YouTube Video Recommendations: Analysis and Link Prediction

Luyao Hou
Department of Computer Science
Stanford University
luyaoh@stanford.edu

Emma Zhong
Department of Computer Science
Stanford University
emmazjy@stanford.edu

Abstract

Link prediction is a common problem in networks and has broad applications, ranging from predicting unobserved interactions to recommending related items. In this paper, unlike most previous works that have focused on social networks, we investigate link prediction over YouTube video network. More specifically, given a specific query video in the graph, we want to find videos that are most relevant to the query. We start by introducing existing work in link predictions and some supervised models as our baseline. We then talk about a weighted random walk algorithm that predicts top related videos for a given video in the network.

1 Introduction

YouTube is a online video community where users can upload and share videos with others. When a user is watching a video, YouTube will recommend related videos for the user to watch next and the quality of such video recommendations directly influences how long a user stays on the site.

Video recommendation is important because: (1) When a new video is uploaded, it has few connections with other nodes in the network. Making an accurate prediction in this case will help improve the video's discoverability and help users navigate through the site. (2) Like social media, if a user likes a particular type of video, then it is likely that he or she wants to watch other videos of the same type as well. Recommending the right type of videos to watch next can help improve users' experiences and help them more efficiently find contents they want to watch.

Video recommendation is also challenging. First of all, there are a huge number of videos uploaded everyday and the video network is changing constantly. As a result, we need a recommendation system that adapts to the dynamic nature of the network and also scales to a graph with billions of nodes. Moreover, length of individual videos and the diversity of videos are also growing, which means that it is harder to make predictions by purely analyzing video content.

In this paper, we build upon the idea of Pixie Random Walk by Eksombatchai et al. [4] to make link predictions for videos in the YouTube network. More specifically, given a video network as undirected graph where each edge denotes a pair of related videos, we first remove a subset of edges from the graph for testing. Then, for each node in the graph, we rank all other nodes by their relevance to the query node, and measure the fraction of times where true related videos are ranked among top K. We think random walk techniques are suitable for video recommendation because: (1) It does not require pre-training and can always run on the newest version of the network. (2) It utilizes the property of video network that most related videos are close to each other in the network. Therefore, the random walk only needs to explore the neighborhood of a node, which lets it run faster. (3) It mostly relies on the structure of the network and is thus more reliable than video content analysis. In the end, we find that the weighted random walk model achieves twice the accuracy than our baseline supervised models. We also evaluate the effect of longer walks and weighted walks based on individual video attributes.

In the rest of the paper, we will introduce existing work in the field of link prediction in section 2. In section 3 we discuss our data collection process and a brief analysis of the YouTube network. We then talk about our methods and results in section 4, 5 and 6. We conclude with possible future directions in section 7.

2 Related Work

There has been various methods proposed and studied for link prediction. In this section we discuss some of these methods and why we pursue the weighted random walk model.

Proximity Measures. In their paper, Liben-Nowell et al. [1] presents and examines methods for the link prediction problem based on network proximity measures adapted from different techniques. The techniques include ones based on neighborhoods, on the ensemble of all paths as well as some higher-level approaches. The authors found that certain subtle measures of the network for detecting node proximity can often outperform more direct measures such as shortest-path distances. Even though such methods are fast, however, they tend to achieve low accuracy as they rely on a single measure which may not be effective for certain graphs.

Supervised Learning. Al Hasan et al. [2] examined link prediction in co-authorship in biology and computer science using supervised learning. The authors examined various features for each node ranging from proximity features like keyword match count, aggregated features like sum of papers to topological features like clustering index. They also compared the performance of different models, including SVM, K-nearest neighbors, Naive Bayes on both datasets and found SVM to have highest accuracy on the two datasets they worked with. Despite the improved accuracies, supervised learning methods can be hard to scale to large graphs as the training time grows with more samples. It can also take longer to find the best supervised learning model as such models usually involve more hyperparameters. Moreover, supervised learning methods always require labeled features, which may not be suitable for ranking problems.

Flow-based Approaches. Lichtenwalter et al. [3] presents in their paper an effective flow-based algorithm as well as a general framework for the link prediction task after investigating issues such as sampling approaches, generality of existing methods and pitfalls these methods have encountered.

