Graph Based Video Recommendation System

Avi Singhal [* 1] **Hemanth Kumar Jayakumar** [* 2] **Patrick Yee** [* 2]

Abstract

In this project we attempt to build a graph based video recommendation system without using graph neural networks. We experiment with three methods of link prediction: 1. using a variant of random walk, 2. generate node embeddings using random walks for link prediction 3. employ a topology-centric framework (GELATO) to achieve desirable results for link prediction. We also analyse the scalability of our approach and try to make existing work scalable to larger graphs. We present a comparison between using non-cluster approach and clustering the graph and running the same methodology for the methods mentioned previously.

Keywords: Graph Neural Networks (GNN), GELATO, embeddings, clustering.

Code Links:Pixie Random Walk, Random walk embedings , Gelato

1. Introduction

Recommendation systems are very popular and an important part for the success of any application. The latency and performance of a recommendation system has direct influence upon the time spent by a user on the application. It helps improve customer retention, helps discover new products, increase sales, enhance user experience and save time. A simple example that highlights all the mentioned benefits is of amazon, the better the recommendation system of amazon is, the more related products it will show to the customers. This will save the time of the customers, and if the recommendation system is good then it will also display new products that the customer might be interested in, this will in turn lead to increased sales for amazon. From a video recommendation perspective, if the recommended videos are relevant then the user is more likely to spend longer on Youtube and he will not have to search for videos.

The task of a recommendation system is analogous to link prediction i.e. if there exists a link between two entities then it means they are related in a way. Video recommendation system is very dynamic, it is constantly growing and has billions of videos in the network and handling such a large graph and performing computations on it in an efficient manner is what makes this task a challenge.

Our project is different from other recommendation system approaches which use collaborative filtering and content based filtering. We employ graphs for the task of building a recommendation system, this is because in the real world the objects are explicitly or implicitly connected with each other. This characteristic is very clearly visible for the case of video recommendation systems where the objects include the videos, users,attributes and are tightly connected with each other and influence each other via various relations(Wang et al., 2021). Graph based learning has demonstrated good results for many tasks i.e graph classification, node classification, link prediction, graph generation. Hence, we try to harness the power of graph based learning for recommendation systems as this task is similar to link prediction.

2. Related Works

(Abbas et al., 2017) propose a context-aware recommender system that keeps track of multiple interests of a user and recommends videos based on their current context only. They proposed this solution to overcome the limitations of hybrid recommendation approach which involves use of both collaborative recommendation and content-based recommendation, this approach may not recommend appropriate videos for the current context that the user is in. (Zhu et al., 2013) proposed a content-based video recommendation framework, called Video Topic, that utilizes a topic model. It decomposes the recommendation process into video representation and recommendation generation. It captures user interests in videos by using a topic model to represent the videos, and then generates recommendations by finding those videos that most fit to the topic distribution of the user interests. (Liu et al., 2020) propose a GNN based tag ranking (GraphTR) framework with video, tag, user and media. They design a novel GNN that combines a multi-field transformer, GraphSAGE and neural FM layers in node aggregation. They also propose a neighbor-similarity-based

[1]Department of Computer Science, Rice University, Houston, Texas, United States [2]Credits to Dr. Arlei Silva .

loss to encode various user preferences into heterogeneous node representations to enhance performance. Our main reference papers include (Huang et al., 2021) where the authors employ random walk-based embeddings for link prediction and use autocovariance as the similarity metric. They highlight the power of autocovariance for link prediction tasks. Another important reference for our project is (Anonymous, 2023), this work is done by Professor Arlei's lab at Rice University. They propose a novel topology-centric framework to enhance link prediction without using a graph neural network. We also refer to (Zhu et al., 2013) for the pixie random walk experiment, this was a project conducted by students at Stanford.

