Improving LightGCN based recommendation system with NLP metadata embeddings

Pratik Acharya
achary17@purdue.edu
collaboration with Vidhi Jain
Thanks to Prof. Bruno Ribeiro

Abstract

In this study we explored various methodologies relating to recommendation systems using Deep Learning algorithms. LightGCN is one of the state of the art matching systems for recommendation based on collaborative filtering. We apply this to a book recommendation problem which has been growing in popularity recently. LightGCN only uses the user book interactions and not any user or book metadata. Along the way we devise a new technique to initialize embeddings of LightGCN nodes using sentence embeddings from an NLP transformer model called MiniLM. This also means we have a content based recommendation along with collaborative filtering. This improves the performance significantly.

1 Motivation

Shopping, social media, streaming websites and apps actively invest in recommendation systems to provide curated content for each user. This state-of-the-art methodology is used less in the publishing category because reading as a habit had been declining over the past few decades. Over the past 2 years (the pandemic years), we have seen a sudden rise in the number of books people read. We offer 2 pieces of evidence to support our claim: a. The annual revenue of the eBook industry has risen to more than a billion dollars in the US alone. b. The subreddit r/suggestmeabook, where users provide book recommendations to each other, has gained 1 million followers since 2020.

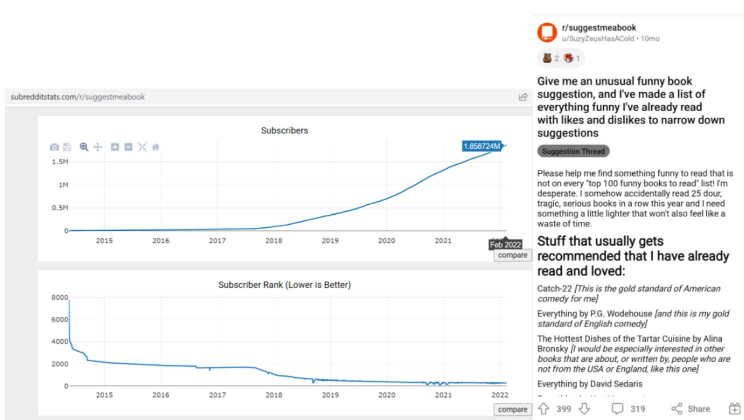


Figure 1: r/suggestmeabook

2 Dataset

We chose the Goodreads books dataset provided by University of California, San Diego. The dataset can be found here. This dataset was collected in the year 2017 and consists of 3 files – 1. Books: Contains information on each book in JSON format. 2. Shelves: Captures the user-book interactions in a CSV format. 3. Reviews: Captures any text-based reviews shared by users. Some specific statistics of the dataset: This dataset contains data pertaining to 2.3M books including 1.5M works, 400k series and 830k authors. It also has dataset pertaining to 870k users and 230M user-book interactions including 112M reading statuses and 104M ratings. An example entry from this dataset is shown in the figure below.


```
{'isbn': '',
 'text_reviews_count': '7',
 'series': ['189911'],
 'country_code': 'US',
 'language_code': 'eng',
# top user-generated shelves for a book, used to define genres by goodreads
 'popular_shelves': [{'count': '58', 'name': 'to-read'},
                     {'count': '15', 'name': 'fantasy'},
                     {'count': '6', 'name': 'fiction'},
                     {'count': '5', 'name': 'owned'}, ...],
 'asin': 'B00071IKUY',
 'is_ebook': 'false',
 'average_rating': '4.03',
 'kindle_asin': '',
# a list of books that users who like the current book also like
 'similar_books': ['19997', '828466', '1569323', '425389', '1176674', '262740', '3743837',
                   '880461', '2292726', '1883810', '1808197', '625150', '1988046', '390170',
                   '2620131', '383106', '1597281'],
 'description': 'Omnibus book club edition containing the Ladies of Madrigyn and the Witches of Wenshar.',
 'format': 'Hardcover',
 'link': 'https://www.goodreads.com/book/show/7327624-the-unschooled-wizard',
 'authors': [{'author_id': '10333', 'role': ''}],
 'publisher': 'Nelson Doubleday, Inc.',
 'num_pages': '600',
 'publication_day': '',
 'isbn13': '',
 'publication_month': '',
 'edition_information': 'Book Club Edition',
 'publication_year': '1987',
 'url': 'https://www.goodreads.com/book/show/7327624-the-unschooled-wizard',
 'image_url': 'https://images.gr-assets.com/books/1304100136m/7327624.jpg',
 'book_id': '7327624',
 'ratings_count': '140',
 'work_id': '8948723',
 'title': 'The Unschooled Wizard (Sun Wolf and Starhawk, #1-2)',
 'title_without_series': 'The Unschooled Wizard (Sun Wolf and Starhawk, #1-2)'}
```


