Heterogeneous Graph Convolution for Book Recommendations

Matteo Santelmo
EPFL, Lausanne, Switzerland
matteo.santelmo@epfl.ch

Stefano Viel
EPFL, Lausanne, Switzerland
stefano.viel@epfl.ch

***Abstract*—This work investigates the use of modern Graph Convolutional Network (GCN) architectures on Heterogeneous Graphs, targeting a recommendation task. Utilizing the GoodBooks-10K dataset, state-of-the-art methods, namely GraphSAGE and Graph Attention Networks, have been used to generate nodes representation employed in an encoder-decoder architecture. The superiority of these methods over traditional baselines for recommender systems such as matrix factorization is shown in our results. Leveraging the flexibility of Heterogeneous Graphs and sampling-based batching for efficient training on large graphs, we demonstrate that integrating additional information to the bipartite graph representing user-item interactions allow a significant improvement in recommendation relevance precision. Our code is made available on GitHub[1].**

I. Introduction

Given the recent growth in popularity of recommendation system powered by graph approaches, this project aims to showcase a particular example of this type of problem. In the state of the art, providing relevant recommendation is a problem that is addressed with very different approaches, from Matrix Factorization [1] to Neural Collaborative Filtering [2] and Graph Convolutional Networks [3]. This variety arises because domain- and application-specific tasks can lead to different problem definitions, highly dependent also on the type of data that is available.

As further discussed in section II-A and II-E, our approach leverages the flexibility of graph representation to enrich the prediction thanks to heterogeneous information. As a consequence, we focused on methods capable of dealing with appropriate structures and scalable to provide recommendations for new users, based on their purchase history, without requiring additional gradient updates.

As shown in [4], graph-based methods allow to incorporate auxiliary information, such as times and user attributes, while also capturing high-order connections. Unlike traditional approaches like Matrix Factorization [1], which focus on direct interactions, graph-based methods allows to process multi-hops relationships thanks to convolution. This broader context enables them to outperform traditional methods. In this work we will experimented with two architectures, namely GraphSAGE [5] and Graph Attention Networks (GAT) [6].

We modeled the recommendation problem as a link regression task over a bipartite graph. The model predicts the edge label between a user node and a book node as a continuous value from 1 to 5, representing the expected user rating for that book. At inference time, we predict the ratings a user would give to all unread books and recommend those with the highest ratings. Among the other common modeling options for this task, link prediction was discarded as it only predicts whether or not the user will buy a certain book and not his degree of satisfaction. Using this approach would either necessitate discarding ratings below a certain threshold or, if these ratings were retained, it would lead to recommending books similar to those a user purchased but didn't like. Link classification has also been considered, but achieved poor results in experiments. This is likely because the model, treating ratings as separate labels, fails to capture the ordinal relationship between them.

II. Methodology

A. Data description

Our model will be evaluated on the GoodBooks-10K dataset[2]. It is a collection of user ratings and metadata for the 10,000 most popular books on Goodreads. The dataset contains ratings from 53,424 users, where each user has 20 or more ratings. While all user are anonymous each book has metadata information (authors, title, original publication year etc.). The dataset consists in 5,976,479 ratings that users gave to books they read, the ratings go from 1 to 5. Moreover, a list of user-defined genre tags (e.g., fictional, fantasy, education) is also available, together with the count of the times each tag was assigned to each book, and finally a users' *to-read* list is available for each user.

B. Baselines

To evaluate the performance of our model two baselines were considered: a Gaussian random baseline and matrix factorization.

The first baseline is a rather simple **normal distribution** with mean and standard deviation derived from the training data. At inference time we randomly sampled ratings from this distribution for each user-book pair in the test set and evaluated the predicted results.

