Edge-RelGCN research plan

2022.03.11
Han, Donghee

index

Modeling User Behavior with Graph Convolution for Personalized Product Search

[Modeling User Behavior with Graph Convolution for Personalized Product Search]()

GCN을 통해 user-item review 데이터를 모델링

사용자 행동 session 단위로 아이템과 연결

text 정보는 LSTM, Transformer 등을 사용해 벡터로 변환

→ 최종 단계에서 아이템 벡터와 함께 연산하여 유저벡터를 구

→ 리뷰 텍스트 벡터를 아이템(노드) 임베딩으로 사용

PS. *ACM Web Conference* 에 흥미로운 연구가 많이 보임

https://www2022.thewebconf.org/accepted-papers/

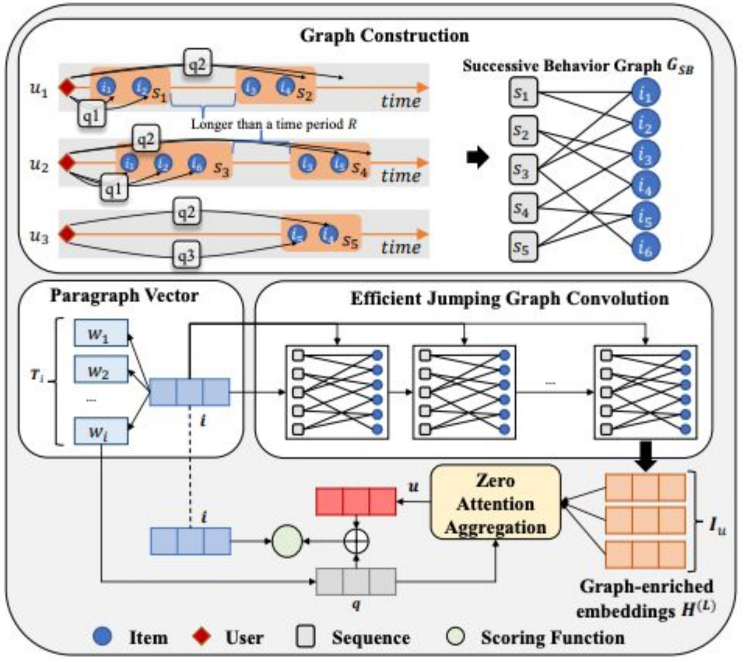
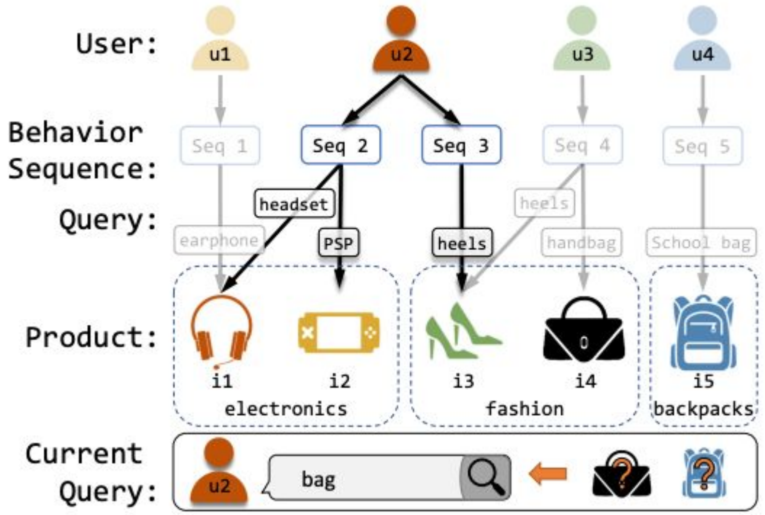


Figure 2: The framework of our proposed SBG model.

Modeling User Behavior with Graph Convolution for Personalized Product Search



데이터셋 상에는 query가 없어 제품 카테고리 정보를 쿼리로 대신 사용

사용자 모델에 따라 같은 'bag'를 쿼리해도, 누군가에게는 백팩을 다른 유저에게는 핸드백을 추천할 수 있음

Amazon Dataset

Modeling User Behavior with Graph Convolution for Personalized Product Search

Amazon review Dataset 만으로 실험 진행

Domain 을 다양화 하여 실험 진행

실험용 데이터셋

Movie - 50만개 최신 리뷰 사용 예정

Clothing - 27만개 최신 리뷰 사용 예정

review 가 최소 5개 이상인 item만 사용

Timestamp 기준으로 Train / Valid / Test 분할

inductive 이므로 unseen item 에 대한 효과를 비교해볼 수 있

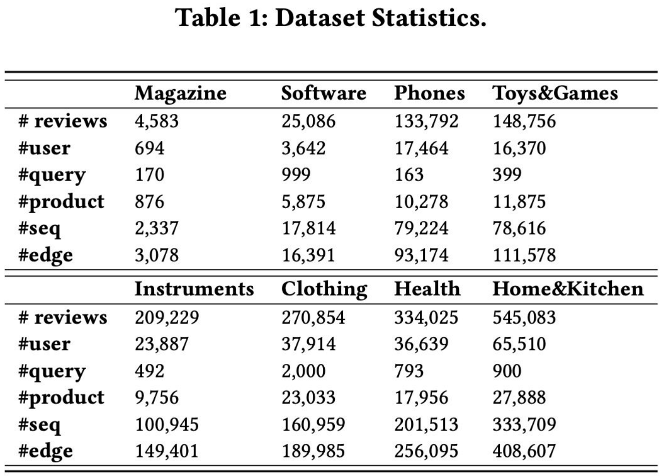


Table 1: Dataset Statistics.

	Magazine	Software	Phones	Toys&Games
# reviews	4,583	25,086	133,792	148,756
#user	694	3,642	17,464	16,370
#query	170	999	163	399
#product	876	5,875	10,278	11,875
#seq	2,337	17,814	79,224	78,616
#edge	3,078	16,391	93,174	111,578
	Instruments	**Clothing**	**Health**	**Home&Kitchen**
# reviews	209,229	270,854	334,025	545,083
#user	23,887	37,914	36,639	65,510
#query	492	2,000	793	900
#product	9,756	23,033	17,956	27,888
#seq	100,945	160,959	201,513	333,709
#edge	149,401	189,985	256,095	408,607

BERT feature extraction

다양한 BERT 기반 모델을 통한 Review feature extraction
Emotion, Sentiment 벡터를 함께 활용 (교수님 제안)

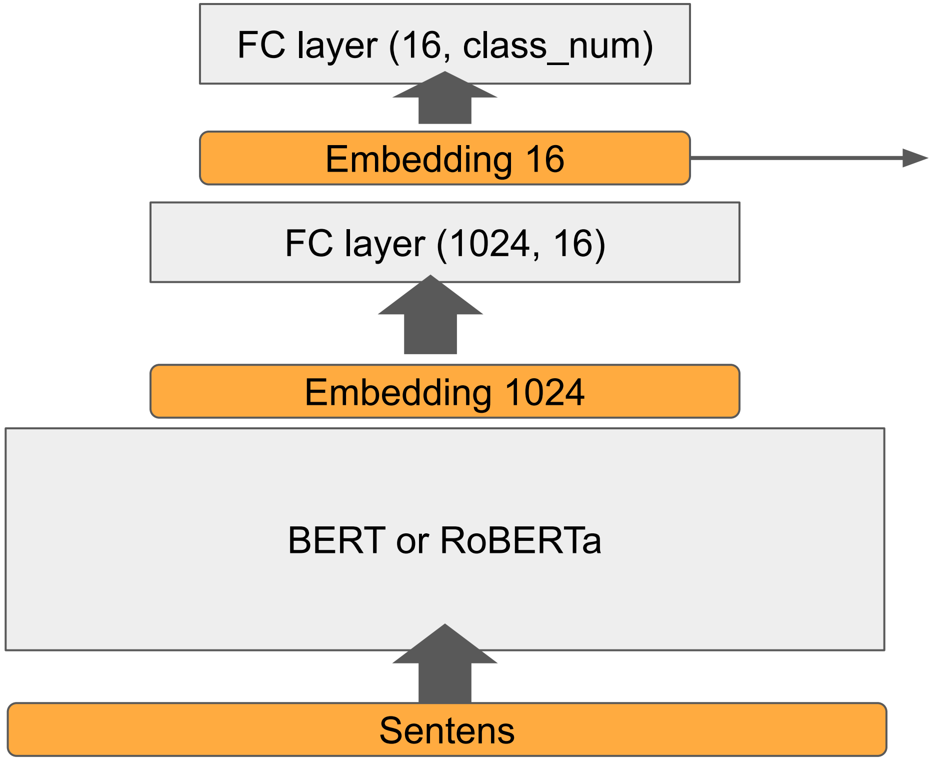
GoEmotion - multi-hot emotion classification
https://huggingface.co/datasets/go_emotions

SieBERT - RoBERTa 기반의 sentiment 분석 (0,1)
https://huggingface.co/siebert/sentiment-roberta-large-english

Negative, Neutral, Positive 로 분류
https://huggingface.co/cardiffnlp/twitter-roberta-base-sentiment

→ BERT, RoBERTa PTM은 각각 768, 1024의 embedding vector size 를 가지고 있음
→ 더 작은 embedding vector 활용을 위해서는 별도로 custom 모델 생성 후, fine-tuning 진행해야함
→ 우선은 emotion vector 만을 사용하는 실험 진행
(현실적으로 메모리에 다 올리기가 어려워짐)

