Graph-based Recommendation Systems: Comparison Analysis between Traditional Clustering Techniques and Neural Embedding

Se Won Jang, Simon Kim, JeongWoo Ha
Department of Computer Science
Stanford University
Stanford, CA 94305
{swjang, spkim, jwha}@stanford.edu

1 Introduction

Measuring similarities between two different items has been a focus of active research in the field of Recommender Systems. Due to the abstract nature of the notion of similarity, many Recommender Systems utilize collaborative filtering techniques to infer similarities between two items. Specific designs of collaborative filtering systems differ greatly between industries and markets, but most of them share a common structure of Market-Basket Model, in which we make an assumption that with large volume of data, we can effectively recommend items based on basket co-occurrence.

A very common approach among such systems is to perform graph traversal techniques on a basket-to-item bipartite graph and generate a set of co-purchased candidates. However, this method has critical drawbacks to consider. First, graph traversal techniques are very sensitive to traversal hyperparameters. How far do we drift apart from the starting item? How do we order the generated set of candidates? How many steps do we perform to find the right balance between exploration and exploitation? More importantly, how do we model the relational information between these items? These are all common, hard challenges that such graph-traversal based recommender systems face. Because Traditional methods such as collaborative filtering and random-walk algorithms have their own limitations, we will compare them with a non-traditional method, Neural Embeddings approach, to propose a different approach to Recommender Systems.

2 Related Works

2.1 Recommendation in Bipartite Graphs

In his paper *Recommendation as link prediction in bipartite graphs*, Li suggests a kernel-based recommendation approach that indirectly inspects customers and items related to user-item pair to predict whether an edge may exist between them. The paper used a set of users U and a set of items I as nodes, and transaction was represented as an intersection graph where an edge between the two signified purchase of the product. The graph kernel was defined on the user-item pairs context, and thus item recommendation problem is equivalent to predicting a link between user-item based on graphical analysis.

The graph kernel model uses a Laplacian kernel to capture node similarities that are related to distances between nodes. They estimate distances between nodes through commute time or transition probabilities of random walks. One limitation of this design, however, is that it can only work with nodes that are indirectly connected. To complement for the limitation, the model in the paper inspects structural commonalities of various parts of a network based on the examination of previous graph kernels. This allowed the model to use random walks not as a distance measure. This graphical representation has been used for training the model on different graphs to devise a better recommendation system for predictions.

As benchmark algorithms, the paper used graphical models that did not utilize the information gained from graph kernel, such as user-based, item-based, and item-popularity graph interpretations. The graph kernel model approach showed improvements on recommendation performance over these benchmark models, which proved that utilizing only the local features from pairs of nodes are limited in yielding high performance. Based on this paper, we can see that using embedded structural representation of graph can yield beneficial results to a recommendation model.

2.2 Word2Vec in Graph

In his paper *Efficient Estimation of Word Representations in Vector Space*, Mikolov et al explore and evaluate a technique to embed natural language words in d-dimensional vector space, now commonly known as the word2vec model.

In the word2vec model, each word is defined by its context words that occur within a certain proximity in sentences. Mikolov et al trains a neural-network on n-gram tuples generated with each word and its n-gram context words so that the context word the network outputs the target word. The paper proposes multiple techniques to achieve this, which the paper describes as the CBOW (Continuous Bag of Words) model and the Skip-gram model. Although the two methods differ in the way the neural network is trained, they share the same philosophy in the sense that the neural network behaves as a form of autoencoder in which one of its layers, the weight matrix, represents a concatenation of vector representation of the natural language vocabulary. More importantly, this paper explores the capability of capturing word analogies. Certain relational properties between natural language words can be described as vector operations, such as (Men-Women) - (King-Queen). The paper has gained significant attention over the years, and has been applied to solve many other interesting industry problems, one of which is Recommender Systems.

2.3 Market Basket Analysis with Graph Mining Techniques

In *Extending market basket analysis with graph mining techniques: A real case*, Ivan F. Videla-Cavieres and Sebastian A. Rios discuss the classical approach of getting information from data in retail and department stores through data mining algorithms to detect clustering that are used in recommendations models, such as the market basket analysis (MBA), which is one of the most applied techniques over transactional data to discover frequent itemsets. Traditional methods include clustering techniques such as K-means and SOM. However, when these techniques are used in real supermarket chain data, the results were of very poor quality with meaningless clusters.