Secondly, we consider **matrix factorization**: ratings are represented in a matrix X where where the element $(X)_{i,j}$ is the rating given by i-th user to the j-th book, or 0 if the

[1]Code available at: https://github.com/matsant01/graph_RecSys.git

[2]https://github.com/zygmuntz/goodbooks-10k/tree/master

user didn't reviewed the book. Our aim is to find two matrices W and Z such that:

$$X \approx W \cdot Z^\top \tag{1}$$

where $X \in \mathbb{R}^{D \times N}$ is usually a very sparse matrix, while $W \in \mathbb{R}^{D \times K}$ and $Z \in \mathbb{R}^{N \times K}$ are such that $K \ll N, D$. The key idea of matrix factorization is indeed to use previous ratings to compute low dimension representations for users (W) and items (Z), whose inner product estimates the rating that the user would give to the book. In order to compute those low-rank matrices, the following loss is optimized:

$$\min L(W, Z) := \tag{2}$$

$$\frac{1}{2} \sum_{(d,n) \in \Omega} [x_{dn} - (W^T Z)_{dn}]^2 + \frac{\lambda_w}{2} ||W||^2_{Frob} + \frac{\lambda_z}{2} ||Z||^2_{Frob}$$

where Ω represents the set of users and books for which we have a rating. The loss was optimized with stochastic gradient descent and a regularization was introduced to prevent overfitting and stabilize the training. Eventually, the recommendations are selected by estimating X as $W \cdot Z^\top$ and taking the items with highest predicted rating.

C. Heterogeneous graph

We construct a bipartite directed graph from the dataset, with users and books as nodes. An edge exists between a user and a book if the user has rated that book. To allow the message passing in the graph neural network from the books to the user, for each edge from a user to a book we also added the reverse edge.

Specifically we loaded our data with a PyTorch Geometric [7] Heterogeneous Graph. This data structure enables to store nodes and edges of different types and as well as saving edge labels, enclosing all the relevant information and preventing the need for custom manipulation of the graph representation.

To apply the architectures described in the next section II-E the embedding of each node needs to be initialized. For *book* nodes, initial features are computed using SBERT-based models [8] to compute the sentence embedding of a book's title and author. This choice is based on the fact that such models are trained to generate semantically meaningful embeddings of sentences, allowing us to include more information in our graph, and thanks to their extensive pre-training phase they should have meaningful representations also for well-known people such as writers. On the other hand, for *user* nodes, we don't have further information, therefore hand-crafted features have been used to at least provide some information from the graph structure. In particular, degree centrality, PageRank and average rating were chosen to create our initial feature vectors for users. One-hot encoding was also a candidate, but given the considerable number of users (∼50k) this option was discarded.

Thanks to the flexibility of the graph data structure, further experiments were conducted to assess the benefit of providing additional information to the model. Specifically, one node for each author and one for each language were added and connected to the corresponding books nodes. The initialization of the authors' nodes features was again generated with SBERT embedding of the author name. Having instead a very limited amount of different languages (26), each language's node was initialized with a one-hot encoded vector. Our hypothesis is that adding information about authors and languages will help even more exploiting the graph structure and improve the performance, as we expect to see certain users more often connected with books of their native tongue and favorite authors, contributing to the improvement of predictions.

D. Data Splits

To guarantee a fair comparison between models, data was divided into a train, validation and test split. Each split is represented by a graph with all users and books nodes, but only a subset of edges from users to books (each associated with a rating). Specifically, the training set contains 80% of user-book edges with corresponding labels (ratings). The validation set includes the same 80% of user-book edges as the training set without labels (to be used only for message passing) and a 10% of user-book edges not appearing in the previous group (with labels). This allows the computation of the graph convolution on the same edges as the training set, while evaluating the predictions on a new set of user-book edges. Similarly, the test set consists of 90% of the user-book edges without labels (80% from training and 10% from validation) for the message passing and the remaining 10% of user-book edges with labels to evaluate the model. Figure 1 visually explain the splitting technique[3].

