Apache S2Graph 기반 머신러닝 모델 환경 구축

글 | 윤도영 shon.0@kakaocorp.com 카카오에서 데이터 엔지니어로 일하고 있고, 데이터 파이프 라인과 머신러닝에 관심이 많은 개발자입니다. 현재 Apache S2Graph(incubating) 프로젝트를 리드하고 있고, 카카오의 그래프 DB팀에서 일하고 있습니다.

서비스에 머신러닝(machine learning)을 적용하기 위해서는 데이터 수집, 데이터 전처리, 모델 생성, 서비스에 API 제공 등 각각의 쉽지 않은 일들을 잘 처리해야 합니다.

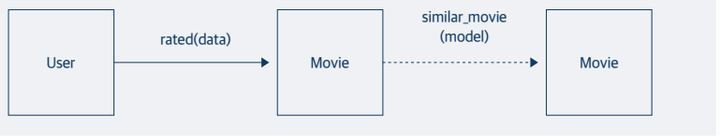
지금까지 좋은 모델을 생성하기 위해서는 어떤 알고리즘을 활용해야 하는지에 대해서는 많이 공유되었습니다. 하지만 모델을 만드는 것만큼 중요하다고 여겨지는, 만들어진 모델을 서비스에 잘 적용하고 쉽게 실험해볼 수 있는 환경을 구축하는 문제에 대한 공유는 많지 않았습니다.

이 글에서는 누구나 쉽게 따라 해볼 수 있도록 실제 데이터가 아닌, 모두에게 공개된 MovieLens(무비렌즈) 데이터 세트를 사용하여 영화 추천이라는 친숙한 도메인 예제를 통해 그래프 데이터베이스를 활용한 머신러닝 모델(machine learning model)과 데이터를 하나로 추상화하는 과정을 살펴보도록 하겠습니다. 이 글의 모든 내용은 깃허브(GitHub)의 MovieLens 데이터 튜토리얼[*1]을 통해 직접 실행해볼 수 있습니다.

The abstraction(개요)

MovieLens 데이터 세트를 S2Graph의 기반 모델인 프로퍼티 그래프(Property Graph) 모델로 모델링해보면 다음과 같습니다.

[그림 1] MovieLens 데이터 세트의 프로퍼티 그래프 모델링



그리고 지금부터는 MovieLens 데이터 세트의 Vertex, Edge의 표현 방식을 자세하게 살펴보도록 하겠습니다.

1) Vertex: 데이터에 존재하는 노드(node)들을 표현합니다.

(1.1) Movie: 사용자들이 MovieLens 데이터 세트에서 평점을 부여한 영화들

(1.1) 예시

movieId, title, genres
1, Toy Story(1995), Adventure|Animation|Children|Comedy|Fantasy
2, Jumanji(1995), Adventure|Children|Fantasy
3, Grumpier Old Men(1995), Comedy|Romance
...

*1 참고 | https://github.com/apache/incubator-s2graph/tree/master/example/movielens

(1.2) User: 영화에 평점을 부여한 사용자들

```
(1.2) 예시
userId
1
2
3
...
```

2) Edge: Vertex와 Vertex 간의 관계를 표현합니다.

(2.1) Rated: 사용자들이 영화에 부여한 평점

```
(2.1) 예시
userId, movieId, rating, timestamp
1, 31, 2.5, 1260759144
1, 1029, 3.0, 1260759179
1, 1061, 3.0, 1260759182
...
```

(2.2) similar_movie: 비슷한 영화들 간의 관계를 표현

이 글에서 가장 중요한 부분으로 ALS 알고리즘을 통해 만들어진 머신러닝 모델이 저장될 관계를 표현합니다. 다른 관계(Edge)들은 MovieLens 데이터 세트에 존재하는 데이터를 저장소에 저장하고, 저장된 원래의 데이터(raw data)를 조회하는 방식이지만, similar_movie는 사용자들의 평점 기록을 통해 ALS 알고리즘이 생성한 모델의 예측 결과를 조회하는 방식이라는 차이가 있습니다.

The technologies stack: 카카오 활용 오픈소스

다음은 실제 카카오에서 활용되는 오픈소스들입니다. 카카오에는 사용자가 검색한 검색어, 사용자가 본 콘텐츠, 사용자가 맺은 친구 관계 등 다양한 종류의 데이터들이 존재합니다. 많은 경우에 이렇게 모인 데이터들을 다음과 같은 기술 스택(technologies stack)으로 저장하여 모델을 생성하고, 생성된 모델을 서비스에 적용하고 있습니다.

