Machine Learning: Sentiment analysis of movie reviews using Logistic Regression

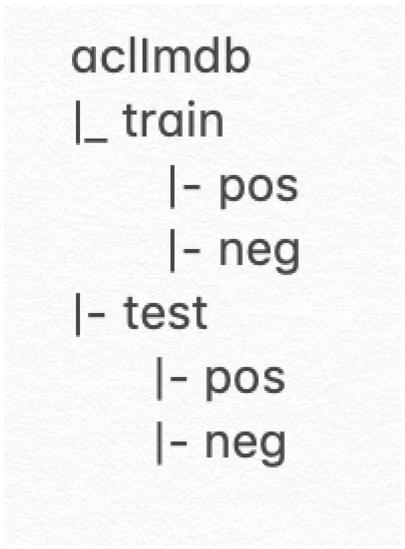
モハマド Meraj モルラ Follow
Oct 11, 2018 · 10 min read

In this article, we will focus on analysing IMDb movie reviews data and try to predict whether the review is positive or negative. Familiarity with some machine learning concepts will help to understand the code and algorithms used. We will use popular **scikit-learn** machine learning framework.

Data-set preparation:

We will use the dataset from here— *http://ai.stanford.edu/~amaas/data/sentiment/*

After downloading the dataset, unnecessary files/folders were removed so that folder structure looks as follows —



```
aclImdb
|_ train
      |- pos
      |- neg
|- test
     |- pos
     |- neg
```

folder structure of dataset

Load data into program:

We will load and peek into train and test data to understand the nature of data. In this case, both train and test data are in similar format.

```
from sklearn.datasets import load_files

reviews_train = load_files("aclImdb/train/")
text_train, y_train = reviews_train.data,
reviews_train.target

print("Number of documents in train data:
{}".format(len(text_train)))
print("Samples per class (train):
{}".format(np.bincount(y_train)))

reviews_test = load_files("aclImdb/test/")
text_test, y_test = reviews_test.data, reviews_test.target

print("Number of documents in test data:
{}".format(len(text_test)))
print("Samples per class (test):
{}".format(np.bincount(y_test)))
```

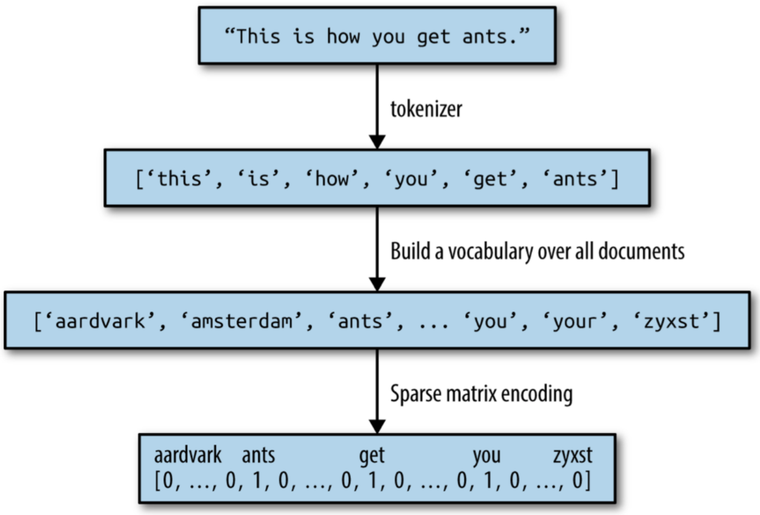
scikit-learn provides **load_files** to read this kind of text data. After loading data we printed the number of documents (train/test) and samples per class (pos/neg) which is as follows —

> *Number of documents in train data: 25000*
> *Samples per class (train): [12500 12500]*
> *Number of documents in test data: 25000*
> *Samples per class (test): [12500 12500]*

We can see total 25000 samples of training and test data with 12500 per class of pos and neg.

Representing text data as Bag of Words:

We want to count the word occurrences as a Bag of Words which include the below steps in the diagram —


"This is how you get ants."
tokenizer
['this', 'is', 'how', 'you', 'get', 'ants']
Build a vocabulary over all documents
['aardvark', 'amsterdam', 'ants', ... 'you', 'your', 'zyxst']
Sparse matrix encoding
aardvark ants get you zyxst
[0, …, 0, 1, 0, …, 0, 1, 0, …, 0, 1, 0, …, 0]

Bag of words processing [1]