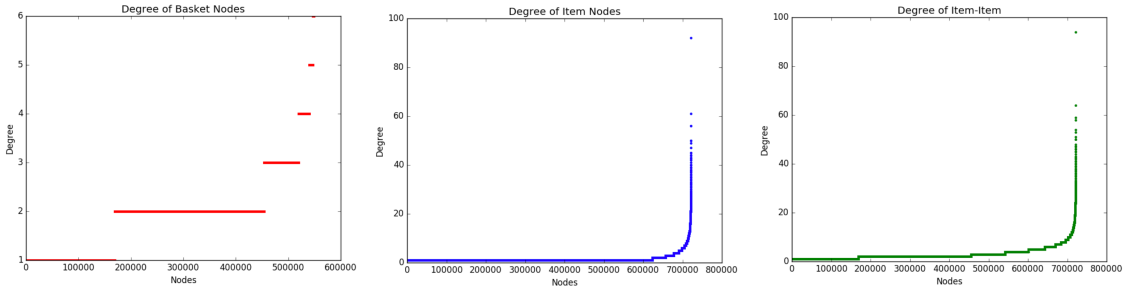
The authors explored a novel approach of performing MBA based on overlapping communities using graph mining techniques. They generated a network of products based only on transactions where each product is linked to others when they appear in the same basket from the same buyer, creating a co-purchased product network. They first built a set of temporal information using transaction set, followed by the transactional product bipartite network where each transaction is linked to the products that are purchased in that particular transaction. Next, they created a co-purchased product network from the bipartite network, where the nodes represent the products and the edges represent the number transactions between the two. Finally, they utilized the overlapping community detection to identify the strongly connected nodes, or clusters. Through this method, they found that the clustering was much more effective, leading to more productive itemsets that much frequent clusters that could provide more insightful recommendation.

3 Method

3.1 Data

We used the Amazon Product Co-Purchasing Network metadata provided by SNAP available at `http://snap.stanford.edu/data/amazon-meta.html`. This dataset holds a variety of product purchase metadata of 721,342 items, including product categories, reviews, and aggregated, filtered lists of co-purchased products ranging from books, music CDs, DVDs, and VHS video tapes. About 72% of the data is books, 3% are DVDs, 20% are music CDs, and 5% are videos. The lists of co-purchased products are not sampled from unique sessions, but is each an aggregation of co-purchase data across multiple user shopping sessions. This dataset does not contain any explicit definition of nodes and edges, and for the purpose of this project we have decided to generate two independent bipartite graphs.

Note that the raw metadata not only contains the product id and the corresponding list of the co-purchased product ids, but also contains various information such as the title of the item, categories, reviews, each of which contains customer rating and votes. These metadata will help us validate our results later on when we compare different algorithms for recommendation systems.



3.2 Collaborative Filtering

Collaborative filtering (CF) has been a well-known algorithm for recommendation system. CF can be divided into two subcategories: memory-based collaborative filtering and model-based collaborative filtering. For the memory-based CF, there are two approaches, one based on items, and the other based on users. The algorithm draws inferences about the relationship between different products based on which ones are purchased together. For the item-based collaborative filtering, we directly compare the similarities between the items by calculating how often the items appeared in the same shopping cart.

In both cases, we create a ratings matrix where the rows represent the users and the columns represents the items. Then we fill up the matrix $R[i][j]$ with the ratings given by user i to the item j. We then measure the similarity between any two pairs.

A common distance metric is cosine similarity, where the ratings are seen as vectors in n-dimensional space and the similarity is calculated based on the angle between these vectors. Another approach is the model-based collaborative filtering. This method is based on matrix factorization, which received more attention, mainly as an unsupervised learning method for latent variable decomposition and dimensionality reduction. It has advantages with scalability and sparsity of the matrix compared to the memory-based approach.

3.2.1 Ratings Matrix and Sparsity

Based on the Amazon Co-Purchase data, we've created the ratings matrix R with all unique user and unique item, where $M \in \mathbb{R}^{m \times n}$ with $m = 1555170$ and $n = 402724$. This is a very large matrix, and the sparsity of the matrix was 0.07%. Based on the given dataset, it is obvious that we have both scalability and sparsity issue, and thus it was more logical to go with the model-based CF.

