# 우버 이츠 **GNN** 메뉴 추천

How Uber uses GNN to recommend you food

# 0. Abstract

- 우버 이츠가 도시별로 User, Restaurant, Menu를 Node로 갖는 그래프를 과거 주문 이력(행동 데이터)을 바탕으로 구축
- 이 그래프에 GraphSAGE를 활용하여 메뉴와 유저의 노드 임베딩 도출
- 도출된 임베딩 간 유사도를 feature로 최종 추천에 활용

결과: 유의미한 on/offline 성능 향상 확인

+) 그래프 feature를 학습 & 서빙하기 위한 데이터 파이프라인 소개

참고영상: https://youtu.be/9O9osybNvyY

참고 블로그: https://eng.uber.com/uber-eats-graph-learning/

# 1. Intro

Agenda

1) Graph representation learning
2) Dish recommendation on Uber  Eats
3) Graph learning on Uber Eats

우버 이츠는 36개국 500개 도시 320,000이상의 레스토랑에서 음식 주문을 중개하는 배달 플랫폼이다.

개별 고객에게 더 나은 고객 경험을 제공하는 추천 시스템을 만들기 위해, 유저-식당-메뉴 그래프를 만들어 임베딩을 도출 후 추천 모델을 학습시켰다.

# 2. Graph learning이란

## Graph Neural Network

그래프 형식의 데이터는 어디에나 존재한다. 이러한 그래프 데이터에서는 다양한 머신러닝 태스크를 수행할 수 있다. 페이스북 친구관계 네트워크에서 사람을 Node, 친구관계를 Edge로 보면 그래프로 나타낼 수 있다.

이러한 그래프를 이용하면 다음과 같은 태스크를 수행할 수 있다.

a) Node Classification: 주어진 노드의 타입을 예측. 성별, 학력 등 다양한 분야 예측이 가능하다.
b) Link Prediction: 두 노드의 연결 여부를 예측. 추천과 밀접하게 연관이 있다.
c) Community Detection: 노드 간 밀집 클러스터 찾기. 여러 사람이 비슷한 패턴을 공유한다면 같은 그룹 혹은 학회에 속해 있을 확률이 높다.
d) Network Similarity: 두 sub-network 간의 유사도를 계산. 유저 그룹 간 유사도를 수치화할 수 있다. 그룹, 페이지 추천 등에도 이용 가능

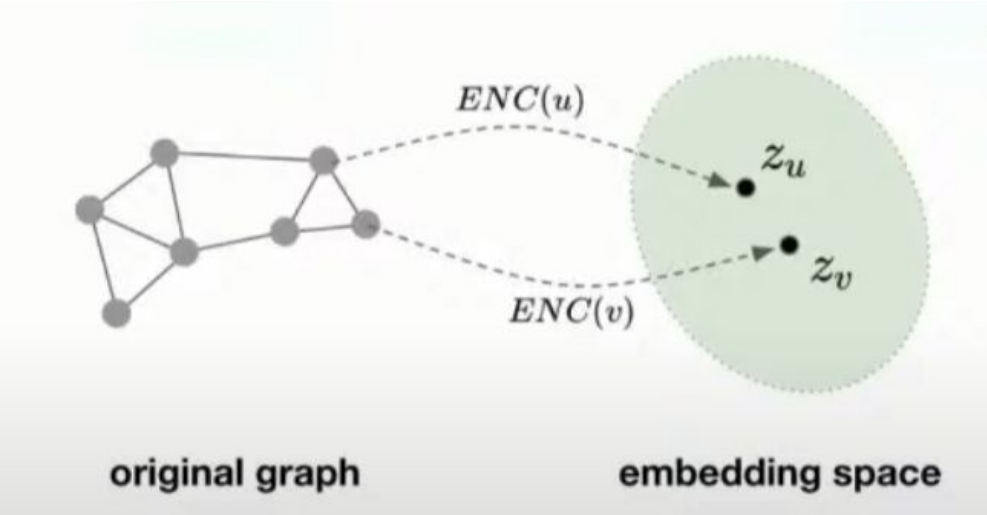
# 2. Graph learning이란

## Learning framework

모든 기계학습 문제는 학습 프레임워크에 데이터를 매핑하는 과정. 그래프 데이터에서는 노드를  특정 임베딩 공간으로 매핑하는 인코더 함수를 정의한다. 여기에 네트워크 구조에 기반한 노드 유사도 함수를 정의하여 그래프 내 유사한 노드 두 개가 있을 경우 이들이 각각 임베딩 공간으로 매핑된 결과물이 유사도 함수에 의해 계산된 유사도가 높게 나오도록 유도하는 방식으로 학습이 이루어진다.

