Predicting Related YouTube Videos

Shantanu Thakoor, Anchit Gupta, Parth Shah

November 11, 2017

1 Introduction

In this project, we consider the problem of YouTube Video Recommendations. Specifically, given a set of videos, we wish to suggest other videos which are related to them - this could be used, for example, to suggest a new video to a user based on videos they have already liked, or to suggest a suitable additional video for a playlist a user has created. We view this problem in two ways. First, we consider it as a network problem, where given a set of nodes we wish to predict another node that is closely related to all the nodes in our set.

Second, we view our problem as a classic problem in the space of recommender systems, where we are trying to recommend an item to a user based on their preferences. We also review the algorithm actually used in YouTube's prediction of related videos, and see how we may improve on that. Finally, we propose a new method to solve this problem, by combining some of these above approaches and improving on them.

2 Related Work

Prediction as a Network Problem

In this view of the problem we look at the underlying graph structure of the youtube video datatset. Given a set of nodes in this graph we wish to predict related nodes to this set.

Link Prediction [LNK03]

We can approach this problem using a classical Link Prediction formulation as described in [LNK03]. In this setting the authors have studied a wide range of mostly heuristic based methods to assign scores to potential edges (u, v). We list some of the ones with the best performance here.

- **Common Neighbors-** Assign a score directly based on the number of common neighbors u and v have.
- **Adamic/Adar** - This is another metric based on the neighborhood sets $N(u), N(v)$. $\sum_{z \in N(u) \cap N(v)} \frac{1}{\log |N(z)|}$, the intuition being that common neighbors with sparse connections need to be weighted heavily as there edges "mean" more.
- **Katz all path metric-** It does a damped sum over all paths between u, v. $\sum_l \beta^l |paths^l_{u,v}|$, where $paths^l$ is the set of all length l paths. So longer paths being weighted less and less.

Though these methods were shown to work well empirically on the citation graph, most of them are computationally very inefficient and also the heuristic based approach might not be able to capture some of the finer features of the network.

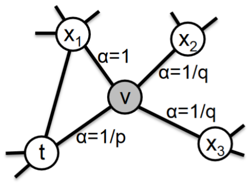
node2vec [GL16]

Another way to do this would be using a system such as node2vec which could theoretically learn much more complicated features than the ones talked about above [GL16]. node2vec basically allows us to learn fixed-length feature representations of nodes in a graph.

It learns a mapping from the set of nodes to a lower dimensional real vector space $\mathbb{R}^d$ such that the likelihood of recovering the neighbors of a node from these vectors is maximized. Their key insight lies in the ability to define a objective function optimizing which gives us the vector

representations. They use a set of biased random-walks to generate the neighborhood of a node. These can be tuned using parameters which defines a notion of a neighborhood for our particular graph.
They extend the Skip-gram model to networks to define a objective function over the neighborhood of a node. They construct the neighborhood of a node in a novel way using second order random walks. Consider a walk that has just traversed (t, v) and is now at v, it transitions using a edge (v, x) with probability $\pi_{vx} = \alpha(t, x)w_{vx}$ where w_{vx} is the edge weight and

$$\alpha(t,x) = \begin{cases} \frac{1}{p} & \text{if } d_{tx} = 0 \\ 1 & \text{if } d_{tx} = 1 \\ \frac{1}{q} & \text{if } d_{tx} = 2 \end{cases}$$



where d_{tx} is the length of the shortest path between t and x. The parameters p and q can be tweaked to influence how far away the random walk would explore and approximately interpolates between a Depth First and Breadth First search. Using these generated neighborhoods we are now able to compute the gradients for out objective function and use SGD to optimize it.

As the above model is trained to predict neighborhoods it is particularly useful in our setting where we would like to predict related videos which might not have been connected by a edge already.
The node2vec is a really novel approach to learn embeddings in a unsupervised fashion, unlike most of the previos approaches it is able to learn representation with impressive performance on down-stream tasks without too much hand engineering. The analysis they provide of the paramter senstivity of the algorithm is also insightful and is often omitted in many ML papers.

In our work we will use node2vec and also investigate whether using some of the heuristic scores from [LNK03] or other state of the art link prediction methods helps us in speeding up learning or in performance.

