STN Project: ’2 Degrees’ Movie Recommendation System

Simon Thorogood (s1881004)

Abstract

Community detection in actor collaboration graphs was explored as basis for movie recommendation. A number of techniques for community detection were implemented and compared in the context of this task. Though it was not possible to fully verify the validity of the approach in the time available, high-level characteristics of the approach were examined and directions for useful future research were identified.

1. Introduction

Movie recommendation systems typically take as input what a user has watched previously (and possibly also up-voted or rated positively) and finds other movies that they might also enjoy based on some measure of similarity. For online video services such as Netflix or Amazon Prime, they are a vital tool for driving the retention of user.

In this project, we explored an approach to recommendation whereby movies are matched based on relationships between the actors that appear in them. Specifically, we hypothesise that the detection of communities of actors in collaboration networks could yield a useful and potentially novel measure of film similarity for the purpose of recommendation.

A community (or cluster) can be defined as being a set of nodes within a network that are more density connected to one another than they are to other nodes within the network. Community detection is a popular topic in network research and many approaches to the problem exist. As part of this project a number of techniques were evaluated with regards to the recommendation problem outlined above.

The project was undertaken jointly between myself, and 2 other students: Sitthinut Kumpalanuwat (s1773493) and Cecilia Cobos Santes (s1826598). In terms of the breakdown of tasks, I was responsible for the initial data extraction and analysis of the collaboration graph, for implementing a clustering method based on DBSCAN and for the development of a recommendation engine that utilised the communities detected in the graph. I also developed an alternative recommendation system based on the Personalised Page Rank algorithm.

2. Related Work

The use of clustering in recommendation task is well-known (Pham et al., 2011), (Mittal et al., 2010) but to date we are not aware of any second-degree content-based approach similar to the one proposed here. Personalised Page Rank has also been applied to the task of recommendation previously (Haveliwala) though again, in a quite different context to the one here.

3. Approach and Methods

The approach taken can be summarised as follows. If a user has watched film **A** staring actor **X**, it should be possible to identify other actors (e.g. actors **Y** & **Z**) who are not in the original film but who share a detected community with actor **X**. These related actors can then be used to identify and recommend new films to the user based on the movies that they have appeared in.

A snapshot of the IMDB dataset (IMDb.com, 2018) was selected for the project. The dataset is updated daily and contains details of over 5 million titles including details of the cast and crew involved in each title. Actors are recorded up to a maximum of 10 per title.

In order to produce an actor collaboration graph of a managable size on which to perform community detection, the following processing and filtering steps were first carried out.

The titles dataset was filtered to exclude non-movie titles (e.g. TV Shows). The date range was also limited to reduce the number of titles to be considered. The cast and crew dataset was filtered to exclude non-acting roles (e.g. directors) and only actors with credits for at least 3 movies were considered. After applying these constraints, a set of actor-actor collaboration graph edges was extracted including a count of the number of movies in which they co-starred. Two graph edge datasets were produced:

1. *movie_star_edges_80s*: A larger set containing edges derived from all movies released in the 1980s. This dataset contains 12,412 nodes (actors) and 121,546 edges
2. *movie_star_edges_1980*: A smaller set used for prototyping containing edges from a subset of movies released in 1980. This dataset contains 3,906 nodes (actors) and 10,036 edges.

NetworkX was used to import the edge data and construct a graph. An initial analysis of the larger *movie_star_edges_80s* dataset (see *Graph Analysis.ipynb*) revealed the following insights. The graph consists of a large connected component consisting of over 98% of the nodes and a number of small disjoint components. The large component was used for all clustering tasks. The large component had a density of 0.16% and an average clustering coefficient of 33% (a result that is significantly lower that that reported in (Watts & Strogatz, 1998) for a similar actor collaboration data source). The diameter of the largest component was 14. The node degrees exhibited a power-law-like distribution.

There are many different approaches that can be taken to community detection in networks. For the purposes of the project, we selected 3 non-overlapping (i.e. partitioning) methods to compare. Each project team member focused on one approach. The methods selected were: Girvan-Newman (Girvan & Newman, 2002); Modularity-based methods e.g. (Blondel et al., 2008); DBSCAN (Ester et al., 1996).

I focused on the application of the DBSCAN algorithm. Density-based spatial clustering of applications with noise (DBSCAN) is a density-based clustering algorithm that first partitions nodes into being core, border or noise nodes and then constructs distinct clusters from the core and border nodes. In the case of clustering in networks, DBSCAN requires the definition of a intrinsic distance metric derived from network topology. The distance metric selected for use in the DSCAN algorithm was the weighted shortest path between actor nodes where the weight was derived as the inverse of the number of collaborations between actors. To put it another way, the distance between 2 actors decreases with the number of movies that they have co-starred in.