# 2. Graph learning이란

노드 u, v가 있을 때, 두 노드 간 similarity는 각 노드의 임베딩 벡터 z_u, z_v 간의 내적과 유사해지도록 유도된다.




$$similarity(u, v) \approx z_v^\top z_u$$

# 2. Graph learning이란

따라서, 노드를 임베딩 공간 내 벡터 representation으로 매핑하여 인코딩 함수를 학습하는 것이 목표라 할 수 있다.

결과적으로는 그래프 공간 상 유사한 노드들이 우리가 결과물로 얻어낸 임베딩 공간에서도 유사한 measure를 갖도록 만들어주는 인코더 함수를 얻고자 하는 것.

요약:

1) '유사함'의 기준 정의

2) 유사도 함수 정의

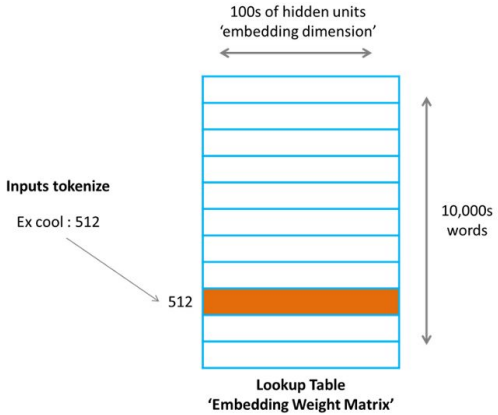
3) 인코딩 함수 정의 후 최적화

# 2. Graph learning이란

## Shallow Encoding

가장 간단한 형태의 인코딩 접근법으로, embedding lookup 테이블을 활용하는 방식.

특정 대상이 있을 때 이미 정해진 임베딩 테이블에서 대상의 index를 사용해 한 벡터를 조회하는 방식으로, 별도의 임베딩 매트릭스가 존재하고 그 매트릭스를 구성하는 한 행은 하나의 . 사이즈는 사전에 정의한 embedding size)



# 2. Graph learning이란

Shallow Encoding 단점

1) 너무 많은 파라미터 수
V개 노드에 대해 O(|V|*임베딩 차원)의 수를 가지는 파라미터를 계산해야한다.

2) unseen node에 대한 임베딩 생성 불가
MF 기반 대부분의 알고리즘의 문제로, 학습 과정에서 등장하지 않는 노드의 임베딩 생성 불가

3) 노드 feature 활용 불가
추천에 유용한 정보를 줄 수 있으며 그래프 임베딩 학습에 있어 중요 시그널이 될 수 있는 노드 feature를 활용할 수 없다.

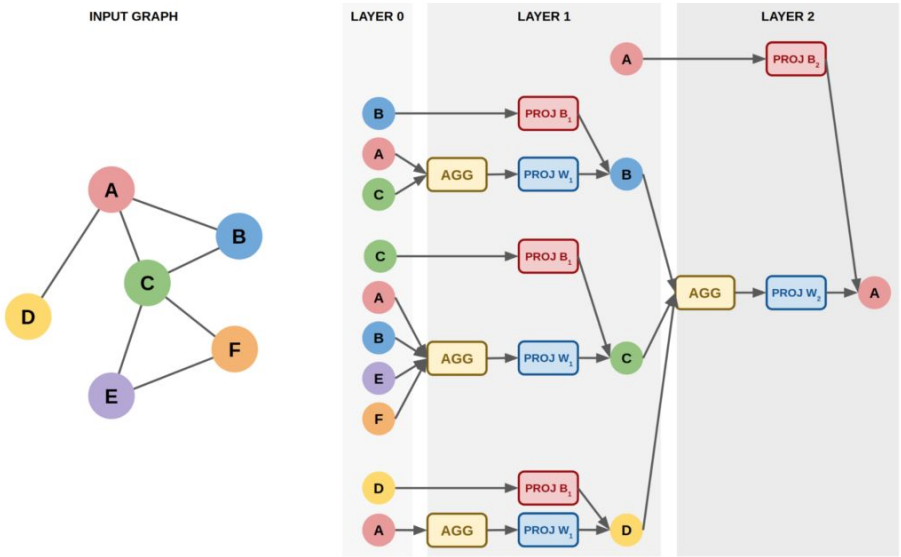
>> 새로운 방법론 필요

# 2. Graph learning이란

Graph Neural Network

Spatial GNN의 한 종류인 GraphSAGE의 핵심 아이디어는 다음과 같다.
**Node representation을 얻기 위해, 이웃으로부터 제한된 BF를 통해 재귀적으로 정보를 aggregate하기 위해 neural netw**

