Double weighted graph convolutional networks for recommender systems

SHUSHAN HE, KAREEM ABDELFATAH, WEI HAN, MADHAV SIGDEL, MENGSHU LIU, and MOHAMMED KORAYEM, CareerBuilder, USA

Graph convolutional networks (GCNs) are very powerful in learning graph-structured data by integrating features from node and its local neighborhood. GCN-based methods can encode features of nodes (items) together with the collaborative signals of user-item interactions to learn the embedding. However, it suffers from cold-start problem as the collaborative signals would be missing for any new nodes. We propose a novel weighted GCN-based representation framework for recommender systems by constructing content and behavioral data into a double connected and weighted hyper-network. Empirical studies on job recommendation scenario shows the effectiveness of our method in representing all nodes in the graph and eliminating the cold-start problem.

CCS Concepts: • **Computing methodologies** → *Semantic networks.*

Additional Key Words and Phrases: graph convolution network, neural networks, embedding, recommender system

ACM Reference Format:
Shushan He, Kareem Abdelfatah, Wei Han, Madhav Sigdel, Mengshu Liu, and Mohammed Korayem. 2022. Double weighted graph convolutional networks for recommender systems. In *CompJobs-WSDM '22: The First International Workshop on Computational Jobs Marketplace, Feb 25, 2022, Phoenix, AZ*. ACM, New York, NY, USA, 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Personalised recommender systems have become a crucial component in any online services including e-commerce and job marketplace. The complex relation between users, items and their interactions in online services can be essentially captured in a graph structure. There has been growing interests in leveraging this inherent graph structure for high-level representation of users and items, and applying it to downstream applications such as recommendation.

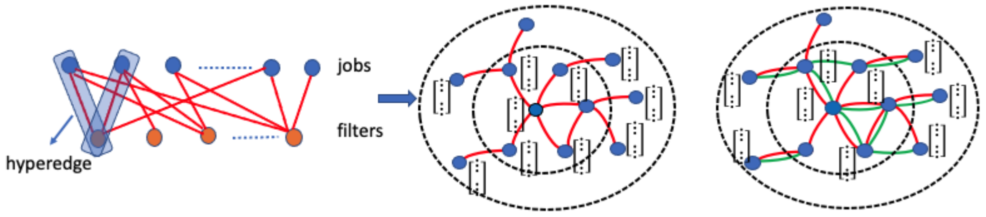
Several studies have been done to model and train graph-structured data. Recently, Graph Convolutional Networks (GCNs) [2, 4, 5] have gained increasing attentions and have been shown to be very powerful in representation learning. The original GCN algorithm [4] is designed in a transductive setting and [5] requires known full graph Laplacian. The extended GCN framework GraphSage [5] is in an inductive setting to learn the embedding function which can be generalized to unseen nodes and subgraphs. Here, we further extend GraphSage to a weighted inductive model to mostly reduce the impact of noises contained in the graph structure.

In recommendation applications, GCNs provide powerful and systematic tools to explore multi-hop relationships on the graph-structure data incorporating abundant data sources. The abilities to explicitly encode the crucial collaborative signal (i.e. user-item interactions) is one of the biggest reasons for the remarkable success of GCNs in recommender systems. However, if the interaction data is simply represented by a bipartite graph between user and item nodes only [1, 7], cold-start problem becomes crucial limitation since the association between items to users would be missing for new items. In this research, we first propose a novel double connected and weighted graph structure to encode the content-based data sources and collaborative signals, then apply our weighted GCNs over the customized graph for a unified perspective to sufficiently benefit from GCNs and eliminate cold-start problem. We validate the effectiveness of our method (referred to Double Weighted GCN, DWGCN for short) on real-world data for job recommendation.