user_id	book_id	is_read	rating	is_reviewed
678442	7381	1	5	0
135202	6166	1	3	0
126691	262380	1	5	0
3224	237810	1	4	0
274050	6955	1	5	0

Figure 2: Goodreads book graph by UCSD

3 Implementation

3.1 Training data generation

To get genuine items, we remove NaNs, winsorize top and bottom 5% of items (books and users) with unusual values, mean, std, count of different attributes, and analysis is done only on English books. Further, remove books with <10 reviews, description length <50 characters, <100 ratings. After this we get around 10 million user-book interactions left, then we draw a random sample to get 2000 unique users, 4500 unique books to add to our graph data structure with 500,000 edges.
To make the graph we used a library called snap in python https://snap.stanford.edu/snappy/doc/reference/graphs.html. This was useful to return a k-connected-graph for pytorch geometric. To do the train test split, we use a function called RandomLinkSplit in the pytorch geometric library, which does an inductive split as shown in the figure, the nodes are same in train, val and test splits. Only the edges are distributed.

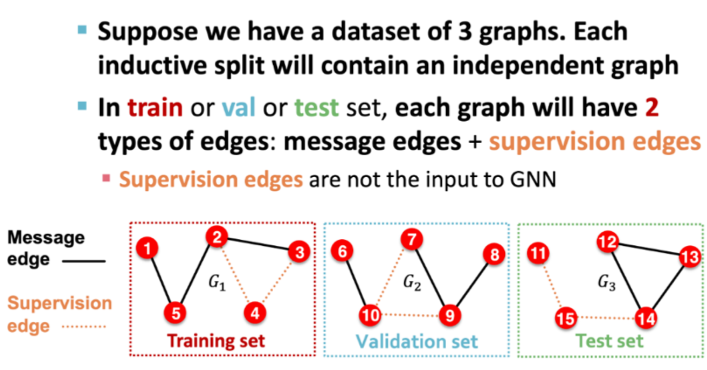


Figure 3: inductive train split

3.2 LightGCN

LightGCN is a type of GCN which is not as computationally expensive. A comparison of the propagation rules is given in the figure below. In LightGCN, we remove activations and other weight matrices except layer 0 embeddings which are learnt, and simplify the formula for aggregation. This means the model is much faster than usual GNN and will suit our large samples from the dataset. For each node, we take the average of embeddings of all the layers. Number of layers here represent maximum degrees of separation between users/books that are averaged to generate the final embeddings. We didn't see a significant change in the performance on adding more layers so we chose to keep 3 layers for our model which aids computation. It is a very simple model where we just make lightgcn layers and take mean after propagation.