Apache S2Graph[*2]

카카오에서 개발하고 Apache 소프트웨어 재단에 오픈소스로 제공한 그래프 데이터베이스로, 이 튜토리얼에서는 MovieLens 데이터 세트 전체를 저장하는 저장소로 사용됩니다. S2Graph는 S2GraphQL이라는 REST 인터페이스(REpresentational State Transfer Interface)를 통해 저장된 원래의 데이터뿐 아니라 머신러닝 모델도 조회하는 기능을 제공합니다.

*2 참고 | https://s2graph.apache.org

Apache Spark[*3]

Apache Spark(아파치 스파크)를 활용해 MovieLens 데이터 세트를 가공하고, MLLib이라는 머신러닝 라이브러리(machine learning library)의 ALS 알고리즘을 활용하여 모델을 생성합니다. 현재까지는 하나의 라이브러리로 Apache Spark을 활용했지만, 다른 머신러닝 라이브러리들도 충분히 활용 가능합니다.

Annoy4s[*4]

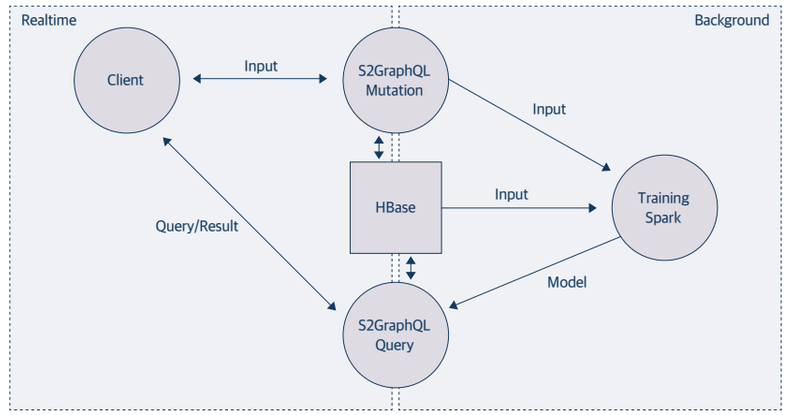
Spark MLLib의 ALS 알고리즘을 통해 만든 모델을 그대로 사용하지 않고, annoy[*5]라는 라이브러리를 통해 approximate nearest neighbors(근사 근접 이웃)를 찾는 인덱스(index)를 생성하고, 이 인덱스를 최종적으로 저장소에 저장하여 사용자들이 좋아할 만한 영화를 찾아주는 태스크(task)에 활용합니다.

The architecture(구조도)

전체 프로세스 설명

[그림2]와 같은 형태로 모델링되는 MovieLens 데이터 세트를 가지고 간단하게 영화 추천 서비스를 구성하는 과정을 설명해보도록 하겠습니다.

[그림 2] Apache S2Graph의 구조도[*6]



*3 참고 | https://spark.apache.org
*4 참고 | https://github.com/annoy4s/annoy4s
*5 참고 | https://github.com/spotify/annoy
*6 참고 | https://github.com/apache/incubator-s2graph/tree/master/example/movielens#the-architecture
*7 참고 | https://github.com/apache/incubator-s2graph/tree/master/example/movielens#description

이 글에서는 협업 필터링(collaborative filtering)의 한 종류인 ALS 알고리즘을 이용해 사용자들이 매긴 영화 평점 데이터(rated edge) 입력값(input)을 토대로 비슷한 영화를 알려주는 모델을 구축하고, 이를 기반으로 사용자의 반응을 분석하여 선호할 만한 영화들을 추천해주는 평점 기반의 추천 서비스를 소개하려고 합니다. 실제 튜토리얼 정보는 MovieLens 데이터 세트의 디스크립션(description)[*7] 부분을 참고하시면 됩니다.

(1) Schema 생성

먼저 사용자의 데이터를 그래프로 모델링하는 과정을 가장 먼저 고민해야 합니다. 대부분 직관적으로 그림을 그리고 관계를 연결하다 보면 자연스레 모델링이 되는데, 이 부분이 그래프 데이터베이스의 가장 큰 장점입니다. 실제 데이터에서 존재하는 Vertex를 schema로 생성하고, 이 Vertex들 간의 관계도를 구축하고 나면 데이터를 수집·저장할 준비가 됩니다.