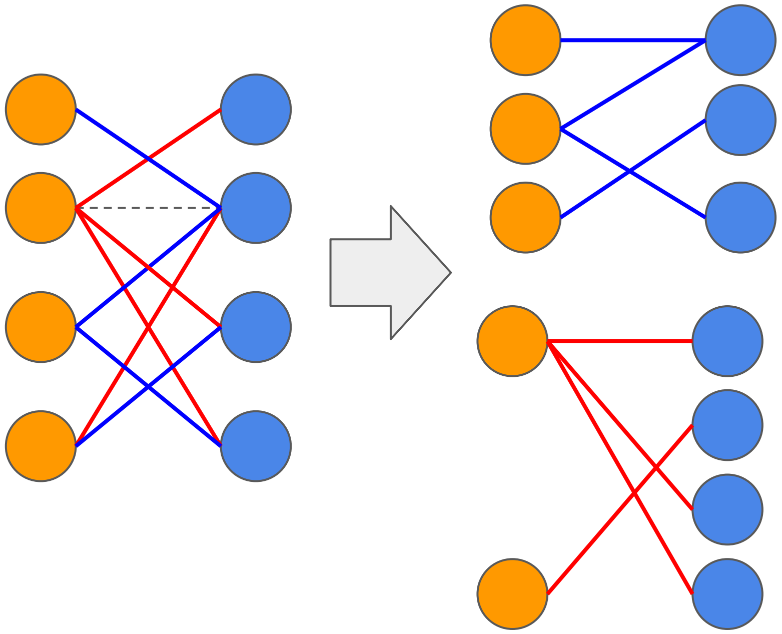
BERT feature extraction



classification layer 직전의
embedding vector를 edge feature 로 사용

Edge-RelGCN

R-GCN 모델에 Edge feature 를 더하는 형태



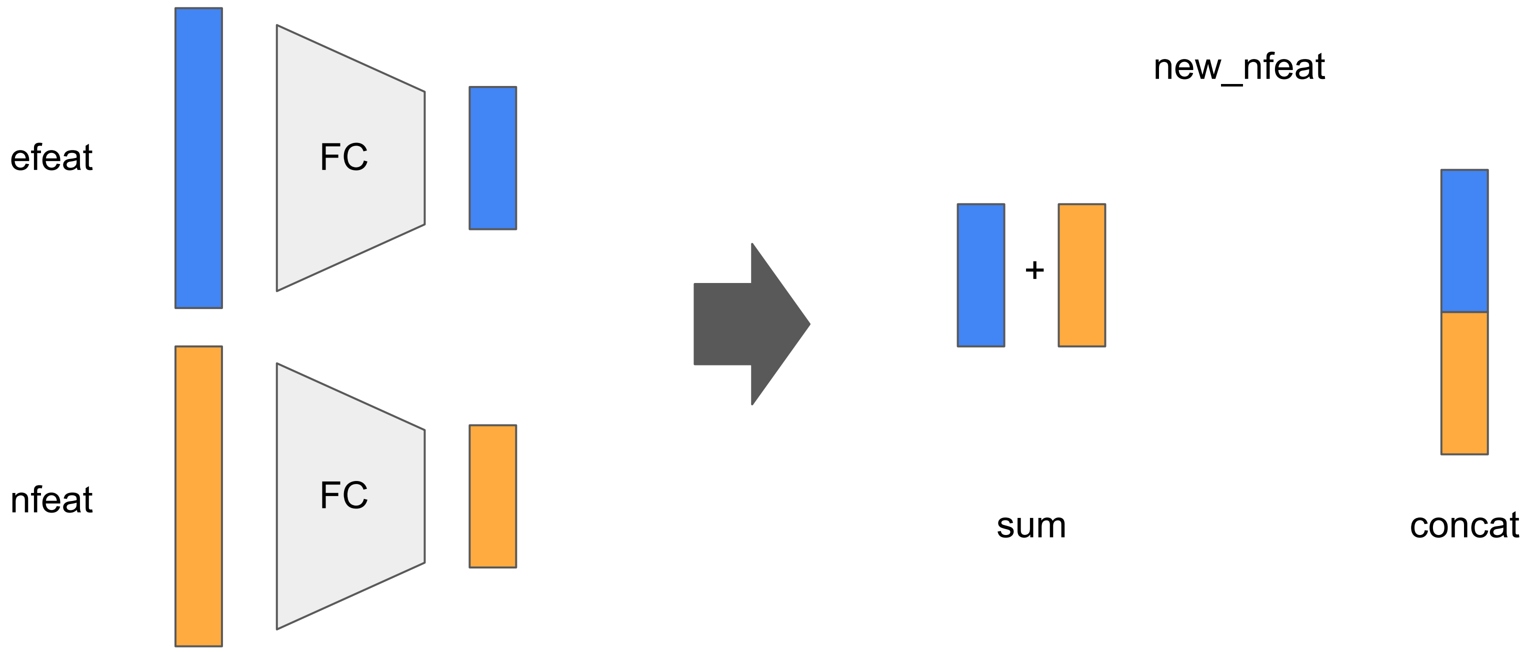
$$h_i^{l+1} = \sigma\left(W_0^{(l)} h_i^{(l)} + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)}\right)$$

$$h_i^{l+1} = \sigma\left(W_0^{(l)} h_i^l + \sum_{r \in R} \sum_{j \in N_i^r} \frac{1}{c_{i,r}} (W_r^{(l)} h_j^l + \boxed{W_{re}^{(l)} e_{ij})}\right)$$

sum or concat?

Edge-RelGCN

sum or concat



TODO

1. CIKM paper search
2. amazon review data preprocessing / build user-item graph
3. Graph 기반 Baseline 선정 및 실험 (GraphSage, IGMC)
4. emotion vector (28) 실험 (edge_GCN, edge_RGCN)
5. PTM fine-tuning (emotion, sentiment)
6. emotion/sentiment embedding vector preprocessing
7. embedding vector (16+16) 실험 (edge_GCN, edge_RGCN)

주요 학회 일정

- RecSys
  - Paper submission deadline: May 3rd, 2022
- **CIKM**
  - **Full & Applied Papers Final Deadline: 16 May 2022**
- ICDM
  - Full conference paper submissions: June 10, 2022
- IEEE BigData
  - Electronic submission of full papers: Aug 20, 2022

http://www.conferencelist.info/upcoming.html

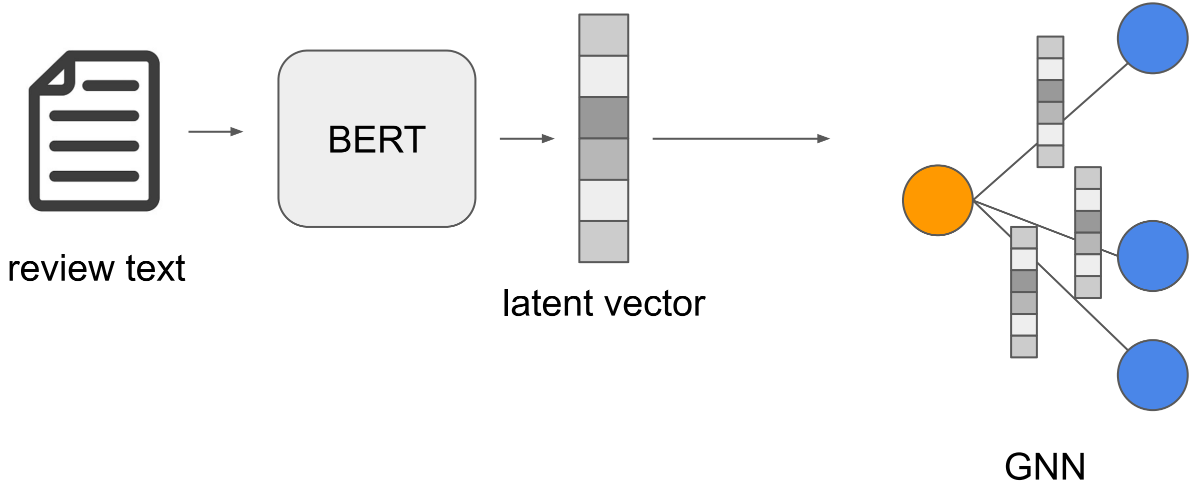
Edge GNN & DGL Implement

2022.02.23

Han, Donghee

Edge GNN

- 기존의 R-GCN 기반의 모델(GraphSAGE, IGMC 등)은 Edge에 Vectorized feature 반영이 어려웠음
- item에 user가 남긴 review text 를 vectorize 하여 embedding
- review text 를 bert를 통해 latent vector로 변환
- message passing layer에 edge embedding vector 추가

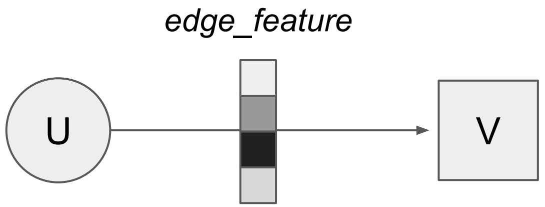


Edge feature

- message passing with edge features
  - 기존 : edge_type 에 따라 레이어를 분리하여 진행 (R = [0,1,2,3,4,5])
  - 제안 : 하나의 레이어를 사용하되 edge_feature 에 latent vector 를 사용
  - AGG(Linear(edge_feat) + Linear(h_i_0) + Linear(h_j)) → h_i_1
  - *Neural Message Passing for Quantum Chemistry - Justin Gilmer*

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$

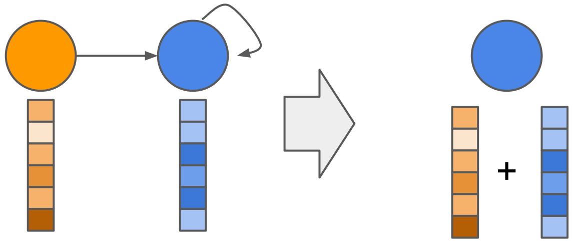
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$



Edge GNN

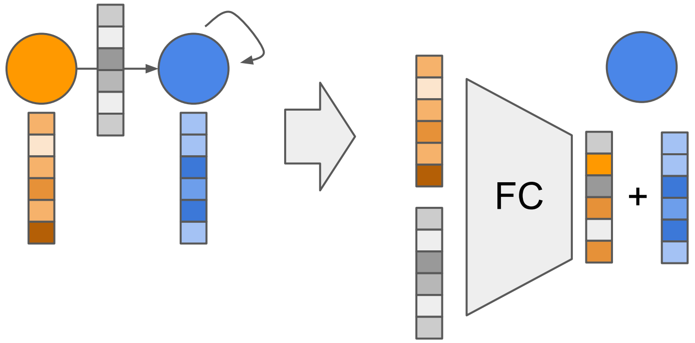
- message passing layer with edge embedding
- merge edge_feat & node_feat through fully-connected layers

$$h_i^{(l+1)} = \sigma\left(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}^r(i)} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)}\right)$$