3.2.2 Singular Value Decomposition (SVD)

The biggest challenge with this model, of course, was the size and sparsity. Thus, we've decided to use singular value decomposition to represent the ratings matrix with smaller dimension matrices. Using the SVD model, we can decompose the ratings matrix R into three different matrices U, Σ, and V^T, where $R^{m \times n}$, $U^{m \times k}$, $\Sigma^{k \times k}$, $Vt^{k \times n}$. Given the value k, SVD will give the minimum reconstruction error (sum of squared errors) of the actual ratings matrix. Thus, if we let $Q = U$ and $P^T = \Sigma V^T$, then we can reconstruct the ratings matrix with our trained k value that will minimize the cost

$$\min_{U,\Sigma,V} \sum_{i,j \in R} (R_{ij} - [U\Sigma V^T]_{ij})^2 \Rightarrow \min_{P,Q} \sum_{i,x \in R} (r_{xi} - q_i \cdot p_x)^2$$

3.2.3 Model Training

To train the model, we first prepared the user-item ratings data from the raw Amazon metadata. Out of 7M total ratings, we divided the data into train, dev, and test dataset with size of 5.5M, 0.5M, and 1M each. Then, with the train dataset, we iterated through different values of k that would minimize the root mean square error between the decomposed matrices and the actual ratings matrix.

3.2.4 Evaluation

Since CF is an unsupervised learning algorithm, we need a test case to see how accurate the predictions are. In the beginning, we separate the train and test data. After successfully learning the model, we evaluate the accuracy of the

predicted ratings with RMSE. Also, we analyze how far the recommended items are compared to the query item from the graph. This would be a good way to compare it with Random Walk model, which is based on the distance between the items. Furthermore, we conduct qualitative analysis on the recommended items to see whether CF model with SVD could indeed recommend sensible items.

3.3 Random Walk

We have a bipartite graph with item nodes on one side and the basket nodes on the other. To get the similarities between two items, it is too costly to iteratively walk through all nodes exhaustively to find the distance between two items. An alternative method to heuristically determine similarity between two items is to perform random walks on two items and find the Jaccard similarity between all the item nodes both walks have landed on.

We have three parameters: α, k, and iterations. α determines the probability of the current node of the random walk to be teleported back to the original item; k determines how many steps to walk from the original node; value of iterations determines the number of such k steps to take to be summed to a counter of all the nodes each iteration has finished on. α will affect the conservativity of the walk so that it does not wander off to a completely unrelated item node. k influences how deep each walk would try to get, and the number of iterations insures the results are normalized across sufficient number of attempts. One of the goals of training this model is to perform hyperparameter tuning to find the optimal composition of these parameters.

3.3.1 Jaccard Similarity

The end of a random walk on each node will create a counter of nodes it has produced. To heuristically determine the relatedness of two items, we performed the length of the intersection of two counters over the length of the union. As higher values of alpha would teleport the walk back to the origin node, we were aware that the Jaccard similarity values of any pair of nodes would be significantly lower as alpha probability of the final nodes will be the origin node. We generated the candidates for each node by listing the top 50 nodes that had the highest jaccard similarity.

3.3.2 Evaluation

Evaluation for the candidates of random walk was similar to that of the neural embeddings. We looked at the average distance between the target node and the candidates to see how successful it was to capture the similarities that may seem to be distant.

3.4 Neural Embeddings Candidate Generation

Neural embeddings, specifically word2vec models, have shown to be very effective in multiple industries for generating a compact representation of objects in high dimensional latent space. In most cases, these models are trained to minimize the cosine similarity between an input object and its target object, where the definition of input and target objects vary greatly across many use cases. With a large amount of data, these models are known to successfully capture some interesting properties of individual objects and inter-object relationships.

In this paper, we explore the effectiveness of Neural Embeddings as a means of co-purchase candidate generation. More specifically, given a co-purchase bipartite graph of baskets and items where each basket holds a flattened representation of multiple user shopping-cart sessions, we analyze the explorative properties of Neural embeddings learned per item.

To do so, we first extract training data from the given bipartite graph with which a word2vec model is trained. Then, upon assigning each item node with its learned high dimensional vector, we run k nearest neighbors over the entire embeddings space to generate the top k candidates. For evaluation, we look at each item and its top k nearest neighbors. Then we look at the distance distribution per position of the neighbors. More specifically, we look at the candidates at each position (sorted in terms of cosine similarity of neural embeddings), and analyze how far they are from the query item on the bipartite graph.