3. Methodology

In this project we have mainly focussed on three methods for link prediction. In method 1, we try to use a variant of random walk for link prediction and run several experiments by varying the hyperparameters. In method two, we generate node embeddings using random walks and use the embeddings for link prediction. In method 3, we employ a topology centric framework for the task of link prediction. For method 2 and 3, we try to scale these methods for large graphs by using clustering. We explain these methods in detail in the upcoming sections.

3.1. Pixie Random Walk

We used Pixie random walk for method 1 in our link prediction task for the video network. Pixie random walk was built and deployed by (Eksombatchai et al., 2017) for Pinterest. The goal was to build a scalable graph-based real-time recommender system for Pinterest. The authors demonstrated that this approach worked well even on graphs with 3 billion nodes and 17 billion edges. They showed that Pixie's random walk helped increase user engagement by 50% for Pinterest. The idea of a pixie random walk is very simple and is similar to conventional random walks, the major difference is that transition from one node to the other is no longer completely random, each node is biased based on its attributes, and in this way the transition is biased. It can be said that the pixie random walk is a biased version of the random walk.

In our project, we use the features of the nodes for biasing purposes. We bias the edges based on the neighbor's proximity to the original source video using metrics such as similarity of genres, video uploaders etc. The probability of choosing node a node is as follows: $b_q = \Pi_i m_i$, here b_q is the bias of a node, m_i is the bias for each feature of the node that we have considered.

For each video query, we sample a subset of its neighbors and this forms the query set, then the pixie random walk is run on each video in the query set. The authors of pixie random walk proposed that meaningful visit counts are obtained when the number of steps are proportional to the degree of a query node. We employed the same equations as given by the authors in the original paper for deciding the length of the random walks. We also tried using a transportation rate, the idea is very similar to that of Pagerank i,e we should spend more time in the locality but we should also be able to explore different neighborhoods. One of the reasons for the success of pixie random walk is that it prefers to recommend videos that are related to multiple videos in the query set. The authors use a unique methodology for aggregating the visit counts of the walks for the videos in the query set, we have also employed the same equations. After performing a pixie random walk on each video in the query set, the visit counts are accumulated using the unique methodology (Eksombatchai et al., 2017).

The advantage of using pixie random walk is that there is no training required and that it is easy to scale to large graphs. In the case when new nodes are added to the graph, we simply need to rerun the random walk to get the recommendations, this is the main reason for scalability unlike supervised learning approaches (SVM, Random Forest) which would have required retraining of the model. Even if we are using graph neural networks, after a certain point of time we would have to retrain the model as the graph changes dynamically and the properties of the graph would have changed hence the embeddings of the node need to be recomputed.

3.2. Embeddings-based Link Prediction

We implemented random-walk-based vertex embeddings as described by (Huang et al., 2021). Embeddings are real vectors that are learned by minimizing differences between inner products of labels and pairwise random-walk-based autocovariance.

This approach has several advantages. First, embeddings can be used for a wide range of tasks other than link prediction, such as vertex classification and regression. Second, embeddings only need to be learned once and give some insight into the structure of the graph. However, our experiments reveal that pure embeddings do not scale well. Learning labels is costly, and additionally, generating predictions after labels have been calculated requires inner products to be computed between every pair of non-adjacent vertices in a sparse graph and then sorted.

We calculated runtimes for graphs of various sizes and fit polynomial curves of the increasing degree to minimize Bayesian (BIC = 733) and Akaike (AIC = 726) information criteria. Our tests reveal that the entire prediction architecture (embeddings, inner products, and

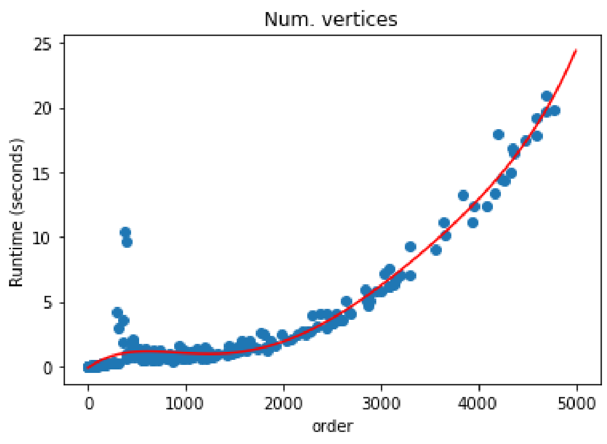

Figure 1. Runtime explodes as graphs get large.