기존 방식

h_i = AGG(W_h * h_j + W_e * e_ij) + h_i



Edge Message Passing

Result

- Result
  - GoEmotion → 28 dim emotion vector
  - Rotten Tomato : test RMSE **0.7773** (previous best 0.8004)
  - only use edge conv (rating layers & edge_feat layers)

- TODO
  - edge-R-GCN
  - edge message passing with graph conv

    AGG(W_rh * h_j + W_re * e_ij + W_0 * h_i) → h_i

DGL implement

- components of GCN layer
  - **message function** - deliver source node feat & edge feat
  - **reduce function** - aggregate all vectors in message
  - **update_all** - process all message passing

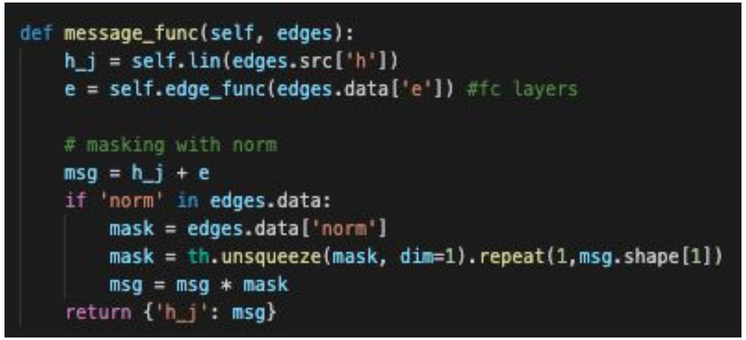
DGL implementation 관련하여 잘 정리되어 있음

https://youtu.be/2aKXWqkbpWg

https://atcold.github.io/pytorch-Deep-Learning/en/week13/13-3/

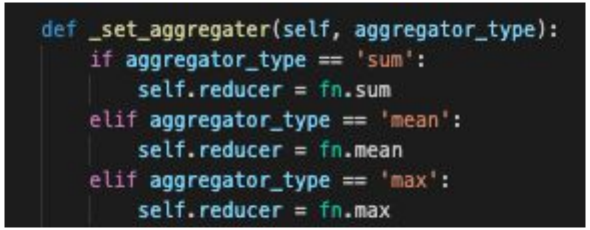
https://docs.dgl.ai/en/0.6.x/guide/message-api.html

DGL implement

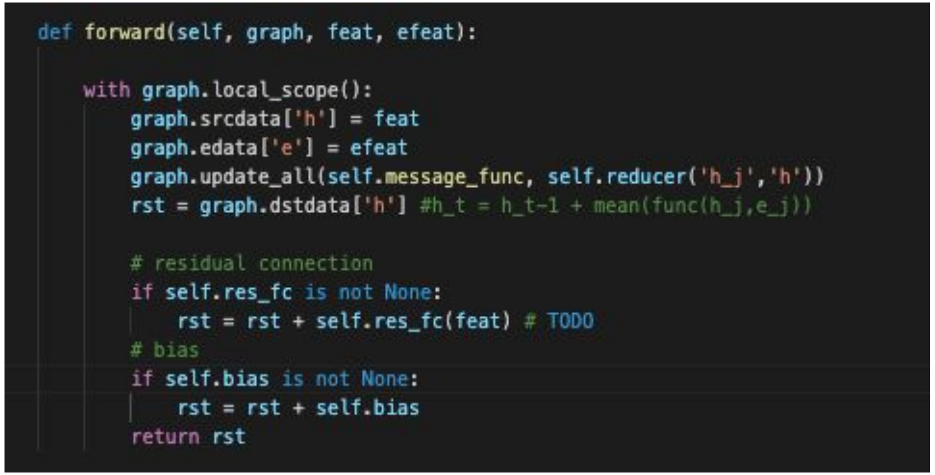


```python
def message_func(self, edges):
    h_j = self.lin(edges.src['h'])
    e = self.edge_func(edges.data['e']) #fc layers

    # masking with norm
    msg = h_j + e
    if 'norm' in edges.data:
        mask = edges.data['norm']
        mask = th.unsqueeze(mask, dim=1).repeat(1,msg.shape[1])
        msg = msg * mask
    return {'h_j': msg}
```



```python
def _set_aggregater(self, aggregator_type):
    if aggregator_type == 'sum':
        self.reducer = fn.sum
    elif aggregator_type == 'mean':
        self.reducer = fn.mean
    elif aggregator_type == 'max':
        self.reducer = fn.max
```



```python
def forward(self, graph, feat, efeat):

    with graph.local_scope():
        graph.srcdata['h'] = feat
        graph.edata['e'] = efeat
        graph.update_all(self.message_func, self.reducer('h_j','h'))
        rst = graph.dstdata['h'] #h_t = h_t-1 + mean(func(h_j,e_j))

        # residual connection
        if self.res_fc is not None:
            rst = rst + self.res_fc(feat) # TODO
        # bias
        if self.bias is not None:
            rst = rst + self.bias
        return rst
```

- message_func : src_node_feat, edge_feat 를 FC layer 에 통과시켜 “h_j” key로 저장
- reduce_func : “h_j” key에 저장된 값 들을 mean aggregation 하여 “h” key로 저장
  https://docs.dgl.ai/en/0.6.x/generated/dgl.function.mean.html#dgl.function.mean

TODO

- Model Variation
  - R-edge-GCN
  - rating-GCN + edge-GCN + R-GCN
  - reinforce node_vector : concat [review_vector, node_vector]
  - other LM (RoBERTa…) latent vector
    https://arxiv.org/pdf/1907.11692.pdf

- Experiment Platform
  - config → yaml file
  - continuous train and logging
  - artifact saving

amazon Q&A + knowledge graph

https://www.amazon.science/conferences-and-events/emnlp-2021

https://www.amazon.science/blog/new-method-improves-knowledge-graph-based-question-answering

https://assets.amazon.science/4a/81/040c7edb48069f520e5967d2e8a2/end-to-end-entity-resolution-and-question-answering-usingdifferentiable-knowledge-graphs.pdf

Graph Neuralnet based Conversational recommender systems

2022.02.11
Han, Donghee

Background

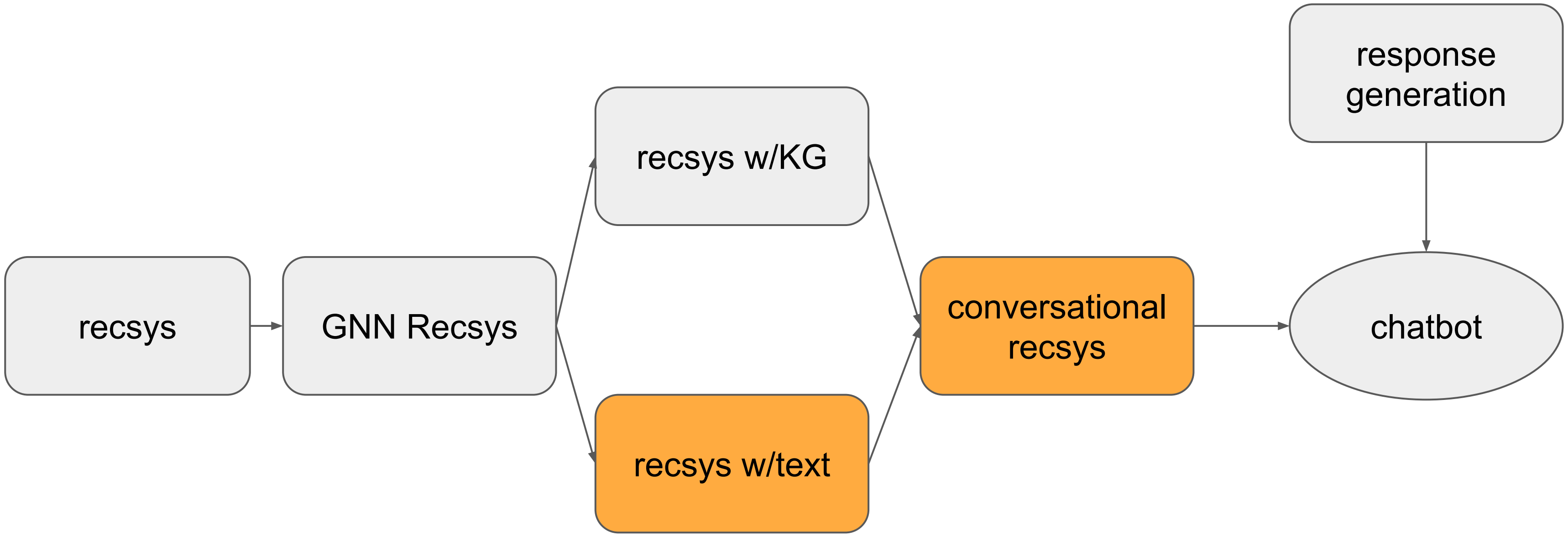
검색, 추천 고도화에 따라 챗봇기반의 서비스 등장

nlp-bert, recsys-graph SOTA 달성

dialog 환경에서의 Task 및 데이터셋 등장 (Wizard of Wikipedia, ReDial)

다양한 Task에 최적화한 모델 needs

Background


recsys
GNN Recsys
recsys w/KG
recsys w/text
conversational recsys
response generation
chatbot

