Building Content-Based Recommender Systems Using Graph Convolutional Networks and NLP

Vivek Trivedy

vivek.trivedy@temple.edu

April 2020

Abstract

With the rise of streaming platforms such as Netflix and Disney+, effective recommendation systems have become a crucial tool for suggesting content to users. In this paper we address the following question: can we improve a content-based recommender system by stacking unsupervised methods to learn more granular node representations. The proposed model uses Doc2Vec, KNN Graphs, and Unsupervised Graph Convolutional Networks to learn node representations and recommendations. The objective is to compare our model with a baseline method of K-Means Clustering. We found that our results validate our initial hypothesis that a stacked model will produce more complex embeddings.

1 Introduction

In order to describe the details of our proposed approach, we will first describe our data and outline some background information that will be a prerequisite for understanding how the approach works. We will also discuss some problems that arise with current recommender systems and how our method attempts to solve these problems.

1.1 Data and Background Information

We use the MovieLens dataset to build our model. The metadata used contains 45,000 movie examples with 24 features including measures such as genre, content reviews, and average

rating. The text features require some embedding to be done during pre-processing which significantly increases the dimensionality of the data. We discuss our approach later.

In order to understand the goals of this paper, it will be helpful to understand recommender systems in general. There are two main approaches to tackling recommender systems - content-based methods and collaborative filtering. Content-based methods leverage similarities in the items themselves. Thus if $User_1$ watches a movie, M_1, which is similar to an unseen movie, M_2, then a system will recommend M_2. On the other hand, collaborative filtering uses similarities between users. Thus if two users, $User_1$ and $User_2$, are similar, then a movie liked by $User_1$ will be recommended to $User_2$. These days many large platforms such as Netflix utilize systems that combine both approaches which is known as a hybrid system. The focus of this paper will be on content-based recommender systems.

1.2 Motivation

The motivation behind our approach comes from wanting to solve some problems that come with content-based recommender systems. These issues include difficulty in performing feature selection given metadata and an inability to leverage information from similar examples when doing feature embedding. We will discuss these problems and our proposed solutions here.

When it comes to data, datasets often have a large amount of metadata per example, but the collected features are not all effective measures for learning representations. Domain knowledge may be of use in this circumstance. Psychological literature points out that genre is one of the most powerful predictors of user preferences [Barza and Memari, 2014]. With this in mind, our data pre-processing focuses on features associated with genre and content. In particular, text data was processed using Doc2Vec [Le and Mikolov, 2014]. It is preferred to simpler methods such as TF-IDF as it achieved state of the art results on the IMDB dataset which is similar to the one we use.

Another issue we address is that the representations generated for examples are often constructed using only the associated data for that example. However, a much more effective approach

would be to aggregate information from similar examples as well. This is achieved by using Graph Convolutional Networks (GCNs). Using this method, node embeddings are learned by aggregating information from a node's surrounding neighborhood. This approach allows nodes to be represented with context from similar items which is a core feature of effective recommendations.

The method and pipeline we propose describes how to combine a series of unsupervised methods in order to produce more complex representations in the final step before recommendation. Another key highlight is that unseen nodes can be quickly queried due to how the aggregator functions are learned during the embedding process. The final results show promising results over the baseline model and through the recommendations made from queried points.

2 Related Work

The method proposed in this paper builds on the work done by many groups in the domains of recommender systems, graph neural networks, and natural language processing.

2.1 Graph Convolutional Networks

The foundation for our proposed method comes largely from Graph Convolutional Networks (GCN) [Kipf and Welling, 2017]. In particular Kipf and Welling were crucial in first describing how features can be propagated through a graph by averaging over the neighborhood of a feature. Their work generalized how the concept of convolutions could be extended from traditional structures such as images to irregular structures such as graphs. Their methods focused on semi-supervised learning for classification but did not yet explore fully unsupervised methods for recommender systems.

An advancement in the domain of GCNs came with the introduction of aggregator functions that could be used in an unsupervised setting to learn node embeddings using neighborhood feature sharing [Hamilton *et al,* 2017]. This method allowed nodes to incorporate information from other nodes by creating a neural network composed of multiple aggregator functions and nonlinearities. In particular, this method included a purely unsupervised approach for

learning embeddings using random walks. This is very helpful when labeled data is not present as the embeddings can then be used for a downstream task.