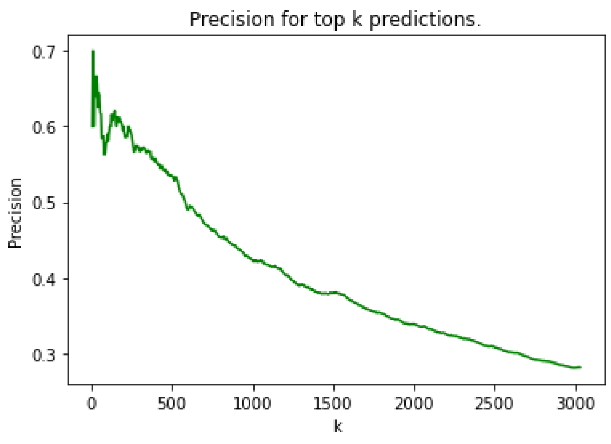

Figure 2. Precision of predictions on G^* for different values of k without clustering.

sorting) runs on an estimated order of $O(N^5)$ which is infeasible for our use case.

To solve this problem, we tried clustering the graph, the number of clusters is a hyperparameter. Clustering will enable us to use the same methodology on a smaller graph. Once we have clustered our original graph into 5 clusters (an example), we will run our embedding-based approach on each of the clustered graphs. Thus, we can use a scalable algorithm for graph clustering/partitioning and make predictions on each component. To test this hypothesis, we reduced the data to a smaller graph G^* on 5701 vertices. In order to preserve overall graph topology, G^* was created by initializing a random vertex and exploding outwards by iteratively adding edges in a process inspired by breadth-first search. Initial experiments reveal that the structure of G^* is indeed amenable to embeddings-based link prediction as shown in figure 2 and so there is theoretical justification for generalizing our results from G^* to the larger dataset. For testing purposes, 20% of edges were removed from G^* at random while preserving connectivity, and predictions were made and recorded using random-walk embeddings ($k = 50$). Then, we split G^* into subgraphs of order < 1500 using METIS partitioning and make predictions in the same way ($k = 5 \times$ #Components). We used METIS as our partitioning algorithm as it was suggested by our advisor.

3.3. Gelato

As a final approach, we attempt to use Gelato for our experiments. Gelato works on a unique pipeline where it attempts to obtain edge weights using a MLP which is used to build an Autocovariance matrix used for link prediction. The MLP is trained end-to-end by n-pair loss calculated from the autocovariance matrix based on predicted links with a bias towards negative edges.

This method involves training over the entire graph on select edges while the prediction depends on the autocovariance as a topological heuristic, allowing the use of both learning from neighbors as local information as well as the topology as global information. This enhances the learning and prediction providing superior results.

Gelato outperforms other approaches for link prediction task on the bench marking dataset, but it does not scale due to the construction of an autocovariance matrix(nxn). We try clustering to improve scalability with METIS clustering to convert the entire graph into an ensemble of multiple graphs each covering its own nodes, splitting up the autocovariance matrix into (c * (pxp)).

We compare this with the PixieWalk method on fixed removed-edges to ensure maintaining an unbiased comparison. The results are recorded in the Results section.

4. Dataset Preparation

We used the youtube dataset provided by (X. Cheng & Liu., 2000), we used data of July 17th 2008. Each file in the dataset gives a list of video ids. For each video in the list, the dataset provides its meta information i.e number of comments, genre , uploader, time since uploaded as well as 20 video ids that are related to it.