Text-based rec-sys : Text + GCN

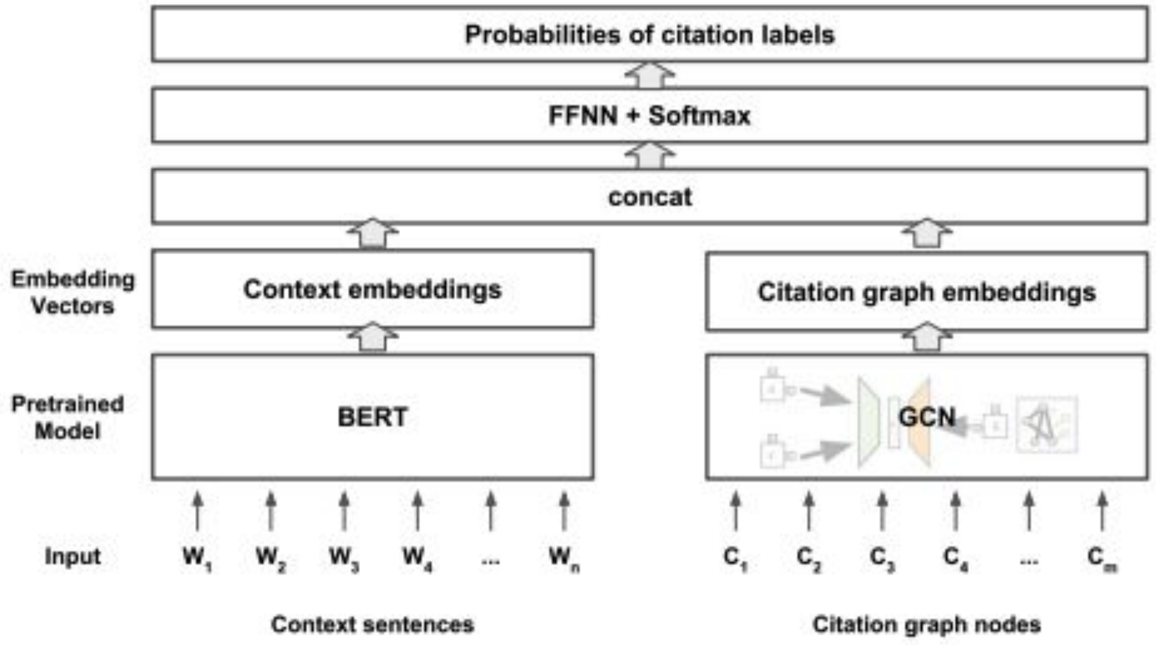
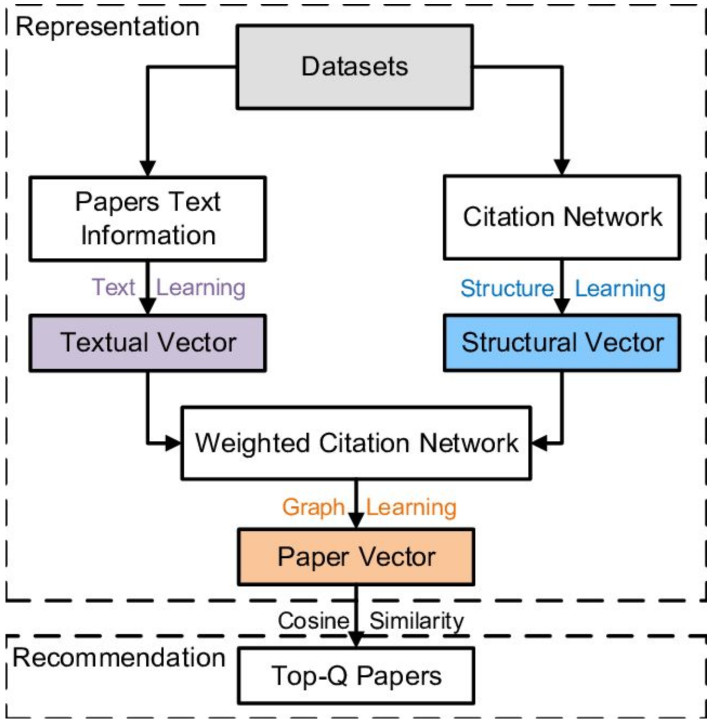

Probabilities of citation labels
FFNN + Softmax
concat
Embedding Vectors
Context embeddings
Citation graph embeddings
Pretrained Model
BERT
GCN
Input
W_1 W_2 W_3 W_4 ... W_n
C_1 C_2 C_3 C_4 ... C_m
Context sentences
Citation graph nodes

Fig. 2 A BERT–GCN model architecture

https://link.springer.com/content/pdf/10.1007/s11192-020-03561-y.pdf

Text-based rec-sys



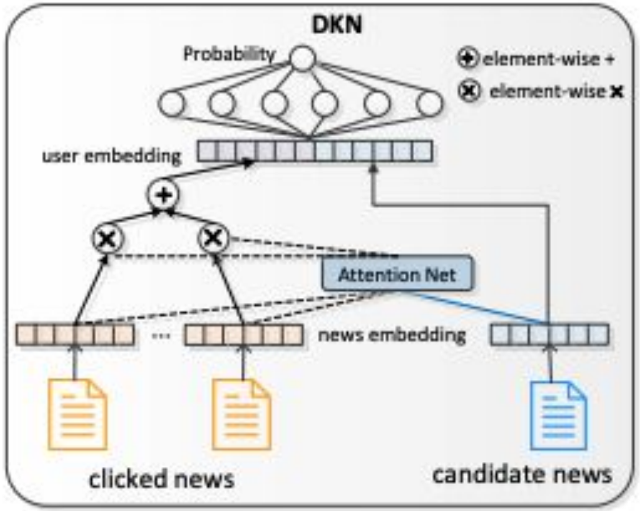


Figure 1: Simple visualization of DKN model.

https://ieeexplore.ieee.org/document/8349947

https://ojs.aaai.org/index.php/AAAI/article/view/4549

Text-based rec-sys

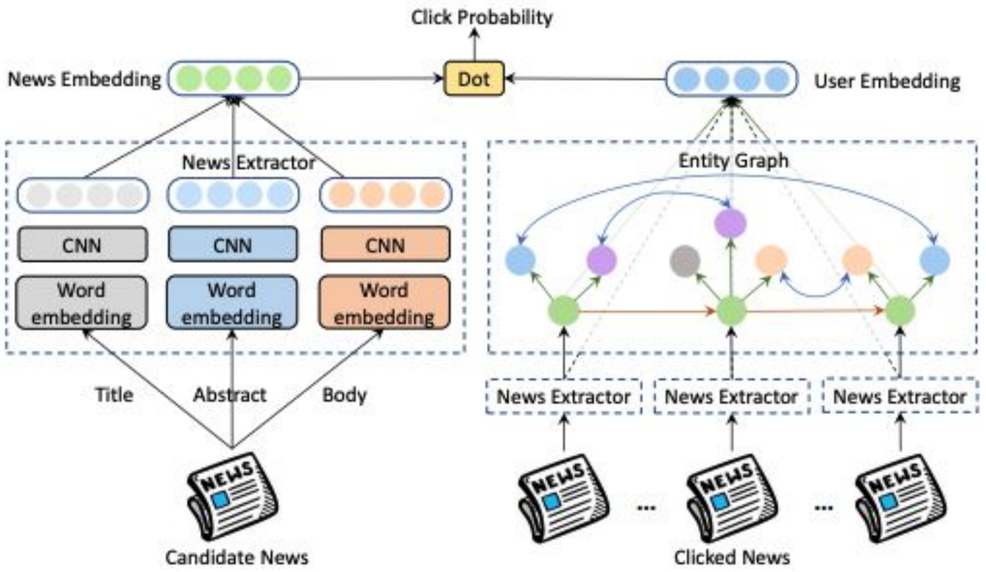


Figure 1: The framework of our model. In the entity graph, green nodes denote news, which are learned from news extractor. Nodes in other colors represent entities. (Best see in colors)

https://dl.acm.org/doi/pdf/10.1145/3442442.3452329

Text-based rec-sys

주로 item 에 대한 review, item text (news 등) 을 활용함

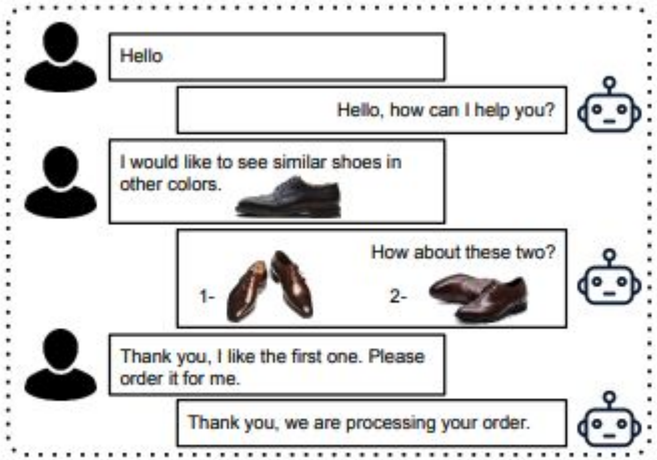
관계정보(graph)와 text 정보를 분리하여 학습

언어모델을 활용하는 path 와 추천 모델을 활용하는 path를 병렬로 사용

최종 embedding을 결합해 user-item 점수 예측

→ ***RQ.1 text embedding을 graph 모델에 direct로 사용할 수 있을까?***
(구체화, end2end)

Conversational Recommender System



Hello
Hello, how can I help you?
I would like to see similar shoes in other colors.
How about these two?
1-
2-
Thank you, I like the first one. Please order it for me.
Thank you, we are processing your order.

Figure 1: An example conversation between a user and an agent in a Conversational Recommender System in e-Commerce domain.