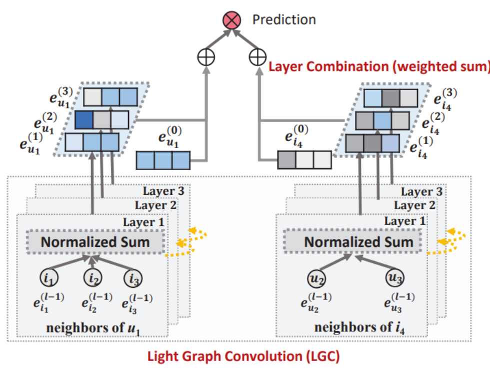


Figure 4: LightGCN Model Architechture from the paper

$$e_u^{(k+1)} = \sigma\left(W_1 e_u^{(k)} + \sum_{i \in \mathcal{N}_u} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(W_1 e_i^{(k)} + W_2(e_i^{(k)} \odot e_u^{(k)}))\right)$$

$$e_i^{(k+1)} = \sigma\left(W_1 e_i^{(k)} + \sum_{i \in \mathcal{N}_i} \frac{1}{\sqrt{|\mathcal{N}_u||\mathcal{N}_i|}}(W_1 e_u^{(k)} + W_2(e_u^{(k)} \odot e_i^{(k)}))\right)$$

N_u: the set of all neighbors of user u (items liked by u)
N_i: the set of all neighbors of item i (users who liked i)
$e_u^{(k)}$: k-th layer user embedding
$e_i^{(k)}$: k-th layer item embedding
W_1, W_2 : trainable weight matrices

Figure 5: GCN propagation rule

$$e_u^{(k+1)} = \sum_{i \in N_u} \frac{1}{\sqrt{|N_u|}\sqrt{|N_i|}} e_i^{(k)} \quad e_i^{(k+1)} = \sum_{u \in N_i} \frac{1}{\sqrt{|N_i|}\sqrt{|N_u|}} e_u^{(k)}$$

Figure 6: LightGCN propagation rule

The loss function used here is Bayesian Personalized Loss. It is commonly used for recommendation problems. Note: here we are also sampling negative edges randomly, since in our training edges exist only between users and items that are rated positively. Negative edges are sampled such that they equal positive edges. BPR loss in PyG is shown in figure below, we use similar score but without regularization.

$$L_{BPR} = -\sum_{u=1}^{M} \sum_{i \in N_u} \sum_{j \notin N_u} \ln \sigma(\hat{y}_{ui} - \hat{y}_{uj}) + \lambda ||E^{(0)}||^2$$

$\hat{y}_{ui}$: predicted score of a positive sample
$\hat{y}_{uj}$: predicted score of a negative sample
λ: hyperparameter which controls the L2 regularization strength

Figure 7: BPR Loss

```
reg_loss = lambda_val * (users_emb_0.norm(2).pow(2) +
                         pos_items_emb_0.norm(2).pow(2) +
                         neg_items_emb_0.norm(2).pow(2)) # L2 loss

pos_scores = torch.mul(users_emb_final, pos_items_emb_final)
pos_scores = torch.sum(pos_scores, dim=-1) # predicted scores of positive samples
neg_scores = torch.mul(users_emb_final, neg_items_emb_final)
neg_scores = torch.sum(neg_scores, dim=-1) # predicted scores of negative samples

loss = -torch.mean(torch.nn.functional.softplus(pos_scores - neg_scores)) + reg_loss
```

Figure 8: BPR Loss in PyG

3.3 MiniLMv2

MiniLM is trained to mimic the self attention module of a transformer model. We have chosen this model specifically over other transformers because our dataset has 2 million books and to generate embeddings for all descriptions, we need a fast model. As we can see in the benchmark comparison, MiniLM is near the top and is one of the best in terms of accuracy.

