

지식 그래프를 활용한 추천 서비스

방 학 4 주 차

이번 주차에서 리뷰할 논문


KGAT: Knowledge Graph Attention Network for Recommendation
(2019, Wang et al.)

추천 시스템의 성공

검색엔진부터 이커머스, SNS, 뉴스, 웹 어플리케이션까지 널리 사용되고 있음

님을 위한 추천상품

더보기 >




블루다이아몬드 아몬드 브리즈 언스위트, 190ml, 24개

13,140원

로켓배송

추가 할인쿠폰




구달 청귤 비타C 잡티세럼 기획 세트, 세럼 30 ml +...

13,270원

로켓와우

추가 할인쿠폰



바닐라코 뉴 클린 잇 렌징 밤 오리지널 대

15,780원

로켓와우

추가 할인쿠폰

기업들의 생각

"협업 필터링을 통해서 사용자에게 추천을 제공하자!"

사용자 행동 데이터를 이용하여 예측하기

Collaborative Filtering (협업 필터링)의 한계점

부수적인 정보를 자체 생산할 수 없음

따라서 **sparse**한 데이터에 대해서는 성능이 저하됨

이를 위해 업계에서 사용한 것

SL (Supervised Learning) 모델을 사용하기

user와 item에 대한 ID를 feature vector로 만들고, 이를 SL 모델에 입력하여 점수를 예측하기

이를 위해 업계에서 사용한 것

대표적인 모델 1 : Wide & Deep (Cheong et al., 2016)

두 가지를 결합해서 과거 데이터 활용과 일반화를 이루어냄

1. 넓은 선형 모델(wide linear model)
2. 깊은 신경망(deep neural network)

이를 위해 업계에서 사용한 것

대표적인 모델 2 : NFM (He et al., 2017)

Factorization Machine에 Neural Network를 결합해서 성능을 높임

이를 위해 업계에서 사용한 것

대표적인 모델 3 : xDeepFM (Lian et al., 2018)

Explicit Interaction을 위한 새로운 구조를 제시함

이를 위해 업계에서 사용한 것

대표적인 모델 4 : Deep Crossing (Shan et al., 2016)

Feature Engineering 없이 예측 모델링을 진행함

이를 위해 업계에서 사용한 것

대표적인 모델 5 : Deep Interest Network (Zhou et al., 2018)

fixed-length vector가 가지는 bottleneck을 해결하기 위한 모델

두 모델이 주목하는 것

(CF) 유사한 유저 VS 유사한 아이템 (SL)

CF 모델들은 같은 아이템을 선호하는 두 유저의 이력에 관심을 가짐

VS

SL 모델들은 같은 속성을 가진 유사한 두 아이템에 주목함

두 모델이 주목하는 것

(CF) 유사한 유저 VS 유사한 아이템 (SL)

