Milestone 2

April 12, 2017

0.0.1 Milestone 2: Assembling training data, due Wednesday, April 12, 2017

We are aware that you have little time this week, due to the midterm. So this milestone is a bit easier to achieve than the others. The goal for this week is to prepare the data for the modeling phase of the project. You should end up with a typical data setup of training data X and data labels Y.

The exact form of X and Y depends on the ideas you had previously. In general though Y should involve the genre of a movie, and X the features you want to include to predict the genre. Remember from the lecture that more features does not necessarily equal better prediction performance. Use your application knowledge and the insight you gathered from your genre pair analysis and additional EDA to design Y. Do you want to include all genres? Are there genres that you assume to be easier to separate than others? Are there genres that could be grouped together? There is no one right answer here. We are looking for your insight, so be sure to describe your decision process in your notebook.

In preparation for the deep learning part we strongly encourage you to have two sets of training data X, one with the metadata and one with the movie posters. Make sure to have a common key, like the movie ID, to be able to link the two sets together. Also be mindful of the data rate when you obtain the posters. Time your requests and choose which poster resolution you need. In most cases w500 should be sufficient, and probably a lower resolution will be fine.

The notebook to submit this week should at least include:

- Discussion about the imbalanced nature of the data and how you want to address it
- Description of your data
- What does your choice of Y look like?
- Which features do you choose for X and why?
- How do you sample your data, how many samples, and why?

Important: You do not need to upload the data itself to Canvas.

```
In [4]: # we will eventually get 2000 movies from IMBD randomly
        # create a top 200 list by sending the query 5 times

        random1 = urllib.urlopen("https://api.themoviedb.org/3/discover/movie?api_k
        random1_json = json.loads(random1.read())
        random_movie_data_json = random1_json["results"]

        pages = range(2,1001)
        np.random.shuffle(pages)
```

```
        sampled_pages = pages[:99]


        # need to sleep in order to not return an error: limitation 40 requests per
        for i in range(len(sampled_pages)):
            if i%39 == 0:
                time.sleep(7)

            tmp_url = "https://api.themoviedb.org/3/discover/movie?api_key=2dc6c9f1
            tmp_page = urllib.urlopen(tmp_url)
            tmp_json = json.loads(tmp_page.read())
            for movie in tmp_json["results"]:
                random_movie_data_json.append(movie)
```

In [34]:
```
random_movie_data_json[0]
```

Out[34]:
```
{u'adult': False,
 u'backdrop_path': u'/6aUWe0GSl69wMTSWWexsorMIvwU.jpg',
 u'genre_ids': [14, 10402, 10749],
 u'id': 321612,
 u'original_language': u'en',
 u'original_title': u'Beauty and the Beast',
 u'overview': u"A live-action adaptation of Disney's version of the classi
 u'popularity': 174.72627,
 u'poster_path': u'/tWqifoYuwLETmmasnGHO7xBjEtt.jpg',
 u'release_date': u'2017-03-16',
 u'title': u'Beauty and the Beast',
 u'video': False,
 u'vote_average': 6.9,
 u'vote_count': 1512}
```

In [26]:
```
# make a df and save
genre_ids, overview, popularity, poster_path, title, vote_average, vote_co
for movie in random_movie_data_json:
    genre_ids.append(movie["genre_ids"])
    overview.append(movie["overview"])
    popularity.append(movie["popularity"])
    poster_path.append(movie["poster_path"])
    title.append(movie["title"])
    vote_average.append(movie["vote_average"])
    vote_count.append(movie["vote_count"])
    release_date.append(movie["release_date"])
    movie_id.append(movie["id"])

data = {'title': title, 'overview': overview, 'popularity': popularity, 'r
ran_df = pd.DataFrame(data = data)
ran_df.to_csv('dataset1.csv', encoding = 'utf-8')
```