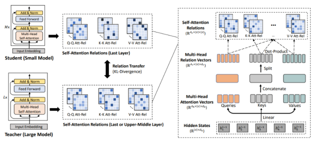


Figure 9: MiniLM Model Architechture from the paper

Model Name	Performance Sentence Embeddings (14 Datasets)	Performance Semantic Search (6 Datasets)	Avg. Performance	Speed	Model Size
paraphrase-MiniLM-L3-v2	62.29	39.19	50.74	19000	61 MB
all-MiniLM-L6-v2	68.06	49.54	58.80	14200	80 MB
multi-qa-MiniLM-L6-cos-v1	64.33	51.83	58.08	14200	80 MB
paraphrase-multilingual-MiniLM-L12-v2	64.25	39.19	51.72	7500	420 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
paraphrase-albert-small-v2	64.46	40.04	52.25	5000	43 MB
distiluse-base-multilingual-cased-v1	61.30	29.87	45.59	4000	480 MB
distiluse-base-multilingual-cased-v2	60.18	27.35	43.77	4000	480 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	290 MB

Figure 10: MiniLM benchmark with other models via huggingface

To check whether the embeddings we get are genuine, we cluseter this using kmeans and DBSCAN. Each of the clusters gave a detailed genre. Examples are shown in figure below. This verifies that the embeddings are making sense.

```
'The Christmas Wedding', 'Merry Ex-Mas', 'A Christmas Present',
'Silver Bells', 'Snow Globe',
'Sleigh Ride (Minnesota Christmas, #2)',
'Gus Was A Friendly Ghost', 'The Fifth Day of Christmas',
'Big Girls Do It on Christmas (Big Girls Do It, #6)',
'12 Terrors of Christmas', 'The Not Very Merry Pout-Pout Fish',
'A Scandal in Battersea (Elemental Masters, #12)',
'Western Kisses', 'The Holiday Bride (Holiday Brides, #2)',
'Silent Night 3 (Fear Street Superchiller, #11)',
'Christmas with an Angel (Harmony Harbor, #1.5)',
'Christmas in Harmony (Harmony, #2.5)',
'Choosing Christmas (Piper Anderson, #2.5)',
'Christmas in Paradise (TJ Jensen Paradise Lake Mysteries #4)',
'The Christmas Bouquet / Bayside Retreat (Chesapeake Shores, #11 & 12.5)',
'Twelve Days of Christmas',
"Christmas at Leo's (Memoirs of a Houseboy, #5)",
"Merry's Christmas: a love story (Redeeming Romance Series)",
'A Regency Christmas: Scarlet Ribbons / Christmas Promise / A Little Christmas'
'Silent Knife (Celebration Bay, #2)',
'Love Finds You in Folly Beach, South Carolina',
'Sprinkles on Top (Sugar Springs, #3)',
"Ruth's First Christmas Tree  (Ruth Galloway, #4.5)",
'I Heart Christmas', 'Waiting in Vain'], dtype='<U246')
```

Figure 11: Books with christmas theme

```
(['Honey So Sweet, Vol. 2', 'White Crane (Samurai Kids, #1)',
  'Tsubasa: RESERVoir CHRoNiCLE, Vol. 13', 'Sanshirō',
  'Maid-sama! Vol. 01 (Maid-sama!, #1)', "La Corda d'Oro, Volume 1",
  'Naruto, Vol. 66: The New Three (Naruto, #66)',
  'Strobe Edge, Vol. 5 (Strobe Edge, #5)', 'Skip Beat!, Vol. 12',
  'Close The Last Door, Volume 01',
  'Twittering Birds Never Fly, Vol.1',
  'The Wallflower, Vol. 3 (The Wallflower, #3)',
  'キスよりも早く1 [Kisu Yorimo Hayaku 4] (Faster than a Kiss #4)',
  'Your Lie in April, Vol. 4', 'Noragami: Stray God, Vol. 12',
  'The Ring of Sky (Young Samurai, #8)',
  'The Fox Woman (Love/War/Death, #1)',
  'Punch!, Vol. 3 (Punch!, #3)', 'Godchild, Volume 04',
  'Bleach, Volume 07', 'The Decay of the Angel',
  'Dragon Ball Z, Vol. 21: Tournament of the Heavens (Dragon Ball Z, #21)',
  'Get Well, Pikachu! (Pokemon Reader #6)',
  'What Did You Eat Yesterday?, Volume 4',
  'Your Lie in April, Vol. 4',
  'A Silent Voice, Vol. 7 (A Silent Voice, #7)',
  'Night of Cake & Puppets (Daughter of Smoke & Bone, #2.5)',
  'Sand Chronicles, Vol. 9', 'A Pale View of Hills',
  'Durarara!! Saika Arc, Vol. 1'], dtype='<U246')
```