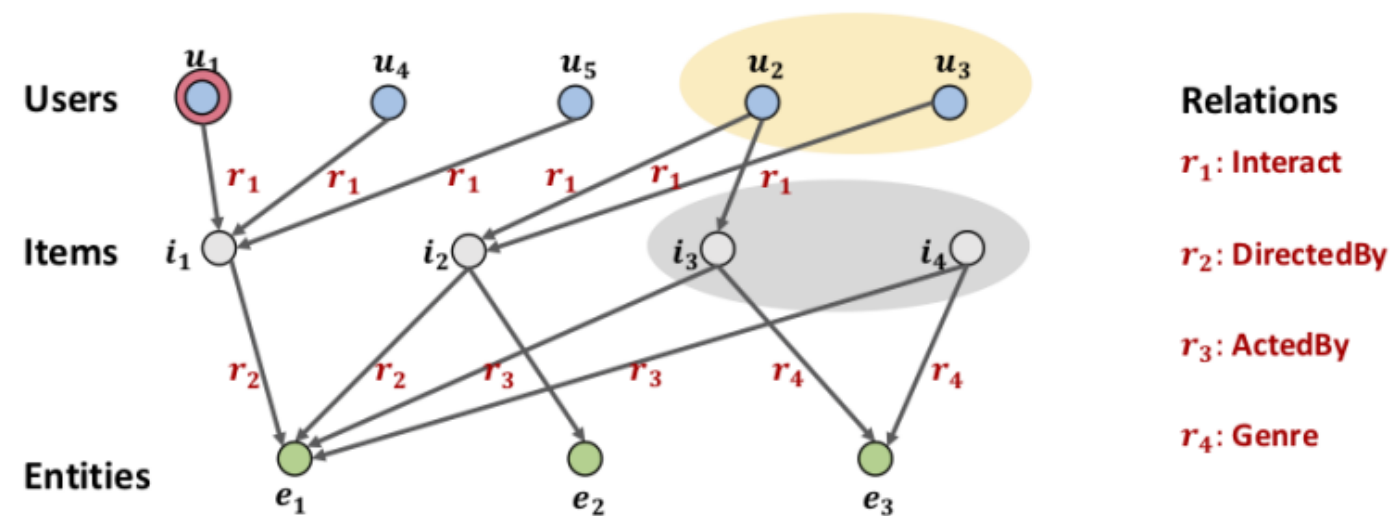
두 모델들은

1. 상호 보완적일뿐만 아니라
2. 사용자와 아이템은 함께 고차원적인 관계를 형성하기도 함

이러한 SL 모델들의 한계점

성능은 나쁘지 않지만, 관계를 고려하지 않음

성능은 강력하지만, 각 상호작용을 독립적인 데이터 인스턴스로 모델링
=> 그 관계를 고려하지 않게 됨



이러한 SL 모델들의 한계점

성능은 나쁘지 않지만, 관계를 고려하지 않음

SL 혼자로서는 이러한 관계를 파악할 수 없어서

(1) 같은 사람이 감독한 다른 영화의 시청 여부

(2) 다른 entity를 기반으로 하는 관계

등을 고려할 수 없음

SL 모델의 한계점을 해결하기 위한 모델

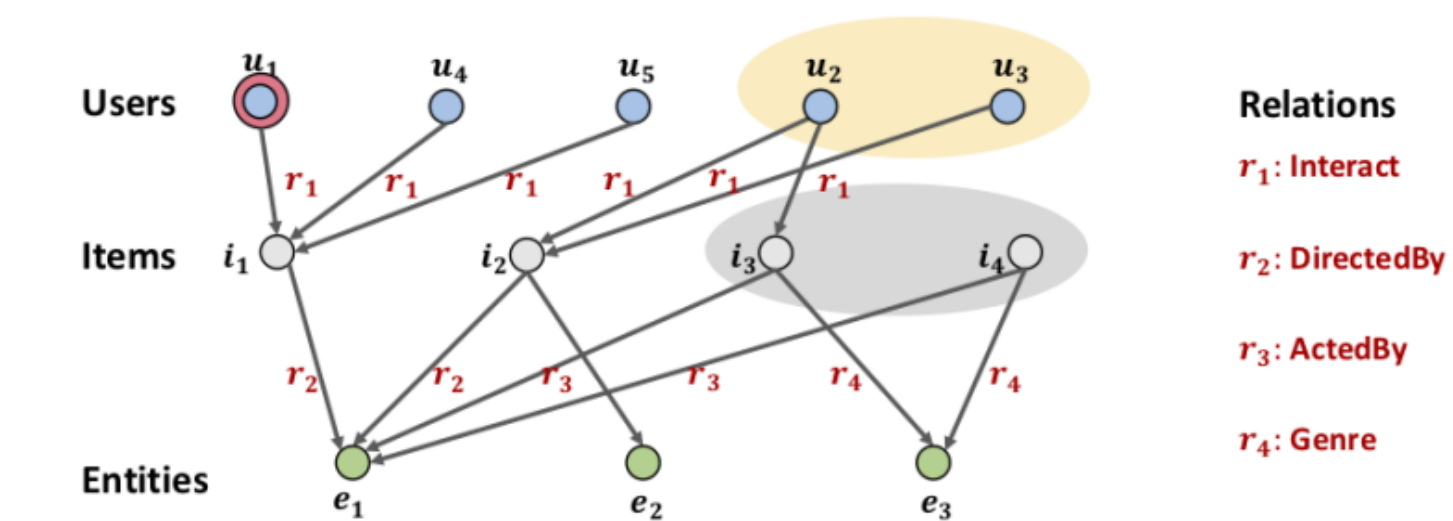
CKG (Collaborative Knowledge Graph)