The authors introduced a new unsupervised prediction method PropFlow based on the probability of a restricted random walk between two nodes in less than l steps. After investigating in methods of constructing dataset and reducing variances, the paper comes to the realization that supervised learning is suitable for the task of link prediction. The framework introduced takes advantage of supervised learning to reduce variance and overcomes imbalance through undersampling. This framework can be applied to any domain without requiring any domain-specific node attributes, and outperforms baseline methods by 30% in terms of AUC.

Weighted Random Walk. In their paper, Eksombatchai et al. [4] presented their work at Pinterest including Pixie Random Walk algorithm, used to generate personalized recommendations for users. This work addresses two big challenges at Pinterest: large size of the dataset(billions of pins and boards) and real-time requirement of recommendations. The input of the algorithm is formulated as a bipartite graph of pins and boards. The algorithm biases the random walk in a user-specific way and allows for multiple dynamically generated query pins with different importance weights to provide entire context for the user's previous behavior. Besides that, special convergence criteria and graph pruning are used to achieve real-time performance.

Both high-quality recommendations and high performance can be achieved using Pixie Random Walk. The authors compared performance of Pixie with two state-of-the-art deep learning content-based recommender methods, and concluded that Pixie was able to generate much better recommendations for all values of hit rate. Overall, Pixie takes less than 60 milliseconds to generate high-quality recommendations. We find that similar weighted random walk approaches can be helpful for video link predictions because they scale to large graphs and incorporate both graph structural features and individual node features.

3 Data Collection

In this section we first talk about the datasets we use and our preprocessing of the data. We then perform analysis on the YouTube network.

3.1 Datasets

The dataset we use is the dataset for "Statistics and Social Network of YouTube Videos" [5]. Each file in the dataset has a list of video ids. For each video in the list, the dataset provides its meta information like number of comments, genre and uploader, as well as 20 video ids that are related to it. In this project, we use data from July, 17th, 2008 to July, 27th, 2008 inclusive.

3.2 Graph Construction

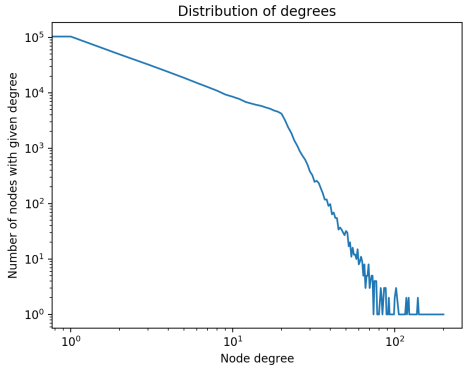
We construct an undirected graph based on the dataset in two steps. During the first step, we add a node for each unique video id in the dataset. We add an undirected edge between two videos if they are related. In total we have 2,188,889 nodes and 4,000,068 edges.

Since the dataset does not give meta information for videos that are in the related video list, in the second step we remove nodes that do not have meta information. After pruning, we have 357,179 nodes and 1,094,417 edges in the graph. Based on our analysis, the pruning process does not affect the structure of the graph much.

3.3 Graph Analysis

We first analyze the degree distribution of our graph, as shown in figure 1. We can observe that most nodes in our graph have degree less than 20. There is a turning point at degree 20 due to the nature of dataset as each video has a related video list of length 20. The number of nodes decreases drastically as degree passes 20 and there are only a few nodes with degree over 100. Even though this graph may not precisely represent the YouTube network as nodes at the end of the video crawl may artificially have degree 1, it still suggests that the YouTube network is concentrated with a small portion of nodes having large degrees.

