
 0x7144578>), 'allf_1': FeatureUnion(n_jobs=1,
       transformer_list=[('text', Pipeline(memory=None,
     steps=[('vect', CountVectorizer(analyzer='word', binary=False, decode_error=u'strict',
        dtype=<type 'numpy.int64'>, encoding='utf-8', input=u'content',
        lowercase=True, max_df=1.0, max_features=None, min_df=2,
        ngram_range=(1...        validate=False))])), ('ling', Pipeline(memory=None, steps=[('ta',
 SentimentVectorizer())]))],
       transformer_weights=None)}
```

Dig out used features

READY

```
%pyspark

# Get support
support =  full_clf.named_steps['chi2_1'].get_support()

a =    full_clf.named_steps['allf_1'].transformer_list[0][1].named_steps['vect'].get_feature_names
b =    "len"
c =    full_clf.named_steps['allf_1'].transformer_list[2][1].named_steps['ta'].get_feature_names()

# Get feature names
allfeats = np.append(np.append(a, b), c)
```

Look at regression coefficient for each category

READY

```
%pyspark

coefs           = pd.DataFrame(columns=['name', 'coef'])

coefs['name']   = allfeats[support]

for cat in range(np.size(labels)):
    print("---  Doing: {}".format(labels[cat]))
    coefs['coef']   = full_clf.named_steps['clf_1'].coef_[cat]

    coefs.sort_values(by='coef', ascending=False, inplace=True)
    cn.show(coefs, mr=10, type='st', fs=90, caption='most positive for label {}'.format(labels[cat

    coefs.sort_values(by='coef', ascending=True, inplace=True)
    cn.show(coefs, mr=10, type='st', fs=90, caption='most negative for label {}'.format(labels[cat

    # coefs.sort_values(by='name', ascending=True, inplace=True)
    # cn.show(coefs, mr=10, type='st', fs=90)
```

```
---  Doing: 1bad

Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 360, in <module>
    exec(code, _zcUserQueryNameSpace)
  File "<stdin>", line 5, in <module>
  File "/usr/lib64/python2.7/site-packages/pandas/core/frame.py", line 3119, in __setitem__
    self._set_item(key, value)
  File "/usr/lib64/python2.7/site-packages/pandas/core/frame.py", line 3194, in _set_item
    value = self._sanitize_column(key, value)
  File "/usr/lib64/python2.7/site-packages/pandas/core/frame.py", line 3391, in _sanitize_column
    value = _sanitize_index(value, self.index, copy=False)
  File "/usr/lib64/python2.7/site-packages/pandas/core/series.py", line 4001, in _sanitize_index
    raise ValueError('Length of values does not match length of ' 'index')
ValueError: Length of values does not match length of index
```

READY

4.4 Save model

READY

No pickle support for LemmaTokenizer

READY

```
%pyspark

# Save to pickle
pk.dump(full_clf, open(modelpath + 'TextAnalysis_Ex3_myclf.data', "w"))
```

```
Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 367, in <module>
    raise Exception(traceback.format_exc())
Exception: Traceback (most recent call last):
  File "/tmp/zeppelin_pyspark-291917852042627838.py", line 360, in <module>
    exec(code, _zcUserQueryNameSpace)
```

```
File "<stdin>", line 1, in <module>
File "/usr/lib64/python2.7/pickle.py", line 1370, in dump
  Pickler(file, protocol).dump(obj)
File "/usr/lib64/python2.7/pickle.py", line 224, in dump
  self.save(obj)
File "/usr/lib64/python2.7/pickle.py", line 331, in save
  self.save_reduce(obj=obj, *rv)
File "/usr/lib64/python2.7/pickle.py", line 419, in save_reduce
  save(state)
File "/usr/lib64/python2.7/pickle.py", line 286, in save
  f(self, obj) # Call unbound method with explicit self
```

READY

Some issues:

- The labels are rather slippery
  - The art categories tend to overlap
  - Publishing is an odd category
    - music 'publishing' is typically under 'music'
    - game 'publishing' is typically under 'games'
    - books or tablets then go to a generic 'publishing'
- Some categories are plain wrong:
  [Actual: food, Predicted: film & video] A look into the smoky world of competition bbq and the people that pour their time, energy, and souls into this truly American sport. American Smoke
  [Actual: comics, Predicted: music] Gilbert and Sullivan's immortal comic opera, directed by Alistair Donkin and produced by Marylhurst's Music and Interior Design Depts. HMS PINAFORE at Marylhurst University!
  => In this case I would agree with the classifier.
- Some descriptions are rather vague and tell little about the project:
  [Actual: food, Predicted: crafts] Skandals Revamp! I am expanding the business and need your help! This means new products, Skandals catalogue, and online store. Skandals expansion!
  [Actual: games, Predicted: film & video] PRELUDE (Canceled)
  => Next to nothing about the product.

READY

5 More advanced code: Loop over model and n-fold

READY

This will take a while (10 min) ... Be patient!

Get full features

READY

```
%pyspark
```

```
fullfeatures    = featpipe.fit_transform(reviews_clean['text']).toarray()
```

Preparation

READY

```
%pyspark

from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import LinearSVC
from sklearn.model_selection import cross_val_score

## Features filter
model_filter        = SelectKBest(chi2, k=nbfeats)

## Models
models = [
    RandomForestClassifier(n_estimators=600, max_depth=8, random_state=0),  # Check parametrisatio
    LinearSVC(),
    BernoulliNB(fit_prior=False),                                # Unigrams mostly
    MultinomialNB(fit_prior=False),                              # Bigrams mostly
    LogisticRegression(random_state=0),
    SGDClassifier(loss='hinge', penalty='l2'),                    # Relying on good sci-kit defaults
    ]
```