Knowledge Graph와 User-Item Graph의 하이브리드 구조

- 형성된 고차적인 관계를 최대한으로 활용함

SL 모델의 한계점을 해결하기 위한 모델

CKG (Collaborative Knowledge Graph)



$$\begin{aligned} & u_1 \xrightarrow{r_1} i_1 \xrightarrow{-r_2} e_1 \xrightarrow{r_2} i_2 \xrightarrow{-r_1} \{u_2, u_3\}, \\ & u_1 \xrightarrow{r_1} i_1 \xrightarrow{-r_2} e_1 \xrightarrow{r_3} \{i_3, i_4\}, \end{aligned}$$

CKG 모델이 해결해야할 문제

고차적인 정보를 잘 활용하기 위해 2가지 과제가 부여됨

- 1) 크기가 커지면서 모델에 계산 과부하가 걸리기 쉽다
- 2) 고차적인 관계는 예측에 불균등하게 기여하므로 신중하게 가중치를 부여해야 한다

CKG 모델을 잘 쓰기 위한 시도들

경로 기반(Path-based)과 정규화(Regularization-based) 기반

1) 경로 기반 방법

- 고차 정보를 전달하는 경로를 추출하여 모델에 제공함

CKG 모델을 잘 쓰기 위한 시도들

경로 기반(Path-based)과 정규화(Regularization-based) 기반

2) 정규화 기반 방법

- 추천 모델의 학습을 regularize하기 위해 모델의 구조를 나타낼 수 있는 추가적인 loss term을 제시함

이 논문의 목적

지식 그래프의 고차적인 정보를 잘 활용할 수 있는 모델의 제작

지식 그래프의 고차적인 정보를

1. 효율적이고
2. 명시적이고
3. end-to-end 방식으로 활용할 수 있도록 개발해야 함

KGAT

Knowledge Graph Attention Network

- 이웃의 임베딩을 기반으로 노드의 임베딩을 업데이트함
- 임베딩 전파를 선형 시간복잡도에서 재귀적으로 수행함
- Attention 매커니즘을 통해 전파 중에 각 이웃의 가중치를 학습함
- 가중치가 곧 고차 연결의 중요성을 나타내게 됨

KGAT의 장점

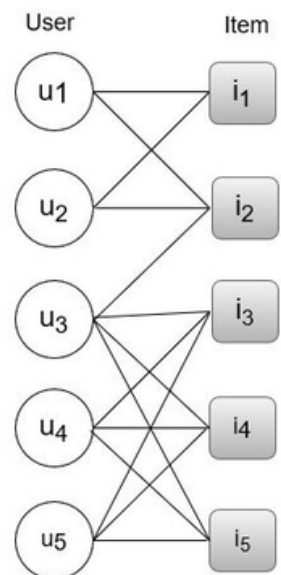
효율성, 편리성, 최적화, 맞춤화

- 경로 기반 방식보다 경로를 구체화하는 프로세스가 더 효율적임
- 좀 더 사용하기 편리함
- 고차 관계를 직접 고려하므로 매개변수가 추천에 최적화되도록 맞춤화됨

User-Item Bipartite Graph

사용자-아이템 상호작용 이력을 그래프로 나타내기

유저 집합의 u , 아이템 집합의 i , 상호작용 여부를 나타내는 y_{ui} 로 구성된 그래프 (u, y_{ui}, i)



Knowledge Graph

실제 엔티티와 관계를 통해서 아이템을 나타내기

subject-property-object로 구성된 튜플 (h, r, t)
ex) (Hugh Jackman, ActorOf, Logan)

Knowledge Graph

추가적으로, item-entity alignment를 정의하기도 함.

