Final Report – Group 50, Crazy Poor Asian

Shuiling Yu, Yekai Chen, Chin-Han Lin, Zheying Lu

1. Briefly describe what the project accomplished.

Our project aims to create a sample movie database including top-100 popular movie from 2008 to 2019, to build up movie box office prediction model and to show visualization radar figure for each movie. The final application is based on this movie database, where user can easily read, edit, delete information of movie, director and actor. Besides, though the prediction model does not show high accurateness, it still gives reference of movie box prediction. And radar figures demonstrate the competitiveness of each movie in our database and give visualized recommendation for users.

2. Discuss the usefulness of your project, i.e. what real problem you solved.

It is difficult to get the answer for a movie whether it is worth watching directly from several data of movie pages on IMDB or other movie websites. However, in our project, we combined information not only from movie itself, but also from directors and actors which including gross, votes, rating, director award and so on. It is useful to help user make direct judgement from radar figures and learn directly the status of each movie in our database. Applying our application, users can save time and get more correct evaluation in multiple dimensions because they do not have to search a lot of pages related to movies and we have done information collection and analysis.

Furthermore, predicting movie box office is a challenging part to show whether a movie will be successful or not. In the project, a regression model considering multi-factors has been established and this model has been evaluated by statistic methods. It considers the fame of actors and director, the quality of the movie as well as the type of movie, which are essential elements affecting the attraction of the movie. In further improvement, this model will also be used by investors to reduce the risk of investment and control cost to maximize the profit.

3. Discuss the data in your database

There are three tables in our database.

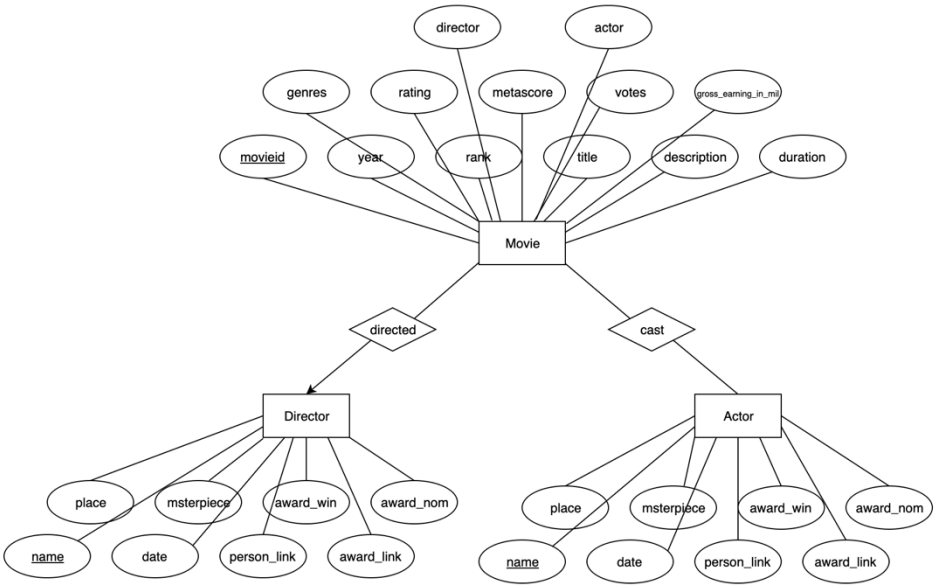
The first table schema is Movie(movieid, year, rank, title, description, duration, genres, rating, metascore, votes, gross_earning_in_mil, director, actor). Total 1100 tuples and 13 columns are in Movie table. And the type of genres is list.

The second table schema is Director(name, date, place, masterpiece, award_win, award_nom, person_link and award_link). Total 639 tuples and 8 columns are in this table. And the type of masterpieces is list.

The second table schema is Actor(name, date, place, masterpiece, award_win, award_nom, person_link and award_link). Total 991 tuples and 8 columns are in this table. And the type of masterpieces is list.

4. Include your ER Diagram and Schema

The ER Diagram:



The schema is shown in part3.

5. Briefly discuss from where you collected data and how you did it (if crawling is automated, explain how and what tools were used)

The data are crawled on IMDB by python package “BeautifulSoup” and established data frames by python package “pandas”. First, the original URL paths should be crawled can be created by paste base URL with detailed year together to determine the real URL. And then in crawl function, we should extract html tags to grasp information. Before the command line to grasp information, we should also to check if it is Null or not. And then append information for each feature into lists. Finally, we can create data frames to combine data.

For director table and actor table, the URL comes from the hyperlinks that has been crawled in movie pages. And the specific methods are similar.