For creating the graph, we add a node for each video in the graph and assign its meta information as the node features. We also add the neighbors of each node according to the 20 video id's given for the node. The dataset does not give meta information for videos that are in the related video list, so we then remove nodes that do not have meta information. For pixie walk we remove 5% of the edges from each node, the removed edges are treated as the test edges.

Model	Acc (%) for K=5	Acc (%) for K=10	Acc (%) for K=20	Acc (%) for K=50	Acc (%) for K=100	Acc (%) for K=1000
PRW(N=1000,T=[0.5])	2.12	4.21	6.91	7.9	9.6	30.53
PRW(N=1000,T=[0.5],bias=genre)	2.22	3.97	6.36	7.13	8.38	28.79
PRW(N=1000,T=[0.5],bias=uploader)	1.61	3.37	5.57	6.81	8	28.8
PRW(N=1000,T=[0.5],bias=views)	1.41	2.69	5.25	6.09	8.67	29.4
PRW(N=1000,T=[0.5],bias=unvisited)	1.86	3.32	6.16	6.89	9.25	28.91
PRW(All Bias) T=[0.5]	1.87	3.02	5.59	6.44	8.81	28
PRW(N=1000,T=[0])	8.29	15.29	25.96	42.89	55.37	75.66
PRW(N=1000,T=[0],bias=genre)	9.39	16.2	26.68	44.3	57.53	76.8
PRW(N=1000,T=[0],bias=uploader)	9.93	17.2	28.5	46.23	58.25	77.23
PRW(N=1000,T=[0],bias=views)	9.52	15.93	26.29	43.7	55.7	75.3
PRW(N=1000,T=[0],bias=unvisited)	9.61	15.8	26.7	44	55.6	74.79
PRW(All Bias) T=[0]	9.46	17.53	28.3	47.86	58.9	76.8

Table 1. Pixie Random Walk Results

5. Results

We first present an analysis of the graph under consideration for pixie. We used a graph with 48781 nodes and 131309 edges. We could not try out our experiments on larger graphs due to computational bottleneck.

Figure 3 top left shows that most of the nodes in our graph have degree less than 20 and after that the frequency quickly decreases to 0. We believe the cause of this behavior is that each node has 20 neighbors in its neighbor list. We also present the shortest path between a pair of related videos after removing the edge between them. The result is shown in figure 3 top right. Negative shortest path implies that the pair of videos becomes disconnected. A 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. For pixie random walk, we will bias based on the node attributes, hence we present some characters of graph i.e % of neighbors with similar genre, uploader etc. In figure 3 bottom left, more than 90% of the nodes have same genre in their vicinity, but there is also a significant portion of nodes that have dissimilar (0%) neighbor nodes with same genre. This statistic can affect the when biasing based on the genre. Figure 3 bottom right, indicates that most of the neighboring videos come from different uploaders.

Pixie Random Walk. For pixie random walk we perform 2 experiments, here "T" is the trasnportation rate:

- N=1000, T=[0.5]
- N=1000, T=[0]

For each of the above experiment, we used the following biasing strategies:

Graph	No clustering	METIS	Pixie Random Walk
G^*	58%	44%	5%
G	NA	64%	0.48%

Table 2. Random Walk Based Node Embeddings for Link Prediction results

Epochs	No clustering	Clustering
50	37.451 %	45.858%
250	36.054 %	38.808%

Table 3. Gelato Results

1. No bias
2. Bias towards node with same genre, biasing factor m = 1.5
3. Bias towards node from same uploader, m = 1.5.
4. Bias towards node with more viewers, m = $\sqrt{log(v+3)}$ where v is the number of views of a video.
5. Bias towards unvisited nodes, m = 1.5.
6. Using all the above mentioned bias together.

To evaluate accuracy, for each source node n in the graph, we perform random walk following the algorithm described in section 3.1 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.