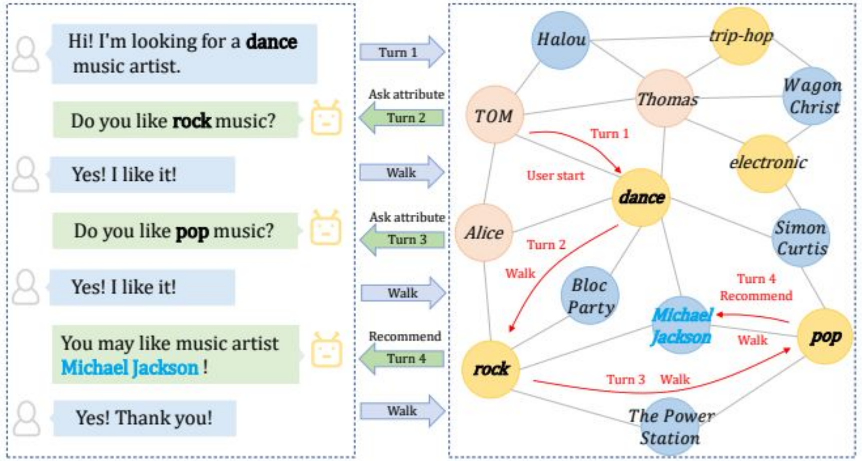


Figure 4: An illustration of interactive path reasoning in the conversational path reasoning (CPR) model. Credits: Lei et al. [100].

[Deep Conversational Recommender System: A New Frontier for Goal-Oriented Dialogue Systems]

https://arxiv.org/pdf/2101.09459.pdf

Conversational Recommender System

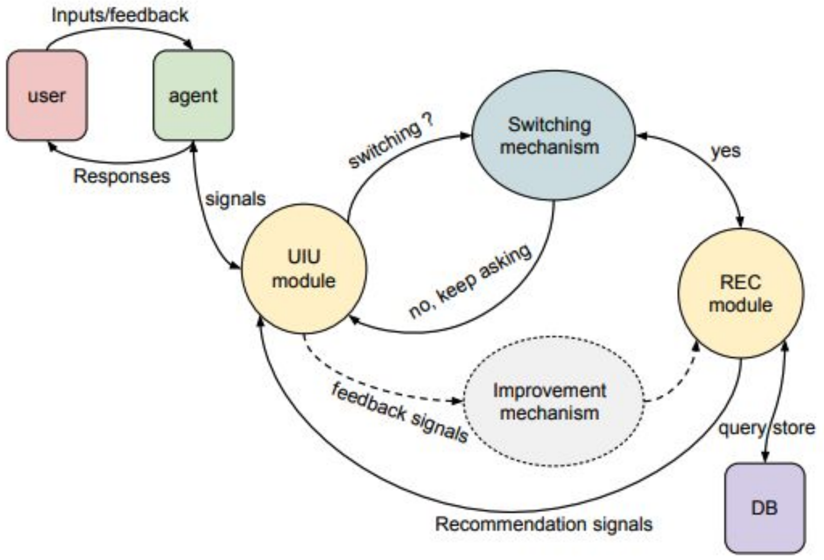
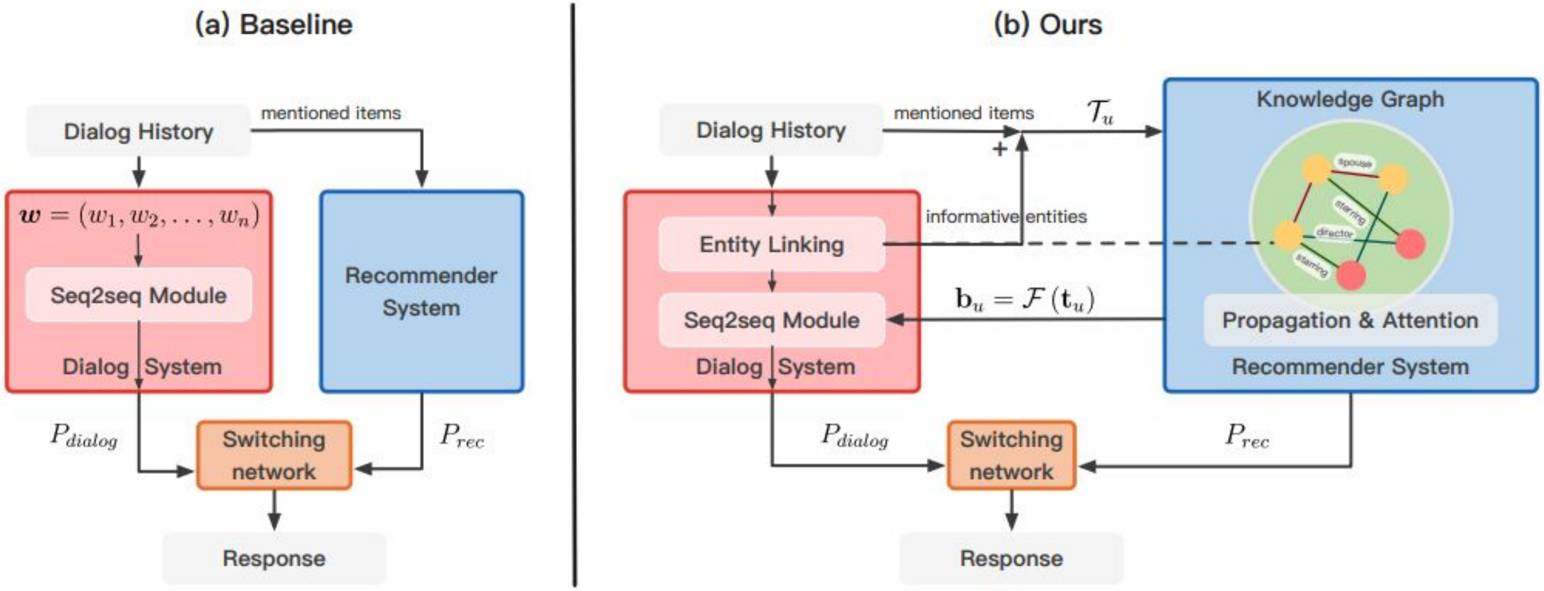


Figure 2: The main components of a DCRS. The dash line shows optional component of a DCRS.

Deep Conversational Recommender System: A New Frontier for Goal-Oriented Dialogue Systems

Conversational Recommender System



KBRD : Bridging the recommender system and the dialog system with knowledge graph

https://arxiv.org/pdf/1908.05391.pdf

Conversational Recommender System : CRSDP

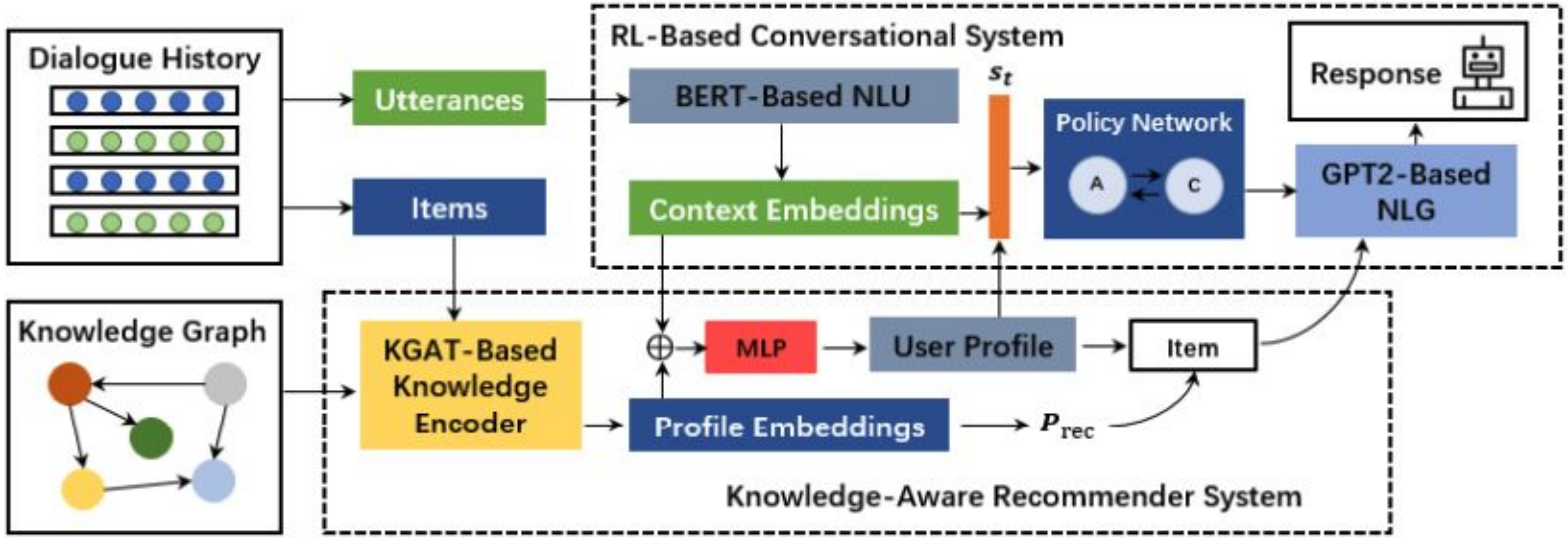


Figure 2: The overview of the proposed CRSDP model. The knowledge-aware recommender system retrieves movies that match user preferences. The RL-based conversational system interacts with the user through natural language.

Conversational rec-sys

Conversation agent 유형에 따라 다양한 case로 구분

- 단순 검색형, KG 탐색형
- 대화에만 의존하는 경우, 외부 지식(KG)를 결합하는 경우

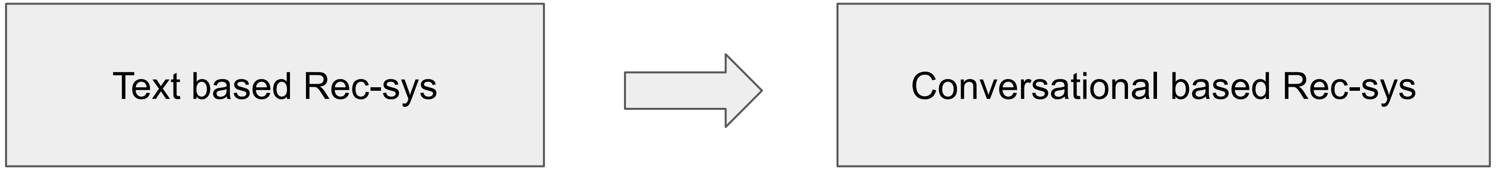
자연어에서 embedding 을 추출 하여 외부지식(KG) 등과 결합하는것은 이전 모델들과 유사

→ ***RQ.2 User Intent 를 어떻게 graph 모델에 결합 할 수 있을까?***
→ ***RQ.3 Conversation 의 결과를 KG 에 어떻게 stacking 할 수 있을까?***
out-dated 된 지식을 depreciate 시킬 수 있을까?