Prediction as a Recommender System Problem

We review the survey on recommender systems as published by [ASA+10], and see how they can be applied to our setting. Formally, we are given a set of users U, a set of items I, and a utility function $u : U \times I \to R$ giving the utility of each item for each user. The task is to extrapolate the function u from a small set of values. Traditional recommender systems can be classified into the following approaches:

- Content Based Recommender Systems: Here, a profile is created based on each user and each item. In our setting, it could be based on video data like category, length, or uploader, or on user data like the genre of videos a user enjoys. Then, given a user each item is given a score based on how well the profiles of the item and user match. Finally, items are ranked according to this score.

- User Collaborative Filtering Systems: Here, we start with utility function values for a set of users and items. Then, when evaluating the value of the utility function for a new user c and item i, we return the a weighted average of $u(c', i)$ for all other users c'. The weights used here are the degree of similarity between c and c' - we should give more weight to users who have tastes similar to c.

- Item Collaborative Filtering Systems: This is similar to the user based collaborative filtering system as detailed above. However, here we use a weighted average of $u(c, i')$ for all other items i', again based on the similarity between i and i'.

For our setting, we believe item based collaborative filtering is the most natural choice. Given a set of videos, we would model it as trying to recommend new videos for a user who has already rated highly all the videos in our set.
We believe off-the-shelf item based collaborative filtering recommender systems would not work well on our problem setting. The key component here is how we calculate the similarity between videos. If we simply use video data like uploader or category, then we may be losing information contained in the rich structure of the network.

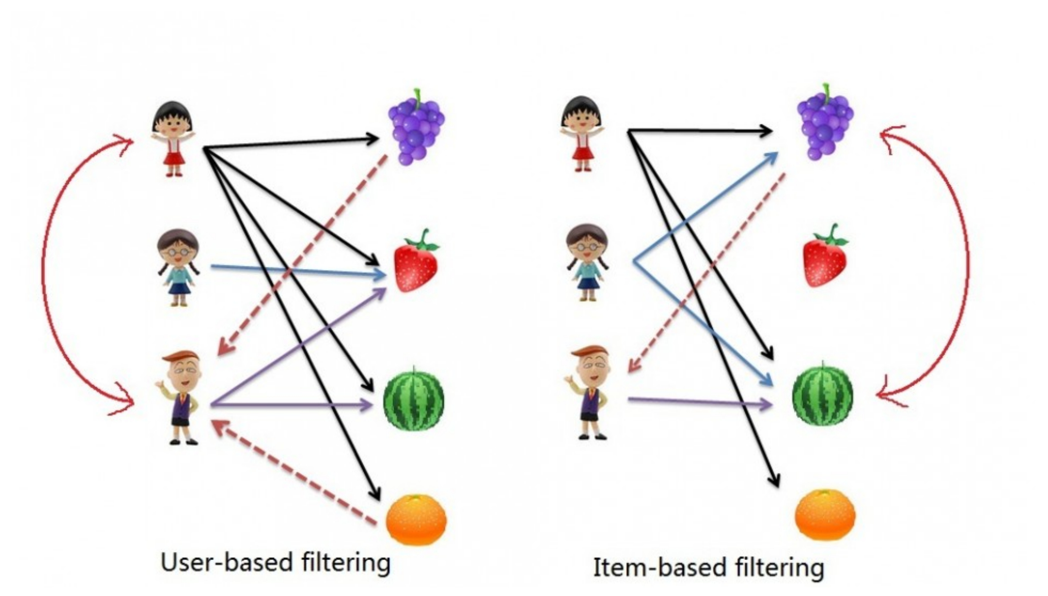


Figure 1: A pictorial representation of collaborative filtering from *salemmarafi.com*

Another problem with this approach is that it does not scale well with a very large number of items, without precomputing a quadratic number of similarity values. Clearly, this would not scale for a dataset the size of YouTube, nor would it work well considering that new videos get added to the platform every day. Hence, we may also try to extend this method to use less precomputed values, and hence scale well to a large setting.

YouTube Video Recommendations

Youtube, one of the most popular video sharing platforms, has its own state of the art recommendation system incorporated on its website. For completeness, here we review one of the papers detailing its design and implementation.