In [33]:
```
ran_df.head()
```

Out[33]:

	genre_ids	movie_id	overview	popularity	poster_path	release_date	title	vote_average	vote_count
0	[14, 10402, 10749]	321612	A live-action adaptation of Disney's version o...	174.726270	/tWqifoYuwLETmmasnGHO7xBjEtt.jpg	2017-03-16	Beauty and the Beast	6.9	1539
1	[36, 16, 35, 10751, 27, 53, 18, 80]	295693	A story about how a new baby's arrival impacts...	125.359764	/67NXPYvK92oQgEQvLppF2Siol9q.jpg	2017-03-23	The Boss Baby	5.7	340
2	[28, 18, 878]	263115	In the near future, a weary Logan cares for an...	83.229122	/45Y1G5FEgttPAwjTYic6czC9xCn.jpg	2017-02-28	Logan	7.6	2240
3	[28, 12, 14]	293167	Explore the mysterious and dangerous home of t...	69.901494	/5wBbdNb0NdGiZQJYoKHRv6VbiOr.jpg	2017-03-08	Kong: Skull Island	6.1	1048
4	[16, 35, 18, 10751, 10402]	335797	A koala named Buster recruits his best friend ...	67.964414	/s9ye87pvq2IaDvjv9x4IOXVjvA7.jpg	2016-11-23	Sing	6.7	1056

In [32]:
```
# now we get the posters into a df
imgs = []
for i in range(len(ran_df.poster_path[:20])):
    if i%39 == 0:
        # sleep
        time.sleep(7)
    url = "https://image.tmdb.org/t/p/w500" + poster_path[i]
    tmp_poster = cStringIO.StringIO(urllib.urlopen(url).read())
    img = Image.open(tmp_poster)
    imgs.append(img)

imgs[:20]
```

Out[32]:
```
[<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x118BA
 <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
 <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
 <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
 <PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
```

```
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
<PIL.JpegImagePlugin.JpegImageFile image mode=RGB size=500x750 at 0x119BC
```

The above is just the top 20. From later ones, we can see that width is always 500 but height can vary.

```
In [35]: # create rgb arrays for these images
         RGB = []

         for img in imgs:
             tmp = img.load()
             pixels = []
             for i in range(img.size[0]):
                 for j in range(img.size[1]):
                     pixels.append(tmp[i,j])
             RGB.append(pixels)
```

```
In [38]: data_img = {'title': title, 'movie_id': movie_id,'genre_ids': genre_ids, '
         img_df = pd.DataFrame(data = data_img)
         img_df.head()
```

```
Out[38]:                                                  RGB  \
         0  [(12, 32, 65), (11, 31, 64), (21, 41, 74), (28...
         1  [(255, 255, 255), (255, 255, 255), (255, 255, ...
         2  [(7, 9, 8), (5, 7, 6), (7, 9, 8), (7, 9, 8), (...
         3  [(140, 51, 17), (135, 46, 12), (133, 44, 10), ...
         4  [(92, 79, 107), (89, 76, 104), (95, 80, 109), ...

                                     genre_ids  \
         0                  [14, 10402, 10749]
         1  [36, 16, 35, 10751, 27, 53, 18, 80]
         2                        [28, 18, 878]
         3                         [28, 12, 14]
         4          [16, 35, 18, 10751, 10402]

                                                          imgs  movie_id  \
```

```
0  <PIL.JpegImagePlugin.JpegImageFile image mode=...    321612
1  <PIL.JpegImagePlugin.JpegImageFile image mode=...    295693
2  <PIL.JpegImagePlugin.JpegImageFile image mode=...    263115
3  <PIL.JpegImagePlugin.JpegImageFile image mode=...    293167
4  <PIL.JpegImagePlugin.JpegImageFile image mode=...    335797

                  title
0  Beauty and the Beast
1         The Boss Baby
2                 Logan
3    Kong: Skull Island
4                  Sing
```

In [39]:
```
# the below is a dataset we created to address the data imbalance issue

# a dataset balanced by genre and release year
genre_movie_data_json = []

# sample from these years
years = [2007, 2008, 2009, 2010, 2011, 2012, 2013, 2014, 2015, 2016]

# all genres
genres = [10752, 80, 10402, 35, 36, 37, 53, 9648, 12, 10770, 14, 16, 18, 9

for i in range(0,len(years)):
    for j in range(0,len(genres)):

        # sleep periodically
        if (i*j*20)%39 == 0:
            time.sleep(7)

        # pull one page, 20 movies with the same [genre, release date]
        tmp_url = "https://api.themoviedb.org/3/discover/movie?api_key=2dc
        tmp_page = urllib.urlopen(tmp_url)
        tmp_json = json.loads(tmp_page.read())
        for movie in tmp_json['results']:
            genre_movie_data_json.append(movie)

len(genre_movie_data_json)
```