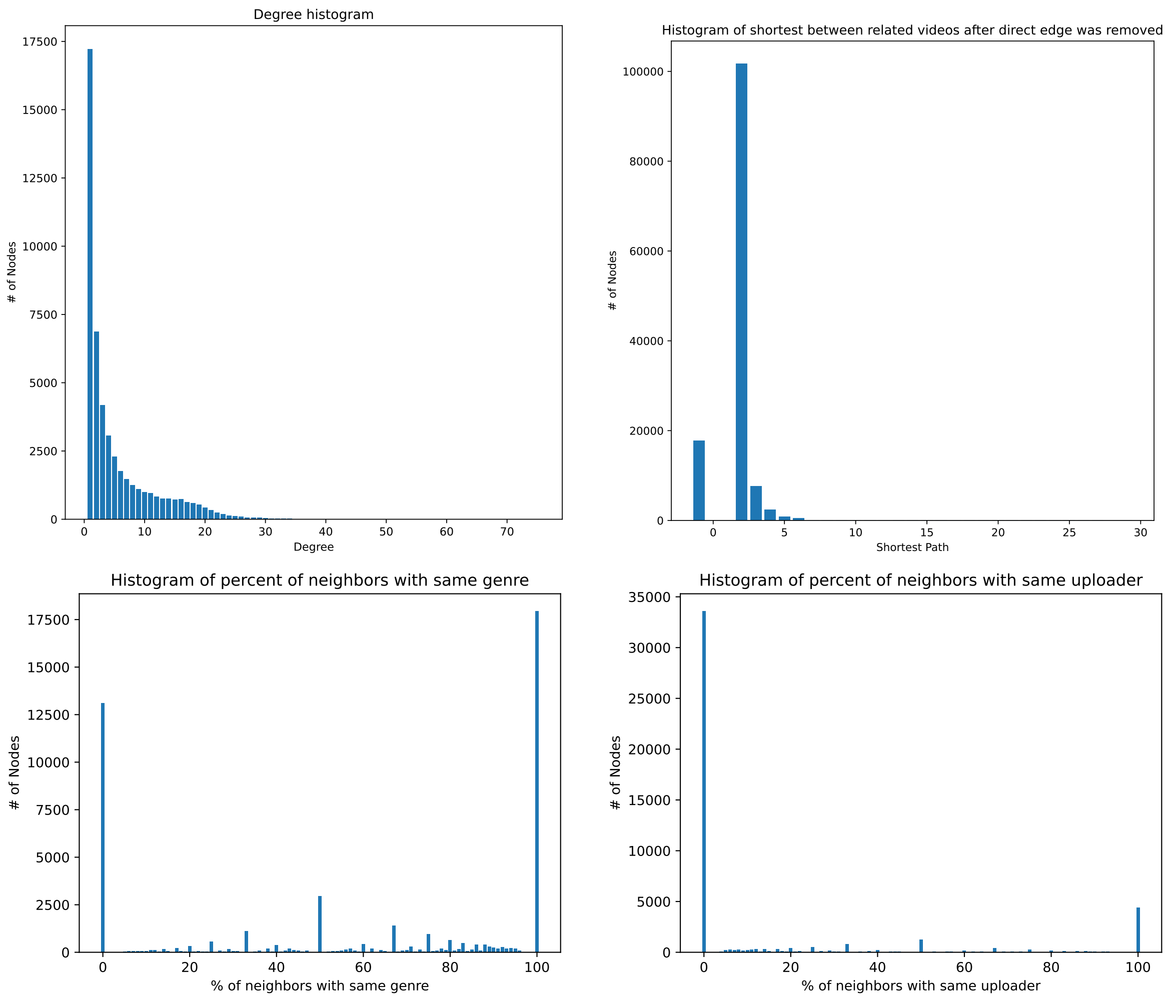

Figure 3. Characteristics of Graph

Table 1 shows the results for the first experiment where we used a transportation rate of 0.5. It can be that there is a trend in the table, as we increase the value of ”K”, the accuracy improves. This is intuitive because increasing ”K” allows for discoverability of more nodes. The accuracy is too low for all the variations we have performed. We hypothesize that the transporation rate could be the cause for such results. This is because the transportation rate can take the walk in a part of the graph which is unrelated to the source video. To confirm our hypothesis, we removed the transportation rate and obtained below results.