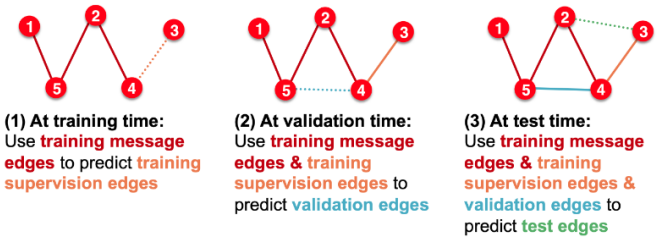

Fig. 1: Graphical explanation of data splitting technique operated on edges.[2]

In the case of the graph with the additional authors and languages nodes, the splitting procedure remains unchanged as it only targets edges between user and book nodes. So, all the books and authors nodes are included in all the splits.

E. Architecture

As anticipated in the introduction, our aim is to compute relevant recommendation by solving an edge regression task. To achieve this, our architecture is composed by an encoder, responsible for the computation of meaningful node embeddings, and a decoder that computes the predicted rating, i.e.

[3]Credits to Zhuoqing Fang for his article available here

the edge label, given the node embeddings of a user and a book.

Two different architectures have been tested and compared for the encoding step, namely GraphSAGE [5] and GAT [6]. Both these techniques can be framed in the message passing framework, where nodes updates their representations iteratively by sending, receiving and aggregating information of their neighborhood. In particular, GraphSAGE key innovation over previous approaches is the possibility to learn the function that aggregates information from a node's local neighborhood. Considering the mean aggregator, that we used in our experiments, the update of the embedding h_u of node v in GraphSAGE is computed as:

$$\mathrm{h}_v^{(k)} = \sigma\left(W^{(k)} \cdot \left[\mathrm{h}_v^{(k-1)} \| \frac{1}{|\mathcal{N}(v)|} \sum_{u \in \mathcal{N}(v)} \mathrm{h}_u^{(k-1)}\right]\right) \quad (3)$$

where $\|$ denotes the concatenation, k refers to the iteration, σ is a non-linear activation function (ReLU in our case) and $\mathcal{N}(v)$ represents a fixed-size uniform sample of nodes from the neighborhood of node v.

The main improvement introduced by GAT comes from the assumption that the influence of neighbors is neither identical nor pre-determined by the graph structure, therefore this method differentiates the contributions of neighbors using attention mechanism and updates the vector of each node by attending over its neighbors.

Moreover, a key issue of simpler GCN and matrix factorization is their inherently transductive nature that doesn't allow them to generalize to unseen data. By learning an aggregation function that summarizes neighborhood information, both GraphSAGE and GAT features have inductive capabilities, which is valuable considering that recommendation systems often deal with dynamic number of users and items.

One linear layer for each node type is employed to project the initial features to the same dimension before applying convolution. Finally, a decoder composed by 3 layer fully connected neural network takes as input the concatenated embeddings from a user node and a book node, and outputs of a scalar representing the predicted rating.

The objective used to trained our model is the same as the one optimized by Matrix Factorization, i.e. the Mean Square Error between the predicted and true ratings, with no regularization. As discussed in the introduction, we also conducted experiments treating the task as edge label classification, using cross-entropy loss with the true rating as the gold label. However, the results were poor, so this approach was discarded.

F. Batching

Given the size of our dataset, computing full gradient updates on the whole dataset is not feasible on standard customer GPUs. To efficiently apply convolutions on big graphs, we employed the batched and sampling approach proposed in [5]. The key idea is to pre-compute smaller graphs (mini-batches) by collecting all the necessary nodes: starting from `batch_size` nodes, a fixed amount of neighbors of each node is sampled and added to the mini-batch, repeating this process `num_conv_layers` times. Appendix A of [5] explains this in detail.

G. Evaluation

To evaluate our models we used a set of classification accuracy metrics, namely *precision@k*, *recall@k*, *F1-score@k* and *MAP@K* (Mean Average Precision at K). It must be noted that these metrics aim to assess the quality of recommendation by focusing on classification between relevant and irrelevant items [9], and don't directly use the ratings value if not for ranking (top-k) and establishing relevance with a threshold. This type of metric is particularly suitable for applications in e-commerce that try to persuade users to purchase products or services [10].