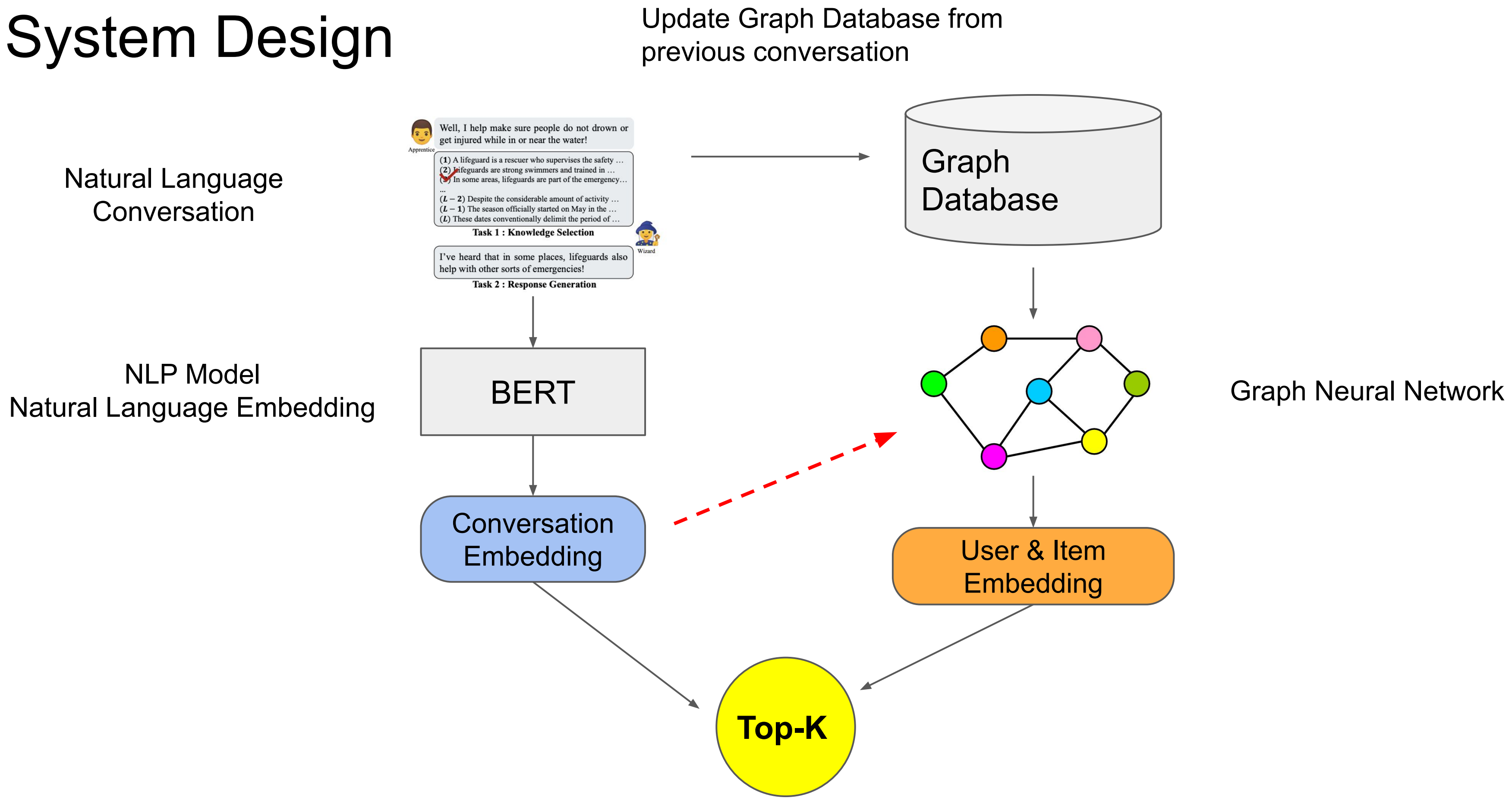
Goal

자연어 interaction 과 Graph Neuralnet을 결합한 추천 모델 개발

- Conversation, Review 등을 추천에 활용
- 기존의 nlp모델을 개선하여 User Intention Understanding 활용
- graph 모델을 결합하여 user-item relation 과 interaction 을 함께 학습
- 이전 대화의 결과를 Graph database 에 추가하여 다음 Session 에 활용하는 방안 연구



System Design



Well, I help make sure people do not drown or get injured while in or near the water!

(**1**) A lifeguard is a rescuer who supervises the safety …
(**2**) Lifeguards are strong swimmers and trained in …
(**3**) In some areas, lifeguards are part of the emergency…
…
(**$L-2$**) Despite the considerable amount of activity …
(**$L-1$**) The season officially started on May in the …
(**L**) These dates conventionally delimit the period of …

Task 1 : Knowledge Selection

I've heard that in some places, lifeguards also help with other sorts of emergencies!

Task 2 : Response Generation

Methods

- Train GNN model with text features
  - Use text embeddings from review or conversation
  - Limit text data to prevent model from becoming too large
- Stacking user interactions on graph data
  - Compress natural language interaction and add to graph data
  - Connect recent data with previous data
  - Update user/item profiles based on previous interactions
- Conversation embedding and User/Item embedding
  - Extract conversation embedding from user-agent conversation
  - Build User-Item embedding from graph database
  - Recommend Top-k items from embeddings

Baselines

ReDial - https://arxiv.org/pdf/1812.07617v2.pdf
https://github.com/RaymondLi0/conversational-recommendations

CRSDP - https://dl.acm.org/doi/pdf/10.1145/3502223.3502225

CRFR - https://aclanthology.org/2021.emnlp-main.355.pdf

KGSF - https://arxiv.org/pdf/2007.04032.pdf
https://github.com/Lancelot39/KGSF

mile-stone

1. Text Information 활용방안
- 자연어로 정의된 관계를 활용 (edge-feature)
- review / conversation based recommendation

2. CRS Baseline 분석
- ReDial, CRSDP 등 기존 모델 분석 및 포팅

3. Conversational data stacking on Graph
- 대화 기록을 축적하여 추천에 활용하는 방안 연구

sub task

1. DGL, Pytorch-Geo 등 gnn framework know-how 축적
2. Conversational recommendation 실험 framework 이해도 향상
3. 관련 데이터셋 분석 및 유형 분류

→ 추후 챗봇기반 검색엔진 개발 기반 역량 확보

dataset

wizard of wikipedia
- https://arxiv.org/pdf/1811.01241.pdf
- https://github.com/facebookresearch/ParlAI/tree/main/projects/wizard_of_wikipedia
- Open-domain multi-turn knowledge-based dialogue dataset
- Chit-chat between Wizard(Wiki IR sys) & Apprentice about a given topic

ReDial
- https://redialdata.github.io/website/
- annotated dataset of dialogues, where users recommend movies to each other

others

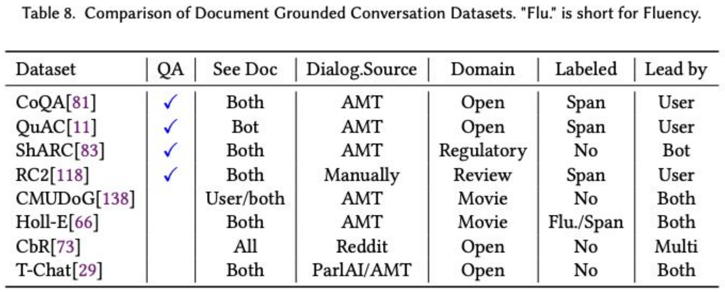


Table 8. Comparison of Document Grounded Conversation Datasets. "Flu." is short for Fluency.

Dataset	QA	See Doc	Dialog.Source	Domain	Labeled	Lead by
CoQA[81]	✓	Both	AMT	Open	Span	User
QuAC[11]	✓	Bot	AMT	Open	Span	User
ShARC[83]	✓	Both	AMT	Regulatory	No	Bot
RC2[118]	✓	Both	Manually	Review	Span	User
CMUDoG[138]		User/both	AMT	Movie	No	Both
Holl-E[66]		Both	AMT	Movie	Flu./Span	Both
CbR[73]		All	Reddit	Open	No	Multi
T-Chat[29]		Both	ParlAI/AMT	Open	No	Both

https://arxiv.org/pdf/2004.13818.pdf

Reference

A Survey of Document Grounded Dialogue Systems (DGDS)
https://arxiv.org/pdf/2004.13818.pdf

Self-supervised Graph Learning for Recommendation
https://arxiv.org/pdf/2010.10783.pdf https://github.com/wujcan/SGL

CORGI: Content-Rich Graph Neural Networks with Attention
https://arxiv.org/pdf/2110.04866.pdf https://eedi.com/projects/neurips-education-challenge 자연어 활용, 학습데이터셋