The first application of GCNs to recommender systems was done by Monti *et al* [2017]. Their approach attemted to aggregate information over user-user and item-item graphs. The idea of applying GCNs to recommender system tasks was extended by STAR-GCN [Zhang *et al,* 2019]. This approach used encoder and decoder layers in order to learn node embeddings. In particular, the paper attempted to combat the cold start problem which is when recommendations are inaccurate at the beginning of the prediction life cycle due to a small amount of data.

2.2 NLP and Feature Selection

Besides the work on GCNs, there was helpful literature in psychology and Natural Language Processing (NLP) which was used during the feature engineering and data pre-processing steps. During feature selection, we focused on measures that captured genre and content as these were found to be strong predictors of user preferences [Barza and Memari, 2014]. When it came to feature engineering, using Doc2Vec was crucial in converting textual descriptions to feature vectors [Le and Mikolov, 2014]. This model is an extension of Word2Vec in that it trains an additional vector that represents each document while also training on the words within each document. This model was very helpful for converting vast quantities of text into feature vectors that could be used as inputs for our model.

3 Methodology

The process of developing this model can be broken down into three main steps: pre-processing, building the baseline model, and building the proposed model. These steps will be outlined here.

3.1 Pre-processing

The first step of pre-processing involved deciding which examples and features to include and remove from the dataset. Uninformative features such as *IMDB_ID* and *PosterURL* were removed. We also made the decision to only include movies that have already been released and to only include English movies. The focus on English movies was to ease the process of converting the text data into embedded vectors which would have been difficult across languages. As mentioned before, Doc2Vec was used in order to transform text data into embedded vectors. Some different dimensions were tested seeing a relatively small difference between choices of output dimensions, we chose to have an embedding space of $n = 25$. The number of genres was relatively small so this was handled using one-hot encoding where examples were allowed to be members of multiple genres.

In order to handle missing values, stochastic imputation was used on continuous features. This method was chosen to avoid transforming the distribution of the underlying feature space. Continuous features were also scaled using the min-max method, again to prevent a distortion of the underlying distribution. Scaling was necessary because a portion of our models is sensitive to feature scale, namely KNN and K-Means Clustering.

3.2 Baseline Model: K-Means Clustering

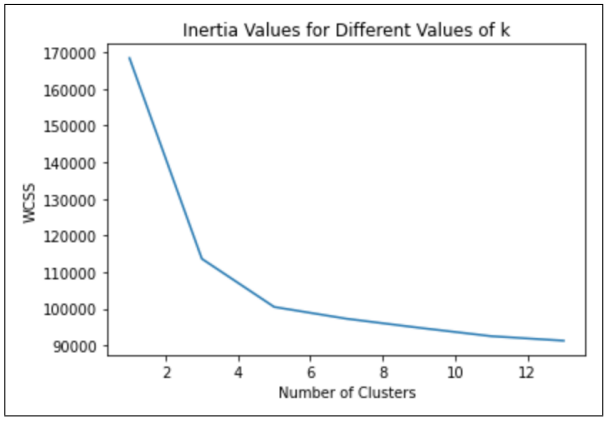

Figure 1: Tuning k with inertia

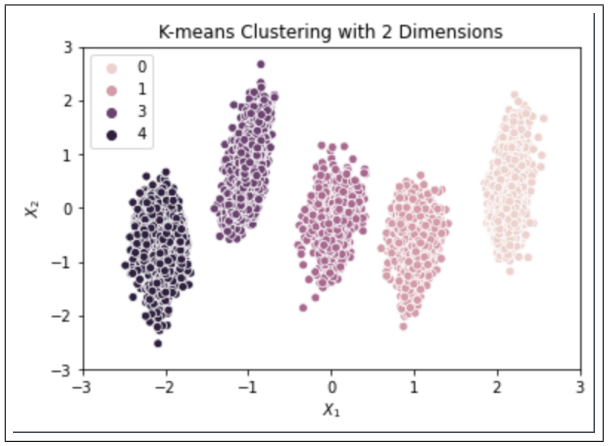

Figure 2: K-Means Visualization using PCA

In order to test our hypothesis, we use a simple K-Means clustering model as a baseline. The

k hyper-parameter was tuned to be 5 by using an inertia based method which can be seen in Figure 1. The K-Means model produces fairly separated clusters as we can see from Figure 2, but the clusters do not capture a great deal of complexity such as low level communities and non-globular clusters. This is one of the downsides of using a simple K-Means approach.