6. Clearly list the functionality of your application (feature specs)

1) Integration of movie, actor, director database
2) Movie box office prediction based on existing records in database
3) Box office analysis and visualization based on different aspects

7. Explain one basic function

There are four basic functions in the project, including inserting, deleting, editing and searching data. Each function is user-friendly. Taking the insert function as an example, our website provides the from with information that user can add if user wants to insert new movie data on our website. The information user can insert is same as the attributes in the movie table, such as Movie Title, Released Year, Movie type and so on. After user submits the form, our website can automatically add the data to our database and show in the movie table on the website. Moreover, we adopt foreign keys to connect each table, so that both of the other tables also create new data in each table and store the new data in the database.

8. Show the actual SQL code snippet

In the basic functions of project, our team uses SQL code to implement the search function. In the search function, users input the key words to look for the relevant results they want. After users click on the search button, the movie, actor and director tables are joined together and the information in these tables is filtered by SQL query to get the necessary data. SQL code snippet is shown below:

```
SELECT m.title AS title, d.name AS name, a.name AS star
FROM pages_director AS d
LEFT JOIN pages_movie AS m ON d.name = m.director_id
LEFT JOIN pages_actor AS a ON a.name = m.actor_id
WHERE m.title LIKE %s
```

Furthermore, both of our advanced functions also adopt SQL query to join these three tables and select the essential information. Then, the filtered information is used to create the radar charts and predict the box office of movie. One of SQL code snippet for the advance function (radar chart) is shown below:

```
SELECT m.title AS title, m.rating AS rating, m.votes AS votes, m.metascore AS metascore,
m.gross_earning_in_mil AS gross, d.name AS name, d.award_win AS d_win, d.award_nom AS
d_nom, a.name AS star, a.award_win AS a_win, a.award_nom AS a_nom
FROM pages_director AS d
LEFT JOIN pages_movie AS m ON d.name = m.director_id
LEFT JOIN pages_actor AS a ON a.name = m.actor_id
WHERE m.title LIKE %s
```

9. List and briefly explain the dataflow, i.e. the steps that occur between a user entering the data on the screen and the output that occurs (you can insert a set of screenshots)

   Data Flow for Search Function:
   1) On the navigation bar, user can input the key words to search the related movie information.

2) After users click on the search button, the search function in the view.py is called by the webserver to join all tables and collect the data fitting to the query from database.

views.py

```
def search(request):
    template = 'recommendation.html'

    query = request.GET.get('q')
    tep = "%%%s%%" % query
    filter_title = Director.objects.raw(
        "SELECT m.title AS title, d.name AS name, a.name AS star \
        FROM pages_director AS d LEFT JOIN pages_movie AS m ON d.name = m.director_id \
        LEFT JOIN pages_actor AS a ON a.name = m.actor_id \
        WHERE m.title LIKE %s", [tep])

    filter_data = Director.objects.raw(
        "SELECT m.title AS title, m.rating AS rating, m.votes AS votes, m.metascore AS metascore, \
        m.gross_earning_in_mil AS gross, d.name AS name, d.award_win AS d_win, d.award_nom AS d_nom, \
        a.name AS star, a.award_win AS a_win, a.award_nom AS a_nom \
        FROM pages_director AS d LEFT JOIN pages_movie AS m ON d.name = m.director_id \
        LEFT JOIN pages_actor AS a ON a.name = m.actor_id \
        WHERE m.title LIKE %s", [tep])

    limit_tuple = filter_data[:1]
    for movie in limit_tuple:
        title = movie.title
        rating = movie.rating
        votes = movie.votes
        metascore = movie.metascore
        gross = movie.gross
        d_award = movie.d_win + movie.d_nom
        a_award = movie.a_win + movie.a_nom
```

3) Then, the search function implements the data processing for the advanced function (radar chart), in that our basic search function is combined with the advanced function. Data after processing is stored in the dictionary data type and connected to the recommendation.html.