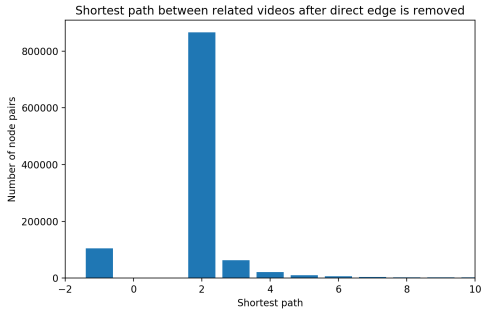
Figure 1: Degree Distribution

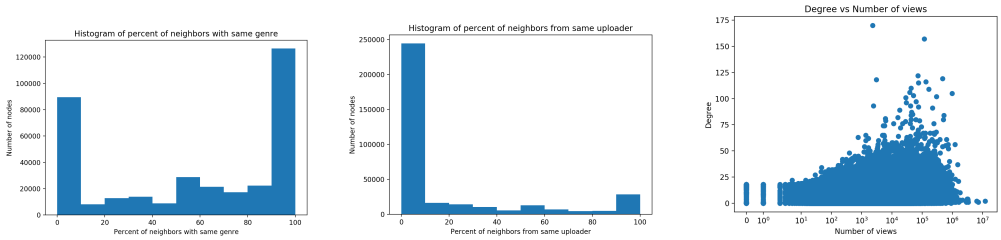


We then analyze the shortest path between a pair of related videos after removing the edge between them. The result is shown in figure 2. Negative shortest path means that the pair of videos becomes disconnected. Note that the majority of related videos have shortest path 2 after the edge between them is removed, which means that related videos often share at least one common neighbor. The result also influences our sampling strategy, as we will discuss in section 4.

In preparation for the weighted random walk, we also analyze the nature of related videos, as shown in Figure 3. Figure 3(a) shows the histogram, across all nodes, of the percent of neighbors with same genre. Even though most nodes fall in the $> 90\%$ bucket, there are large portion of nodes with $< 5\%$ neighbors from same genre, which potentially makes biasing towards nodes with same

Figure 2: Shortest Path





(a) Neighbor with same genre (b) Neighbor from same uploader (c) Degree vs Number of views

Figure 3: Neighborhood Analysis

genre less effective. Figure 3(b) shows the histogram of the percent of neighbors from same uploader. Surprisingly, the majority of nodes have $< 5\%$ neighbors from same uploader, suggesting that the related videos often come from various sources. Figure 3(c) explores the relationship between node degree and node reviews. As the figure shows, there is a slight correlation between node degree and number of views, specifically node with higher number of views tends to have higher degrees as well. We will explore the opportunity of biasing towards nodes with large number of reviews in section 4 and 5.

3.4 Dataset Construction

Our baseline models and random walk model have slightly different ways to construct training, evaluation and test datasets, primarily due to the difference between supervised and unsupervised models. We will thus discuss dataset constructions in section 5.

4 Methodology

In this section we introduce both our baseline models as well as our random walk algorithm.

4.1 Baseline

We use both random forest and SVM as our baselines. We construct a 6-d feature vector for each pair of nodes. In this case, all features we use are network structural features, which include the degree of the first node, the degree of the second node, the Jaccard Index, the Katz Index, The Adamic-Adar Index and the Preferential Attachment Index. We discuss the hyperparameters and evaluation results in section 5. As supervised methods, both random forest and SVM require a large amount of training time and they can be slow at test and evaluation as well. They also do not scale well as the graph and the training set grows.

4.2 Weighted Random Walk

We build upon Pixie Random Walk [4] and adapt the idea of weighted random walk to the YouTube video network. In this section, we introduce the modified biased random walk algorithm, with the following improvements made upon basic random walk.

4.2.1 Biasing the Random Walk

In the original Pixie paper, the random walk is biased by changing the random edge selection to be biased based on user features [4]. This approach is not applicable to our graph consisting of only videos. Thus, we bias the edges based on the neighbor's proximity to the original source video, evaluated using metrics such as similarity of genres and video uploaders, and other structural measures. More specifically, the probability of choosing node q is proportionally to its biasing factor

$$b_q = \Pi_i m_i \tag{1}$$

where m_i is the biasing multiplier for the ith feature. This allows us to prefer edges that match the source video's features, enabling a more personalized random walk. This is implemented in line 9 of Algorithm 1.

4.2.2 Multiple Query Videos with Step Distribution

For each video, we sample a fraction of its neighbors to construct its query set Q. Then we run random walk starting from each video in Q. An important insight from the Pixie paper is that the number of steps required to obtain meaningful visit counts depends on the degree of the query pin [4]. Therefore a scaling factor s_q is introduced for each video q in the query set so that the number of steps assigned to each video increases sub-linearly with its degree

$$s_q = deg(q) \cdot (C - log(deg(q))) \tag{2}$$

where $deg(q)$ is the degree of q and C is the max degree of videos in the query set. The number of steps for each video q is proportional to the scaling factor

$$N_q = N \frac{s_q}{\sum_{v \in Q} s_v} \tag{3}$$

where N is the total number of steps assigned to each source video.