The results of this experiment demonstrate that the hypothesis was correct and that the random walk was indeed going in a unrelated region. The accuracy has significantly increased, the trend of increasing accuracy with Topk which was previously seen is still the same. It can be seen that even after using the different biasing techniques, none of them seem to outperform any other by a significant margin. It seems that the biasing schemes we employed are not making full advantage of the pixie random walk algorithm. It is also possible that the features of the videos do not convey a lot of information about the relationship between the videos and hence the biasing scheme does not work as well as was expected.

Embeddings based Link Prediction. We used precision as our evaluation metric because by and large, we only care that predictions that are made are true links and not that predictions which are not made are false links. In the case of G^*, we observed a 14% decrease in precision when preprocessing with METIS:

Observe that we did not compute embeddings for G without clustering due to the high computational cost. While this is a significant decrease in performance, it is far more scalable and still performs much better than random guessing, where expected precision is $< 0.1\%$ (See figure.)

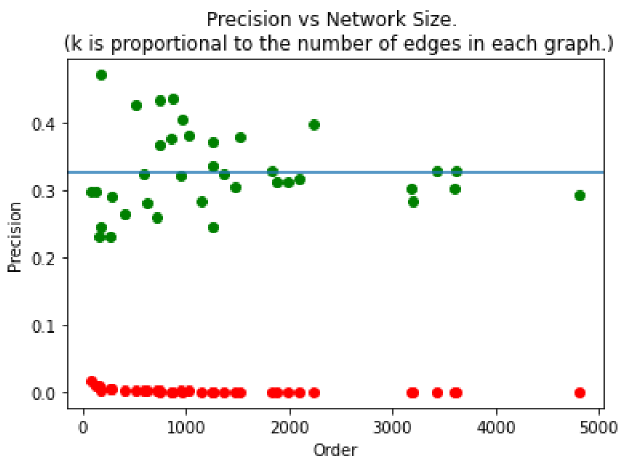

Figure 4. Comparing precision of embeddings against random guessing for different values of k, N.

With the clustering procedure validated empirically, we computed predictions for the Youtube dataset after splitting the original graph into 10 subgraphs of order < 5000. We evaluated our algorithm on the full data by removing 20% of the edges and computing embedding-based predictions (with clustering), which yielded a fantastic performance of precision=64%. We also used the same graphs and performed pixie random walk experiment to compare the performance. Table 2 shows the results, showing that pixie gives a very low precision score for both graphs. The main reason for this is that there are very few true positives and a lot of false positives i.e. for a node which has 2 missing edges if we predict 1 edge correctly, then accuracy will be 50% but the precision@5 will be 20%. As K increases the precision drops further. But the precision values are still better than random guessing. Since all the different variants of pixie give similar results, we used the ”all bias” experiment for comparison in this case.

Gelato. Average precision is used to compare the results of Gelato. Due to the size of the graph, the environments (Kaggle and Colab) used to train and test the pipeline are not capable of building and computing the autocovariance matrix, thus the experiments conducted were limited to a graph of size 4000 nodes. The results shown are for a graph of size 800 nodes with around 2100 edges, training the MLPs for 50 and 250 epochs.