3.3 Unsupervised Stacked Model

The proposed method begins by constructing a directed KNN Graph from the dataset using Euclidean distance. The value of k was chosen taken to be between 5 and 7 in order to mirror the number of recommendations offered on platforms such as Netflix's "Watch More Like This". Note this decision of k determines to what degree more dissimilar examples are incorporated into the feature representations of each example. This method is usually best determined with A/B testing on a user group. The graph was chosen to be directed because similarity is generally not a symmetric relation in the context of recommendations.

In order to learn embeddings for each node, the KNN Graph is fed through the unsupervised variant of GraphSAGE [Hamilton *et al*, 2017]. The model requires a parameter, k, which dictates how many hops to take when considering the features to be considered during aggregation. Thus $k = 2$ will aggregate features up to a neighbor of a neighbor. In the unsupervised setting, embedding is done by first concatenating feature vectors of a node and its neighbors and then learning the weights of an aggregator function using stochastic gradient descent to minimize a custom loss function. This loss function optimizes for two qualities - nodes that are close in the graph should have similar embeddings and nodes that are further away should have different embeddings. The positive sampling is clear and comes from neighboring nodes. Negative sampling is done by performing random walks on the graph and sampling points outside of a node's k-hop neighborhood.

3.4 Recommendation Methods

Once the new embeddings are learned, recommendations can be made. We explore two methods to approach this final recommendation task. The first method is to reconstruct a

KNN Graph using the newly learned features for each node and then providing recommendations based on the new neighbors for each node. A second approach borrows an idea from graph theory known as k-cliques. These are subsets of k nodes that form a sub-graph that is complete. An important property of k-cliques is that they display strong connectivity which indicates a high level of similarity. Nodes that are part of k-cliques are labeled as "strong recommendations" and for nodes not in k-cliques, recommendations are provided using the previously described KNN Graph approach.

4 Results

We found that our results validate our original hypothesis. We also show a sample of recommendation queries and compare them to recommendations from Netflix, Google, and Disney+. Our results are discussed more thoroughly in the following section.

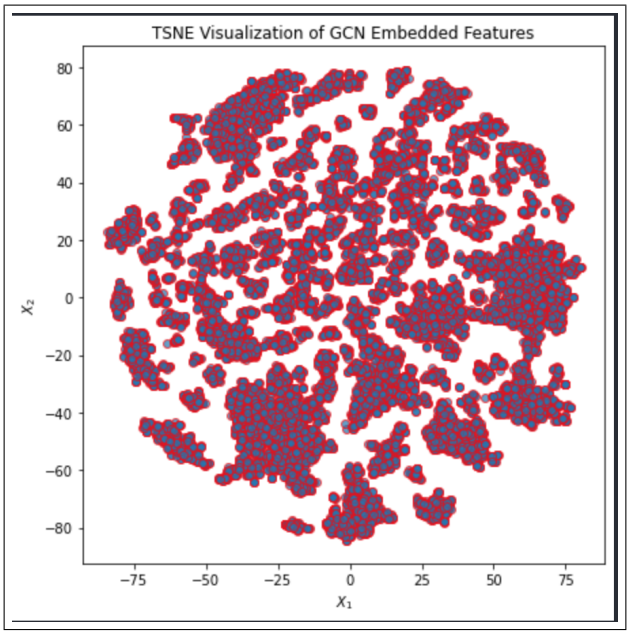

Figure 3: T-SNE visualization of embedded feature space

Figure 3 visualizes the new feature space using T-SNE. We can see a significant increase in the resolution and granularity of node communities compared to our baseline model. While our hypothesis is centered around showing improvement in embedding results, the end goal is to offer recommendations. Thus although we did not have user-level data to

validate recommendations at scale, we offer a subset of recommendation queries and discuss our results in comparison to search results returned by Netflix, Google, and Disney+. The results are displayed in Table 1. Note that the bolded results are "strong recommendations" as described in the recommendation methodology. From an initial glance, our model performs relatively well. Because this task is on unlabeled data without user level information, large scale A/B testing with user groups would generally be the best way to measure the performance. We found that all of our Cinderella results, four of five Harry Potter results, and three of five Interstellar results appear in searches for similar movie using Netflix, Google, and Disney+.