views.py

```


    rating_max = Movie.objects.all().aggregate(rm1 = Max('rating'))
    rating_min = Movie.objects.all().aggregate(rm2 = Min('rating'))
    rating_range = rating_max.get('rm1')-rating_min.get('rm2')
    new_rating = (rating-rating_min.get('rm2'))/rating_range*100

    votes_max = Movie.objects.all().aggregate(rm1 = Max('votes'))
    votes_min = Movie.objects.all().aggregate(rm2 = Min('votes'))
    votes_range = votes_max.get('rm1')-votes_min.get('rm2')
    new_votes = (votes-votes_min.get('rm2'))/votes_range*100

    metascore_max = Movie.objects.all().aggregate(rm1 = Max('metascore'))
    metascore_min = Movie.objects.all().aggregate(rm2 = Min('metascore'))
    metascore_range = metascore_max.get('rm1')-metascore_min.get('rm2')
    new_metascore = (metascore-metascore_min.get('rm2'))/metascore_range*100

    gross_max = Movie.objects.all().aggregate(rm1 = Max('gross_earning_in_mil'))
    gross_min = Movie.objects.all().aggregate(rm2 = Min('gross_earning_in_mil'))
    gross_range = gross_max.get('rm1')-gross_min.get('rm2')
    new_gross = (gross-gross_min.get('rm2'))/gross_range*100

    context = {
        'filter_title': filter_title,
        'limit_tuple': limit_tuple,
        'new_rating': new_rating,
        'new_votes': new_votes,
        'new_d_award': new_d_award,
        'new_a_award': new_a_award,
        'ew_metascoren': new_metascore,
        'new_gross': new_gross
    }
    return render(request, template, context)
```

4) In the recommendation.html, data is processed to the table type and radar chart.

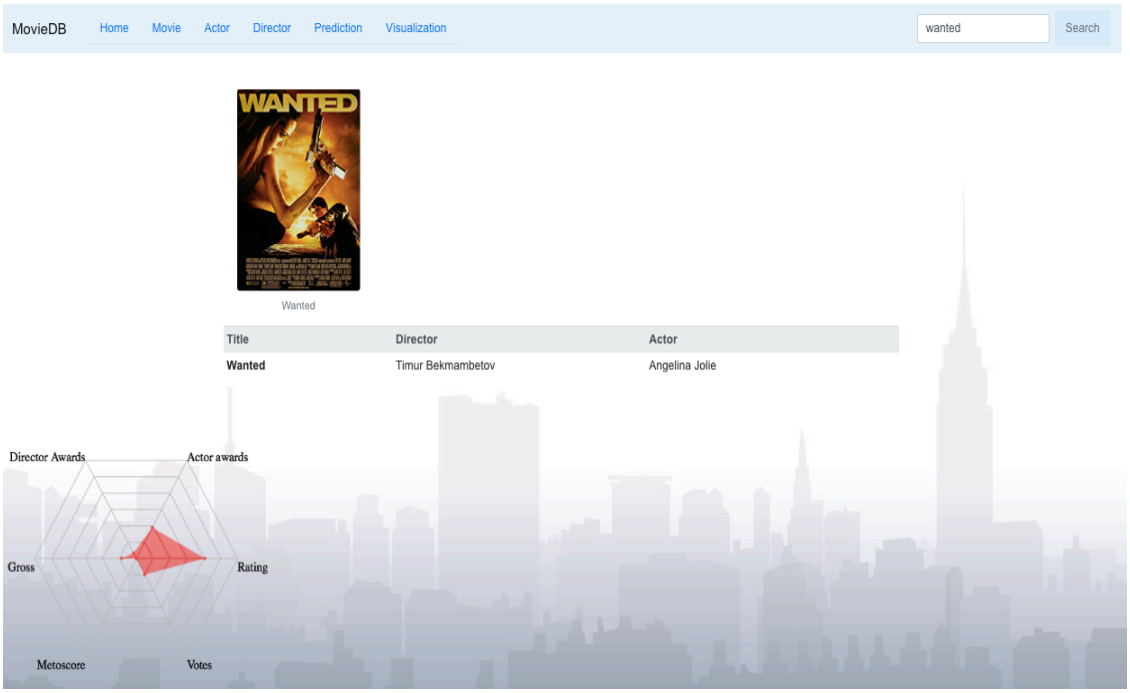
recommendation.html

```
        }
    </style>
    <table class="table table-hover table-sm">
        <thead class="thead-light" data-file-width="5px">
        <tr>
            <th scope="col">Title</th>
            <th scope="col">Director</th>
            <th scope="col">Actor</th>
        </tr>
        </thead>
        <tbody>
        {% for movie in filter_title %}
            <tr>
                <th scope="row"> {{ movie.title }}</th>
                <td> {{ movie.name }}</td>
                <td> {{ movie.star }}</td>
            </tr>
        {% endfor %}
        {% for movie in filter_title %}
            {% if movie.title %}
            <figure class="figure">
              <img src="{% get_static_prefix %}images/poster/{{movie.title}}.jpg" class="figure-img img-fluid rounded" alt="MoviePoster" hspace="20">
              <figcaption class="figure-caption" style="...">{{movie.title}}</figcaption>
            </figure>
            {% endif %}
        {% endfor %}
        </tbody>
    </table>
        <h3>{{ filter_title.name }}</h3><br>
{% else %}
```