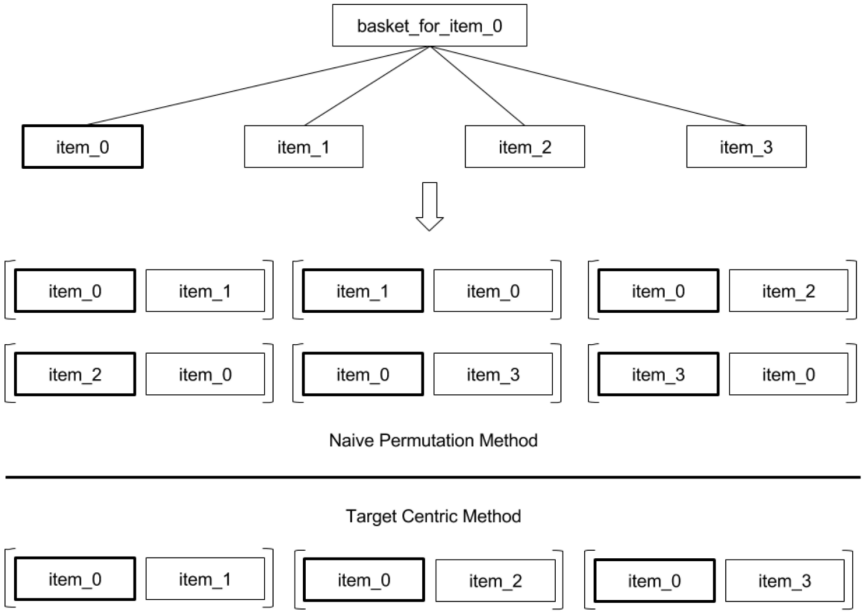
3.4.1 Training Data Generation

Since the product co-purchase information was aggregated from multiple user sessions, and hence does not capture temporal co-purchase information, our basket-item graph inherently has a low item to item degree distribution. This means that normal skip-gram word2vec model will not perform well, due to the lack of training data. This required us to implement two different train data generation techniques.

First, given a set of co-purchased items 0, 1, 2 n for a basket, we apply binary permutation on the set, acquiring the set (0, 1), (1, 0), (0, 2), (n, n -1) . The reason that we generate permutation instead of combination is because no autoencoders are guaranteed to be symmetric by nature. This means that the model may learn to successfully output 1 given 0, but not the other way round. To handle this, we chose to include all binary permutations of co-purchased set items per basket in our training data.

Second, we applied a different technique where for each basket a target item was chosen. This is a direct reflection of the property of our dataset, where each co-purchase list was generated for a specific item. Given a target item, and a set of co-purchased items, we generate bi-directional tuples between the target item and each co-purchased item, so that as mentioned above, we capture symmetry in the training data.

We trained two models for each of these training data sampling methods.



3.4.2 Model

For the neural embeddings model, we have adopted the word2vec model with noise-contrastive estimation. This is because the vocabulary size of all Amazon products can be much larger than that of a natural language vocabulary. As seen in the previous section, the particular graph that we are dealing with has over 710000 item nodes and 540000 basket nodes. Applying softmax over several hundreds of thousands of multinomial classes can be extremely inefficient. By modifying the multinomial classification problem into a binomial one allows us to train the model much faster. Empirically, we have seen x1.2 speed gains by using noise-contrastive estimation loss, compared to using softmax cross-entropy loss for model training.

Dimension	Epochs	Learning Rate	Loss	Optimization	Negative Sample
128	20	0.03 ~ 0.0001 decayed exponentially by 0.95	Nce loss	Stochastic Gradient Descent	128

3.4.3 Candidate Generation

Once we train the object embeddings, we generate recommended candidates by running k-nearest neighbors search on the 128 dimensional space, in terms of cosine similarity. This is a very expensive process, and will likely be very hard to push to a real-time serving system without applying LSH techniques or a large distributed cluster. For the sake of

this project however, we simply normalized the embeddings matrix so that each row has norm of 1.0, and ran batched dot product between rows to compute cosine similarities, and generate top 50 candidates.

3.4.4 Evaluation

After generating top 50 candidates per item in sorted order, we iterate through each items candidate set from position 1 to position 50 (lower position has higher cosine similarity), and check the following information.

First, we record the distance between the query item and the candidate item at position i on the item to basket bipartite graph. Due to the nature of basket-item bipartite graphs, we divide the distance by 2 since there is no edge between any two item nodes (every item is only directly connected to basket nodes).