Aggregation Rule >



# 2. Graph learning이란

AGG 함수의 파라미터를 조정해 나가는 과정에서 학습 발생

즉, 각각의 노드에 대해서 임베딩을 학습하는 Shallow Embedding 방식이 아닌,
AGG를 구성하는 파라미터를 최적화 함으로써 임베딩 값을 계산하는 'Rule'을 생성

>> 새로운 노드에 대한 임베딩 계산 가능

이러한 학습 방식 덕분에 Inductive한 예측 수행 가능
(데이터 사례에서 보편 룰을 도출)

# 2. Graph learning이란

Neighborhood Aggregation

전체 모델 네트워크는 여러 개의 레이어를 가지게 되는데, 각 레이어는 BFS상 깊이의 한 레벨을 의미한다. 이때 노드들은 각 레이어마다 임베딩을 가지게 된다. (레이어1의 노드 b 임베딩 != 레이어2의 노드 b 임베딩)

특정 노드의 레이어별 임베딩은 다를 수 있지만 다른 노드로 유입되는 노드의 임베딩 웨이트는 레이어마다 같다.
(레이어1에서 노드 a로 유입되는 노드 b의 임베딩 웨이트 = 레이어1에서 노드 c로 유입되는 노드 b의 임베딩 웨이트)

# 2. Graph learning이란

이러한 구조로 인해 scalability를 얻게 되고, 그래프 내 노드 개수에 학습이 영향을 받지 않게 된다. 노드 수보다 뉴럴넷 내 뉴런 수에 더 큰 영향을 받는 것 (모든 뉴럴넷 파라미터가 전체 노드에서 공유되므로)

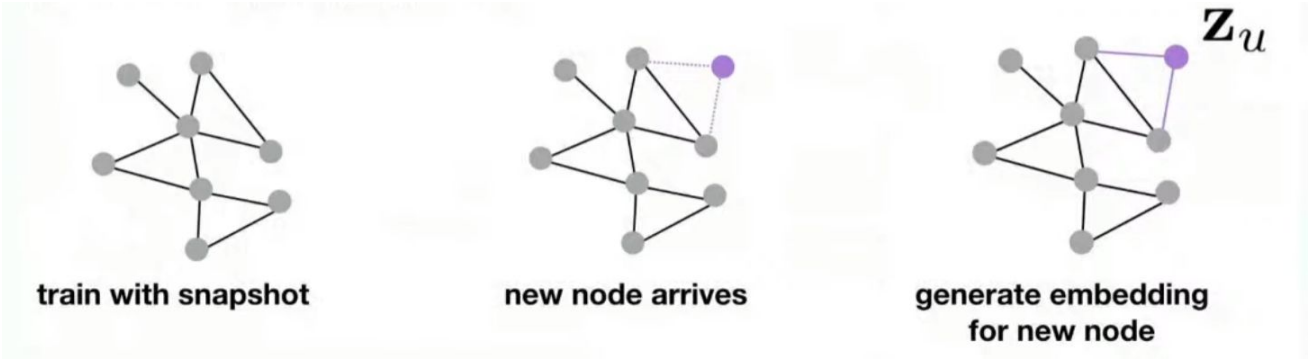
특정 노드가 레이어0에서 갖는 임베딩 값은 그 자체의 입력피쳐
> 입력값 자체에서 임베딩을 사용한 것이 아님
> Aggregation Rule만 레이어 층에서 업데이트