recommendation.html

```
        // plot ploygon
        function drawPolygon(ctx){
          ctx.save();

          ctx.strokeStyle = mColorPolygon;
          var r = mRadius/ mCount; //unit radiu
          //画6个圈
          for(var i = 0; i < mCount; i ++){
              ctx.beginPath();
              var currR = r * ( i + 1); //current radiu
              //画6条边
              for(var j = 0; j < mCount; j ++){
                  var x = mCenter + currR * Math.cos(mAngle * j);
                  var y = mCenter + currR * Math.sin(mAngle * j);

                  ctx.lineTo(x, y);
              }
              ctx.closePath()
              ctx.stroke();
          }

          ctx.restore();
        }

      //Draw lines
      function drawLines(ctx){
          ctx.save();

          ctx.beginPath();
          ctx.strokeStyle = mColorLines;

          for(var i = 0; i < mCount; i ++){
              var x = mCenter + mRadius * Math.cos(mAngle * i);
              var y = mCenter + mRadius * Math.sin(mAngle * i);

              ctx.moveTo(mCenter, mCenter);
              ctx.lineTo(x, y);
          }

          ctx.stroke();

          ctx.restore();
      }
```

5) According to the above step, the results are shown on the recommendation page.



10. **Explain your two advanced functions and why they are considered as advanced**. Being able to do it is very important both in the report and final presentation.

1) Box office prediction using linear regression

In this function, we considered different aspects of movie such as genres and rating system to build a linear regression model, we then evaluated the model based on mean square error and R^2 score. Once related information is inserted into our database, predicted box office can be calculated based on existing model.

As for the importance of this functionality, looking from the perspective of movie industry, being able to predict box off based on known information is essential for movie development. As one of the most commonly-used model, linear regression can achieve relatively high performance under simply implementation requirements.

2) Movie aspects visualization

In this function, for each movie in our database, we considered related award count, voting system or other import factors cross different database entities to find out the relationship between movie box office and these key factors. Specifically, a radar chart of user-specified movie will be shown after internal data preprocessing steps (e.g. data normalization).

As for the importance of this functionality, it can provide user a more intuitional way to know the relationship between movie box office and other key factors. This function integrates data not only from movie table but also from direct and actor table to provide more general information.

11. **Describe one technical challenge that the team encountered**. This should be sufficiently detailed such that another future team could use this as helpful advice if they were to start a similar project or were to maintain your project. **Say you created a very robust crawler - share your knowledge. You learnt how to visualize a graph and make an interactive interface for it - teach all! Know how to minimize time building a mobile app - describe!**

1) Data crawling process

When trying to get the original actor and direct information from IMDB website, the URL for these information changes sometimes from one pattern to another pattern, and only one pattern is useful for our project. We tried several times to come across the right URL and then crawl the data we want.

2) Data preprocessing

Crawled data sometimes have empty value or special characters (i.e. identical person name with different serial number). We didn't check the data clearly before saving data into database, therefore there were some errors when running the server, some debugging time was spent to identify the actual bug, and we have to go back to clear the crawled data, some error handling states were also added to run the server successfully.

3) Django framework challenges

Since all the team members have no experience in front-end and back-end development, the main challenge we faced is learning how to use Django at the beginning. And we also faced some tricks in Django, for example, defining foreign key in an object is different from regular process. Another example is about passing raw query into Django for binary operations of Movie and Director objects, we can only compile the program by using "Director.objects.raw([query])" rather than "Movie.objects.raw([query])".

12. State if everything went according to the initial development plan and proposed specifications, if not - why?

In general, things went according to the planned development, we get the crawled data, integrated them into database, then conducted some data analysis. The attributes for each entity reduced slightly after we considered the accessibility and usefulness of these attributes. One of our advanced function changed from movie recommendation to data visualization, since we didn't implement a user login system and we think the overall implementation procedure of recommendation would be the similar to that of prediction, so we changed our direction to another direction - movie aspects visualization.

13. Describe the final division of labor and how did you manage team work.

Final labor division:

- Yekai Chen: front-end and backend development, basic function
- Chin-Han Lin: front-end and backend development, basic function
- Shuiling Yu: front-end and backend development, advanced function (prediction)
- Zheying Lu: web-crawling, advanced function (visualization)

We use Github as the team collaboration tools for our project, detailed code can be found at https://github.com/CYKAKAKA/CS411_Project