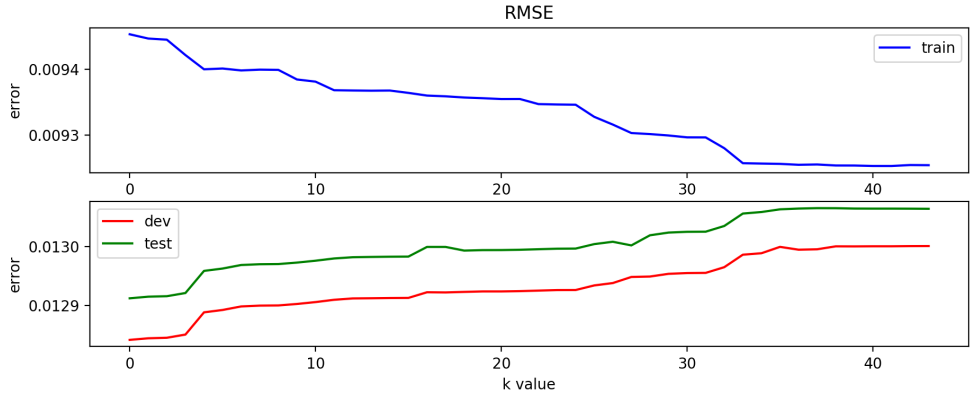
Second, during this process, we also look at the distribution of candidate items that arent reachable from the query item. This is a very crucial information we must capture since our hypothesis is that neural embeddings can capture co-purchasability of items that are not connected on the graph generated from a limited dataset.

4 Result

4.1 Collaborative Filtering

4.1.1 Accuracy

We tested out different k values that minimized the RMSE between the prediction ratings matrix from the SVD model and the actual train data ratings matrix. We noticed that when $k > 40$, the change in error for all train, dev, and test dataset was negligible. Thus, we chose $k = 50$ as the dimension of the sigma matrix from the SVD model.



4.1.2 Distance and Reachability Metrics

	Total queries	Fully reachable	Partially reachable	Not reachable	Mean Distance	Std Distance	Max Distance	Min Distance
Naive	721342	0	562	720780	8.231	15.736	89	1
Targeted	721342	0	703	720639	7.939	12.102	53	1

The results show that the collaborative filtering model has a high maximum distance and standard deviation. This implies that the model could potentially recommend items that Random Walk model cannot. However, it could also mean that the CF model is not as accurate as the Neural Embedding or Random Walk models, especially when the ratings matrix is very large and sparse. Out of 721342 nodes, only 562 and 703 candidate sets were partially reachable from the query item, which shows that over 99.9% of all candidate sets had no reachable nodes from the query node. Thus, this result shows that the CF model may provide insightful recommendations that Random Walk model can't easily, yet the size and sparsity of the given data makes it difficult to always accurately provide appropriate recommendations.

4.2 Examples of Qualitative Analysis

The table of qualitative analysis shows some good cases of appropriate recommendations and some bad. From the first row, we can see that although there are some unrelated ones such as 'Seabiscuit,' many of the recommended books are in the scope of some social issues or historical events that are related to social movements. Also, it was pleasing to see that for the second query, the resulting recommendation included almost all books related to some sort of guide. However, many of the recommendations was also very unhelpful like the last row; this shows the crucial weak point of SVD leveraged CF, which is that it could provide bad recommendations due to the significant reduction in the matrix dimension.

Query	1	2	3	4	5
A Cruel Paradise: Journals of an International Relief Worker, Group [Book]	Lords of the Lake: The Naval War on Lake Ontario, 1812-1814 [Book]	Open Society and Its Enemies (Vol 1) [Book]	PLACE AT THE TABLE: THE GAY INDIVIDUAL IN AMERICAN SOCIETY: [Book]	Seabiscuit: An American Legend [Book]	Defying Male Civilization: Women in the Spanish Civil War [Book]
Leopard and Fat-Tailed Geckos: Reptile Keeper's Guide (Reptile Guidebook Series) [Book]	Kitchens: A Design Sourcebook [Book]	Taking Up Riding as an Adult (Horse-Wise Guides Series) [Book]	Poetry As Prayer, Emily Dickinson (The Poetry As Prayer Series) [Book]	Complete Guide to Lock Picking [Book]	Insight Guide Puerto Rico (Puerto Rico, 3rd ed) [Book]
Science As Inquiry: Active Assessment Strategies to Enhance Student Learning [Book]	The Art of Living: The Classic Manual on Virtue, Happiness, and Effectiveness [Book]	Becoming a Chief Home Officer [Book]	Always & Forever: The Classics [Music]	Devotion [Music]	The Opal (Studies in Austrian Literature, Culture, and Thought Translation Series) [Book]