Query	Cinderella	Harry Potter and the Chamber of Secrets	Interstellar
	Return to Neverland	**Fantastic Beasts and Where to Find Them**	**The Martian**
	'Twas the Night	Percy Jackson Sea of Monsters	**Frank Herbert's Children of Dune**
	The Slipper and the Rose	**The Last Mimzy**	**Voyage to the Bottom of the Sea**
	The Princess and the Frog	**Harry Potter and the Order of the Phoenix**	**Doctor Who**
	Cinderella II: Dreams Come True	**Harry Potter and the Philosopher's Stone**	**The Lost World**

Table 1: Results showing recommendations for select queries

5 Discussion

We will now discuss the interpretation and value of our results along with ideas for future directions to take this project.

5.1 Interpretation, Efficiency, and Significance

In terms of creating a more complex embedding space for our features, the proposed method clearly has an advantage over a simple K-Means approach, validating our hypothesis. One

thing to note in particular is that we get many well formed communities. This observed community structure motivated the idea of using k-cliques to power final recommendations. In contrast to the K-Means method which works well with globular clusters, our proposed method seems to be able to capture structure of arbitrary shape and size which allows non-globular structures to be formed. This is a promising result as it shows a method of using purely unsupervised methods to produce useful embeddings for content-based recommendations.

An important point to note is that the proposed method is very efficient for querying unseen data points. This is because, node embeddings are learned using aggregator functions so the model does not need to be fully retrained upon introduction of a new data point.

The sample queries show that our recommender is able to effectively identify sequels and movies with very similar genres as shown by the examples of Cinderella and Harry Potter. In general, it seems that our model is able to match both genre and content very well. The results for Interstellar were interesting in that although the results were "strongly recommended" as they formed a k-clique, the genres and contents were not an exact match, but had general similarities. This behavior is something that can be explored through large scale testing and in future extensions of this project. Our proposed approach may be suitable to help hybrid recommender systems overcome the cold start problem by proposing similar movies using learned embeddings and offering recommendations from different communities in order to build up a significant user profile to kick-start the collaborative filtering component of the hybrid system.

5.2 Future Work

Working through this project and collecting results brought up many directions to take this project forward. One of the primary goals is to incorporate user-level data in order to allow for more large scale testing. This would also help convey how suitable our approach is for a hybrid recommender system. One idea is to use a bipartite structure as that seems to be a natural solution to a recommender system's matching problem. Another exciting avenue to explore is the application of network science tools such as link prediction to find relationships between users and movies. This seems promising because it allows recommender systems to

learn the preferences of a user without seeing previous examples of these preferences.

Acknowledgements

I would like to thank and acknowledge Dr. Zoran Obradvoic for fantastic classes in both data mining and social network analysis that gave me many of the tools necessary to carry out this project. I would also like to thank Dr. Longin Jan Latecki for helping me become more confident with research over the last year through working in his lab.

References

[Barza and Memari, 2014]. Saham Barza and Mehran Memari. Movie Genre Preference and Culture. *Procedia − Social and Behavioral Sciences*, 2014.

[Hamilton et al., 2017] William Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. 2017.

[Kipf and Welling, 2017] Thomas N Kipf, and Max Welling. Graph convolutional matrix completion. arXiv preprint $arXiv$: 1706.02263, 2017.

[Koren et al., 2009] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 2009.

[Monti et al., 2017] Federico Monti, Michael Bronstein, and Xavier Bresson. Geometric matrix completion with recurrent multi-graph neural networks. $NIPS$, 2017.

[Le and Mikolov, 2014] Q. Le, T. Mikolov. Distributed Represenations of Sentences and Documents. $InProceedingsofICML$, 2014.

[Zhang *et al*, 2019] J. Zhang, X. Shi, S. Zhao, and I. King. STAR-GCN: Stacked and reconstructed graph convolutional networks for recommender systems, $arXivpreprintarXiv$: 1905.13129, 2019.

Appendix

A Github repository of the code and data used to complete this project is included here: `https://github.com/VTrivedy/GCNRecommenderSystem`. This page will be updated as this project pursues new directions.