아이템과 엔티티 사이의 매핑 관계를 의미함

아이템 i 가 entity e 와 정렬될 수 있음을 나타내는 (i, e)

Collaborative Knowledge Graph

User 행동과 Item에 대한 지식을 통합한 관계 그래프

user 행동을 User-Item Bipartite Graph로 설정하면

Item-entity alignment에 따라 (h, r, t) 로 표현할 수 있게 됨
(단, r 은 relationship과 interact의 합집합이 가진 원소)

High-Order Connectivity

고품질의 추천을 하기 위해서는 고차 연결성을 활용해야 함

이를 위해 사용하는 개념이 바로 L-order Connectivity임

L-order Connectivity

multi-hop relation path를 통해 정의함

$e_1 \xrightarrow{r_1} e_2 \xrightarrow{r_2} \dots \xrightarrow{r_L} e_L$

- i 번째 triplet : (e_{i-1}, r_i, e_i)
- L : sequence의 길이

User Preference를 추론하기

CF 모델은 유저와 유사한 사용자의 채택을 바탕으로 선호도를 예측함.

SL 모델은 유저가 채택한 아이템과 유사한 것을 바탕으로 선호도를 예측함.

그러나 SL 모델은 관련성을 드러내지 못하므로 연결을 완전히 탐색하지 못하고 그대로 유지함

이 논문에서 제시하는 방법

end-to-end 방식으로 고차 관계를 활용하기

3가지 주요 요소로 구성되어 있음

- 1) Embedding Layer : CKG의 구조를 보존하여 각 노드를 vector로 매개변수화
- 2) Attentive embedding propagation Layer : 업데이트를 통해 이웃으로 반복적으로 임베딩을 전파
- 3) Prediction Layer : 각 AEP Layer에서 user와 item 값들을 집계하고, 이를 바탕으로 예측

Embedding Layer

지식 그래프 임베딩은 Embedding Layer의 역할을 하기 효과적임

널리 사용되는 TransR을 사용해서 임베딩을 진행함

- 지식그래프가 있을 때 traslation principle을 최적화하는 방향으로 각 entity와 relation을 임베딩함

- translation principle : $\mathbf{e}_h^r + \mathbf{e}_r \approx \mathbf{e}_t^r$

Embedding Layer

주어진 (h, r, t) 를 이용해 plausibility score을 구함

$$g(h, r, t) = \|\mathbf{W}_r \mathbf{e}_h + \mathbf{e}_r - \mathbf{W}_r \mathbf{e}_t\|_2^2$$

- 점수가 낮으면 triplet이 실제 존재할 가능성이 크다고 봄

Embedding Layer

Pairwise Ranking Loss

$$\mathcal{L}_{\text{KG}} = \sum_{(h, r, t, t') \in \mathcal{T}} -\ln \sigma(g(h, r, t') - g(h, r, t))$$

- TransR의 train 과정은 valid한 것과 아닌 것의 상대적인 순서를 고려하고 이를 pairwise ranking loss를 통해 구별을 도움
- $(h, r, t, t') = (h, r, t)$ 는 valid하지만, (h, r, t') 은 valid하지 않음
- 이 계층은 Regularizer로 작동할 뿐만 아니라 모델의 표현 능력을 향상시킴