```
/usr/lib64/python2.7/site-packages/sklearn/tree/tree.py:40: RuntimeWarning:
numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
/usr/lib64/python2.7/site-packages/sklearn/ensemble/weight_boosting.py:29: DeprecationWarning:
numpy.core.umath_tests is an internal NumPy module and should not be imported. It will be removed i
n a future NumPy release.
/usr/lib64/python2.7/site-packages/sklearn/ensemble/gradient_boosting.py:34: RuntimeWarning:
numpy.dtype size changed, may indicate binary incompatibility. Expected 96, got 88
```

Do n-fold on full features

READY

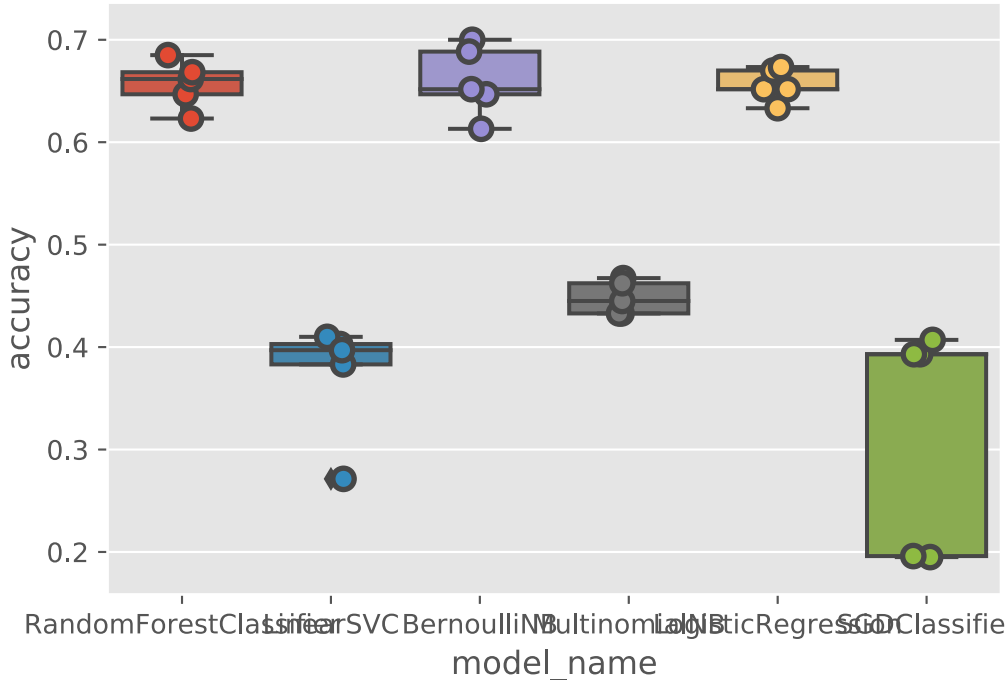
```
%pyspark

# NB Folds
CV  = 5

# nb feats

## Prepare output
cv_df   = pd.DataFrame(index=range(CV * len(models)))
entries = []

# Whether you want to run this or not
doCVModels = True

if doCVModels:
    # Loop over candidate models
    for model in models:
        model_name = model.__class__.__name__
        print("Doing: {}".format(model_name))

        # A fairly simple Pipeline
        full_clf = Pipeline([
            #('allf',   featpipe),   # Feature preparation - should be done externally so as not t
            ('chi2',   model_filter),
            ('clf',    model),
            ])

        accuracies = cross_val_score(estimator=full_clf, X=fullfeatures, y=reviews_clean['score'],

        for fold_idx, accuracy in enumerate(accuracies):
            entries.append((model_name, fold_idx, accuracy))

    cv_df = pd.DataFrame(entries, columns=['model_name', 'fold_idx', 'accuracy'])
```

```
    # Save to pickle
    pk.dump(cv_df, open(modelpath + 'TextAnalysis_Ex3_cv_df.data', "w"))
else:
    pass
    # Just read from pickle file
    cv_df = pk.load(open(modelpath + 'TextAnalysis_Ex3_cv_df.data', "r"))
```

```
Doing: RandomForestClassifier
[CV]  ................................................................
[CV]  ................................................................
[CV]  ................................................................
[CV]  ................................................................
[CV]  ................................................................
[CV] ........................... , score=0.646766169154, total=   3.8s
[CV] ........................... , score=0.661691542289, total=   3.8s
[Parallel(n_jobs=-1)]: Done   2 out of   5 | elapsed:    4.9s remaining:    7.3s
[CV] ..................................... , score=0.685, total=   3.6s
[CV] ........................... , score=0.668341708543, total=   3.6s
[CV] ........................... , score=0.623115577889, total=   3.5s
[Parallel(n_jobs=-1)]: Done   5 out of   5 | elapsed:    5.7s finished
Doing: LinearSVC
[CV]  ................................................................
[CV]  ................................................................
[CV]  ................................................................
[CV]
```

```
%pyspark

fig1, ax1 = plt.subplots(figsize=(6, 4))

sns.boxplot(ax=ax1, x='model_name', y='accuracy', data=cv_df)
sns.stripplot(ax=ax1, x='model_name', y='accuracy', data=cv_df, size=8, jitter=True, edgecolor="gr

cn.show(fig1)
```

READY




```
%pyspark
```

READY

```
cv_df.groupby('model_name').accuracy.mean()
```

```
model_name
BernoulliNB               0.660003
LinearSVC                 0.372882
LogisticRegression        0.656003
MultinomialNB             0.448064
RandomForestClassifier    0.656983
SGDClassifier             0.316817
Name: accuracy, dtype: float64
```

```
%pyspark
```

READY