Figure 12: Japanese Fiction

Using the book descriptions, we get sentence embeddings of size 384 for each book. Now we use dimensionality reduction to get 10 dimensional embeddings for books. To make embeddings for users, we take a rating weighted average of book embeddings, of books read by that user. Now we standardize these embeddings and this gives initial user and book embeddings for all nodes. The process is shown below.

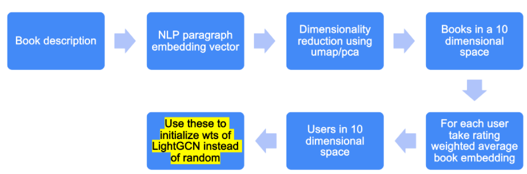


Figure 13: Process of making the embeddings

4 Results

We find that the model trains quickly 30 minutes for 30 epochs considering that the dataset is large. The validation recall quickly reaches the asymptotic value. With increase in embedding size of each node, the validation recall increases slightly.So we fix a size of 10 here in the 3 layers deep model.

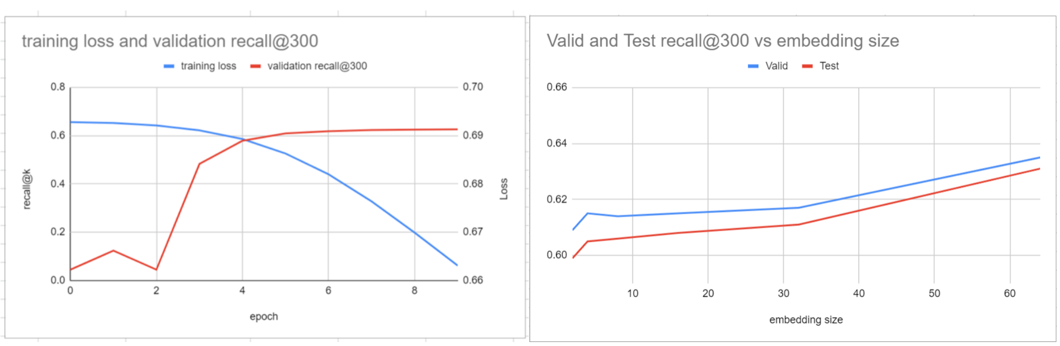


Figure 14: Training LightGCN for goodreads data

Now after initializing node embeddings with NLP embeddings from MiniLM(dimension reduced to 10), we can see 2 advantages immidiately. The convergence is very quick, almost in 1 epoch vs 10-30 for normal sampled embeddings.

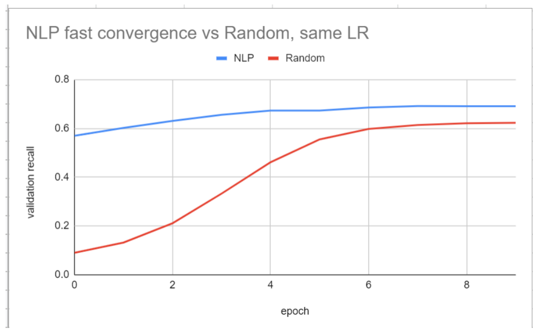
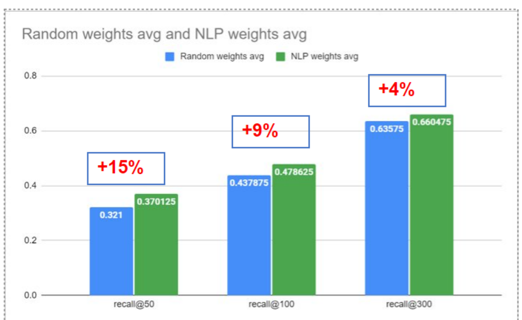


Figure 15: Fast convergence for NLP embeddings

Another very big advantage is the performance improvment in the recall score after convergence. We can see the results of 8 samples in the table below and the corresponding paired t tests. With a very low p value, the recall@50 has increased as much as 15%, which is huge considering that LightGCN is already state of the art. The performance has increased significantly for all recall scores.