4.3 Random Walk

4.3.1 Hyperparameter Tuning

To observe the effects of changing α, weve initially fixed k to be 10 and performed 100,000 iterations with varying alpha. The resulting distances of two candidate nodes are as follows. Note that weve removed pairs with distance 0.

alpha	Total queries	Mean Distance	Std Distance	Max Distance	Min Distance
0.05	721342	8.02	3.02	10	1
0.1	721342	6.52	3.41	10	1
0.25	721342	3.78	2.77	10	1
0.5	721342	1.99	1.39	10	1
0.8	721342	1.25	0.56	9	1

k	Total queries	Mean Distance	Std Distance	Max Distance	Min Distance
1	721342	1	0	1	1
5	721342	4.09	1.41	5	1
10	721342	6.51	3.41	10	1
100	721342	9.99	9.49	100	1

You can see that α sets the boundary random walks can explore out to. Higher α values prevent the walker from exploring far, and thus the candidates are likely to conservatively explore closer nodes more than further ones.

To observe the effects of changing k, weve fixed α to be 0.1 and performed 100,000 iterations for each value of k. As k increases, the runtime cost rises exponentially because each step increases the number of travelled nodes exponentially.

As k increases, you can see the value approaches $\frac{1}{\alpha}$, which is as expected. Therefore, we can see that k determines how conservative the model should be within the boundary set by α. By setting $k = 100$, we could insure two distant nodes have a higher chance of being selected as candidates for similar items, but because we have around 720k nodes the runtime cost was far too costly for the model to be run with $k = 100$. As a result, for the purpose of this paper weve set the value of k to be 10.

The graph had an average of 2.93 degree in item-to-item relations. The above tables show the effects of changing different parameters. As will be discussed below, recommendations by random-walk are limited to only nodes that are directly reachable from the origin node. This prevents the recommendation from suggesting items that are not directly connected, but still are similar. Below, this paper will show why Neural Embeddings, a non-traditional method, is more successful in specific cases.

4.3.2 Examples of Qualitative Analysis

Query	1	2	3	4	5
Prettybelle (1974 Original Cast)	That Travelin' Two-Beat/Sings the Great Country Hits	Beethoven: Symphony No. 9 "Choral"	Passtrak Series 7: General Securities Representation	The Best of Natalie Cole [EMI-Capitol Special Markets]	These Twelve Days: A Family Guide to After-Christmas Celebrations
Guitar Country/ More of That Guitar Country	Finger Style Guitar/ Stringin' Along with Chet Atkins	Hum & Strum / Other Chet Atkins	Most Popular Guitar/Down Home	Mister Guitar/Chet Atkins in Three Dimensions	Essential [Music]
Making Bread: The Taste of Traditional Home-Baking	That Travelin' Two-Beats/Sings the Great Country Hits	Jewish Family and Life : Traditions, Holidays, and Values for Today's Parents and Children	Mathematics of the Securities Industry	The Best of Natalie Cole [EMI-Capitol Special Markets]	These Twelve Days: A Family Guide to After-Christmas Celebrations
World War II Allied Fighter Planes Trading Cards	That Travelin' Two-Beat/Sings the Great Country Hits	Beethoven: Symphonies Nos. 3 "Eroica" & 8	Mathematics of the Securities Industry	The Best of Natalie Cole [EMI-Capitol Special Markets]	These Twelve Days: A Family Guide to After-Christmas Celebrations
Brittas Empire (Vols. 4-6)	Brittas Empire (Vols. 1-3)	That Travelin' Two-Beat/Sings the Great Country Hits	Beethoven: Symphonies Nos. 3 "Eroica" & 8	The Best of Natalie Cole [EMI-Capitol Special Markets]	These Twelve Days: A Family Guide to After-Christmas Celebrations

These qualitative analysis show clearly the strengths and limitations of random walk model. From positive results we can see it is successful in selecting candidates that are in close distance from the origin item node. For items with sufficient neighboring item-item degrees, the random walk can generate candidates with visible similarities between the pairs.

However, the clustering effect can bring detrimental results to random walks. If an item has no associated items such that it is connected to a single basket, and the basket is also connected only to the origin item node, then regardless of how the random walk is progressed it will not yield significant results, as seen by the first two negative results. The third negative result also shows the limitations of random walk. This is the case where the two books Brittas Empire (Vols. 4-6) and Brittas Empire (Vols. 1-3) are mutually connected via single basket, and these two items are the only ones directly connected. With no teleportation to random nodes, the random walks will be trapped in the cluster of these wo items, and thus apart from recommending one another it will not yield any significant results, as seen from the top candidates. This shows that for network with sparse item-item network, random walk is not the best model to generate similar candidates. This is why this paper suggests Neural Embeddings, as shown below.

