Progress Report

Recasting Movies Using IMDB Data

Gaurav Mishra , Piyush Khemka , Pulkit Sharma

110828660 , 110828688 , 110900867

{gamishra,pkhemka,pusharma}@cs.stonybrook.edu

1 Background

Problem Statement Given an input movie, can we suggest a suitable replacement for the lead actor, if the movie were to be remade in the year specified by the user.

The method we had proposed was to represent each actor as a feature vector and then cluster them using the same. Then find the most suitable replacement in the cluster based on age and genre fit (details outlined in the proposal attached) and suggest best five replacements.

2 Data

IMDB provides data of actors and movies in form of text[1] files, which in its raw form is not friendly for the task at hand of retrieving information of movies and actors. To make it usable, we used IMDbPY[2], an open source library to dump data from the text files into a SQL database.

2.1 Extraction

To construct the feature vector of each actor for our baseline model, we wanted to extract their age, height, average rating of all their movies and average number of votes.

The SQL database comprised of 21 tables with each table containing very niche data such as the table `name` containing only the names of actors, the table `person_info` containing biographical information of all the actors and the table `cast_info` containing movie ids and the ids of people who worked in it.

To get the data in our desired format, we wrote multiple SQL queries to fetch data from the above mentioned tables. Example queries are pasted below:

```
INSERT INTO movie_info_pruned_movie
(select * from movie_info where info_type_id = 8 and info = 'USA'
and
movie_id in (
Select id from title where
kind_id = 1
));
```

Listing 1: This query creates a table of only movies made in USA. This was done in order to save time on complex multiple joins.

```
Select avg(temp.info), temp.person_id
From
( select
CI.person_id,
CI.movie_id,
CI.nr_order,
CI.role_id,
CI.note,
```

[1] `http://www.imdb.com/interfaces`

[2] `http://imdbpy.sourceforge.net/`

```
MI.info_type_id,
MI.info
from cast_info as CI, movie_info_idx as MI
where CI.movie_id = MI.movie_id
and MI.info_type_id = 101
and CI.person_id in
( select distinct person_id from cast_info where nr_order = 1 and role_id = 1 and
  note is NULL and movie_id in (select movie_id from movie_info_pruned))
and CI.nr_order = 1
and CI.role_id = 1
and CI.note is NULL
) as temp
group by temp.person_id;
```

Listing 2: This query fetches the average rating for all the movies made in USA for a lead actor

For our baseline model we only considered movies which were produced in USA. We found this to be a good filter to weed out movies made in languages other than English. We used this, instead of explicitly filtering over the language of the movie, because foreign movies are often dubbed in English and this was heavily skewing our model.

We also considered only top billed actors for our model because IMDB provides data for the complete cast and crew. This includes actors who only appear in the background of a scene for a small amount of time. As expected, the number of such actors are much more than the handful of A - list stars. Naturally, a test user of our system won't be happy to see his favourite movie star being replaced by an Extra[3]. Therefore we filtered our query on the basis of billing position and extracted only the top billed actor for each movie. This reduced our corpus to roughly 8000 actors.

We also considered only those movies which had theatrical releases i.e. we removed TV movies, TV shows and award shows from our dataset.

2.2 Data Cleaning

To make the data consistent, we had to do a lot of data cleaning. For instance some of the actors had their heights listed in cms while the others had it in inches. We converted all of them to cm to bring them to the same scale.

Date of Birth also required cleaning. Some actors didn't have their Date of Birth listed in IMDB while the others had it listed in inconsistent format such as 30-12-1980 vs 30/12/80. We extracted their current age by subtracting their year of birth from 2016.

2.3 Data Normalisation

Our features - age, height, average rating and average voting all had different scales and range. We had to normalise it to bring them to the same scale. To do so, we converted all of them to their z-scores and used that for our baseline.

3 Baseline Model

We took only four features for our baseline model. Age and Height represents personal metadata for each of the actors while average rating and average number of votes (Votes is the number of people who rated the movie on IMDB) is a representation of the quality of work and popularity of the actor respectively.