Out[39]:
```
3799
```

In [41]:
```
genre_ids, overview, popularity, poster_path, title, vote_average, vote_co
for movie in genre_movie_data_json:
    genre_ids.append(movie["genre_ids"])
    overview.append(movie["overview"])
    popularity.append(movie["popularity"])
    poster_path.append(movie["poster_path"])
```

```
            title.append(movie["title"])
            vote_average.append(movie["vote_average"])
            vote_count.append(movie["vote_count"])
            release_date.append(movie["release_date"])
            movie_id.append(movie["id"])

        data = {'title': title, 'overview': overview, 'popularity': popularity, 'r
        genre_df = pd.DataFrame(data = data)

        # check for duplicates
        # print len(genre_df.movie_id.unique())

        # delete duplicates based on movie_id
        genre_clean = genre_df.loc[genre_df['movie_id'].isin(genre_df.movie_id.dro
        # print len(genre_df.loc[genre_df['movie_id'].isin(genre_df.movie_id.drop_

        genre_df.head()
```

Out[41]:

	genre_ids	movie_id	overview	popularity	poster_path	release_date	title	vote_average	vote_count
0	[28, 12, 14, 10752]	9703	As the Roman empire crumbles, young Romulus Au...	2.154325	/8K4WWwFew1CzCGVkgmKdamCA6kz.jpg	2007-04-19	The Last Legion	5.1	161
1	[18, 10752]	11600	Redacted is a film written and directed by Bri...	1.688185	/59SCyrGk5KtTtqZg16QmHx7BInt.jpg	2007-01-01	Redacted	6.0	29
2	[18, 10752]	7862	The story Jewish counterfeiter, Salomon Sorowi...	1.535367	/bRQddrgVemZtFdnrPy9AxTpkhpj.jpg	2007-02-09	The Counterfeiters	7.3	123
3	[10752, 35, 18]	6172	Hitler no longer believes in himself, and can ...	1.316710	/g0qAFteXp0V8QbFPxdx9bo058IP.jpg	2007-01-11	My Führer	5.4	20
4	[18, 36, 10752]	13614	An examination of the Soviet slaughter of thou...	1.222725	/yNOxe78EYZO5LcQTfDwTeIuvDcK.jpg	2007-09-21	Katyn	6.6	41

^ our balanced dataset

1 Discussions

- Discussion about the imbalanced nature of the data and how you want to address it

We noticed in our data that there is not an equal representation of each genre in our data set. To address this problem, we created an additional data set filtering for the genres. Therefore, we created a stratefied sample where each genre had equal weights in our data. Another method we thought about was to bootstrap our data during the modeling process to have equal weights for each genre.

- Description of your data

We have 3 separate data sets. The first two metadata sets contain "overview," "popularity," "poster_path," "title," "vote_average," "vote_count," "release_date," "id," and "genre_ids." The third data set contains the RGB pixels of each movie poster. These two data sets are linked using "id."

- What does your choice of Y look like?

Our choice of Y is a list of genre labels.

- Which features do you choose for X and why?

We chose "overview," "popularity," "title," "vote_average," "vote_count," and "release_date." We are going to do text analysis on the overview data because the plots of the movies are probably indicative of movie genres. The other genres are there because they could also convey information about the movie genres. For example, actions movies probably tend to be more popular and have higher "vote_average."

- How do you sample your data, how many samples, and why?

For the first dataset, we used the GET /discover/movie method to get 2000 movies with English as its original language. The GET /discover/movie method returns 1000 pages with 20 movie objects on each page. We picked the first page and 99 other pages by random, and grab all 20 movie objects on each of those 100 pages.

For the second dataset, we used the /discover/movie method to get 80 movies from every genre, 20 each from the years 2013, 2014, 2015, 2016. We then deleted the duplicates according to movie ids and ended up with 976 movies. This is a more balanced dataset.

```
In [ ]:
```