4.4 Neural Embeddings Candidate Generation

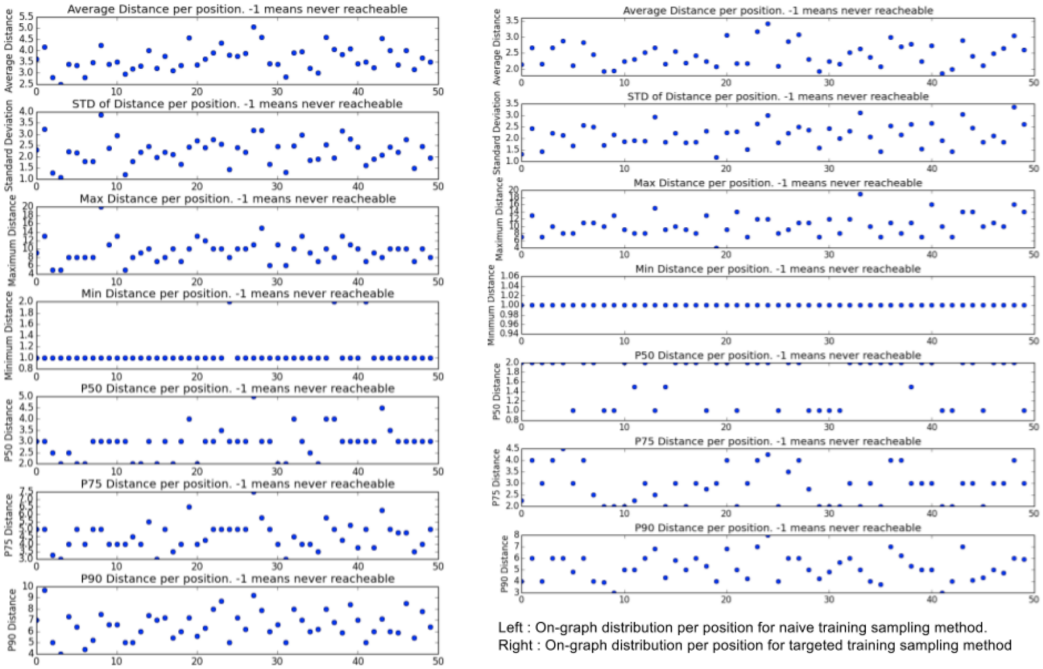
4.4.1 Distance and Reachability Metrics

The evaluation results show that Neural Embeddings trained from the two training sampling methods are capable of generating candidates that are unreachable from the query node on the basket-item graph. Out of 721342 total items, only 1102 and 1728 candidate sets contained candidates that were reachable from the query item node. There were 0 candidate set where all of the candidates were reachable from the query node, and over 99.5% of all candidate sets had absolutely none of its candidates reachable from the query node. Even considering the fact that this dataset includes some partial information (some items are only mentioned in the co-purchase list, and do not contain any other

metadata), this is an extremely high number. These numbers suggest that using word2vec embeddings and cosine similarity as the comparator function allows us to capture some complex information that is invisible from the graph structure. In other words, it may be capable of encoding proximity and similarity information that cannot be captured by traditional graph analysis techniques.

	Total queries	Fully reachable	Partially reachable	Not reachable	Mean Distance	Std Distance	Max Distance	Min Distance
Naive	21342	0	1102	7202240	3.637	2.396	20	1
Targeted	721342	0	1728	719614	2.533	2.329	19	1

Only looking at candidates that were reachable on the graph, the distribution of candidate distances per position looks quite uniform throughout the position indices (top 50 candidates), with average distance around $2.5 \sim 3.6$ and standard deviation of 2.3. There are two interesting things to note from this data. First, the distance distribution is not uniform across distances. Second, the distance distribution is quite uniform across candidate positions, at around 3.5 hops from the query node. This means that the word2vec model does not completely ignore the distances between nodes on the graph, but is in fact capable of capturing the proximity of two items on the bipartite graph. If the word2vec model had failed to learn meaningful information, this distribution would have shown uniform-like distribution (BFS tree depth per node for this graph can be over 150).