We clustered the actors using K-means with the aid of our feature vector. This yielded a cluster of actors who were similar to each other. To find the best replacements of the actor, we used a Euclidean distance metric to find a list of top actors who were nearest to the given actor in the cluster

Age Fit: As mentioned in our proposal we applied age fit on top of our distance function. After finding the list of most similar actors, we refine our results further by suggesting only those actors as

[3]background actor or extra is a performer in film who appears in a non-speaking capacity usually in the background

replacements who satisfy the age criterion of the role being recast. For instance if we recast Inception in the 1960s, the suggested replacement must be of a similar age group to Leonardo Dicaprio's age at the time of Inception's release. This is because we cannot suggest a replacement for Leonardo Dicaprio in the 1960s who wasn't even born then.

```
kmeans_model = KMeans(n_clusters = 5, random_state = 1)
y_pred=kmeans_model.fit_predict(vec)
center= kmeans_model.cluster_centers_
```

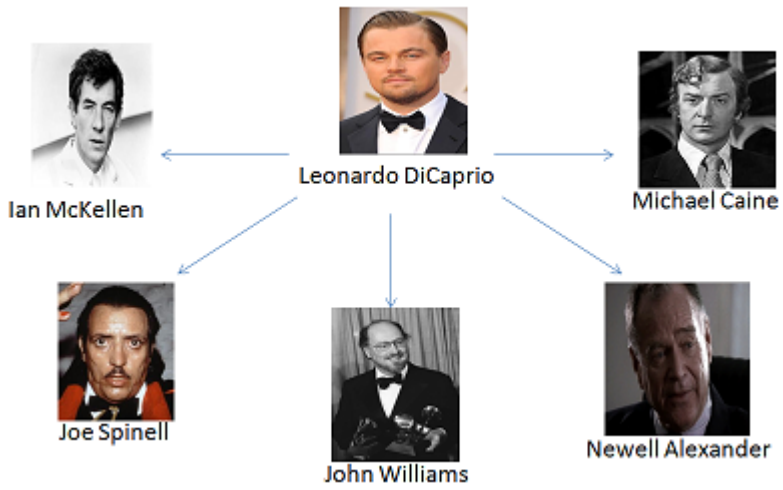
Listing 3: This code snippet does the k-mean clustering of the actor feature vectors.

```
# Convert to arrays
z = np.array(a)
# Apply euclidean distance formula on all the variables and get the values in
  squareform
d = squareform(pdist(z, 'euclidean'))
# Make a dataframe out of the result for easy viewing
d = pd.DataFrame(data = d,columns=[master["name"]])
d.index = master["name"]
# 0.00 is the distance of one country with itself. So, it is replaced with NaN.
d.replace(to_replace=0.0000,value=np.NaN,inplace=True)
d.head()
```

Listing 4: Code for finding the nearest actors within a cluster

4 Results

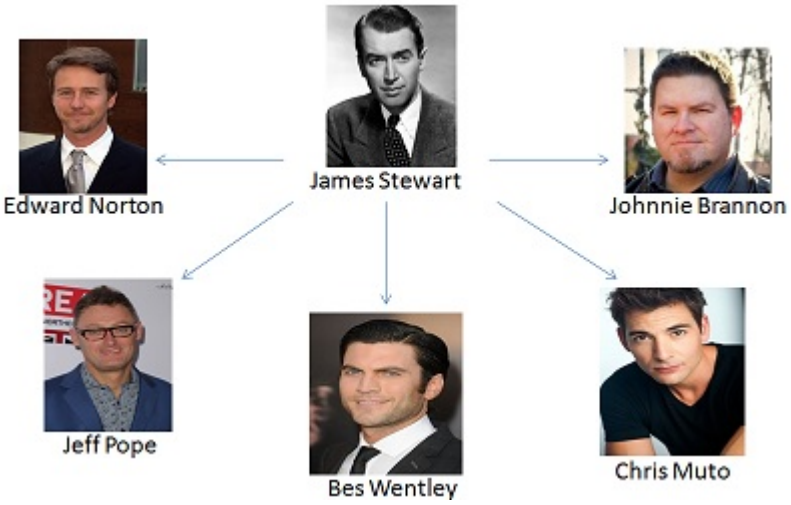
4.1 Remake 'Inception' (2010) in 1960.