The Scikit-Learn implementation of DBSCAN (Scikit-learn, 2018) was used in conjunction with a pre-computed distance matrix for the nodes in the graph. Batch jobs running on DICE compute clusters were used to address the relatively high time-complexity of some of the calculations required. See *Example - DBCSAN Clustering.ipynb* for an example of DBSCAN clustering.

In order to evaluate the performance of different clustering approaches in a recommendation context, I next developed a prototype recommendation engine that utilised the communities identified in the previous step to make movie recommendations. The operation of the engine can be summarised as follows:

1. For a given input movie find the set of actors that starred in it.
2. For each actors, attempt to identify a community that they belong to and retrieve their closest actor neighbours within the community.
3. For each related actor, find a list of movies that they have starred in.
4. Return the list of movies as suggested recommendations.

The *Example - Recommendation Engine (Cluster).ipynb* notebook provides a simple test harness for the engine.

By way of a contrast to the precomputed clustering approaches described above I also developed a prototype of a recommendation engine based on the Personalised Page Rank algorithm. This algorithm is based on the standard page rank algorithm (Page et al., 1999) but takes as an input a subset of nodes from which random walks in the graph commence. In the context of a recommendation task, the nodes associated with the actors that are in the input movie are used as the starting points for the random walks and the nodes (excluding the input nodes) with the highest resulting page rank are used to find movies to recommend (in this prototype, the top 10 highest ranked actors were selected). This approach can be thought of conceptually as building a personal cluster (or clusters) for the input actors 'on the fly'. The prototype was developed using the NetworkX pagerank implementation with equal probability being assigned to each of the input actor nodes as a starting point and the weights used in the previous approach also being accounted for in the random walk edge selection. The *Example - Recommendation Engine (PPR).ipynb* notebook provides a simple test harness for the engine.

4. Results

In terms of the problem at hand, DBSCAN has some attractive characteristics: It doesn't require the number of expected clusters to be provided in advance and it is robust to the existence of noise in the graph (i.e. data point that do not naturally align with any particular cluster). Conversely, the performance of the algorithm is highly dependent of the selection of hyper-parameters and these hyper-parameters do not account for situations where the density is different in different parts of the graph. The hyper-parameters required are:

Neighbourhood Radius (ϵ) - this parameter determines the maximum distance from a node that another node can be in order to be considered a neighbour.

Minimum Samples - the number of other nodes that must be within the neighbourhood radius in order for a node to be considered 'core'.

A grid search was performed on these hyper-parameters in order to ascertain the optimal values for the recommendation task. The selection of hyper-parameters was informed by the following anticipated desirable characteristics.:

- To avoid very generic recommendations, a large number of small clusters is considered more desirable than for example, 1 or 2 very large clusters.
- As many nodes as possible should be assigned to clusters to allow recommendations to occur, though a level of unassigned (i.e. noise) nodes is acceptable.

The graphs in Figure 1 summarise how the clustering algorithm performs with different hyperparameter settings. It can be seen that using a neighbourhood radius greater than 1 results in almost the entire graph being combined into a single 'mega-community'. On the other hand setting the neighbourhood radius too low results in too many nodes not being classified at all. Similarly, setting the minimum samples parameter too high results a a small number of large clusters being identified.

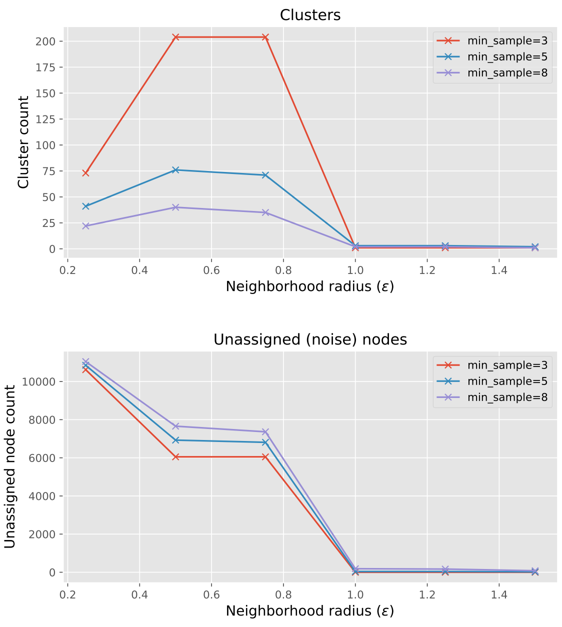

Figure 1. DBSCAN clustering performance with respect to hyperparameters

The optimal values for neighbourhood radius and minimum sample were determined to be 0.75 and 3 respectively, with around 200 distinct communities being returned. It should be noted though that at these 'optimal' values, around 50% of the nodes remain unclassified and that a number of the clusters detected were large in size (> 1000 nodes) and therefore too generic for effective recommendation tasks (see Figure 2).