Left : On-graph distribution per position for naive training sampling method.
Right : On-graph distribution per position for targeted training sampling method

4.4.2 Examples and Qualitative Analysis

Its very hard to measure the quality of a recommender system without a way to measure the online performance. So in this paper, we provide a few good and bad examples to demonstrate the type of recommendations that this model makes. These samples were chosen from top 10 candidate sets in terms of their distances away from the query item. Here, the average distance from the query items are between 10 and 20 ($10 \sim 20$ random hops away from the query item). We can say however, that they are well beyond the discovery scope of traditional graph recommender technique.

Query	1	2	3	4	5
Creative Thinking in Photoshop [Book - Design]	Dragons: A Book of Designs [Book - Design]	Resonant Power Converters [Book - Engineering]	Security Analysis on Wall Street [Book - Finance]	Construction Qa/Qc Systems that Work [Book - Engineering]	The Self as Agent [Book - Self motivation]
Spanish Complete Course : Basic-Intermediate , Complete Compact Edition [Book - Education]	Crazy in Alabama [Book - Crime]	Tutorials in Introductory Physics and Homework Package [Book - Education]	A Cat in the garden 2002 calendar [Book - Calendar]	Great Expectations [Book - Classics]	The Canoe - A Living Tradition [Book - Education / Nonfiction]
Muslim Rulers and Rebels [Book - History]	Kanzi - the ape at the brink of human mind [Book - Anthropology]	Why the Frog has Big Eyes [Book - Childrens]	Greatest Hits [Music]	Meaning and Mental Representation [Book-Nonfiction]	One Hundred Years of Collectible Jewelry [Book - Antiques]
Air Pollution Engineering Manual [Book - Engineering]	The Complete Guide to Editing Your Fiction [Book - Reference]	The Roses - Pierre-Joseph Redoute [Book - Arts]	Equine Dentistry [Book - Medicine]	Benjamin Smoke [DVD - Documentary]	Life is a Dream [Book]
The Donna Summer Anthology [Music - Disco,Pop,R&B]	Frida Kahlo [Book - Autobiography]	Ian Anderson: Divinities - Twelve Dances with God [Music - Rock]	The Addams Family [Video - Fantasy]	Employee's Entrance [Video - Drama]	Wireless Video Communications [Book - Engineering]

5 Analysis

Random-walk algorithm is limited for sparse graph structures, as seen above. It is limited to suggesting candidates that are directly connected by edges, and also cannot travel to nodes that are further away in distance yet still may be similar. Also, it is vulnerable to clustering effects. If a basket only contains two items, and each of the items is only connected to one another, the random-walk algorithm will be limited to suggesting only each other as the most similar candidate and fail to explore to look at others.

Collaborative filtering is more successful in attempting to capture similarities of two items that are not necessarily connected, as seen by the results above. However, we can see that this model is vulnerable to sparse dataset. If the ratings matrix is too sparse, then even with the SVD the model will be unsuccessful in capturing meaningful similarities between two items due to enormous amount of noise that arise.

From the results we can see that the neural embeddings model is successful in capturing similarities between two items that are not directly connected to each other. The distance between the target item and the candidate nodes are significantly small enough to suggest meaningful similarities, as seen by our qualitative analysis of the results. In addition, it is successful in capturing similarities between partially reachable item nodes as well. Therefore, we can conclude that for dataset with sparse ratings matrix, a significant number of disconnected nodes, and clustering effects, neural embeddings model can perform better than the traditional models.

6 Conclusion

This paper looked into traditional recommender system models, such as Collaborative Filtering and Random-Walk algorithms. We could see that for sparse datasets, these models are not successful in capturing similarities of items that are not directly connected by graph structures. As a result, we have looked into an alternative to recommender systems - Neural Embeddings, which we could see is more successful in capturing embedded similarities between items that are not directly connected. We believe this paper will be a stepping stone to providing an alternative model to traditional models.

References

[1] Li, X. & Chen, H. (2013). Recommendation as link prediction in bipartite graphs: A graph kernel-based machine learning approach. *Decision Support Systems*, 54(2), 880-890.

[2] Mikolov, Tomas, et al. (2013). Efficient Estimation of Word Representations in Vector Space. [1301.3781]

[3] Videla-Cavieres, Ivan F., and Sebastian A. Rios. (2014). Extending market basket analysis with graph mining techniques: A real case. *Expert Systems with Applications*, 41(4), 1928-1936.

[4] Gutmann, M. & Hyvarinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. *PMLR*, 9:297-304.

[5] Zhou, Y., et al. (2008). Large-Scale Parallel Collaborative Filtering for the Netflix Prize. *AAIM*, 5034:337-348.