4.2.3 Mixed Teleportation Rate

Inspired by the Personalized PageRank algorithm, we add teleportation to our algorithm so that at each step, the surfer is teleported to one of the neighbors of the source node. We also build upon this idea to allow the surfer to have different teleportation rates during a random walk. The idea is to allow the surfer to spend the majority of time exploring nodes nearby but also be able to visit nodes farther away. We define a list of teleport rate T and split the number steps assigned to each q, N_q, evenly across all teleport rates in T. This is implemented in line 3 of Algorithm 1.

4.2.4 Multi-hit Booster

We adopt this innovation of the Pixie algorithm such that we prefer to recommend videos that are related to multiple videos in the query set. This is achieved by changing the aggregation of visit counts $V_q[v]$ from summation to the following way which reward videos visited multiple times from different query videos

$$VisitCounts[v] = (\sum_{q \in Q} \sqrt{V_q[v]})^2 \tag{4}$$

At the end of the algorithm, we sort nodes in the graph by top visits. We then treat the top visited nodes as related videos to the source node. We will discuss more details in section 5.

Algorithm 1: Modified Pixie Random Walk with teleportation

```
PixieRandomWalk
 (q: query video, G: graph of videos, F: video-specific features, N_q: int, T: teleport rates)
for β ∈ T do
    for i=[1:N_q/len(T)] do
        currentVideo = q
        V[currentVideo]++
        if teleports with probability β then
            currentVideo = chooseNeighbor(q)
        else
            currentVideo = personalizedNeighbor(G, F)
        end
    end
end
return V
```

Algorithm 2: Pixie recommendation for mutiple videos

```
PixieRandomWalkMultiple
 (Q: query videos, G: graph of videos, F: video-specific features, N: int, T: teleport rates)
for q ∈ Q do
    N_q = Equation (3)
    V_q = PixieRandomWalk(q, G, F, N_q, T)
end
for v ∈ G do
    VisitCounts[v] = Equation (4);
end
return VisitCounts
```

5 Evaluation

In this section we first introduce how we construct the training and evaluation datasets for our models. We then evaluate and compare the performances of both our baseline models and our random walk model. We also discuss parameter tuning and the importance of different features.

5.1 Supervised Methods

5.1.1 Dataset Construction

We use the following process to find labeled node pairs where a pair of nodes is labeled positive if there exists an edge between them and is labeled negative otherwise. We obtain positive samples by uniformly randomly selecting edges from the graph, adding it to the set, and then removing the edge from the graph. For negative samples, based on Figure 2, since most related nodes have distance 2 even if the edge between them is removed, our negative samples focus on pairs of nodes where the shortest path between them has length 2. In particular, 10% of our negative samples are constructed by uniformly randomly selecting two nodes from the graph and using the pair if they not directly related. The rest 90% are constructed by selecting a random node and one of its distance-2 neighbors where the two nodes are not directly related. Using this process, we find 360,000 samples for training set, 20,000 samples for evaluation set and 20,000 samples for test set. Each dataset has an even split between positive and negative samples, and has the distribution of negative samples as described above.

5.1.2 Hyperparameters

For SVM, we use a rbf kernel and a max iteration of 50,000. We observe that increasing the number of iterations does not help improve the accuracy much. The SVM model does not converge after 50,000 iterations.

For random forest, we find that 500 trees and a max depth of 5 for each tree gives us the highest accuracy.

5.1.3 Evaluation

To evaluate accuracy and be able to compare the performance with the random walk model, we use a different accuracy measurement than F1 scores. For each source node, since most related videos are close to each other, we first find all nodes that are within 4 hops from the source node. Then we rank all nodes within 4 hops by their probability of being related to the source node in descending order. We then look at the true related videos of the source node and find the percentage of them ranked in top K. We ignore any related videos that are more than 4 hops away. Here the K values we use are 5, 10, 20, 50, 100 and 1000. The results are summarized in table 1 and 2.