(2) Data import

Schema를 생성한 뒤에는 S2Graph의 API를 활용하여 데이터를 저장소에 저장합니다. S2Graph는 실시간으로 발생하는 사용자 데이터를 뛰어난 성능으로 저장할 수 있는 저장소로, 카카오 내에서 널리 활용하고 있습니다. 특히, S2Graph는 대량으로 작업이 가능한 벌크로드(bulkload) 기능을 제공함으로써 100억 건 이상 되는 대량의 데이터도 한 번에 저장할 수 있게 해줍니다. 이를 통해 물리적으로 분리되어 있는 여러 데이터베이스의 대량의 데이터를 한곳에 그래프 형태로 저장할 수 있습니다. 이 글의 MovieLens 데이터 세트에서는 평점(rated) 데이터가 가장 큰 데이터로 이를 벌크로드 기능을 통해 저장하면 아래와 같은 기본적인 조회가 가능해집니다.

[그림 3] 1번 사용자가 평점을 매긴 영화 10편(왼쪽), 제목에 Toy를 포함한 영화 5편(오른쪽)

1번 사용자가 평점을 매긴 영화 10편

```
query {
  movielens {
    User(id: 1) {
      rated(limit: 10) {
        Movie {
          title
        }
      }
    }
  }
}
```

제목이 Toy를 포함한 영화 5편

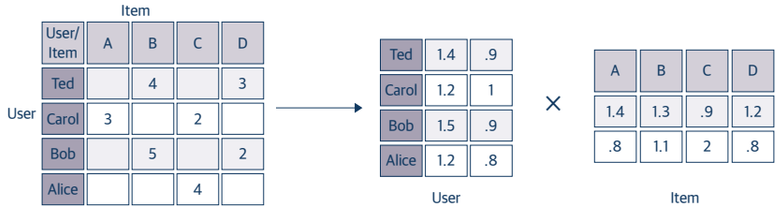
```
query {
  movielens {
    Movie (search: "title: *Toy*", limit: 5) {
      title
    }
  }
}
```

(3) ALS 알고리즘을 활용한 비슷한 영화 모델 생성

모델을 생성하기 위해서는 ALS 알고리즘을 이용하여 사용자들의 평점 데이터를 행렬분해(Matrix Factorization)합니다. 우리의 목표는 비슷한 영화를 찾는 것이기에 행렬분해 결과 중 항목(item)에 해당하는 행렬(matrix)만 annoy 알고리즘의 입력값(input)으로 사용합니다.

다음으로 annoy 알고리즘을 활용하여 근사 근접 이웃 탐색(approximate nearest neighbor search)이 가능한 annoy 지수(index)를 생성합니다.

[그림 4] 추천 데이터 계산*8



*8 참고 | https://mapr.com/ebooks/spark/08-recommendation-engine-spark.html

(4) 생성 모델의 schema 등록

위에서 생성한 annoy 인덱스로 S2GraphQL의 update schema API를 통해 schema를 업데이트 합니다. 업데이트가 완료된 후 similar_movie 관계에 대해 조회를 해보면, 저장된 원래 데이터를 조회하는 대신 annoy 인덱스를 활용해서 3번에서 만든 모델에 기반한 비슷한 영화 목록을 추천해 주도록 시스템이 변경된 것을 볼 수 있습니다.

(5) GraphQL을 통해 데이터 조회 및 API 제공

[그림 5]처럼 2에서 저장된 원 데이터와 4에서 등록된 모델 데이터를 합쳐 데이터를 조회할 수 있습니다. 특히 눈여겨볼 부분은 similar_movie를 조회하는 부분인데, 데이터를 조회하는 사용자는 해당 데이터가 직접 저장한 원래의 데이터인지, 모델이 다이나믹하게 만들어내는 데이터인지 알지 못하고 사용하게 됩니다.

보통 일반 그래프 데이터베이스는 사용자가 미리 관계를 저장해놓은 데이터를 조회하는 기능만을 지원합니다. 그러나 앞에서처럼 모델 서빙(model serving)을 지원하는 S2Graph는 미리 준비한 관련 데이터가 없어도 GraphQL을 통한 완벽한(seamless) 인터페이스로 실시간(dynamic) 유사도 계산과 직관적인 조회를 지원합니다.