# 2. Graph learning이란

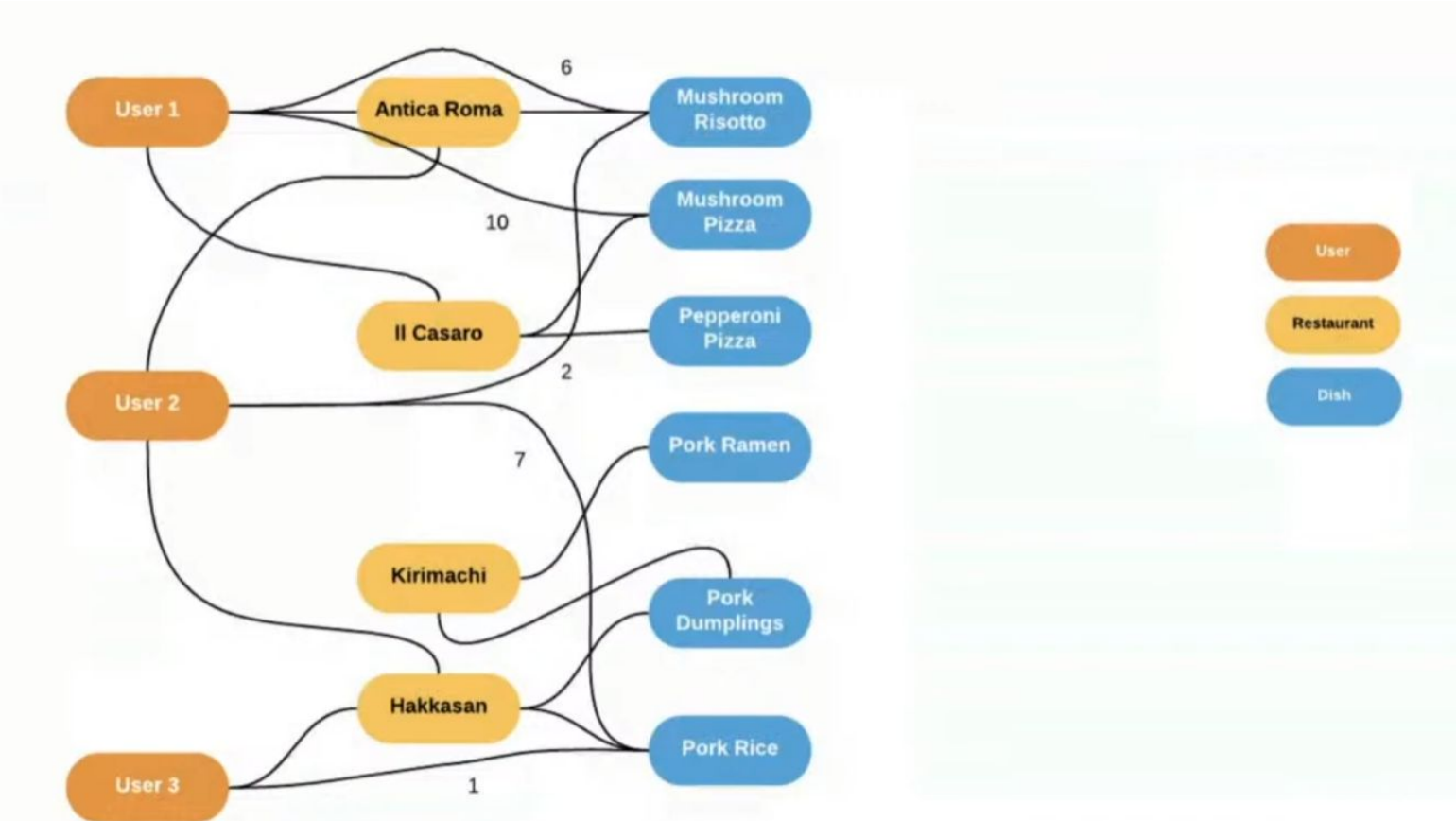
Inductive capability

Unseen Node에 대해서 재학습 과정 없이 임베딩을 부여할 수 있는 특성

임베딩을 만들어내는 함수의 파라미터를 기존 그래프 snapshot으로 학습하여 unseen node에 대한 임베딩을 만들어 낼 수 있도록 디자인



# 3. 우버 이츠 Graph learning 기반 추천



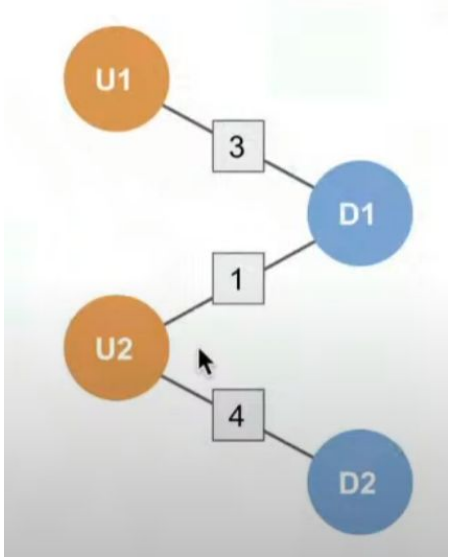
# 3. 우버 이츠 Graph learning 기반 추천

Bipartie graph for dish recommendation

유저와 메뉴의 interaction으로부터 bipartie graph 생성
> 두 노드(유저, 메뉴)를 연결하는 edge에는 음식 주문 빈도가 weight로 사용됨