2 METHODOLOGIES

2.1 Model Description

2.1.1 Behavioral and Content based networks. Let $G_{IF} = (V_I \cup V_F, E_c)$ be a bipartite content-based network with two separate parts V_I and V_F where V_I is the set of all items, V_F is the set of all considered filters on items, and E_c is the set of pairs (i, f) such that the item $i \in V_I$ satisfies the filter condition $f \in V_F$. Clearly, while there is a new item, we can easily add it to the network G_{IF}. We also consider the engagement network $G_{IU} = (V_I \cup V_U, E_u)$ where V_U is the set of all users and E_u is the set of pairs (i, u) such that user u has some interactions with the item i. Various filters on the contextual information can be defined to characterize item nodes for reducing the impacts of noises due to different writing and formatting difference. And the filters can be over item nodes (e.g. education level, skills requested) or node pairs (e.g. distance<15 miles). Since G_{IF} is bipartite, it identifies a unique multigraph with vertex set V_I, edge set E_I such that $i_1 i_2 \in E_I$ if and only if items i_1 and i_2 satisfy some common filter, and multiplicity function $\mu : E_I \longrightarrow \mathbb{N}$ such that $\mu(i_1, i_2)$ is the number of common filters satisfied by items i_1 and i_2 (see Figure 1a). We denote this multigraph by $G_I = (V_I, E_I, \mu)$. Similarly, the bipartite graph G_{IU} also identifies one unique multigraph $G'_I = (V_I, E'_I, \mu')$ where $\mu' : E'_I \longrightarrow \mathbb{N}$ such that $\mu'(i_1, i_2)$ is the number of common user-item interactions of i_1 and i_2. Finally, we get a behavioral and content based multigraph defined by a 5-tuple $G = (V_I, E_I, E'_I, \mu, \mu')$ (see Figure 1b).



(a) Content based H-network **(b)** Double connected and weighted

Fig. 1. Illustration on (a)content based H-network, and (b) double connected and weighted H-network based on content (red) and behavioral (green) connections.

2.1.2 Weighted H-network. To allow the importance of filters and user-item interactions, we could further define a weighted H-network. Over the multigraph $G_I = (V_I, E_I, \mu)$, let $F : E_I \longrightarrow \mathcal{F}$ be the filter mapping such that $F(i_1 i_2)$ is the set of common passed filters by i_1 and i_2 where $\mathcal{F}$ is the family of considered filters on items, and define the weight function $\omega_{\mathcal{F}} : E_I \longrightarrow \mathbb{R}$ such that $\omega_{\mathcal{F}}(i_1 i_2) = \sum_{f \in F(i_1 i_2)} \omega_f$ where ω_f is the weight for filter f. Note that if $\omega_f = 1$ for any $f \in \mathcal{F}$, then $\omega_{\mathcal{F}}$ is equal to the multiplicity μ. Similarly, over the multigraph $G'_I = (V_I, E'_I, \mu')$, let C be the family of interactions of users on items (eg. click, application) and $\mu_C = \{multiplicity \;\; \mu_c : c \in C\}$ be the multiplicity family where $\mu_c : E'_I \longrightarrow \mathbb{N}$ such that $\mu_c(i_1 i_2)$ is the number of common user-item interactions c between i_1 and i_2. Define the weight function $\omega : E'_I \longrightarrow \mathbb{R}$ with $\omega(i_1 i_2) = \sum_{c \in \mathcal{F}} \omega_c \mu_c$ where ω_c is the weight for interaction c. Finally, we get two weighted multigraphs $G_I = (V_I, E_I, F, \omega_{\mathcal{F}})$ and $G'_I = (V_I, E'_I, \mu_C, \omega_C)$ for the weighted H-network.

2.1.3 Sparse weighted H-network. In the item-filter network, we identify the connections between items in G_I in different aspects. We expect the connection to be strong enough to show the similarity of items. While $|V_F| << |V_I|$ in G_{IF}, the graph G_I will be very dense. To sparse the graph, we can define a hard filter family $\mathcal{F}_h$ to remove weak connections. For example, we can set thresholds τ_f, then modify the graph G_I by removing connections between two items with less than τ_f common filters. The hard filters can be also defined according to some business logic. For example, we could only consider the items with distance less than a predefined threshold τ_l.