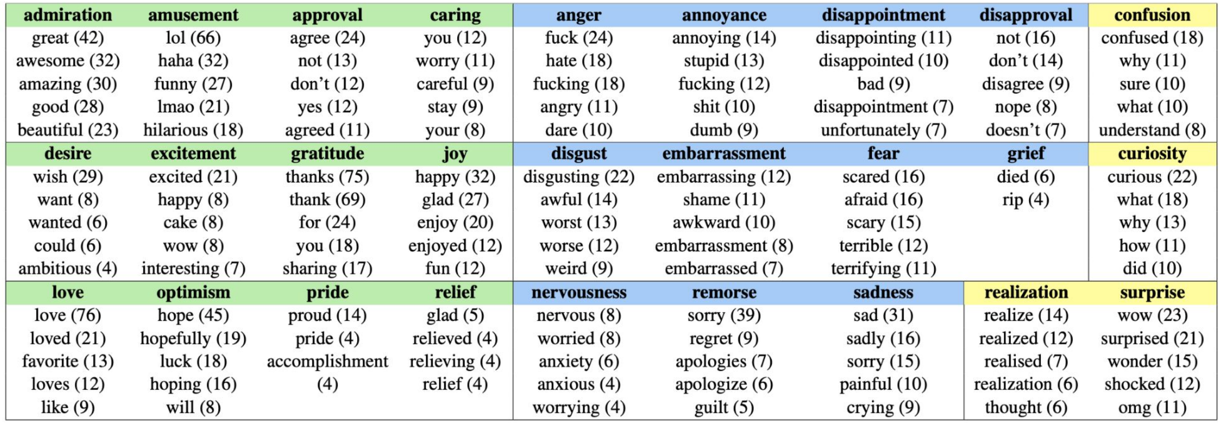
PinnerSage: Multi-Modal User Embedding Framework for Recommendations at Pinterest
https://arxiv.org/pdf/2007.03634.pdf
https://medium.com/pinterest-engineering/pinnersage-multi-modal-user-embedding-framework-for-recommendations-at-pinterest-bfd116b49475 멀티모달

SEQUENTIAL LATENT KNOWLEDGE SELECTION FOR KNOWLEDGE-GROUNDED DIALOGUE
https://arxiv.org/pdf/2002.07510.pdf Wizard of Wikipedia 에서 SOTA 달성

2021.12.13

GoEmotions: A Dataset of Fine-Grained Emotions

- https://arxiv.org/pdf/2005.00547.pdf 데이터셋 논문
- https://github.com/google-research/google-research/tree/master/goemotions TF로 되어있는 레포
- 12개의 긍정, 14개의 부정, 4개의 애매한 감정으로 분류됨



admiration	**amusement**	**approval**	**caring**	**anger**	**annoyance**	**disappointment**	**disapproval**	**confusion**
great (42)	lol (66)	agree (24)	you (12)	fuck (24)	annoying (14)	disappointing (11)	not (16)	confused (18)
awesome (32)	haha (32)	not (13)	worry (11)	hate (18)	stupid (13)	disappointed (10)	don't (14)	why (11)
amazing (30)	funny (27)	don't (12)	careful (9)	fucking (18)	fucking (12)	bad (9)	disagree (9)	sure (10)
good (28)	lmao (21)	yes (12)	stay (9)	angry (11)	shit (10)	disappointment (7)	nope (8)	what (10)
beautiful (23)	hilarious (18)	agreed (11)	your (8)	dare (10)	dumb (9)	unfortunately (7)	doesn't (7)	understand (8)
desire	**excitement**	**gratitude**	**joy**	**disgust**	**embarrassment**	**fear**	**grief**	**curiosity**
wish (29)	excited (21)	thanks (75)	happy (32)	disgusting (22)	embarrassing (12)	scared (16)	died (6)	curious (22)
want (8)	happy (8)	thank (69)	glad (27)	awful (14)	shame (11)	afraid (16)	rip (4)	what (18)
wanted (6)	cake (8)	for (24)	enjoy (20)	worst (13)	awkward (10)	scary (15)		why (13)
could (6)	wow (8)	you (18)	enjoyed (12)	worse (12)	embarrassment (8)	terrible (12)		how (11)
ambitious (4)	interesting (7)	sharing (17)	fun (12)	weird (9)	embarrassed (7)	terrifying (11)		did (10)
love	**optimism**	**pride**	**relief**	**nervousness**	**remorse**	**sadness**	**realization**	**surprise**
love (76)	hope (45)	proud (14)	glad (5)	nervous (8)	sorry (39)	sad (31)	realize (14)	wow (23)
loved (21)	hopefully (19)	pride (4)	relieved (4)	worried (8)	regret (9)	sadly (16)	realized (12)	surprised (21)
favorite (13)	luck (18)	accomplishment	relieving (4)	anxiety (6)	apologies (7)	sorry (15)	realised (7)	wonder (15)
loves (12)	hoping (16)	(4)	relief (4)	anxious (4)	apologize (6)	painful (10)	realization (6)	shocked (12)
like (9)	will (8)			worrying (4)	guilt (5)	crying (9)	thought (6)	omg (11)

Table 3: Top 5 words associated with each emotion (positive , negative , ambiguous). The rounded z-scored log odds ratios in the parentheses, with the threshold set at 3, indicate significance of association.

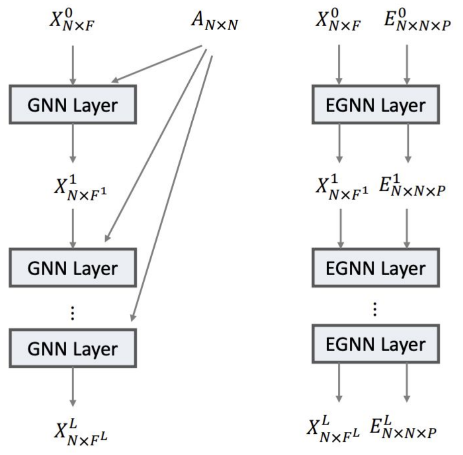
go emotion - pytorch

- https://github.com/monologg/GoEmotions-pytorch → torch 구현 레포
- https://github.com/monologg/GoEmotions-pytorch/blob/master/model.py 모델 부분
- forward 메서드에서 latent vector 추출 가능

  https://github.com/monologg/GoEmotions-pytorch/blob/b75f9d3a8b76bc1060687925b04ca98a1b8ea66b/model.py#L17-L46
-

Edge feature&weight

- Exploiting Edge Features in Graph Neural Networks
  https://www.arxiv-vanity.com/papers/1809.02709/
- End-to-end learning of latent edge weights for Graph Convolutional Networks
  https://www.ingwb.com/binaries/content/assets/insights/themes/empowering-with-advanced-analytics/internship-theses-at-wbaa/end-to-end-learning-of-latent-edge-weights-for-graph-convolutional-networks.pdf
- Exploiting Edge Features in Graph Neural Networks
  https://arxiv.org/pdf/1809.02709.pdf
  edge enhanced graph neural network (EGNN)



edge enhanced graph neural network (EGNN)

- Doubly stocahstic normalization of edges
  - j 관점에서 normalization * i 관점에서 normalization
  - 정확한 이유에 대해서는 스터디 필요
- EGNN(A): Attention based EGNN layer
  -

$$\tilde{E}_{ijp} = \frac{\hat{E}_{ijp}}{\sum_{k=1}^{N} \hat{E}_{ikp}}$$

$$E_{ijp} = \sum_{k=1}^{N} \frac{\tilde{E}_{ikp}\tilde{E}_{jkp}}{\sum_{v=1}^{N} \tilde{E}_{vkp}}$$

$$X^l = \sigma \left[\Big\Vert_{p=1}^{P} \left(\alpha^l_{\cdot\cdot p}(X^{l-1}, E^{l-1}_{\cdot\cdot p}) g^l(X^{l-1}) \right) \right].$$

TODO

- pseudo label…
- bert로 아마존 데이터 multi-label emotion 생성
  - latent vector 같이 저장하도록 수정
- rating 을 edge feature 로 embedding (e weight?)
- emotion 을 edge feature 로 embedding
- edge wieght, edge feature mp test 작성
  - https://docs.dgl.ai/en/0.6.x/guide/message-edge.html
  -

2021.12.13

Edge feature

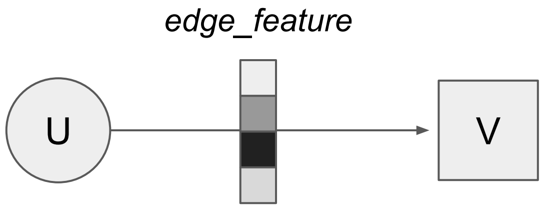
- 기존의 rating 추정 모델들은 rating을 edge_type 으로 사용
    - rating 이라는 것은 numeric value 인데 categorical 한 형태로 임베딩시킴
    - 이렇게 한 이유에 대해서 언급하지 않음. 최종결과는 확률 분포에 따른 기대값 사용
    - 같은 rating 값을 지닌 relation 별로 별도의 GCN을 학습

- ‘사용자의 리뷰’ 내부에는 다양한 dim 이 존재한다고 볼 수 있음
    - BERT 의 감정 분류 결과를 multi-hot 방식으로 적용
    - BERT 의 분류 결과가 아닌 중간의 latent vector 를 사용

Edge feature

- message passing with edge features
  - 기존 : edge_type 에 따라 레이어를 분리하여 진행 (R = [0,1,2,3,4,5])
  - 제안 : 하나의 레이어를 사용하되 edge_feature 에 latent vector 를 사용
  - AGG(Linear(edge_feat) + Linear(h_i_0) + Linear(h_j)) → h_i_1
  - *Neural Message Passing for Quantum Chemistry - Justin Gilmer*

$$m_v^{t+1} = \sum_{w \in N(v)} M_t(h_v^t, h_w^t, e_{vw})$$
$$h_v^{t+1} = U_t(h_v^t, m_v^{t+1})$$



Edge embedding

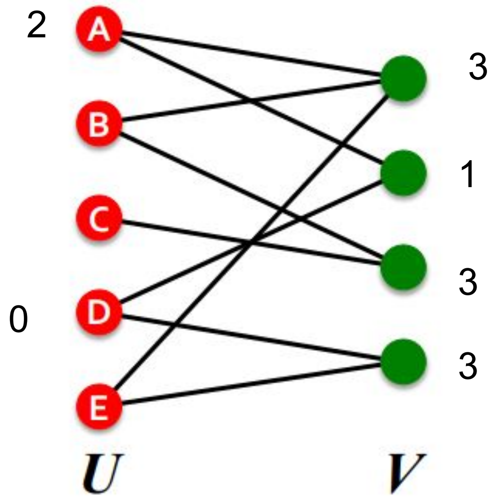
- Edge embedding + edge features
  - Edge2Vec 등을 활용해 edge embedding 생성
  - edge feature와 edge embedding 을 함께 연산하여 결과 추론