Model	K=5	K=10	K=20	K=50	K=100	K=1000
SVM(max-iter=50000)	26.81%	31.27%	34.05%	42.35%	54.95%	94.07%
Random Forest(n=500,depth=5)	24.98%	38.17%	50.06%	60.27%	68.77%	95.95%

Table 1: Evaluation of Baseline Models (Training Set)

Model	K=5	K=10	K=20	K=50	K=100	K=1000
SVM(max-iter=50000)	25.86%	29.95%	32.89%	41.53%	54.00%	94.30%
Random Forest(n=500,depth=5)	24.77%	38.19%	50.01%	60.66%	68.59%	96.08%

Table 2: Evaluation of Baseline Models (Test Set)

The two models achieve similar accuracy when K=1000 but the random forest has much higher accuracy when $K \leq 100$. We also analyze the importance of each feature used by the random forest model, as shown in Table 3.

	Degree1	Degree2	Jaccard	Katz	Adamic Adar	Preferential Attachment
Importance	0.046	0.184	0.074	0.155	0.271	0.270

Table 3: Importance of Features in Random Forest

The analysis shows that the Jaccard index has low importance in the decision trees, while the most important features are all related to node degrees. This suggests that nodes with higher degrees are more likely to be recommended as related videos and the network prefers to recommend popular videos rather than less popular videos even though they may share more commonalities with the source node.

5.2 Weighted Random Walk

5.2.1 Dataset Construction

We use 5% of edges in the graph as evaluation set and 5% of edges in the graph as test set. In particular, we iterate over each node in the graph in random order and for each node, we uniformly randomly remove 5% of its edges for evaluation and 5% of its edges for testing. After the dataset construction process, we have 993,904 edges remaining in the graph.

5.2.2 Evaluation

We test the following three random walk models, where T is the teleport rate array discussed in section 4:

- N = 1000, T = [0.5]
- N = 5000, T = [0.5]
- N = 5000, T = [0.75, 0.25]

For each of these three models, we test the following five biasing strategies:

- No bias
- Bias towards node with same genre, biasing factor m = 1.5.
- Bias towards node from same uploader, m = 1.5.
- Bias towards node with more viewers, $m = \sqrt{log(v+3)}$ where v is the number of views of a video.
- Bias towards unvisited nodes, m = 1.5.

To evaluate accuracy, for each source node n in the graph, we perform random walk following the algorithm described in section 4 with neighbors of the source node as query set. We then rank the visited nodes by aggregating visited counts in descending order. Finally, we look at test edges where one of the end points is n and find the percentage of them that are ranked top K. The K values we use are 5, 10, 20, 50, 100, 1000.

The test results are summarized in table 4 and we include results from baseline here again for comparison. For simplicity we use RW to denote the random walk model.

Model	K=5	K=10	K=20	K=50	K=100	K=1000
SVM(max-iter=50000)	25.86%	29.95%	32.89%	41.53%	54.00%	94.30%
Random Forest(n=500,depth=5)	24.77%	38.19%	50.01%	60.66%	68.59%	**96.08%**
RW(N=1000,T=[0.5])	51.61%	61.89%	71.45%	80.65%	84.69%	87.42%
RW(N=1000,T=[0.5],bias=genre)	51.44%	61.83%	71.34%	80.48%	84.47%	87.19%
RW(N=1000,T=[0.5],bias=uploader)	51.83%	62.11%	71.52%	80.61%	84.63%	87.31%
RW(N=1000,T=[0.5],bias=views)	51.54%	61.84%	71.36%	80.54%	84.55%	87.21%
RW(N=1000,T=[0.5],bias=unvisited)	51.15%	61.64%	71.25%	80.48%	84.65%	87.61%
RW(N=5000,T=[0.5])	53.59%	**63.79%**	73.06%	81.93%	85.89%	89.38%
RW(N=5000,T=[0.5],bias=genre)	53.36%	63.61%	72.99%	81.93%	85.83%	89.30%
RW(N=5000,T=[0.5],bias=uploader)	**53.67%**	63.76%	**73.07%**	**82.08%**	**86.13%**	89.54%
RW(N=5000,T=[0.5],bias=views)	53.32%	63.56%	72.89%	81.85%	85.88%	89.39%
RW(N=5000,T=[0.5],bias=unvisited)	53.34%	63.57%	72.85%	81.85%	85.82%	89.42%
RW(N=5000,T=[0.75,0.25])	53.15%	63.43%	72.73%	81.72%	85.74%	89.40%
RW(N=5000,T=[0.75,0.25],bias=genre)	52.94%	63.22%	72.54%	81.73%	85.78%	89.48%
RW(N=5000,T=[0.75,0.25],bias=uploader)	53.47%	63.59%	72.88%	81.90%	85.91%	89.61%
RW(N=5000,T=[0.75,0.25],bias=views)	52.93%	63.13%	72.56%	81.71%	85.79%	89.52%
RW(N=5000,T=[0.75,0.25],bias=unvisited)	53.21%	63.44%	72.74%	81.80%	85.82%	**89.67%**