METIS clustering was then used to cluster the graph into 7 subgraphs and Gelato was run on each of these. This provided superior running time and provided comparable, and at times, better results. The clustering technique allows minimal edge loss in the subgraphs allowing gelato to independently predict links, maintaining low subgraph sizes while providing better results. When comparing the same test graph, with the edges, experiments were conducted to compare Pixie random walk, we believe that average precision is less relevant as Pixie Random Walk provides the visit counts of a node rather than proposing probabilities of the existence of edges independently. Average precision requires the probability of each edge, whereas Pixie Random Walk provides the number of visits throughout the walks on each node. This doesn't work for average precision as the normalization leads to dependency of each node on the other nodes in the random walk, summing to 1 across the edges, whereas average precision requires the probability of an edge's existence, which does not depend on the other edges. Using the normalized visit probability described, the average precision of the Pixie Random Walk turns out as 97%.

6. Conclusion and Future Work

We tried three different methods for implementing a video recommendation system, all of the methods employed the topological characteristics of the graph and did not employ graph neural networks. For pixie random walk we observed that using the transportation rate gave low accuracy and the reason was that it took us to a part of the graph which was unrelated to source video. Pixie random walk requires no training and is scalable to large graphs, however for the video recommendation dataset we used, it seems that the different bias schemes we employed did not give significantly different results. It is possible that the number of features in the dataset are too low, the features in the dataset did not convey much information about the videos or there exists another biasing scheme which is more appropriate.

We observe that random-walk based embeddings outperforms pixie in both G and G^* under all conditions. While Pixie is far more scalable than embeddings-based predictions in a vacuum, we can offset this advantage using fast clustering algorithms such as METIS for high quality results with a negligible difference in speed.

Pixie Random Walk performs poorly compared to all the methods, and while a proper comparison between Pixie and Gelato is not feasible/logical due to the metrics mismatch. Clustering Gelato provides better training and inference time, as well as comparable accuracy.

Clustering approach in random-walk based embeddings as well as Gelato makes the methods scalable to large graphs and improves performance. Clustering seems to be the way forward for low latency and inference in graph based recommendation systems.

In future work, similar comparisons can be made between clustered and non-clustered predictions for other tasks such as vertex classification and regression. We also plan to run the same experiments on the complete dataset given the computation resources as that would give a better picture of the performance of our approach.

References

Abbas, M., Riaz, M. U., Rauf, A., Khan, M. T., and Khalid, S. Context-aware youtube recommender system. In *2017 International Conference on Information and Communication Technologies (ICICT)*, pp. 161–164, 2017. doi: 10.1109/ICICT.2017.8320183.

Anonymous. Link prediction without graph neural networks, 2023.

Eksombatchai, C., Jindal, P., Liu, J. Z., Liu, Y., Sharma, R., Sugnet, C., Ulrich, M., and Leskovec, J. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time, 2017.

Huang, Z., Silva, A., and Singh, A. A broader picture of random-walk based graph embedding. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery &amp Data Mining*. ACM, aug 2021. doi: 10.1145/3447548.3467300. URL https://doi.org/10.1145%2F3447548.3467300.

Liu, Q., Xie, R., Chen, L., Liu, S., Tu, K., Cui, P., Zhang, B., and Lin, L. Graph neural network for tag ranking in tag-enhanced video recommendation. In *Proceedings of the 29th ACM International Conference on Information Knowledge Management*, CIKM '20, pp. 2613–2620, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450368599. doi: 10.1145/3340531.3416021. URL https://doi.org/10.1145/3340531.3416021.

Wang, S., Hu, L., Wang, Y., He, X., Sheng, Q. Z., Orgun, M. A., Cao, L., Ricci, F., and Yu, P. S. Graph learning based recommender systems: A review, 2021.

X. Cheng, C. D. and Liu., J. Statistics and social network of youtube videos, 2000.

Zhu, Q., Shyu, M.-L., and Wang, H. Videotopic: Content-based video recommendation using a topic model. In *2013 IEEE International Symposium on Multimedia*, pp. 219–222, 2013. doi: 10.1109/ISM.2013.41.