Attentive Embedding Propagation Layers

GCN과 GAN을 통해서 계층의 목적을 수행함

GCN : Graph Convolution Network

- 고차 연결성을 통해 임베딩을 반복적으로 전파함

GAN : Graph Attention Network

- 전파의 가중치를 생성해서 연결이 가진 중요성을 보여줌

Attentive Embedding Propagation Layers

Information Propagation

한 entity는 여러 triplet에 속할 수 있음

따라서 entity는 여러 triplet 사이에 다리가 되어주고, 정보를 전파해줌

Attentive Embedding Propagation Layers

Information Propagation

ego network : valid한 지식 그래프

ego graph의 선형 결합 :
$$\mathbf{e}_{N_h} = \sum_{(h,r,t) \in N_h} \pi(h,r,t) \mathbf{e}_t$$

- $\pi(h, r, t)$ 가 전파가 얼마나 전파되었는지를 제어함

Attentive Embedding Propagation Layers

Knowledge-aware Attention

a relational attention mechanism

$$\pi(h, r, t) = (\mathbf{W}_r \mathbf{e}_t)^\top \tanh\left((\mathbf{W}_r \mathbf{e}_h + \mathbf{e}_r)\right)$$

- 이를 통해 Attention score가 두 entity의 거리에 의존적이게 됨
=> 가까울수록 더 많은 정보를 전파하게 됨

Attentive Embedding Propagation Layers

Knowledge-aware Attention

그 다음에는 softmax를 통해 정규화함

$$\pi(h, r, t) = \frac{\exp(\pi(h, r, t))}{\sum_{(h, r', t') \in N_h} \exp(\pi(h, r', t'))}$$

- 최종 attention score는 어떤 이웃 노드에 더 많은 attention이 이루어져야 하는지를 제안할 수 있게 됨

Attentive Embedding Propagation Layers

Information Aggregation

entity h 의 새로운 표현방식으로서 entity 표현 e_h 와 에고 네트워크 표현을 집계하는 것임

$$\mathbf{e}_h^{(1)} = f(\mathbf{e}_h, \mathbf{e}_{\mathcal{N}_h}).$$

이를 위해 3가지 유형의 집계기를 사용함

Attentive Embedding Propagation Layers

Information Aggregation

GCN Aggregator

$$f_{\text{GCN}} = \text{LeakyReLU}\left(\mathbf{W}(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\right)$$

두 표현을 합치고 비선형 변환을 함

Attentive Embedding Propagation Layers

Information Aggregation

GraphSage Aggregator

$$f_{\text{GraphSage}} = \text{LeakyReLU}\left(\mathbf{W}(\mathbf{e}_h || \mathbf{e}_{\mathcal{N}_h})\right)$$

비선형 변환을 통해 두 표현을 연결함

Attentive Embedding Propagation Layers

Information Aggregation

Bi-Interaction Aggregator

$$f_{\text{Bi-Interaction}} = \text{LeakyReLU}\left(\mathbf{W}_1(\mathbf{e}_h + \mathbf{e}_{\mathcal{N}_h})\right) + \text{LeakyReLU}\left(\mathbf{W}_2(\mathbf{e}_h \odot \mathbf{e}_{\mathcal{N}_h})\right)$$

두 표현 사이의 feature interaction을 고려해서 design함

- 이를 통해 affinity가 높을수록 더 많이 전파가 될 수 있게 해줌

Attentive Embedding Propagation Layers

High-order Propagation

앞서 한 내용들을 여러 계층으로 일반화하는 과정임.

고차 연결을 탐색하기 위해 더 많은 전파 계층을 쌓고 수집함.

각 단계에서 반복적으로 entity의 표현을
$$\mathbf{e}_{N_h}^{(l-1)} = \sum_{(h,r,t) \in N_h} \pi(h,r,t) \mathbf{e}_t^{(l-1)}$$
로 구함.

이를 통해 l 레이어의 h 엔티티를 표현하는데 도움을 줌.

결국, 이를 통해 속성 기반으로 연결된 노드들을 학습 과정에 끊어짐 없이 넣을 수 있도록 도와줌.

