Fall 2018

Jupyter: Intro to Data Science - Lecture 12 Clustering and Topic Modeling

Grant Long
CUNY City College

NYC Tech-in-Residence Corps

Data Dive 12: Clustering and Topic Modeling

This week we'll use movie data from [the movie database (tMDB) (https://www.themoviedb.org/?language=en-US)](https://www.themoviedb.org/?language=en-US) available on [Kaggle (https://www.kaggle.com/tmdb/)](https://www.kaggle.com/tmdb/).

> The Movie Database (TMDb) is a community built movie and TV database. Every piece of data has been added by our amazing community dating back to 2008. TMDb's strong international focus and breadth of data is largely unmatched and something we're incredibly proud of. Put simply, we live and breathe community and that's precisely what makes us different.

Tech Specs

Among some familiar tools from recents data dives, we'll be using [`StandardScaler` (https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html)](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) and [`KMeans` (https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html)](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html) from `scikit-learn` . As we discussed in class, we'll also be using `altair` , `gensim` , and `pyLDAvis` .

In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.feature_extraction.text import CountVectorizer
```

Instruct Colab (or your personal machine) to install non-standard packages.

In [ ]:
```
imUsingColab = False

if imUsingColab:
    !pip install gensim
    !pip install pyLDAvis
    !pip install vega
    !pip install altair
```

Import non-standard packages.

In [ ]:
```
from gensim import corpora
from gensim.models.ldamodel import LdaModel

import pyLDAvis.gensim
```

Attempt to load `altair`. Will not work without `altair` and `vega` pre-installed.

In [ ]:
```
try:
    import altair as alt
    if imUsingColab:
        alt.renderers.enable('colab')
    else:
        alt.renderers.enable('notebook')
    imUsingAltair = True
    print('Altair successfully loaded.')

except ModuleNotFoundError:
    imUsingAltair = False
    print('Altair loading failed. Will default to matplotlib.')
```

In [ ]:
```
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
warnings.simplefilter(action='ignore', category=DeprecationWarning)
pd.options.mode.chained_assignment = None

random_state = 20181126
```

Load data from web

In [ ]:
```
df = pd.read_csv('https://grantmlong.com/data/tmdb_5000_movies.csv')
```

Drop `na` values, filter for recent movies produced in English.

In [ ]:
```
print(df.shape)
cluster_vars = ['budget', 'popularity', 'revenue', 'runtime', 'vote_average
df = df.dropna(subset=cluster_vars+['release_date'])
print(df.shape)
```

In [ ]:
```
df['release_year'] = df.release_date.str.slice(0,4).astype(int)
samp_df = df.loc[(df.release_year>=1900) &
                 (df.original_language=='en')
                ].reset_index()

titles = {samp_df.title.loc[i] : i for i in samp_df.index.values}

print(samp_df.shape)
```

In [ ]:
```

```

Create Vector of Normalized Data for Clustering.

In [ ]:
```
scaler = StandardScaler()
scaler.fit(samp_df[cluster_vars].astype(float))
X = pd.DataFrame(scaler.transform(samp_df[cluster_vars].astype(float)), col
```

Cluster with `KMeans`

To start off, we'll create 5 clusters. Obviously we'll want to play with this to see what kinds of results we get.

In [ ]:
```
kmeans = KMeans(n_clusters=5,
                random_state=random_state)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)
samp_df['cluster'] = (y_kmeans+1).astype(str)
```

In [ ]:
```

```

Visualize Results

I've included code for `altair` hear for those interested. It's a lot nicer to use in this case because we'll be able to add tooltips to see which movies land where. If that's not working for you, you can always use `matplotlib` as a fall back.

In [ ]:
```
x_var = 'budget'
y_var = 'revenue'

if imUsingAltair:

    chart = alt.Chart(
        samp_df,
        width=650,
        height=400
    ).mark_circle(
        size=80
    ).encode(
        x=x_var,
        y=y_var,
        color='cluster',
        tooltip=['title', 'revenue', 'release_date']
    ).interactive()

else:
    chart = plt.figure(figsize=[14,7])
    chart = plt.scatter(samp_df[x_var], samp_df[y_var], c=samp_df.cluster.a
    chart = plt.xlim([0, 300000000])
    chart = plt.ylim([0, 300])
    chart = plt.xlabel(x_var)
    chart = plt.ylabel(y_var)

chart
```

In [ ]:
```

```

In [ ]:
```

```

In [ ]:
```

```

Topic Modeling the Movies

Our movie data also contains brief overviews of movie contents which we can use in topic modeling! Here, we'll transform the raw text into tokens for use in the bag-of-words style analysis we need to do topic modeling.

First, let's take a look at a few overviews. Ideally, for the richest analysis, our documents would be a bit longer, but these should nonetheless give us some interesting results.

In [ ]:
```
for i in np.random.choice(df.index.values, 3):
    print()
    print(df.overview.loc[i], '\n')
```

Let's both vectorize and tokenize our text. I've filled in a lot for you already, but suffice to say that tokens are the preferred way to `gemsim` takes its input.

In [ ]:
```
vectorizer = CountVectorizer(max_df=.5,
                             min_df=8,
                             max_features=1000,
                             stop_words='english'
                            )

X_text = pd.DataFrame((vectorizer.fit_transform(samp_df.overview)>0).toarra
all_words = vectorizer.get_feature_names()
d = {i : all_words[i] for i in range(len(all_words))}

tokens = [[d[j] for j in X_text.columns[X_text.loc[i]].tolist()] for i in r

print(np.random.choice(tokens, 2))
```

Next, we'll use `gensim` 's built in functionality to create a dictionary and bag of words for us.

In [ ]:
```
dictionary = corpora.Dictionary(tokens)
corpus = [dictionary.doc2bow(text) for text in tokens]
```

Finally, we train our model and output our results.

In [ ]:
```
ldamodel = LdaModel(corpus, num_topics=10, id2word=dictionary, passes=30, r
topics = ldamodel.print_topics(num_words=6)
for topic in topics:
    print(topic)
```

Let's see how our model classifies some of our favorite movies.

In [ ]:
```
i = titles['X-Men: Days of Future Past']

print(samp_df.title.loc[i])
print(samp_df.overview.loc[i])
ldamodel.get_document_topics(corpus[i])
```

Using `pyLDAvis` , we can also build an interactive visualization of our topic model.

In [ ]:
```
lda_display = pyLDAvis.gensim.prepare(ldamodel, corpus, dictionary, sort_tc
pyLDAvis.display(lda_display)
```

In [ ]:

In [ ]:

In [ ]:

In [ ]: