FourthBrain

GNN Based E-Commerce Recommender System

Ying Kang

Outline

- Opportunity - ecommerce conversion
- Solution - a better recommender
- Data + Model
- Demo
- MLOps Stack/Iterations
- Conclusions
- Future Work

Appendix

- Model architecture and evaluation matrix
- Production infrastructure
- Lessons Learned

Opportunity

- Worldwide E-commerce sales: Over $ 5.2 trillion
  - Conversion Rate in the US: ~2.3% for 2Q 2022
  - Conversion Rate in UK: 4%
- Improve from 2.3% to 4% => 73.9% increase in sales
- Big opportunity

- **Hypothesis**: Personalized recommendation will improve conversion rate
  - Maximize the probability of buying

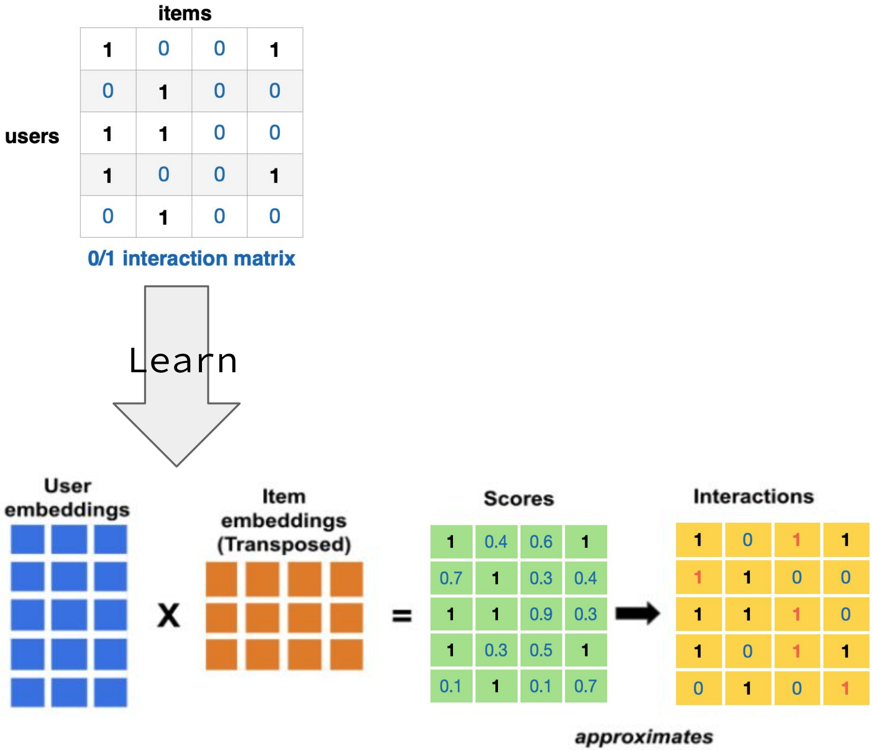
Solutions

- Better recommender => higher conversion rate => more sales!
- Use Graph Neural Network LightGCN
- Based on a research paper in 2020 He, Xiangnan, et al. “LightGCN: Simplifying and powering graph convolution network for recommendation.” 2020
- Outperforms widely used Matrix Factorization models by 35%

Collaborative Filtering(CF) + Matrix factorization(MF)

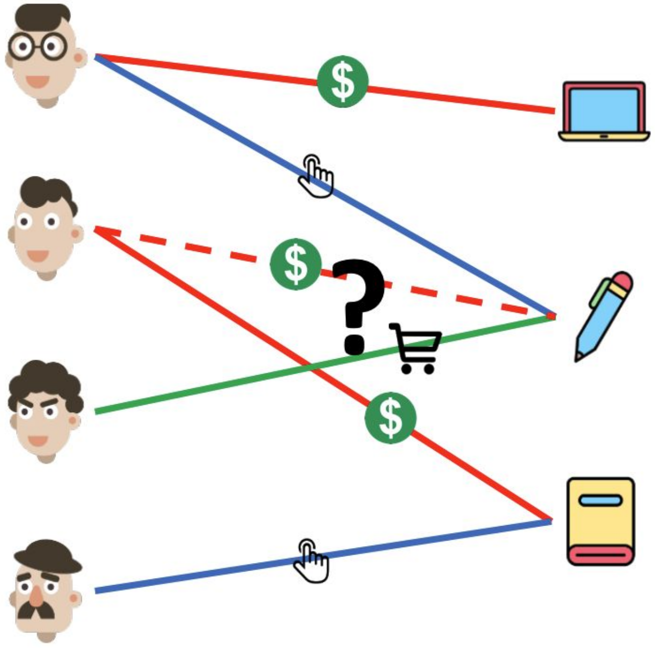
Classic RecSys uses CF with simple MF

- Use MF to learn User and Item embeddings from user-item interaction matrix
- Use learned embeddings to estimate probability of new user-item interactions
- Recommend items to user where estimated user-item probability is high



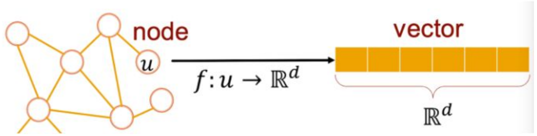
E-commerce is Multi-behavior

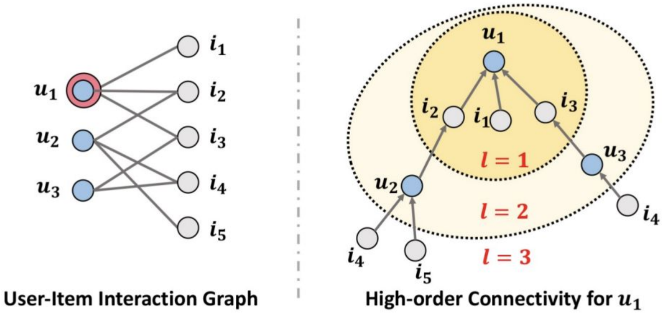
- A consumer interacts with items in multiple forms: view, add to cart, remove from cart, purchase, etc
- **Input**: user-item interactions of different forms
- **Output**: user-item interaction probability on target behavior
  - Purchase



Why GNN

- GNN learns User and Item purchase probability by leveraging graph structure.
- Extends reach to k-hop neighbors.
- Addresses data sparsity issue in e-commerce.
  - i.e. a consumer only interacts with a tiny percentage of items





Figures are from:
Wang et al. Neural Graph Collaborative Filtering, SIGIR 2019

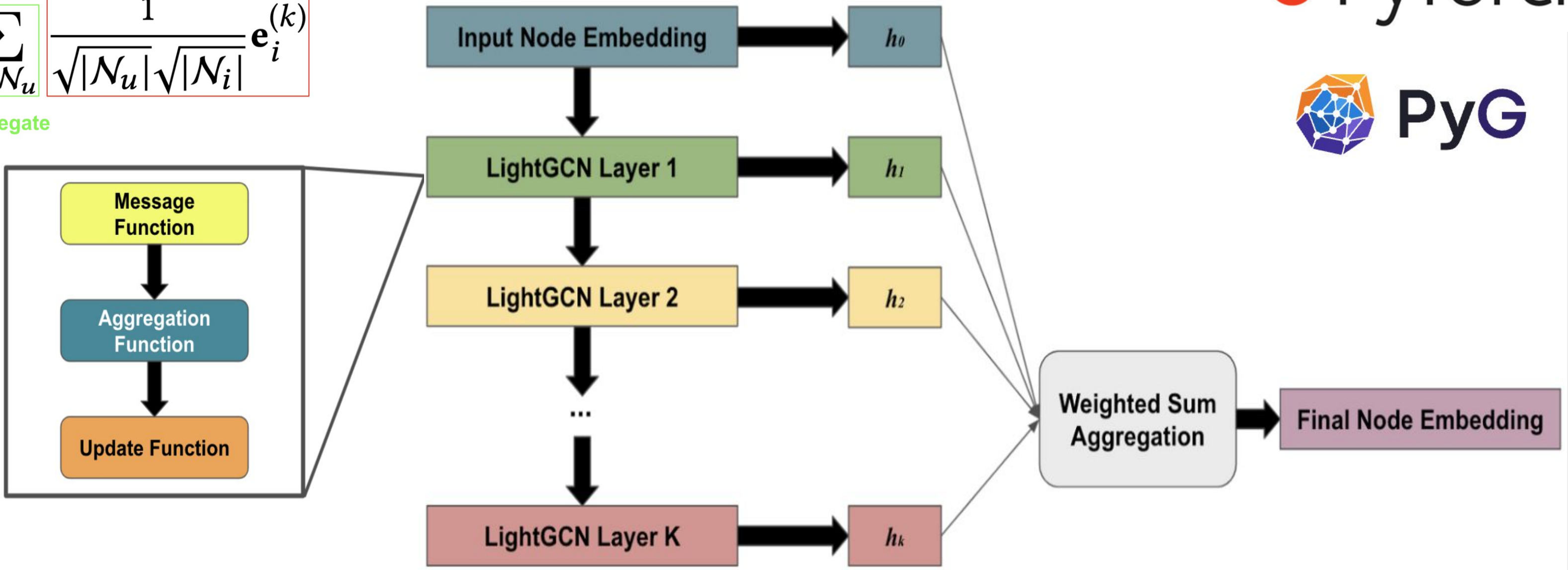
LightGCN Model architecture

Objective Function: $$BPR(u) = \frac{1}{|E| \cdot |E_{neg}|} \sum_{(u,v_{pos})\in E(u)} \sum_{(u,v_{neg})\in E_{neg}(u)} -\log\left(\sigma(f_\theta(u, v_{pos}) - f_\theta(u, v_{neg}))\right)$$

$$\mathbf{e}_u^{(k+1)} = \underbrace{\boxed{\sum_{i\in\mathcal{N}_u}}}_{\text{Aggregate}} \overbrace{\boxed{\frac{1}{\sqrt{|\mathcal{N}_u|}\sqrt{|\mathcal{N}_i|}}\mathbf{e}_i^{(k)}}}^{\text{Message}}$$

PyTorch

PyG



A high level outline of the LightGCN model and aggregation of outputs.

Data

- Kaggle eCommerce Events History in Cosmetics Shop
  - Oct 2019 - Jan 2020
- Number of consumers: 1.64 million
- Number of products: 54.6 thousand
- 4 types of events: view, add to cart, remove from cart, purchase.
- 20 million events in total

Evaluating Model performance

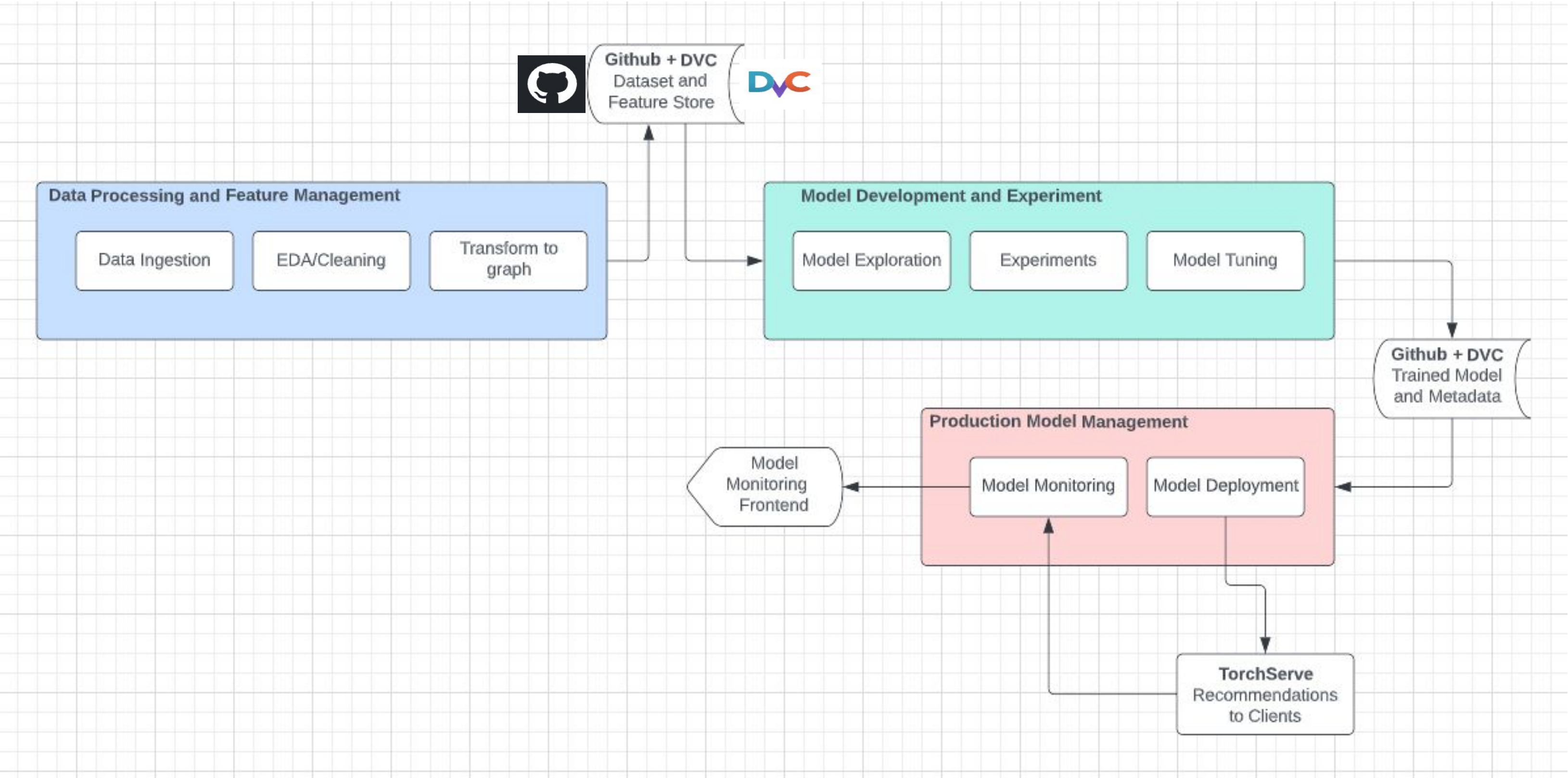
- Evaluation Metric (Recall@K):
  - LightGCN 0.171, Matrix Factorization 0.127
  - Recall@20 - Out of actual bought items, what percentage are in the recommended list of 20