[그림 5] 1번 사용자가 평점을 부여한 10편 영화와 비슷한 5편의 영화(왼쪽), 코미디 장르의 성격을 가지고 제목에 '1955'를 포함하는 5편의 영화(오른쪽)

1번 사용자가 평점을 매긴 영화 10편과 비슷한 영화 5편(item based cf)

```
query {
  movielens {
    User(id: 1) {
      rated(limit: 10) {
        Movie {
          title
          similar_movie(limit: 5) {
            Movie {
              title
            }
          }
        }
      }
    }
  }
}
```

Comedy가 포함된 장르를 가지고 있고, 제목에 1995를 포함하는 영화들과 비슷한 영화들 5편

```
query {
  movielens {
    Movie(search: "genres: *Comedy* AND title:
*1995*", limit: 5) {
      title
        genres
          similar_movie(limit: 5) {
            Movie {
              title
              genres
            }
          }
    }
  }
}
```

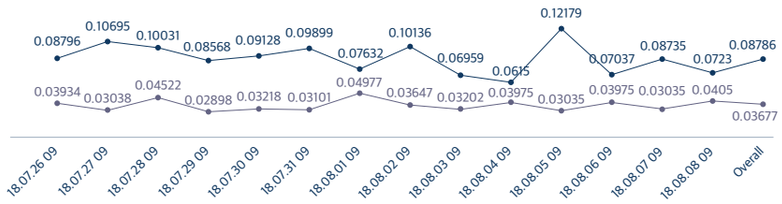
(6) API를 활용한 A/B 실험, 모델 개선

사용자가 좋아할 만한 영화들을 찾는 태스크(task)는 여러 가지 방식(서비스 로직(logic), 알고리즘)으로 시도해볼 수 있는데, 이 시도들 가운데 실제로 사용자들의 반응률이 가장 좋은 방식을 데이터에 기반해 결정할 수 있도록 S2Graph에서는 A/B 테스트 환경을 자체적으로 제공합니다.

인기 있는 상위 10편의 영화를 모든 사용자에게 보여주는 기존 로직 A가 있다고 가정합니다. 그리고 새롭게 ALS 알고리즘을 통해 만든 로직 B를 제공하려고 합니다. 두 개의 로직 중 실제 사용자들의 선호가 높은 방법을 판단하기 위해 5번처럼 로직 B 쿼리(query)를

S2Graph 어드민(admin)에 등록합니다. 쿼리를 등록한 이후에는 각각의 로직에 전체 트래픽 중 몇 %를 할당할지 정할 수 있습니다. 또한 A/B 각각의 쿼리들은 자신의 고유한 ID를 발급받습니다. 실제 사용자가 추천 결과에 노출, 클릭 등의 반응을 하면, 해당 고유 ID의 피드백(feedback)을 클라이언트(client)로부터 S2Graph 서버로 전달받습니다. 이런 과정을 거쳐 다음과 같이 어떤 로직이 효율적이고 좋은 반응을 얻었는지 확인 가능합니다.

[그림 6] A/B 두 가지 로직을 동시에 실험하여 각 로직별로 기존 대비 효율을 확인하는 예



마치며

카카오에서는 앞서 설명한 Apache S2Graph라는 그래프 데이터베이스를 활용하여 많은 부문에서 데이터 기반 의사결정과 액션이라는 목표를 달성해오고 있습니다. 단순히 추천이라는 애플리케이션 외에도 검색, 광고, 사용자 타게팅 등 카카오 서비스 내 여러 분야에서 활용되고 있는 Apache S2Graph가 궁금하면 깃허브의 내용을 참조해 주시길 바랍니다.[*9]

머신러닝 기술을 카카오 서비스에 적용하는 데이터 파이프라인(data pipeline)을 개발, 운영하면서 몇 가지 느낀 점이 있습니다. 간단한 알고리즘 · 모델이라도 서비스에 쉽고 빠르게 적용하고, 기존보다 끊임없이 효율을 개선하는 과정이 중요합니다. 그리고 무엇보다 데이터와 모델을 서비스에 쉽게 적용하고 테스트해볼 수 있는 환경을 갖추는 것이 더 나은 서비스를 제공하기 위해 필수적임을 말씀드리며 글을 마무리하겠습니다.

〈Apache S2Graph 기반 머신러닝 모델 환경 구축〉 브런치로 연결되는 QR 코드입니다.

*9 참고 | https://github.com/apache/incubator-s2graph