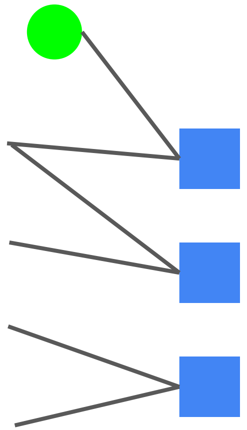
edge_feature + edge_embedding

cold start - entity node 기반의 해결

- graph 기반의 한계인 cold start problem을 entity data 기반으로 해결
- entity data 기반으로 추가된 아이템의 mp 경로를 확보 할 수 있음
- 기존의 관계를 통해 entity data를 생성하도록 모델을 만들 수 있다 (latent meta?, cold start 케이스는 아님)

user node item node entity node





Session + Graph

- 시간에 따라 변화하는 사용자의 관심사 추적
-

Bipartite Graph viz

http://jmcauley.ucsd.edu/data/amazon/

- 아마존 review / rating data
- 제품군 별로 다양 → 데이터 양이 많아 특정 제품군으로 제한하여 진행해야 할듯

https://towardsdatascience.com/graph-based-recommendation-engine-for-amazon-products-1a373e639263

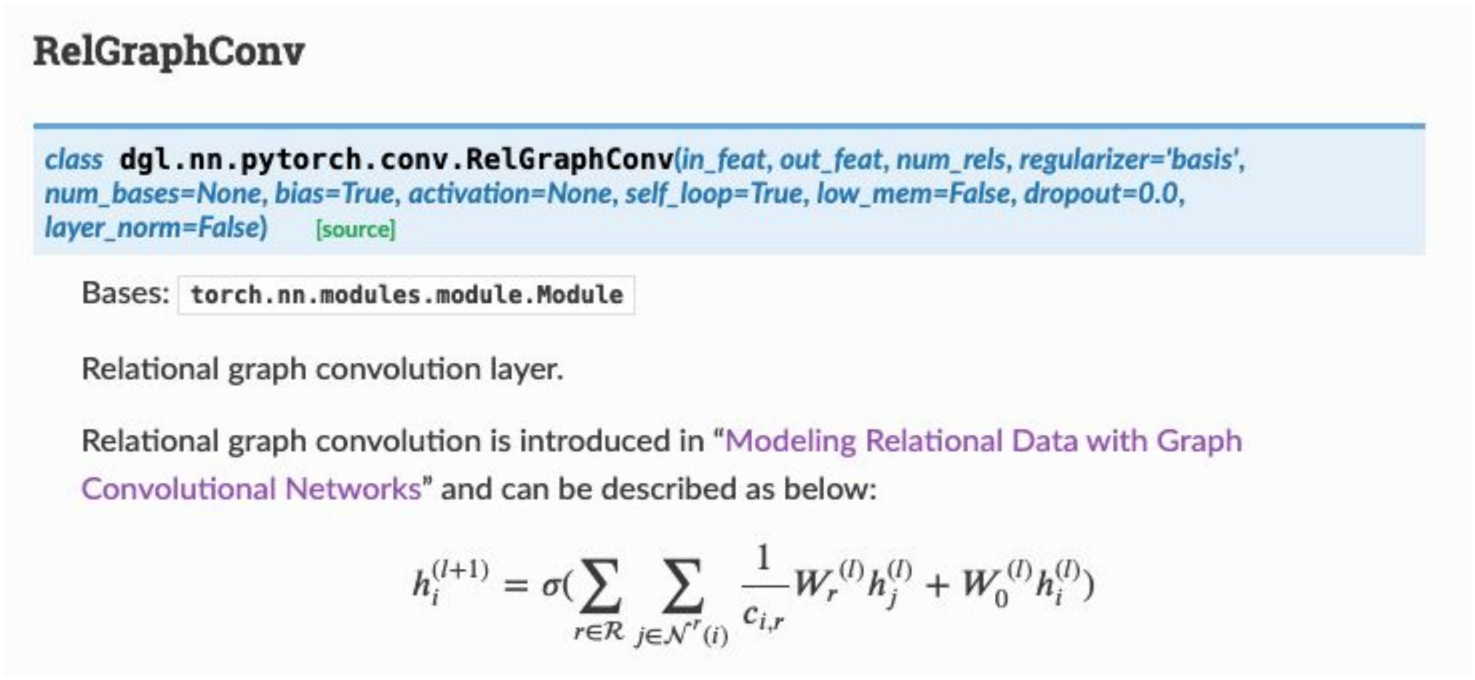
리뷰데이터를 e_feat 로 활용

http://jmcauley.ucsd.edu/data/amazon/

- 아마존 review / rating data
- 제품군 별로 다양 → 데이터 양이 많아 특정 제품군으로 제한하여 진행해야 할듯

모델 구현체 비교

R-GNN



RelGraphConv

class **dgl.nn.pytorch.conv.RelGraphConv**(*in_feat, out_feat, num_rels, regularizer='basis', num_bases=None, bias=True, activation=None, self_loop=True, low_mem=False, dropout=0.0, layer_norm=False*) [source]

Bases: `torch.nn.modules.module.Module`

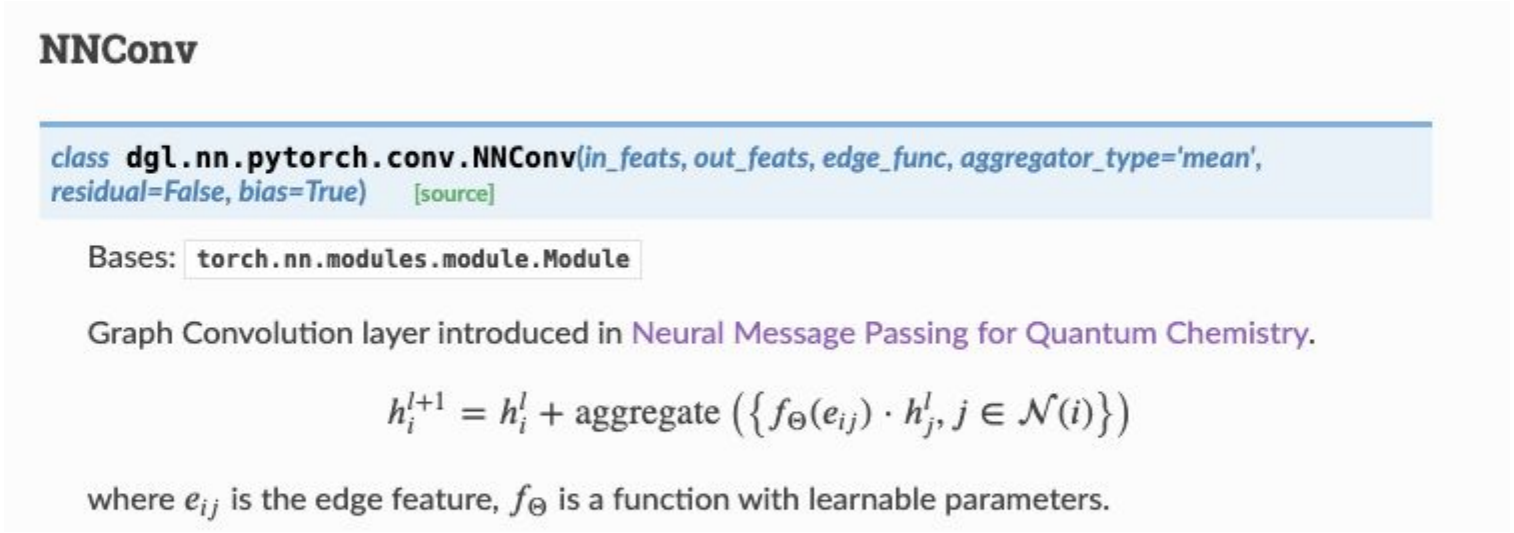
Relational graph convolution layer.

Relational graph convolution is introduced in "Modeling Relational Data with Graph Convolutional Networks" and can be described as below:

$$h_i^{(l+1)} = \sigma(\sum_{r\in\mathcal{R}} \sum_{j\in\mathcal{N}^r(i)} \frac{1}{c_{i,r}} W_r^{(l)} h_j^{(l)} + W_0^{(l)} h_i^{(l)})$$

GCMC, IGMC 방식 / edge_type별로 다른 W_r 사용

MPNN



NNConv

`class dgl.nn.pytorch.conv.NNConv(in_feats, out_feats, edge_func, aggregator_type='mean', residual=False, bias=True)` [source]

Bases: `torch.nn.modules.module.Module`

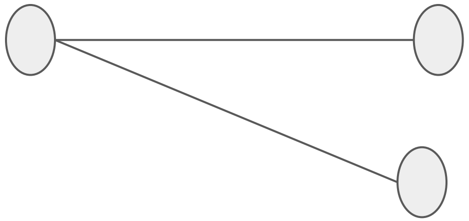
Graph Convolution layer introduced in Neural Message Passing for Quantum Chemistry.

$$h_i^{l+1} = h_i^l + \mathrm{aggregate}\left(\left\{f_\Theta(e_{ij}) \cdot h_j^l, j \in \mathcal{N}(i)\right\}\right)$$

where e_{ij} is the edge feature, f_Θ is a function with learnable parameters.

edge_feat e_ij 를 massage passing 함수에 포함함

구현



1. **Edgetype 을 edge_feats 에 포함해서 학습**
    a. IGMC 구현에서 RelGraphConv 를 NNconv로 대체
    b. R을 사용하지 않는 대신, edge_feat matrix 사용 (feat_dim, r_dim)
    c. AGG(alpha*Linear(edge_feat) + Linear(h_i) + Linear(h_j)) → h_i
    d. edge_feat W 는 (r_dim, 1) → 변환 후, (feat_dim, 1)

2. R-GNN 에 edge_feats 포함
    a. RelGraphConv 에서 Massage Passing 함수에 edge_feat 추가
    b. AGG(W_rh * h_j + W_re * e_ij + W_0 * h_i) → h_i