2.1.4 Neighbor importance selection. GCNs are neighborhood aggregation schemes, but the size of node neighborhoods is irregular in the H-network. In order to keep the computational footprint in the training, we convolve on node over its selected neighborhood of fixed size. Moreover, in the H-network, both $\omega_{\mathcal{F}}$ and ω_C measure the strength of connections. The stronger the connection is, the more reliable the neighbor information is going to be. However, we note that ω_C would be missing for new items. To avoid cold-start problem, we only use $\omega_{\mathcal{F}}$ in the GCNs to weight the importance of neighbors. Thus, we define a neighbor importance selection function $\mathcal{N}^* : V^I \times \mathbb{N} \to V^I$ to generate the neighborhood for nodes such that $\mathcal{N}^*(v, k)$ is a set of neighbors of node v with size k which are randomly selected by weights $\omega_{\mathcal{F}}$. Note that, we only select neighbors related to edges in E_I. For notation simplicity, let $\mathcal{N}_v^{(k)}$ be the selected k-hop neighborhood of node v and $\mathcal{N}_v$ be the whole neighborhood defined by the filter edge set E_I. Given neighborhood sizes k, if $|\mathcal{N}_v| < k$, then the neighbors of v are sampled with replacement in $\mathcal{N}^*(v, k)$. More details are shown in Algorithm 1.

Algorithm 1 Neighborhood importance selection on node u

```
1: Input: Node $u$, filter edge set $E_I$, filter weight function $\omega_{\mathcal{F}}$, depth $K$, neighborhood sizes $s_k$ for $k = 1, 2, \cdots, K$.
2: Initialization: $\mathcal{N}_u^{(0)} := \{u\}$ and $\mathcal{N}_u^{(k)} := \emptyset$ for $k = 1, 2, \cdots, K$.
3: for $k = 1, \cdots, K$ do
4:     for $v \in \mathcal{N}_u^{(k-1)}$ do
5:         $N \leftarrow$ Randomly selected $s_k$ neighbors from $\mathcal{N}_v := \{w : vw \in E_I\}$ by weights $\{\omega_{\mathcal{F}}(vw) : w \in \mathcal{N}_v\}$ with
           replacement if $|\mathcal{N}_v| < s_k$, and without replacement otherwise.
6:         $\mathcal{N}_u^{(k)} \leftarrow \mathcal{N}_u^{(k)} \cup N$
7:     end for
8: end for
9: Output: $k$-hoop neighborhood $\mathcal{N}_u^{(k)}$ for $k = 1, 2, \cdots, K$.
```

2.1.5 Model Architecture. As GCNs, the core of DWGCN algorithm is also to use the localized convolutional modules to generate embeddings for nodes. It starts with node features and its localized graph structures in the H-network, and then learn the weighted GCNs that transform and aggregate features of the node neighborhood. Any node features available as text could be transformed to numerical vectors by some general or pre-trained text to vector embedding models, and combined with other numeric features. Let c_u be the current embedding for node $u \in V_I$, then the task is to generate a new embedding n_u to optimally represent the node's features. This procedure is detailed in Algorithm 2.

2.2 Model Training

2.2.1 Weighted Energy-based Loss Function. We train the DWGCN in a fully unsupervised manner which assumes that two connected nodes in the H-network have more similar representations compared to disconnected or less weighted node pairs. Note that, different from the step for neighborhood sampling which only considers filtering edges in E_I, the customer behavioral edge set E_I' is also considered here. Since the user-item interaction information is only used to train the model on the reliable connections in the backward step, the algorithm is designed for applications which fully overcome the cold-start problem. Define the weight $\omega_{u_1u_2} := 0.5 * \left(\omega_{\mathcal{F}}(u_1u_2) + \omega_C(u_1u_2)\right)$ for $u_1u_2 \in E_I \cup E_I'$, and $\omega_{u_1u_2} := 0$ otherwise. For an edge $uv \in E_I \cup E_I'$, consider the weighted loss function for their node embeddings n_u, n_v

$$\mathcal{L}(n_u, n_v) = -\omega_{uv} \log\left(\sigma(n_u \cdot n_v)\right) - N * \mathbb{E}_{w \sim P_u}(1 - \omega_{uw}) \log\left(\sigma(-n_u \cdot n_w)\right)$$

where P_u denotes the distribution of negative samples for item u, N is the number of negative samples, and $\sigma(\cdot)$ is the sigmoid function. It introduces weights on the cost of missing a positive or negative sample.