Precision@k is the fraction of recommended items in the top-k set of relevant items. In our case an item is relevant if it has a ground truth rating greater or equal to 4 (which serves as our threshold). The *precision@k* is calculated individually for each user and then averaged across all users. So, for each user we select the top-k books with true ratings greater than the threshold. Then, we compute how many of the predicted ratings for this set of books are higher than the threshold. This count is divided by the number of relevant items, capped at k. In symbols:

$$\text{Precision@k} = \frac{\text{\# of recommended items @k that are relevant}}{\text{\# of relevant items @k}} \quad (4)$$

Recall@k is the proportion of relevant items found in the top-k recommendations. Like *Precision@k*, it is first calculated for each user and then averaged across all users. For a given user, we assess how many of the top-k recommendations (based on predicted ratings) are actually relevant (have a ground truth rating higher than the threshold). Mathematically:

$$\text{Recall@k} = \frac{\text{\# of recommended items @k that are relevant}}{\text{total \# of relevant items}} \quad (5)$$

In order to summarize the values of *recall@k* and *precision@k* into a single metrics we define *F1-score@k* as the harmonic mean of these two.

To avoid bias toward specific values of k, we employed *MAP@k* which computes the average of the *Precision@k* for all the k up to a certain value, as follows:

$$\text{MAP@k} = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{\min(k, n_u)} \sum_{i=1}^{k} \text{Precision@}i(u) \quad (6)$$

where U is the number of users and n_u is the number of relevant items for user u.

We chose *MAP@K* to be our reference metric for two main reasons. Firstly, *precision@k* itself is more relevant than *recall@k*, in fact, if the number of relevant items for

a certain user is greater than k, which is a reasonable case, then there will be an upper bound for the *recall@k*, namely $\frac{\text{k}}{\text{total \# of relevant items}}$, making this metric less appropriate to quantify the relevance of recommended items. Secondly, *precision@k* is strictly dependent on k, therefore it's more robust to consider an average over different number of recommendations.

All of the metrics described above will be computed on a set of user-book pairs for which we already have rating. While this might appear a biased approach, as we are predicting ratings and evaluating them on books that we already know the user will purchase, this choice is needed as we lack ground truth ratings for other potential edges. An alternative approach would have been possible only by defining the problem as an edge prediction task, but this would have required non-trivial negative sampling mechanisms that could have biased the results as well. Moreover, at inference time we can expect the distribution of books that a user will consider to buy to resemble those they have already bought. Thus, the metrics used are justified, as they are computed on a similar distribution during training.

III. Experiments and Results

To assess the best hyperparameters for our model, a grid search has been performed over a subset of our dataset. The dataset was created by sampling a portion of 2,000 users, while retaining all the books and the ratings connected to the chosen users, leading to a sub dataset with distribution of ratings and ratings-per-user significantly close to the original ones. A training, validation and test splits have been derived as described in II-D and used for fair comparison. The parameters of interest for this search are the number of convolution layers, the encoder architecture type, the dimension of node embeddings and the learning rate. The values of *MAP@k* for all the tested models, reported in figure 2, show that GraphSAGE is outperforming GAT and also that increasing the number of convolutions is not improving performance, likely because considering longer distance connection is less informative than just 2-hops relationship (e.g. two users are in the same neighborhood only if they reviewed one common book).

The settings achieving the best performance, except for the learning rate as its linked to the batch size, have been used to train models on the full dataset, both with and without authors and languages nodes. For this last case the number of SAGEConv layers has been increased to allow higher order connectivity, fundamental to consider relations between users and authors or users and languages. The results, presented in Table I, show a significant increase in performance of our GraphSAGE approach compared to baseline, and a further improvement when using additional data.