Model Prediction

최종적인 output을 내는 단계

앞서 다룬 여러가지 레이어들을 통해 유저 노드 u 와 item node i 에 대한 다양한 표현들을 획득함.

**각 레이어의 출력은 그 레이어의 u 와 i 에 대한 트리 구조로 만들어진 메시지 집합이므로,
각 출력은 서로 다른 순서의 연결 정보를 강조하게 됨.**

Model Prediction

최종적인 output을 내는 단계

layer-aggregation mechanism을 사용해서 각 표현을 합침

$$\mathbf{e}_u^* = \mathbf{e}_u^{(0)} \parallel \cdots \parallel \mathbf{e}_u^{(L)}, \quad \mathbf{e}_i^* = \mathbf{e}_i^{(0)} \parallel \cdots \parallel \mathbf{e}_i^{(L)}.$$

이를 통해 초기 임베딩을 풍부하게 할 뿐만 아니라 L 을 조정해서 전파 강도도 제어할 수 있음

Model Prediction

최종적인 output을 내는 단계

마지막으로 사용자 표현과 아이템 표현의 내적을 구해서 일치 점수를 예측함

$$\hat{y}(u, i) = \mathbf{e}_u^{*T} \mathbf{e}_i^*$$

Optimization

BPR loss를 통해 추천 모델을 최적화함

관찰된 상호작용은 더 많은 사용자 선호도를 나타내므로 관찰되지 않은 것보다 더 높은 예측 값을 할당해야 한다는 가정 하에 아래의 수식을 계산함.

$$\mathcal{L}_{CF} = \sum_{(u, i, j) \in O} -\ln \sigma(\hat{y}(u, i) - \hat{y}(u, j))$$

O : (u, i) 는 observed이고 (u, j) 는 unobserved인 (u, i, j) 로 구성된 Training set

Optimization

KGAT에서는 Pairwise Ranking Loss와 BPR loss를 이용한 새로운 지표를 정함

$$\mathcal{L}_{\text{KGAT}} = \mathcal{L}_{\text{KG}} + \mathcal{L}_{\text{CF}} + \lambda \|\Theta\|_2^2$$

(theta는 파라미터 집합임)

Optimization – Training

Adam을 이용한 training

LKG와 LCF를 대안적으로 최적화함과 동시에 mini-batch와 Adam을 사용해서 임베딩과 예측에서의 손실을 최소화함

Adam은 학습 속도에 대하여 기술기의 절대값에 대한 학습속도를 적합하게 제어할 수 있음.

특히 무작위로 샘플링된 (h, r, t, t') 배치에 대해 모든 노드에 대한 임베딩을 업데이트하고, 무작위로 (u, i, j) 배치를 샘플링한 다음 L 단계 후의 전파를 표현한 후, 예측 손실(LCF)의 기울기를 통해 모델 파라미터를 업데이트함.

Optimization – Time Complexity Analysis

alternative optimization strategy를 채택함에 따른 시간 비용은 두 부분에서 발생함

- Embedding Layer : Translation principle에서 복잡도가 다음과 같음
 - $O(|\mathcal{G}_2|d^2)$.
- Attentive Embedding Propagation Layer : 각 레이어의 곱이 가지는 복잡도가 다음과 같음
 - $O(|\mathcal{G}|d_ld_{l-1})$
- Training 과정 : 마지막 prediction 단계에서 training을 진행하는 시간 비용은 다음과 같음
 - $O(\sum_{l=1}^L |\mathcal{G}|d_l)$
- 따라서 총 training 복잡도는 다음과 같음 : $O(|\mathcal{G}_2|d^2 + \sum_{l=1}^L |\mathcal{G}|d_ld_{l-1} + |\mathcal{G}|d_l)$

방학 4주차

Experiments

과제 1

EXPERIMENTS 읽고 정리하기

방학 4주차

Conclusion

과제 2

CONCLUSION AND FUTURE WORK 읽고 정리하기

가천대학교 이다

THANK YOU

KGAT

가천대학교 이다