Table 4: Test Accuracies (Top K)

First of all, we can observe that when $K \leq 10$, the accuracy of random walk model is about twice of the baseline models and the random walk model always has higher accuracy for $K \leq 100$. This suggests that the random walk algorithm is good at ranking related videos higher in the list. However, at $K = 1000$, both baseline models outperform the random walk model. One explanation is that when evaluating baseline models, we only look at pairs of nodes whose distance are no greater than 4. This helps limit the number of nodes that the baseline models need to rank, which means that they are solving an easier problem. Also, both baseline models are supervised models, which means that they will be able to assign a positive probability to each node, even though the node may be ranked low in the list. In contrast, if a random walk does not visit a node, then the visit count for that node is 0 and the node will never appear in the top K list for any K.

We can also observe from the results that the most significant improvement in random walk accuracy comes from increased number of steps. This makes sense because as the number of steps increases, the visit counts converge to the stationary distribution and become more accurate. Increased number of steps also increases the number of unique nodes visited. In contrast, however, biasing the random walk does not seem to improve the accuracy much. The difference in accuracy between any biasing strategies at any K are at most 0.5%. The model performance is consistent, though, in the sense that biasing towards the same uploader always gives the highest accuracy for $K \leq 100$ than any other tested biasing strategies. This may seem counter-intuitive, because from figure 3(b), most nodes have $< 5\%$ neighbors from the same uploader. One explanation is that even though a pair of nodes may not come from the same uploader, they often have a common neighbor that has the same uploader as one of the edge endpoints and thus that common neighbor acts as a bridge between the pair of nodes.

It also makes sense that at K=1000, biasing towards unvisited nodes gives the highest accuracy because at large K, recall is more important than precision. Having a combination of different teleport rates also does not improve the accuracy. This could be due to the fact that based on Figure 2, most related videos have short distance between each other and having a long walk with lower teleport rate may add unnecessary noise.

5.3 Error Analysis

In this section we explore the following two questions for our random walk models:

- Why is the accuracy at K=1000 not perfect?
- Does the combination of different teleport rates help?

5.3.1 Nodes not in the top 1000 list

To answer the first question, we look at our best performing random walk model (N=5000,T=0.5,bias=uploader) and categorize nodes that should be in the top 1000 list but are not into three categories. We also include the number of nodes in each category.

- Visited but ranked above 1000: **4**.
- Not visited but reachable from source node: **3325**.
- Not visited and unreachable from source node: **7174**.

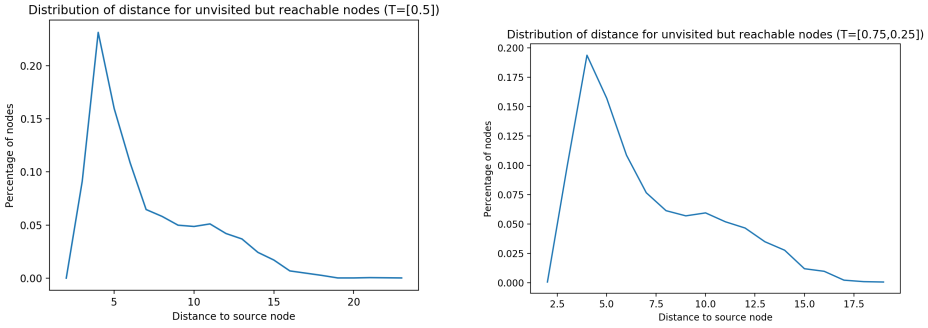
First of all, the results again shows that the random walk model is good at ranking related videos higher because there are only 4 nodes that are visited but ranked above 1000. Also, about 66% of related videos that are not in the top 1000 list are actually not reachable from source node. This reflects one disadvantage of random walk models because disconnected nodes will never be considered. Future work may explore ways to consider disconnected nodes as well, for instance, by randomly teleporting to a node with the same genre. With that said, we think the large number of disconnected nodes are due to the way we construct our dataset. Since the dataset is crawled by BFS, nodes on the edge of BFS tend to have small degrees, as show in Figure 1. As a results, nodes may become disconnected when all of its edges are removed to be test edges. Since the real YouTube video network is more connected, we expect this to be less of a problem in the real network.