Algorithm 2 Node neighborhood aggregator on node u

```
1:  Input: H-network $G = (V_I, E_I \cup E_I', \omega_{\mathcal{F}}, \omega_C)$, current embedding set $\{c_v : v \in V_I\}$, activation function $\sigma(\cdot)$, depth $K$, $k$-hoop neighborhood $\mathcal{N}_u^{(k)}$,weights $\mathbf{W}^{(\mathbf{k})}$, biases $\mathbf{b}^{(k)}$, aggregator functions $g^{(k)}$ for $k = 1, 2, \cdots, K$.
2:  Initialization: $h_v^{(K)} := c_v$ for $v \in \mathcal{N}_u^{(K)}$ and $\mathcal{N}_u^{(0)} := \{u\}$.
3:  for $k = K, \cdots, 1$ do
4:      for $z \in \mathcal{N}_u^{(k-1)}$ do
5:          Convolve over neighborhood: $h_z^{(k-1)} \leftarrow g^{(k)}\left(\{h_v^{(k)} : \forall v \in \mathcal{N}_z \cap \mathcal{N}_u^{(k)}\}\right)$
6:          Convolve with self-embedding: $h_z^{(k-1)} \leftarrow \sigma\left(\mathbf{W}^{(k)} \cdot CONCAT(h_z^{(k-1)}, c_z) + \mathbf{b}^{(k)}\right)$.
7:          Normalization: $h_z^{(k-1)} \leftarrow h_z^{(k-1)}/||h_z^{(k-1)}||_2$.
8:      end for
9:  end for
10: Output: New embedding $n_u := h_u^{(0)}$ for node $u$.
```

2.2.2 *Strategy on negative sampling.*

To enforce the model to learn the parameters to capture the difference between strong and weak connections, instead of uniformly sampling negative instances from the entire set of nodes, we consider the connections with small weights as the hard negative samples.

3 EXPERIMENTS

We test and evaluate the generated embeddings on a sample dataset from a real job marketplace for job recommendation task. The goal is to find the closest jobs (for example, top 100) in the embedding space to the most recently applied job (source job) to recommend to the user. To test our approach in this domain, we first construct the H-network $G = (V_I, E_I \cup E_I', \omega_{\mathcal{F}}, \omega_C)$ following Algorithm **??**. Let V_I be the set of active jobs. To simplify the evaluation, we considered only the jobs covered by enough co-apps i.,e co-apps$\geq$ 20. The "apply" behavior is considered in C and filters related to job titles, categories, skills, location, etc. are considered for $\mathcal{F}$. More specifically, if a user u applied to both the jobs j_1 and j_2, then $j_1 j_2 \in E_I'$ and its weight $\omega_C(j_1 j_2)$ is measured by the no. of co-applications (co-apps). On the other hand, we define connections $j_1 j_2 \in E_I$ if jobs j_1, j_2 have the same job title, or normalized job title [6], or common skills [3], or sim-score ≥ 0.9, where the sim-score is the cosine similarity score using the content-based embeddings by DLEM [8]. The weight function $\omega_{\mathcal{F}}$ is defined by an aggregated score on filter scores and distance, where the parameters are decided by the parameter tuning. We normalize both $\omega_{\mathcal{F}}$ and ω_C, then use $\omega_{\mathcal{F}}(uv) + \omega_C(uv) < 1$ for $uv \in E_I \cup E_I'$ as hard filters to filter out weak connections. The job set V_I is split into training (jobs posted over a week), and testing (jobs posted in the next 3 days after the week). For testing, we consider all the test jobs as new and do not have any interaction signal. Overall we use 615,000 pairs of positive training examples and 20 negative examples per node.