In order to represent the input dataset as Bag of words, we will use **CountVectorizer** and call it's **transform** method. **CountVectorizer** is a transformer that converts the input documents into sparse matrix of features.

```
from sklearn.feature_extraction.text import CountVectorizer

vect = CountVectorizer(min_df=5, ngram_range=(2, 2))
X_train = vect.fit(text_train).transform(text_train)
X_test = vect.transform(text_test)

print("Vocabulary size: {}".format(len(vect.vocabulary_)))
print("X_train:\n{}".format(repr(X_train)))
print("X_test: \n{}".format(repr(X_test)))

feature_names = vect.get_feature_names()
print("Number of features: {}".format(len(feature_names)))
```

CountVectorizer is used with two parameters —

1. **min_df** (= 5): defines the minimum frequency of a word for it to be counted as a feature
2. **ngram_range** (= (2,2)): The ngram_range parameter is a tuple. It defines the minimum and maximum length of sequence of tokens considered. In this case, this length is 2. So, this will find sequence of 2 tokens like—'but the', 'wise man' etc.

Each entry in the resultant matrix is considered a feature. Output from above code snippet is as follows —

```
Vocabulary size: 129549
X_train:
<25000x129549 sparse matrix of type '<class 'numpy.int64'>'
 with 3607330 stored elements in Compressed Sparse Row
format>
X_test:
<25000x129549 sparse matrix of type '<class 'numpy.int64'>'
 with 3392376 stored elements in Compressed Sparse Row
format>
Number of features: 129549
```

In total, it found 129549 features.

Model development:

We will use **LogisticRegression** for model development as for high dimensional sparse data like ours, **LogisticRegression** often works best.

While developing model, we need to do two other things —

1. **Grid Search**: for paramater tuning of LogisticRegression. We want to determine what value of coefficeint '**C**' provides better accuracy.
2. **Cross validation**: in order to avoid overfitting of data.

To learn more about **GridSearch** and **Cross-validation** please refer to [2].

```
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression


param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best cross-validation score:
{:.2f}".format(grid.best_score_))
print("Best parameters: ", grid.best_params_)
print("Best estimator: ", grid.best_estimator_)
```

Here, we use 5-fold cross validation with **GridSearchCV**. After fitting train data we see the best_score_, best_params_ for 'C', and the best_estimator_ (the model we are going to use).

The output from above code snippet is as

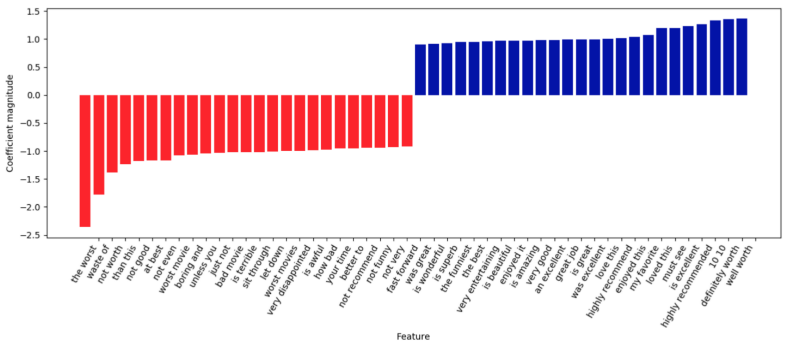
```
Best cross-validation score: 0.88
Best parameters:  {'C': 1}
Best estimator:  LogisticRegression(C=1, class_weight=None,
dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100,
multi_class='warn',
          n_jobs=None, penalty='l2', random_state=None,
solver='warn',
          tol=0.0001, verbose=0, warm_start=False)
```

We have a model with ‘C’ = 1 and with 88 percent accuracy.

We want to plot the best and worst top 25 features.

```
import matplotlib.pyplot as plt
import mglearn


mglearn.tools.visualize_coefficients(grid.best_estimator_.co
ef_, feature_names, n_top_features=25)
plt.show()
```



frequent best and worst features

* mglearn is a library that comes with the book [1]. You can download it from—https://github.com/amueller/mglearn

Making prediction:

Now we are going to make prediction over our test data using the trained model.

```
lr = grid.best_estimator_
lr.fit(X_train, y_train)
lr.predict(X_test)
print("Score: {:.2f}".format(lr.score(X_test, y_test)))
```

Output of prediction shows a score of 88% over test data.

```
Score: 0.88
```

For checking how our model performs on individual data, we will make one prediction with positive movie review and one with negative.

```
pos = ["I've seen this story before but my kids haven't. Boy
with troubled past joins military, faces his past, falls in
love and becomes a man. "
       "The mentor this time is played perfectly by Kevin
Costner; An ordinary man with common everyday problems who
lives an extraordinary "
       "conviction, to save lives. After losing his team he
takes a teaching position training the next generation of
heroes. The young troubled "
       "recruit is played by Kutcher. While his scenes with
the local love interest are a tad stiff and don't generate
enough heat to melt butter, "
       "he compliments Costner well. I never really
understood Sela Ward as the neglected wife and felt she
should of wanted Costner to quit out of "
       "concern for his safety as opposed to her selfish
needs. But her presence on screen is a pleasure. The two
unaccredited stars of this movie "
       "are the Coast Guard and the Sea. Both powerful
forces which should not be taken for granted in real life or
this movie. The movie has some "
       "slow spots and could have used the wasted 15 minutes
to strengthen the character relationships. But it still
works. The rescue scenes are "
       "intense and well filmed and edited to provide
maximum impact. This movie earns the audience applause. And
the applause of my two sons."]


print("Pos prediction: {}".
format(lr.predict(vect.transform(pos))))
```