Higher Recall@K → higher Conversion Rate → higher sales

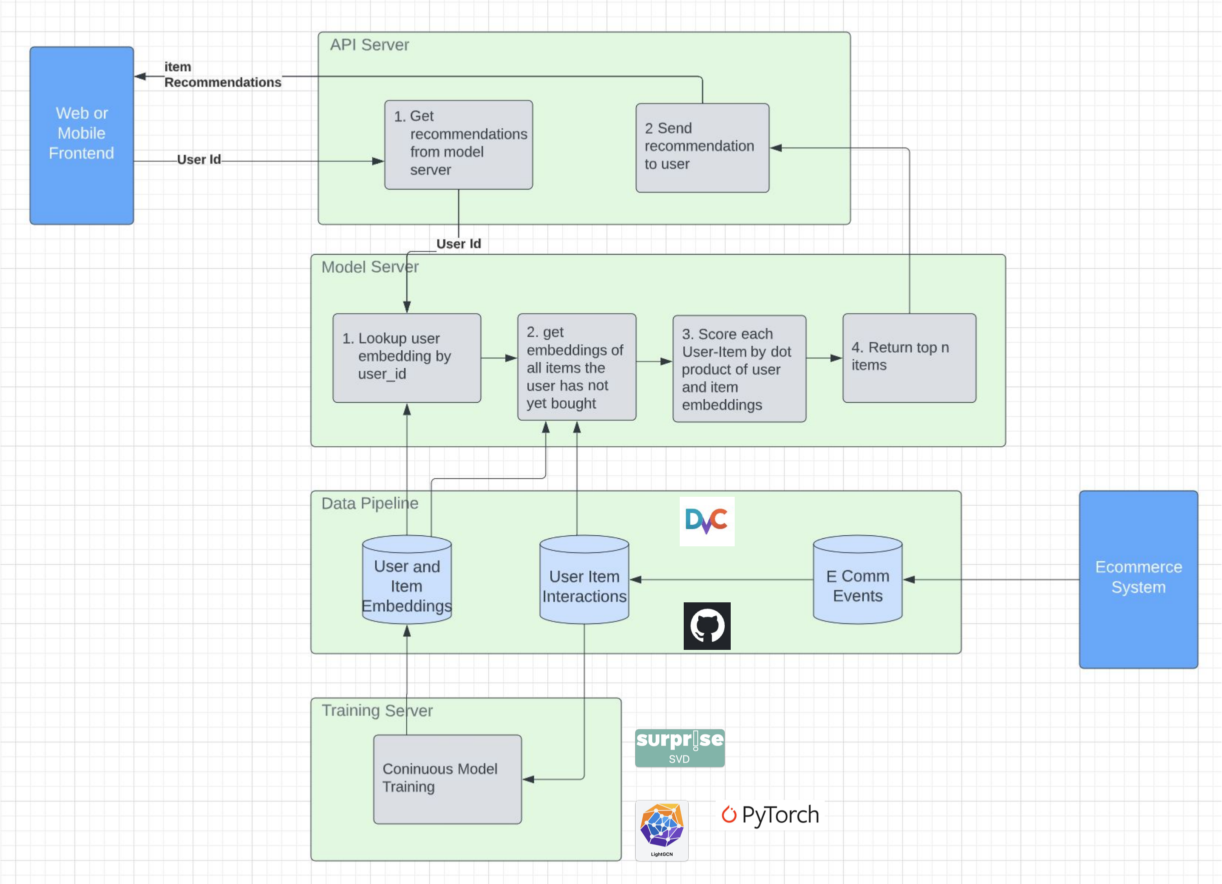
Demo

- Visual of the consumers, products and their connectivity
- Pick one consumer, make a call to the recommender deployed on AWS to recommend 20 products
- Explainability: the paths from consumer to recommended items.

MLOps Stack


Github + DVC
Dataset and
Feature Store
Data Processing and Feature Management
Data Ingestion
EDA/Cleaning
Transform to
graph
Model Development and Experiment
Model Exploration
Experiments
Model Tuning
Github + DVC
Trained Model
and Metadata
Production Model Management
Model Monitoring
Model Deployment
Model
Monitoring
Frontend
TorchServe
Recommendations
to Clients

Production Infrastructure



Conclusions

- state-of-the-art GNN eCommerce recommender based on very recent paper
- Experimented on real world dataset
- 35% better performance than classic Matrix Factorization model.
- Better recommendation => higher conversion rate!
- Recommendations visually explainable
- End-to-end integrated data and ML workflow, using Git and DVC.
- Highly scalable production TorchServe on AWS
- Cost: only 4 hours of sleep every night

Future Work

- What we would do next…
  - Optimize model with other metrics, and use ensemble of multiple models
  - GNN's strong power to learn from graph-structured data(item category, Price bucket)
  - Take Session Feature into consideration

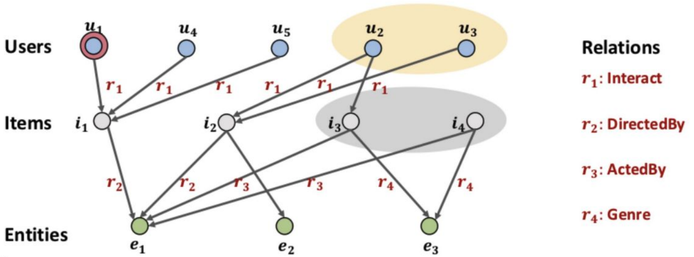
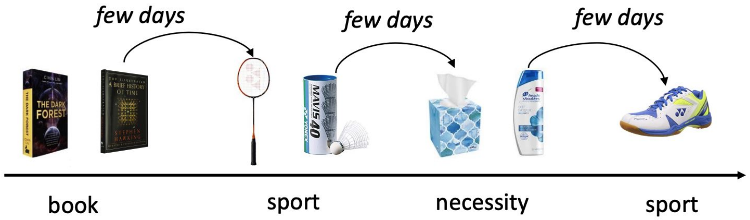