For related videos that are not visited but are actually reachable, we plot the distribution of distance between each node in this category and its source node. The result is shown in Figure 4(a). Note that there is a peak around distance 5 because: (1) Most related videos have distance no greater than 5. (2) Because the teleport rate is 0.5, the algorithm will visit most nodes 2 and 3 hops away but cannot cover all nodes that are more that 3 hops away.

5.3.2 Combination of Teleport Rates

Even though during our evaluation, a combination of teleport rates does not help much with improving the top K accuracy, in this section we explore how a combination of high and low teleport rates change the distribution of nodes not in the top 1000 list.

Here we look at the random walk mode (N=5000,T=[0.75,0.25],bias=uploader). Similar to section 5.3.1, we categorize nodes not in the top 1000 list into three categories:



(a) T=[0.5]

(b) T=[0.75,0.25]

Figure 4: Distribution of distance for unvisited but reachable nodes

- Visited but ranked above 1000: **12**.
- Not visited but reachable from source node: **3178**.
- Not visited and unreachable from source node: **7253**.

Despite randomness across runs, we find that the number of reachable but unvisited nodes decreases with the split teleport rates, which suggests that more nodes are visited by the algorithm. We also plot the distribution of distance between each of unvisited but reachable nodes to its source node, as shown in Figure 4(b). Compared to 4(a), even though the general shape remains the same, in 4(b), the peak at distance 5 decreases to below 20% and the distance to farthest unvisited node decreases to below 20, suggesting that having the teleport rate 0.25 does help improve exploration of the algorithm. The fact that all models with T=[0.75,0.25] have higher top K=1000 accuracy than corresponding models with T=[0.5] also supports this conclusion.

6 Conclusion

In this paper we introduce the random walk algorithm to find most relevant videos in the YouTube network. Unlike traditional supervised models, the random walk model can adapt to the rapidly changing network and requires little information from the video contents to work effectively. We show that the top K accuracy of random walk algorithm is always higher than supervised learning models when K is small, suggesting that the random walk model is better at ranking relevant videos higher. We also explore the relationships between number of steps, biasing strategy, mixed teleport rates and the accuracy of random walk model. Specifically, higher number of steps will lead to higher top K accuracy and mixing high and low teleport rates will allow the model to visit more nodes, leading to higher accuracy with large K without sacrificing much accuracy when K is small.

As discussed in the previous sections, future work could focus on strategies to visit disconnected nodes in the random walk algorithm. Another future direction is to build upon the mixed teleport rates and explore how lower teleport rates (e.g. T=[0.01, 0.5]) affects the number of nodes visited by the algorithm.

7 Individual Contributions

All members of the team did equal work for this project.

References

[1] D. Liben-Nowell and J. Kleinberg. The link-prediction problem for social networks. *Journal of the American society for information science and technology*, 58(7):1019–1031, 2007.

[2] M. Al Hasan, V. Chaoji, S. Salem, and M. Zaki. Link prediction using supervised learning.

[3] R. Lichtenwalter, J. T. Lussier, and N. V. Chawla. New perspectives and methods in link prediction. In *KDD*, pages 243–252, 2010.

[4] C. Eksombatchai, P. Jindal, J. Z. Liu, Y. Liu, R. Sharma, C. Sugnet, M. Ulrich and J. Leskovec. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. *arXiv:1711.07601*.

[5] X. Cheng, C. Dale and J. Liu. Dataset for "Statistics and Social Network of YouTube Videos". *http://netsg.cs.sfu.ca/youtubedata*