This outputs —

```
Pos prediction: [1]
```

Here, 1 means it predicted a positive review.

```
neg = ["David Bryce\'s comments nearby are exceptionally well written and informative as almost say everything "
       "I feel about DARLING LILI. This massive musical is so peculiar and over blown, over produced and must have "
       "caused ruptures at Paramount in 1970. It cost 22 million dollars! That is simply irresponsible. DARLING LILI "
       "must have been greenlit from a board meeting that said \"hey we got that Pink Panther guy and that Sound Of Music gal... "
       "lets get this too\" and handed over a blank cheque. The result is a hybrid of GIGI, ZEPPELIN, HALF A SIXPENCE, some MGM 40s "
       "song and dance numbers of a style (daisies and boaters!) so hopelessly old fashioned as to be like musical porridge, and MATA HARI "
       "dramatics. The production is colossal, lush, breathtaking to view, but the rest: the ridiculous romance, Julie looking befuddled, Hudson "
       "already dead, the mistimed comedy, and the astoundingly boring songs deaden this spectacular film into being irritating. LILI is"
       " like a twee 1940s mega musical with some vulgar bits to spice it up. STAR! released the year before sadly crashed and now is being "
       "finally appreciated for the excellent film is genuinely is... and Andrews looks sublime, mature, especially in the last half hour......"
       "but LILI is POPPINS and DOLLY frilly and I believe really killed off the mega musical binge of the 60s..... "
       "and made Andrews look like Poppins again... which I believe was not Edwards intention. Paramount must have collectively fainted "
       "when they saw this: and with another $20 million festering in CATCH 22, and $12 million in ON A CLEAR DAY and $25 million in PAINT YOUR WAGON...."
       "they had a financial abyss of CLEOPATRA proportions with $77 million tied into 4 films with very uncertain futures. Maybe they should have asked seer "
       "Daisy Gamble from ON A CLEAR DAY ......LILI was very popular on immediate first release in Australia and ran in 70mm cinemas for months but it failed "
       "once out in the subs and the sticks and only ever surfaced after that on one night stands with ON A CLEAR DAY as a Sunday night double. Thank "
       "god Paramount had their simple $1million (yes, ONE MILLION DOLLAR) film LOVE STORY and that $4 million dollar gangster pic THE GODFATHER "
       "also ready to recover all the $77 million in just the next two years....for just $5m.... incredible!"]

print("Neg prediction: {}".
format(lr.predict(vect.transform(neg))))
```

This outputs —

```
Neg prediction: [0]
```

Here, 0 means it predicted a negative review.

Full Source code:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_files
from sklearn.model_selection import GridSearchCV
import numpy as np
import mglearn
import matplotlib.pyplot as plt

reviews_train = load_files("aclImdb/train/")
text_train, y_train = reviews_train.data,
reviews_train.target

print("Number of documents in train data:
{}".format(len(text_train)))
print("Samples per class (train):
{}".format(np.bincount(y_train)))

reviews_test = load_files("aclImdb/test/")
text_test, y_test = reviews_test.data, reviews_test.target

print("Number of documents in test data:
{}".format(len(text_test)))
print("Samples per class (test):
{}".format(np.bincount(y_test)))


vect = CountVectorizer(min_df=5, ngram_range=(2, 2))
X_train = vect.fit(text_train).transform(text_train)
X_test = vect.transform(text_test)

print("Vocabulary size: {}".format(len(vect.vocabulary_)))
print("X_train:\n{}".format(repr(X_train)))
print("X_test: \n{}".format(repr(X_test)))

feature_names = vect.get_feature_names()
print("Number of features: {}".format(len(feature_names)))

param_grid = {'C': [0.001, 0.01, 0.1, 1, 10]}
grid = GridSearchCV(LogisticRegression(), param_grid, cv=5)
grid.fit(X_train, y_train)

print("Best cross-validation score:
{:.2f}".format(grid.best_score_))
print("Best parameters: ", grid.best_params_)
print("Best estimator: ", grid.best_estimator_)