The work done by [DLL+10] details one of the approached used for the recommendation engine. The system is learned from data containing user browsing sessions. For a given pair of videos (v_i, v_j), the system uses the normalized covisitation counts to measure their relatedness score $r(v_i, v_j)$. Here c_{ij} is the covisitation count, ie the number of sessions in which both the videos were watched together, and $f(v_i, v_j)$ is a normalizing factor based on popularities of v_i and v_j.

$$r(v_i, v_j) = \frac{c_{ij}}{f(v_i, v_j)}$$

Based on this relatedness score, they predict the top related videos for a given video, R_i. To predict a recommendation for a user, they begin with a seed set S which is based on the user activity, the videos he has liked, has seen or marked favorite. Given this, they construct the related set based on the union of related sets for each video in the set S.

$$C_1(S) = \cup_{i \in S} R_i$$

They further consider the possibility that the set of recommended videos may not be very diverse since they are all closely related. To broaden the recommendation, they also consider videos which are some distance away - by iteratively computing $C_i(S) = \cup_{j \in C_{i-1}} R_j$, and returning C_n as the final set of recommendations. These videos are then ranked based on the several signals from the user, the video quality, and constrained to provide a small diversification factor.

While our setting is similar, it is not exactly the same. For example, we do not consider the problem of diversification in our recommendation set. Further, we make predictions based only on information about each video, and a small dataset of video relatedness information - we do not assume knowledge of user browsing histories for every pair of videos in our dataset. Hence, we review this work for completeness but do not directly extend or compare against it.

3 Project Proposal

Extending Existing Systems

We believe that a combination of the two approaches detailed in the previous section, may be a good method for this problem.

Node2Vec to improve accuracy: One way to represent the set of videos for which we wish to find related content, would be the average of the node2vec transformations of each node. We expect for this vector to capture to commonalities between all the videos. We can then either train a deep neural network classifier on top of this to output probabilities of other videos being related, or output a vector and take the closest video as our related video.

Node2Vec to improve scalability: (i) Traditional recommender systems lose network information which we believe are a crucial part of the problem. Also they are either very slow or require a large number of values to be precomputed and saved. Now instead of precomputing a quadratic number of values for similarity between videos, we train a neural network to compute the similarity between two videos based on their node2vec vector representations augmented with potentially other useful meta data from the dataset. This would require precomputing only a linear number of values, and can also be computed on the fly quickly. We would also have the added advantage of not losing the network information. (ii) A further tweak for speed up would be first doing community detection, in order to find a smaller subset of nodes which are likely to be related to one another. We could then run our system calculating similarity only on these nodes. This would further reduce time taken for inference, while not penalizing results much since we expect related nodes to come from the same community (which may mean genre of videos, videos of same uploader, etc).

Dataset Description

We use the dataset provided by [CDL08]. This network consists of information regarding a large number of videos - on the order of 1 million. Each video has the following data associated with it:

- Username of the uploader
- Number of day between the date the video was uploaded and the date of crawl
- Video category, as chosen by the uploader
- Number of views
- Rating of the video
- Related IDs: up to 20 IDs of related videos

The network we analyze will be built from this data, with each video being a node and there being an edge from video u to video v if video v is present in the related video list for video u.

Analysis and Evaluation

We intend to use following metrics for analyzing the network and evaluating the performance of our system:

- Prediction of Deleted Edges: We will pick a random sample of nodes from the original graph, and remove some of their outward edges. We will evaluate our system based on the degree of overlap between the removed edges, and the edges we deleted.
- Distance from the seed set: Given a seed set, our system return a set of related videos from the corpus. Intuitively, we expect that related videos should be at a small distance from our seed set in the graph. Measuring the average distance of a related video from our original set would be a good way to see how well our system captures the notion of relatedness.
- Category Inspection: Since our dataset contains information for each video including the video category, we expect that our recommendations should mostly belong to similar categories. We will analyze in what cases a set of videos all of the same category, has a related video from a different category. This may help find some unintuitive links between categories or communities.

- Comparisons: We will try to compare our results with results obtained only node2vec, or collaborative filtering systems without using network information, to better understand how combining the approaches can improve quality of recommendations.

References

[ASA$^+$10] Dhoha Almazro, Ghadeer Shahatah, Lamia Albdulkarim, Mona Kherees, Romy Martinez, and William Nzoukou. A survey paper on recommender systems. *CoRR*, abs/1006.5278, 2010.

[CDL08] Xu Cheng, Cameron Dale, and Jiangchuan Liu. Statistics and social network of youtube videos, 07 2008.

[DLL$^+$10] James Davidson, Benjamin Liebald, Junning Liu, Palash Nandy, Taylor Van Vleet, Ullas Gargi, Sujoy Gupta, Yu He, Mike Lambert, Blake Livingston, and Dasarathi Sampath. The youtube video recommendation system. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 293–296, New York, NY, USA, 2010. ACM.

[GL16] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. *CoRR*, abs/1607.00653, 2016.

[LNK03] David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management*, CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.