	Difference of recall means in favor of NLP weights, paired t tests			
			95%CI	
	Difference of means	p-value	conf.low	conf.high
recall@50	0.049	0.000088**	0.034	0.063
recall@100	0.040	0.0002**	0.026	0.054
recall@300	0.024	0.0213*	0.004	0.044

Test				
seed		recall@50	recall@100	recall@300
5	Random weights	0.317	0.442	0.646
	NLP weights	0.378	0.49	0.673
50	Random weights	0.326	0.453	0.652
	NLP weights	0.381	0.484	0.668
500	Random weights	0.32	0.429	0.637
	NLP weights	0.342	0.44	0.611
5000	Random weights	0.309	0.432	0.632
	NLP weights	0.373	0.484	0.668
50000	Random weights	0.303	0.414	0.633
	NLP weights	0.372	0.482	0.6648
10	Random weights	0.311	0.439	0.64
	NLP weights	0.366	0.477	0.659
100	Random weights	0.34	0.443	0.619
	NLP weights	0.378	0.489	0.672
1000	Random weights	0.342	0.451	0.627
	NLP weights	0.371	0.483	0.668
	Random weights	0.321	0.437875	0.63575
	NLP weights avg	0.370125	0.478625	0.660475

Figure 16: Comparison of both embeddings performance

5 Conclusion and Discussion

Using LightGCN+NLP embeddings initialization we were able to recommend books with a recall@300 score of 66%, which means that 66% of the relevant books in test data are recommended in top 7% recommendations. This is a very good score. Using these embeddings as initial values in LightGCN model, we were able to improve the recall@50 performance by as much as 15%(statistically significant for 8 samples), which is a lot considering that this model is sota for recommendation tasks.

6 Comments from the presentation

Yes this is a very simple permutation invariant model.
I haven't made the LightGCN architecture or trained the MiniLM model. It is pretrained, downloaded from huggingface.
My idea is used to improve the performance of sota LightGCN for book recommendation purpose.
For this i try a new idea to generate embeddings from the book description which are later used to initialize embeddings for the user and book nodes of LightGCN.

7 Future ideas

- Test this technique on other graph based recommendation networks
- Does the performance of the transformer change recall score
- Try different dimension reduction techniques
- Add metadata other than book description by adding sentences like this book is long/short etc. based on number of pages, also add written by this author
- Include authors in the graph as nodes

References

[1] Wang, W., Bao, H., Huang, S., Dong, L., Wei, F. (n.d.). MINILMv2: Multi-Head Self-Attention Relation Distillation for Compressing Pretrained Transformers ArXiv:2012.15828v2 [cs.CL] 27 Jun 2021 https://arxiv.org/pdf/2012.15828.pdf

[2] He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., Wang, M. (2020, July 7). LIGHTGCN: Simplifying and powering graph convolution network for recommendation. https://arxiv.org/abs/2002.02126

[3] Wang, X., He, X., Wang, M., Fung, F., Chua, T.-S. (n.d.). Neural graph collaborative filtering - https://arxiv.org/pdf/1905.08108.pdf

[4] Gao, C., Zheng, Y., Li, N., Li, Y., Qin, Y., Piao, J., Quan, Y., Chang, J., Jin, D., He, X., Li, Y. (n.d.). Chen Gao* , Yinfeng Li , Yingrong Qin ... - Graph Neural Networks for Recommender Systems: Challenges, Methods, and Directions https://arxiv.org/pdf/2109.12843.pdf