mglearn.tools.visualize_coefficients(grid.best_estimator_.co
ef_, feature_names, n_top_features=25)
plt.show()

lr = grid.best_estimator_
```

```
lr.predict(X_test)
print("Score: {:.2f}".format(lr.score(X_test, y_test)))

pos = ["I've seen this story before but my kids haven't. Boy
with troubled past joins military, faces his past, falls in
love and becomes a man. "
       "The mentor this time is played perfectly by Kevin
Costner; An ordinary man with common everyday problems who
lives an extraordinary "
       "conviction, to save lives. After losing his team he
takes a teaching position training the next generation of
heroes. The young troubled "
       "recruit is played by Kutcher. While his scenes with
the local love interest are a tad stiff and don't generate
enough heat to melt butter, "
       "he compliments Costner well. I never really
understood Sela Ward as the neglected wife and felt she
should of wanted Costner to quit out of "
       "concern for his safety as opposed to her selfish
needs. But her presence on screen is a pleasure. The two
unaccredited stars of this movie "
       "are the Coast Guard and the Sea. Both powerful
forces which should not be taken for granted in real life or
this movie. The movie has some "
       "slow spots and could have used the wasted 15 minutes
to strengthen the character relationships. But it still
works. The rescue scenes are "
       "intense and well filmed and edited to provide
maximum impact. This movie earns the audience applause. And
the applause of my two sons."]
print("Pos prediction: {}".
format(lr.predict(vect.transform(pos))))

neg = ["David Bryce\'s comments nearby are exceptionally
well written and informative as almost say everything "
       "I feel about DARLING LILI. This massive musical is
so peculiar and over blown, over produced and must have "
       "caused ruptures at Paramount in 1970. It cost 22
million dollars! That is simply irresponsible. DARLING LILI
"
       "must have been greenlit from a board meeting that
said \"hey we got that Pink Panther guy and that Sound Of
Music gal... "
       "lets get this too\" and handed over a blank cheque.
The result is a hybrid of GIGI, ZEPPELIN, HALF A SIXPENCE,
some MGM 40s "
       "song and dance numbers of a style (daisies and
boaters!) so hopelessly old fashioned as to be like musical
porridge, and MATA HARI "
       "dramatics. The production is colossal, lush,
breathtaking to view, but the rest: the ridiculous romance,
Julie looking befuddled, Hudson "
       "already dead, the mistimed comedy, and the
astoundingly boring songs deaden this spectacular film into
being irritating. LILI is"
       " like a twee 1940s mega musical with some vulgar
bits to spice it up. STAR! released the year before sadly
crashed and now is being "
       "finally appreciated for the excellent film is
genuinely is... and Andrews looks sublime, mature,
especially in the last half hour......"
       "but LILI is POPPINS and DOLLY frilly and I believe
really killed off the mega musical binge of the 60s..... "
       "and made Andrews look like Poppins again... which I
believe was not Edwards intention. Paramount must have
collectively fainted "
       "when they saw this: and with another $20 million
festering in CATCH 22, and $12 million in ON A CLEAR DAY and
$25 million in PAINT YOUR WAGON...."
       "they had a financial abyss of CLEOPATRA proportions
with $77 million tied into 4 films with very uncertain
futures. Maybe they should have asked seer "
       "Daisy Gamble from ON A CLEAR DAY ......LILI was very
popular on immediate first release in Australia and ran in
70mm cinemas for months but it failed "
       "once out in the subs and the sticks and only ever
surfaced after that on one night stands with ON A CLEAR DAY
as a Sunday night double. Thank "
       "god Paramount had their simple $1million (yes, ONE
MILLION DOLLAR) film LOVE STORY and that $4 million dollar
gangster pic THE GODFATHER "
       "also ready to recover all the $77 million in just
the next two years....for just $5m.... incredible!"]
print("Neg prediction: {}".
format(lr.predict(vect.transform(neg))))
```

More things to consider:

Somethings more to consider for text analysis are—**Lemmatization**, **Stemming**, and ***term frequency–inverse document frequency* (tf–idf)**, etc. You can learn how to use these on the web and also from [1]. But for this example project purpose, I found these techniques increasing the execution time a lot without giving any significant improvement in accuracy.

I hope this write-up was helpful to some if not many. This is my first write-up on machine learning topic and I am no expert in this field, kind of still learning. If you like this article, please follow me here or on twitter.

References:

[1] http://shop.oreilly.com/product/0636920030515.do

[2] http://nbviewer.jupyter.org/github/rhiever/Data-Analysis-and-Machine-Learning-Projects/blob/master/example-data-science-notebook/Example%20Machine%20Learning%20Notebook.ipynb

[3] https://medium.com/@rnbrown/more-nlp-with-sklearns-countvectorizer-add577a0b8c8