Uber Eats Graph property

1) Graph is dynamic: 새로운 유저와 음식이 매일 추가됨
2) 각 노드의 음식 이름은 word2vec 값과 같은 피쳐를 가짐



# 3. 우버 이츠 Graph learning 기반 추천

손실 함수: Max Margin Loss

수치적인 유사도보다는 상대적인 랭킹에 초점을 맞춘 손실 함수

$$L = \Sigma_{(u,v)\in E} max(0, -z_u^\top z_v + z_u^\top z_n + \Delta)$$

v: 유저 u에게 positive한 노드 / n: 별도 샘플링한 negative 노드

delta라는 margin을 붙인 이유
> Positive pair인 (z_u, z_v)와 Negative pair인 (z_u, z_n) 사이의 유사도 점수의 차이가 최소 delta만큼 발생

# 3. 우버 이츠 Graph learning 기반 추천

위와 같은 방식으로 샘플들 간 절대적 유사도보다는 negative와 positive 샘플 간의 상대적인 유사도를 기준으로 학습 수행

# 3. 우버 이츠 Graph learning 기반 추천

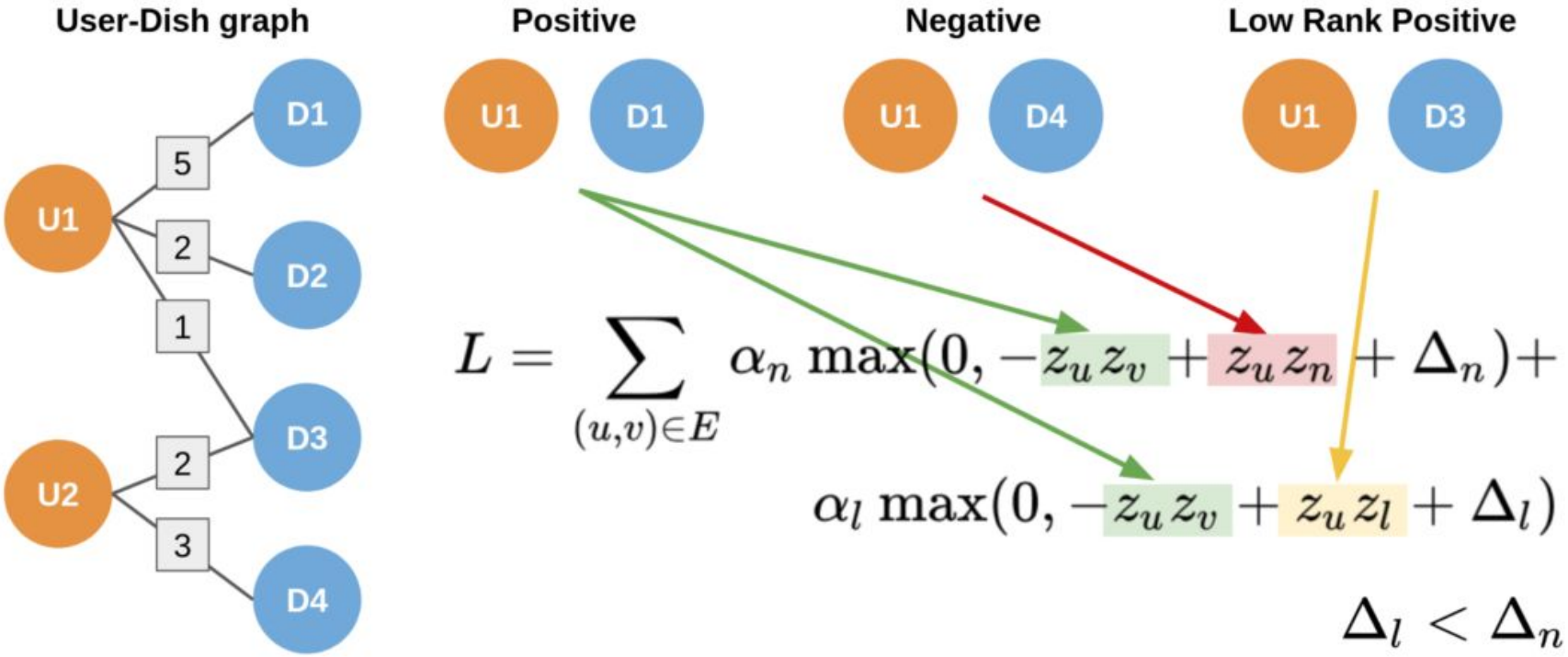
New loss with Low rank positives

Low rank positives는, negative는 아니지만 positive라고 하기 애매한 정도의 샘플을 의미한다. 음식 주문에 있어, 주문한 적은 있지만 특정 이유로 자주 주문하지 않는 경우가 이에 해당한다.

negative와 positive를 비교하는 것 뿐만 아니라 low rank positive와 positive 또한 비교하도록 학습을 설계함으로써 성능 향상을 유도한다.