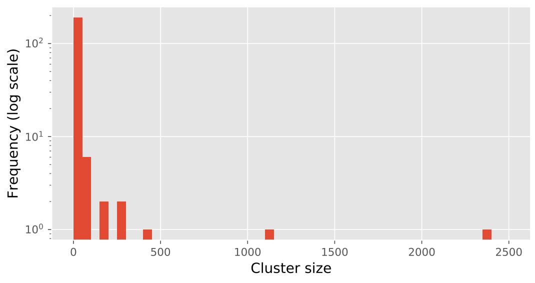

Figure 2. Distribution of DBSCAN cluster sizes for optimal hyperparameters

It can be speculated that the reason for this inability to find a good balance between a small number of unassigned (noise) nodes and a large number of clusters is an insufficient degree of variance in the shortest path distance metric used in the DBSCAN algorithm. This is despite utilising the strength of actor collaboration to define an edge weight.

Recommendation engine	**DBSCAN**	**PPR**
Avg. exc. time	0.02s	8.65s
% no recommendation found	40.0%	17.6%
Avg. number of recommendations	2899.8	711.2

Table 1. Comparison of DBSCAN and PPR recommendation engines. *% no recommendation found* is the % of samples for which no recommendations were found. *Avg. number of recommendations* excludes results with no recommendations

Due to time constraints it was not possible to quantitatively compare all the different cluster techniques developed. However, the 2 engines described in this report were compared in terms of the number of results returned and speed of execution for a random sample of 250 movies. See Table 1.

Both engines can potentially return a lot of results. The PPR engine is able to return results in more cases, which is likely due to the large number of unclassified nodes in the DBSCAN model. Perhaps unsurprisingly the PPR implementation is much slower to execute since it must calculate a set of actors page ranks 'on the fly'. In a real-world scenario though this may be acceptable since recommendations likely change infrequently and could be calculated e.g. once a day.

5. Conclusions & Future Work

Without any meaningful quantitative comparison of the different techniques outlined here, it is hard to draw many conclusions about the approach, however the following areas of future research are proposed.

The number of recommendations can be large and is proposed that work to rank the recommendations would be valuable (i.e. so that the 'Top X' results can be returned). In the clustering case, the strength of actor relations could be used. In the case of PPR, the page rank of the related actors could be used.

To properly test the viability of this approach, it would be necessary to run a multivariate test of the different approaches against a control (e.g. an existing recommendation implementation). Success of a given implementation would likely be measured in terms of the average user interaction with the movies that are recommended.

Another perhaps simpler means of validation would be to compare the recommendations form this approach with those from e.g. a user-simlarity based approach.

References

Blondel, Vincent D., Guillaume, Jean-Loup, Lambiotte, Renaud, and Lefebvre, Etienne. Fast unfolding of communities in large networks. *Journal of Statistical Me-*

chanics: Theory and Experiment, 2008(10):P10008, oct 2008. ISSN 1742-5468. doi: 10.1088/1742-5468/2008/10/P10008.

Ester, Martin, Ester, Martin, Kriegel, Hans-Peter, Sander, Jörg, and Xu, Xiaowei. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. pp. 226—-231, 1996.

Girvan, M and Newman, M E J. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–6, jun 2002. ISSN 0027-8424. doi: 10.1073/pnas.122653799.

Haveliwala, Taher H. Topic-Sensitive PageRank. Technical report. URL http://ilpubs.stanford.edu:8090/573/1/2002-6.pdf.

IMDb.com, Inc. Imdb dataset, 2018. URL https://www.imdb.com/interfaces/.

Mittal, N., Nayak, R., Govil, M. C., and Jain, K. C. Recommender system framework using clustering and collaborative filtering. In *2010 3rd International Conference on Emerging Trends in Engineering and Technology*, pp. 555–558, Nov 2010. doi: 10.1109/ICETET.2010.121.

Page, Lawrence, Brin, Sergey, Motwani, Rajeev, and Winograd, Terry. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. URL http://ilpubs.stanford.edu:8090/422/. Previous number = SIDL-WP-1999-0120.

Pham, Manh Cuong, Cao, Yiwei, Klamma, Ralf, and Jarke, Matthias. A Clustering Approach for Collaborative Filtering Recommendation Using Social Network Analysis. *J. Ucs*, 17(4):583–604, 2011. ISSN 0958695X. doi: 10.3217/jucs-017-04-0583.

Scikit-learn. Scikit-learn dbscan, 2018. URL https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

Watts, Duncan J. and Strogatz, Steven H. Collective dynamics of 'small-world' networks. *Nature*, 393(6684):440–442, jun 1998. doi: 10.1038/30918.