We compare the performance of DWGCN with DLEM [8] which is the content-based method to embed job title, description, and skills. DLEM combines 150-dimensional job embeddings with 3-dimensional Cartensian coordinates transformed from the spherical coordinates (latitude and longitude). Furthermore, we conduct ablation studies and consider several variants of aggregator functions defined in [2] for DWGCN: Mean, Meanpool and GCN. For all these variants, we set the depth $K = 2$, sample sizes $s_1 = 25, s_2 = 10$, and the hidden and output dimensions to be both 128.

Table 1 shows the results of the head-to-head comparison between DWGCN compiled with three different aggregator functions and the baseline DLEM [8]. For a source job, a method wins over the other if the overlap between the recommended jobs and the expected top jobs (based on the co-app) is higher. Results show that the embeddings generated by DWGCN outperformed DLEM in finding the most relevant jobs to recommend to the users. Figure 2 illustrates the t-distributed stochastic neighbor embedding (t-SNE) plots of test sample jobs' embeddings obtained from DWGCN-GCN colored with the job categories. This shows that our approach is also effective in capturing the content information of jobs as the job cohorts with different colors are clearly clustered in different regions.

Methods	Win	Lose	Draw
DWGCN-Mean	**54.64%**	29.96%	15.40%
DWGCN-Meanpool	**48.80%**	35.51%	15.68%
DWGCN-GCN	**61.49%**	15.77%	22.75%

Table 1. Head-to-head comparison with DLEM for source to recommended jobs based on co-apps

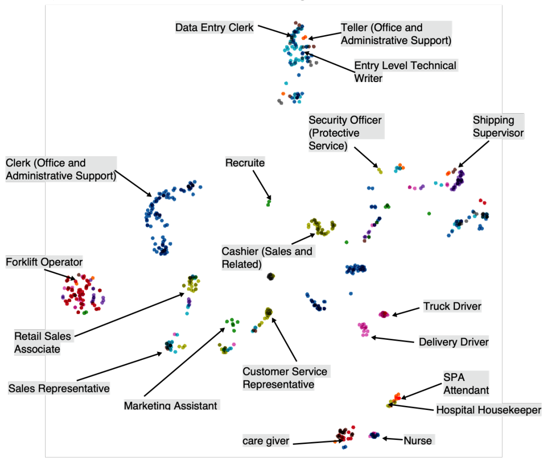


Fig. 2. t-SNE plot of job embedding vectors from DWGCN-GCN which are color-labeled by job categories and annotated with normalized job titles

4 CONCLUSION AND FUTURE WORK

We proposed a weighted GCN-based approach to encode item features and user-item interaction information into the graph structure to represent items. The new embeddings are able to capture content features of items, as well as the behavioral information of users on the related items. Moreover, it can fully overcome cold-start problem. In this work, we only discussed the representation learning for items, but the framework can be easily extended to representation learning on users, or on users and items in the shared space. We are going to to evaluate the performance of our model with A/B experiments and compare with other state-of-art methods.

REFERENCES

[1] Chantat Eksombatchai, Pranav Jindal, Jerry Zitao Liu, Yuchen Liu, Rahul Sharma, Charles Sugnet, Mark Ulrich, and Jure Leskovec. 2018. Pixie: A system for recommending 3+ billion items to 200+ million users in real-time. In *Proceedings of the 2018 world wide web conference*. 1775–1784.

[2] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[3] Faizan Javed, Phuong Hoang, Thomas Mahoney, and Matt McNair. 2017. Large-scale occupational skills normalization for online recruitment. In *Twenty-ninth IAAI conference*.

[4] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[5] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[6] Jingya Wang, Kareem Abdelfatah, Mohammed Korayem, and Janani Balaji. 2019. DeepCarotene-Job Title Classification with Multi-stream Convolutional Neural Network. In *2019 IEEE International Conference on Big Data (Big Data)*. IEEE, 1953–1961.

[7] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. 2018. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 974–983.

[8] Jing Zhao, Jingya Wang, Madhav Sigdel, Bopeng Zhang, Phuong Hoang, Mengshu Liu, and Mohammed Korayem. 2021. Embedding-based Recommender System for Job to Candidate Matching on Scale. *arXiv preprint arXiv:2107.00221* (2021).