아이디어를 요약하자면, 더 어려운 문제를 주어 세심함을 높인다 라고 정리할 수 있다.

# 3. 우버 이츠 Graph learning 기반 추천




$$L = \Sigma_{(u,v)\in E} \alpha_n max(0, -z_u^\top z_v + z_u^\top z_n + \Delta_n) + \alpha_l max(0, -z_u^\top z_v + z_u^\top z_l + \Delta_l)$$

# 3. 우버 이츠 Graph learning 기반 추천

negative와 positive를 비교하는 Max Margin Loss에 low rank positive를 비교하는 Max Margin Loss 항을 추가한 것으로, margin delta는 전자에서가 후자에서보다 크다.

$$\Delta_n > \Delta_l$$

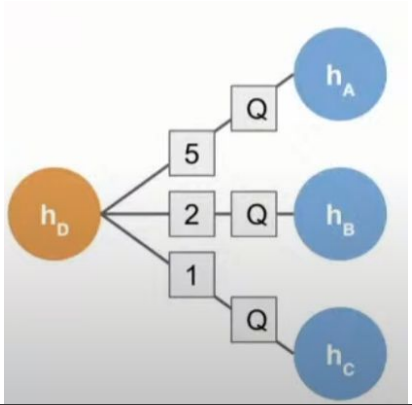
# 3. 우버 이츠 Graph learning 기반 추천

Weighted pool aggregation

edge의 weight에 기반하여 이웃의 임베딩을 aggregation하는 방법

즉, 특정 노드의 임베딩을 swallow 방식이 아닌 weight parameter의 계산 결과로써 획득한 뒤, 주변의 임베딩 결과값을 결합하는 식

*Q는 Fully Connected Layer




$$AGG = \Sigma_{u \in N(v)} w(u, v) Q h_u^{k-1}$$

# 3. 우버 이츠 Graph learning 기반 추천

Offline evaluation

위 방식으로 Downstream personalized ranking을 그래프 노드 임베딩을 사용해 학습했고, 다음과 같은 성능을 얻었다.