Good Results: The result includes actors like Ian Mckellen and Michael Caine. The filmography and stature of both the actors makes them suitable for the role of Leonardo DiCaprio in Inception in 1960.

Goof ups: The results also include some unknown actors like Newell Alexander, John Williams and Joe Spinell who do not share a great deal of similarity with DiCaprio. This is likely due to missing features in our feature vector like - Gross of the movies they act in, whether they act in high budget or low budget movies, number of awards won etc. This problem should get corrected in our final model when we include these features.

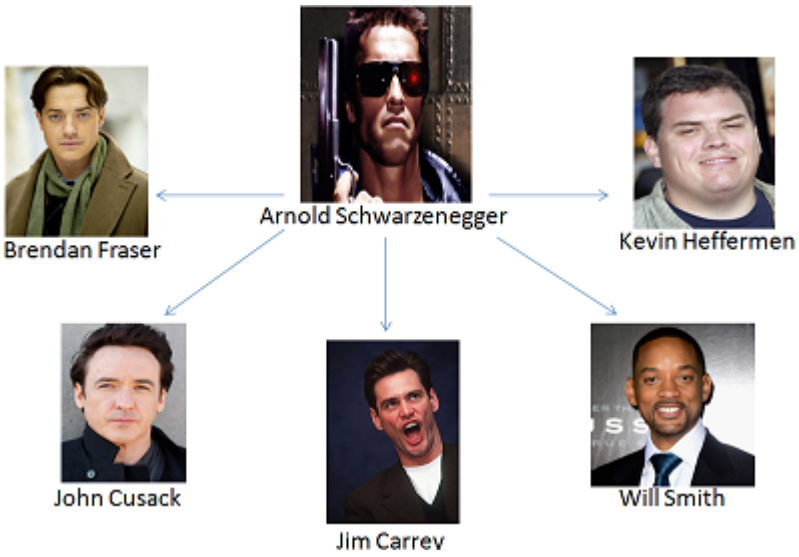
4.2 Remake 'It's a Wonderful Life'(1946) in 2016.



Good Results: Edward Norton and Wes Bentley are good outputs. They are both versatile actors who would be good replacement for James Stewart's role.

Goof ups: The other three results are unknown actors.

4.3 Remake 'Terminator'(1984) in 2010.



Good Results: Our baseline model suggests Will Smith as a probable replacement which seems satisfactory considering his popularity. Though it can be disputed that he isn't of the same build as Arnold and other actors who are more muscular could be better replacements.

Goof ups: The model finds Jim Carrey to be similar to Arnold Schwarzenegger. This is a very bad result considering that Jim Carrey predominantly acts in Comedy movies. The goof up is due to the fact that both are of similar age,height and even their movies have similar average ratings. Since genre is not currently taken into account, they are found to be similar. Once genre fit is taken into account, this problem should get corrected.

5 Future Work

1. **More Features:** We plan to add more features to our feature set. The following features will help us make our model even better-

   - Average gross of all movies
   - Average budget of the movies
   - Average Salary
   - Awards won
   - Genres of movies acted in

2. **Genre Fitness:** Apply genre fit on top of distance function. Similar to age fit, genre fit will be applied to ensure that the actors who have acted in similar movies in the past will be suggested as replacements. As seen in one of the current results, Jim Carrey was suggested as a replacement for Arnold Schwarzenegger. This problem will be corrected on applying genre match.

3. **Actress:** Extend the above system to work for actresses too. Currently, the baseline is trained for male actors only.

4. **Graph Embeddings:** Use Graph Embeddings to improve our model and use it to better visualize results.