It must be noted that the values for metrics shown in Table I are considerably lower than the ones achieved on the subset of our dataset (Fig. 2). That's because the recommendation task becomes easier with fewer books if k is fixed, as a relevant book is more likely to appear in the top k. But since the distribution of ratings per user remains consistent our results on the subset dataset generalize to the full dataset.

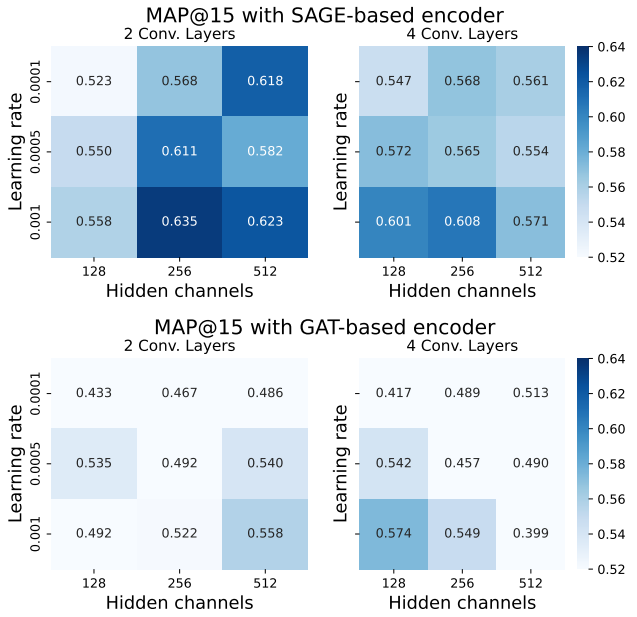

Fig. 2: Mean Average Precision at 15 among different configurations of hyper-parameters, compared over a subset of the GoodBooks-10K dataset.

Model	MAP@15	Precision@5	Recall@5	F1@5
Random Baseline	0.471	0.472	0.332	0.379
Matrix Factorization	0.489	0.494	0.312	0.371
EncDec with SAGE	0.551	0.552	0.347	0.414
*EncDec with SAGE + Additional nodes	**0.593**	**0.596**	**0.380**	**0.450**

TABLE I: Final Experiments results

IV. Conclusions and Future Developments

In this work, we explored the application of modern GCN architectures to recommender system. In particular, we showed that the powerfulness of such methods in conjunction with the flexibility of heterogeneous graphs allow to out perform other approaches. The next step for this work would certainly be a more extensive tuning of hyper-parameters, which was not feasible given our limited amount of computational resources. Future work might then explore the integration of nodes representing other additional features (e.g. genre and tags), which could further improve the performances. Finally, it would be interesting to approach the same problem as a link prediction task, and compare how the performances of the two techniques compare.

References

[1] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, 2009.

[2] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *Proceedings of the 26th international conference on world wide web*, 2017, pp. 173–182.

[3] R. Ying, R. He, K. Chen, P. Eksombatchai, W. L. Hamilton, and J. Leskovec, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, ser. KDD '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 974–983. [Online]. Available: https://doi.org/10.1145/3219819.3219890

[4] ——, "Graph convolutional neural networks for web-scale recommender systems," in *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, 2018, pp. 974–983.

[5] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in neural information processing systems*, vol. 30, 2017.

[6] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," *arXiv preprint arXiv:1710.10903*, 2017.

[7] M. Fey and J. E. Lenssen, "Fast graph representation learning with PyTorch Geometric," in *ICLR Workshop on Representation Learning on Graphs and Manifolds*, 2019.

[8] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," 2019.

[9] F. Hernández del Olmo and E. Gaudioso, "Evaluation of recommender systems: A new approach," *Expert Syst. Appl.*, vol. 35, pp. 790–804, 10 2008.

[10] G. Schröder, M. Thiele, and W. Lehner, "Setting goals and choosing metrics for recommender system evaluations," in *UCERSTI2 workshop at the 5th ACM conference on recommender systems, Chicago, USA*, vol. 23, 2011, p. 53.