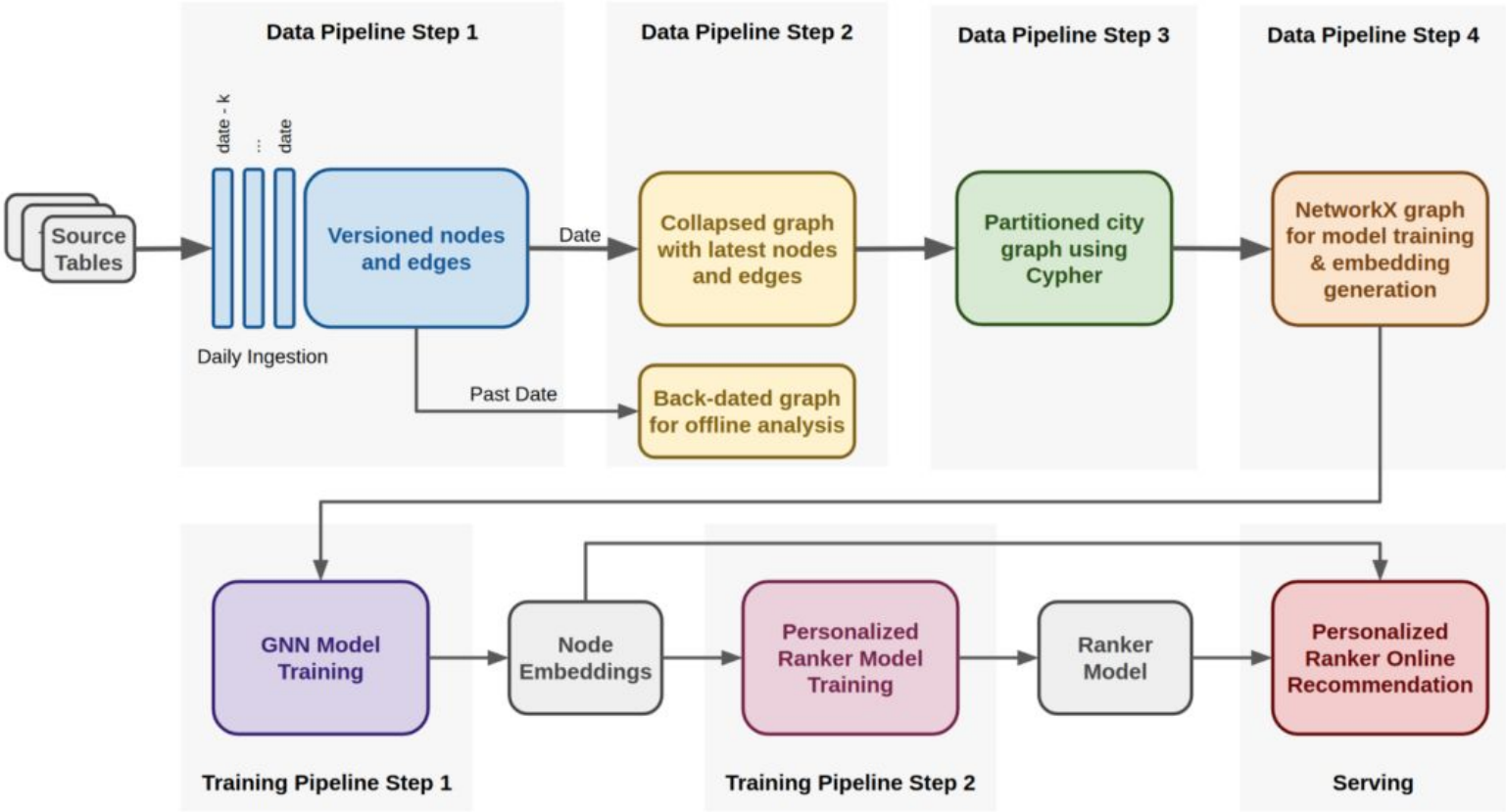
| Model | Test AUC |
| --- | --- |
| 이전의 프로덕션 모델 | 0.784 |
| graph embedding을 추가한 모델 | **0.877** |

# 3. 우버 이츠 Graph learning 기반 추천

## Online evaluation

샌프란시스코 내 A/B 테스트 결과 CTR 측면에서 유의미한 uplift 확인
> 전 지역 확대 예정

# 4. 데이터 및 학습 파이프라인



# 4. 데이터 및 학습 파이프라인

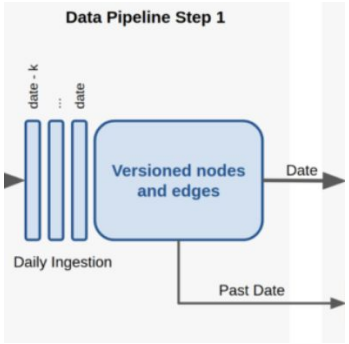
Data Pipeline Step 1

첫번째 파이프라인. Apache Hive Table로부터 데이터를 load하는 다수의 job 실행

Parquet 파일 형태로 테이블을 ingest하여 HDFS로 옮김

이 parquet 파일 내에는 노드와 엣지 정보가 포함되어있음

각 노드와 엣지 정보는 timestamp에 따라 버저닝 된 properties를 가지고 있으며 과거 시간을 기준으로 그래프를 구축하는

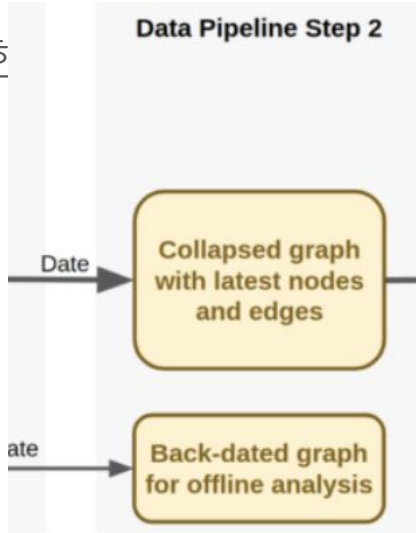


# 4. 데이터 및 학습 파이프라인

## Data Pipeline Step 2

특정 날짜가 주어졌을 때 각각의 노드와 엣지의 최신 특성을 유지하고, 이들을 HDFS에 Cypher 포맷을 이용해 저장한다.