Figure from:
Wang et al. KGAT: Knowledge Graph Attention Network for Recommendation, KDD 2019



Lessons learned

- Start early, really early
- Set up branching and versioning process in the very beginning, and stick with it.
  - Will save you from wasting many many hours
- Do a trial training with the whole dataset as soon as possible.
  - Need to optimize GPU memory use in code
- Library code may have bugs.
  - Be prepared to debug into third party code
- Continuous refactoring
  - Don't leave critical code in Jupyter Notebook for too long.
  - Organize code in classes and functions => Object Oriented programming
- Prepare for production release early
  - don't need a best tuned model to begin
- Tell your cat upfront you won't have time for him in the coming weeks.

Thank You!
Questions?

Appendix

Model use cases

Recommend items for consumers who has buying or viewing history.

For New consumers, new items: popularity recommender, random recommender, content filtering recommender

Data lineage

Raw Data
(20 Million * 9 columns)

(20692840, 9)

	event_time	event_type	product_id	category_id	category_code	brand	price	user_id	user_session
18439902	2019-10-14 07:49:19 UTC	view	5889092	1720400165430363096	NaN	NaN	3.65	560038527	d380fec1-4eee-4b40-940e-25ff5305571b
16072835	2019-11-28 10:55:53 UTC	remove_from_cart	5793077	1487580007776650194	NaN	NaN	0.75	478239488	f6d4d746-424a-4acb-9bc5-ac2ea7ac05a1
2834254	2020-01-22 11:03:23 UTC	cart	5786837	1783999068909863670	NaN	smart	5.56	585185095	94cdcb42-9f6f-45d8-90b9-3770d39f3251
16844423	2019-10-02 13:20:46 UTC	view	5848407	1487580007675986893	NaN	freedecor	0.79	533866480	7e922394-c369-4ae9-99a0-fc1b8495ba2d
15635685	2019-11-25 12:25:40 UTC	view	5886802	1487580013170524342	NaN	NaN	5.56	509522345	69bbbf16-fb51-4ec9-bf50-8aa14ea5c5cb
1298770	2020-01-11 21:36:56 UTC	view	5916489	1487580013279576251	NaN	relouis	4.13	599906722	a55de4b7-0a87-4ae1-b5a0-f8f657764ef6
2691355	2020-01-21 11:33:24 UTC	view	5899164	1487580005293622112	NaN	opi	26.95	586476743	e292d58c-ba70-4342-81cf-81c2a14fe5a8
2927998	2020-01-23 04:14:22 UTC	remove_from_cart	5857673	1487580007399162817	NaN	artex	4.37	422178927	688e954e-668a-452a-a752-43af00635f0a

User-item Interaction Matrix
(10 Million * 4 columns)

	user_id	product_id	event_type	weight
0	465496	5769989	[view]	0.01
1	465496	5865524	[view]	0.01
2	465496	5865526	[view]	0.01
3	1120748	5240	[view]	0.01
4	1180452	5881337	[view]	0.01
...	...	...	...	...
10157403	622090043	5850628	[view]	0.01
10157404	622090052	5688691	[view]	0.01
10157405	622090052	5931986	[view]	0.01
10157406	622090098	5650609	[view]	0.01
10157407	622090237	5754853	[view]	0.01

10157408 rows × 4 columns

surprise

SVD

Edge Index Format

```
tensor([[ 769,  168,  326,  ..., 1683, 2006,  989],
        [1192, 1272, 1085,  ...,  601,  621,   59]])
```

LightGCN

Ethical Considerations

Some ethical considerations of a recommender are:

- Does the system recommend inappropriate products?
- Privacy consideration: can sensitive information about a user be inferred by observing the recommendations that the system generates?
- Does the system limit recommendations in a way that nudges users in particular directions? It is possible to add an experiment to this project to introduce some randomness in the recommendation and see how a small % of randomness effects model accuracy.