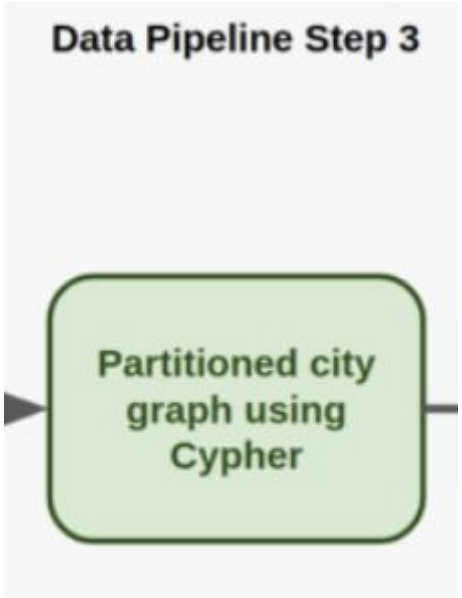
프로덕션 모델을 학습할 때는 현재 날짜를 특정 날짜로 사용한



# 4. 데이터 및 학습 파이프라인

Data Pipeline Step 3

Apache Spark 수행 엔진 내에서 Cypher Query 언어를 사용해 도시별로 파티션된 복수의 그래프를 생성한다.



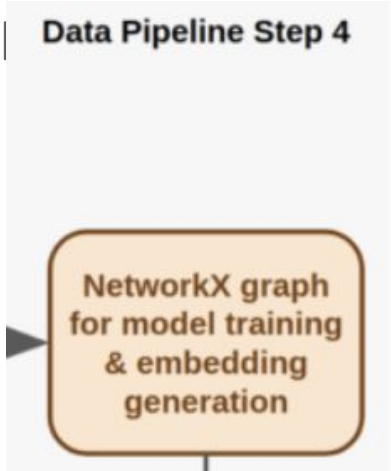
# 4. 데이터 및 학습 파이프라인

Data Pipeline Step 4

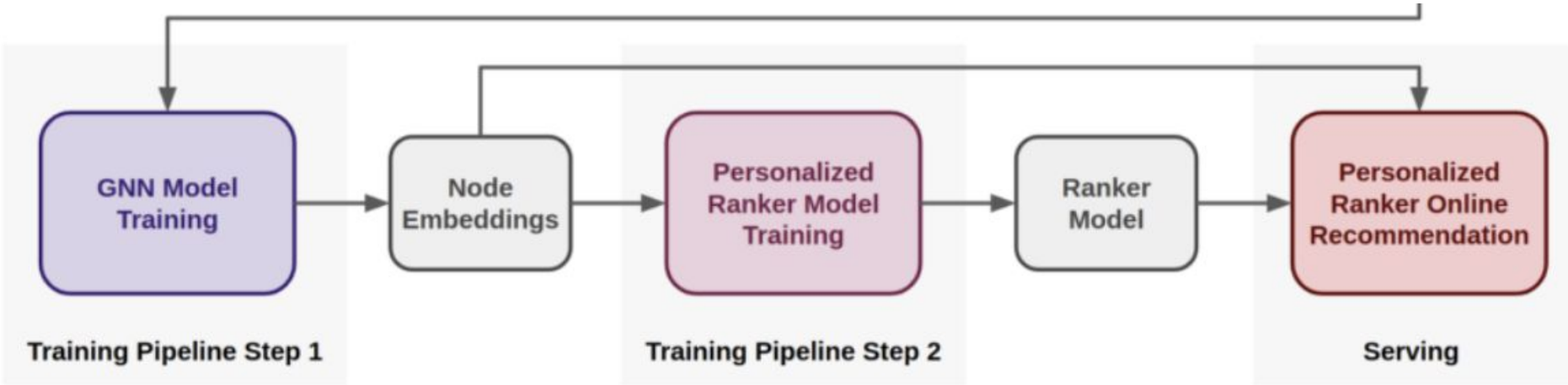
도시별 그래프를 networkx 그래프 포맷으로 변환하여 모델 학습과 임베딩 생성 과정에 투입

이 과정을 텐서플로우를 기반으로 수행되며 GPU를 이용

생성된 임베딩은 request 발생 시 조회되는 lookup 테이블에 저장된다.



# 4. 데이터 및 학습 파이프라인



# 5. 마치며

GNN에서는 link prediction으로 직접적인 추천을 진행하는 것만을 생각했었는데, 우버 이츠가 사용한 방식처럼 feature를 생성하는데에 GNN을 사용한다면 unseen data에 대한 cold start 문제도 무난하게 처리 가능할 것이라는 생각이 들었다.

아이디어가 상당히 직관적이고 간단하게 느껴질 정도로 깔끔한 세미나였기에 꼭 봐보았으면 한다.